vmware Tanzu Docs

Pivotal Container Service (PKS) Documentation v1.6

Published: November 14, 2019

Note: The contents of this PDF may have fallen out of date. For current documentation, see https://docs.pivotal.io/pks/1-6

VMware Enterprise PKS

In this topic

Page last updated:

VMware Enterprise PKS enables operators to provision, operate, and manage enterprise-grade Kubernetes clusters using BOSH and Pivotal Ops Manager.

Overview

Enterprise 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 Enterprise 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.

On Pivotal Platform, you can run Enterprise PKS standalone or alongside Pivotal Application Service.

What Enterprise PKS Adds to Kubernetes

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

Feature	Included in K8s	Included in Enterprise 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

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

Feature Support by IaaS

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

* 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 Linux Workloads*.

Enterprise 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 Enterprise PKS Cluster Management.

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

Enterprise PKS Concepts

For conceptual information about Enterprise PKS, see Enterprise PKS Concepts.

Enterprise PKS Prerequisites

For information about the resource requirements for installing Enterprise 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 Enterprise PKS on vSphere with NSX-T
- GCP Prerequisites and Resource Requirements
- AWS Prerequisites and Resource Requirements
- Azure Prerequisites and Resource Requirements

Preparing to Install Enterprise PKS

To install Enterprise PKS, you must deploy Ops Manager. You use Ops Manager to install and configure Enterprise PKS.

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

Consult the following table for compatibility information:

IaaS	Ops Manager v2.6.16+ or v2.7.6+	NSX-T	Harbor
vSphere	Required	Available	Available
GCP	Required	Not Available	Available
AWS	Required	Not Available	Available
Azure	Required	Not Available	Available

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

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

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

Installing Enterprise PKS

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

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

Upgrading Enterprise PKS

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

Managing Enterprise PKS

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

Managing Kubernetes Clusters and Workloads

For information about managing Enterprise PKS-provisioned Kubernetes clusters and deploying workloads, see Managing Kubernetes Clusters and Workloads.

Backing Up and Restoring Enterprise PKS

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

Enterprise PKS Security

For information about security in Enterprise PKS, see Enterprise PKS Security.

Diagnosing and Troubleshooting Enterprise PKS

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

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

Release Notes

In this topic

Page last updated:

This topic contains release notes for VMware Enterprise PKS v1.6.

warning: Before installing or upgrading to Enterprise PKS v1.6, review the Breaking Changes below.

v1.6.2

Release Date: April 29, 2020

Features

New features and changes in this release:

- Bumps Kubernetes to v1.15.10.
- Bumps UAA to v73.4.20.
- Bumps Percona XtraDB Cluster (PXC) to v0.22.
- Bumps Windows Stemcell to v2019.15.
- Bumps ODB to v0.38.0.
- Bumps Apache Tomcat (in PKS API) to v9.0.31.
- [**Security Fix**] UAA bump fixes blind SCIM injection vulnerability, CVE-2019-11282.
- [**Security Fix**] UAA bump fixes CSRF attack vulnerability.
- [**Security Fix**] PXC bump fixes cURL/libcURL buffer overflow vulnerability, CVE-2019-3822.
- [**Bug Fix**] Improves the behavior of the `pks get-kubeconfig` and `pks get-credentials` commands during cluster updates and upgrades. You can now run the `pks get-kubeconfig` command during single- and multi-master cluster updates. Additionally, you can run the `pks get-credentials` command during multi-master cluster upgrades.
- [**Bug Fix**] New UAA version includes Apache Tomcat bump that fixes SAML login issues.

Product Snapshot

Element	Details
Version	v1.6.2
Release date	April 29, 2020
Compatible Ops Manager versions	See Pivotal Network
Xenial stemcell version	See Pivotal Network
Windows stemcell version	v2019.15
Kubernetes version	v1.15.10
On-Demand Broker version	v0.38.0
Compatible NSX-T versions	v2.5.1, v2.5.0, v2.4.3
NCP version	v2.5.1
Docker version	v18.09.9
Backup and Restore SDK version	v1.17.0

UAA version	v73.4.20

vSphere Version Requirements

For Enterprise PKS installations on vSphere or on vSphere with NSX-T Data Center, refer to the VMware Product Interoperability Matrices.

Upgrade Path

The supported upgrade paths to Enterprise PKS v1.6.2 are from Enterprise PKS v1.5.0 and later.

Breaking Changes

All breaking changes in Enterprise PKS v1.6.2 are also in Enterprise PKS v1.6.0. See Breaking Changes in Enterprise PKS v1.6.0.

Known Issues

All known issues in Enterprise PKS v1.6.2 are also in Enterprise PKS v1.6.0. See Known Issues in Enterprise PKS v1.6.0.

v1.6.1

Release Date: Jan 13, 2020

Features

New features and changes in this release:

- [**Security Fix**] Secures traffic into Kubernetes clusters with up-to-date TLS (v1.2+) and approved cipher suites.
- [**Security Fix**] Bumps UAA to v73.4.16. This update prevents logging of secure information and enables the PKS UAA to start with the `env.no_proxy` property set.
- [**Bug Fix**] Resolves an issue where if you are using Ops Manager v2.7 and PKS v1.6 as a fresh install, enabling Plans 11, 12, or 13 does not enable Windows worker-based clusters. It creates Linux-based clusters only. For more information, see Enterprise PKS Creates a Linux Cluster When You Expect a Windows Cluster.
- [**Bug Fix**] Resolves an issue where applying changes to Enterprise PKS fails if Plan 8 is enabled in the Enterprise PKS tile. For more information, see Applying Changes Fails If Plan 8 Is Enabled.
- [**Bug Fix**] Resolves an issue where the `pks update-cluster --network-profile` command sets `subnet_prefix` to 0 in the ncp.ini file if the network profile does not have `pod_subnet_prefix`. For more information, see Network Profile for “pks update-cluster” Does Not Use the Defaults from the Original Cluster Manifest.
- [**Bug Fix**] Resolves an issue where trying to create a cluster with a long network profile causes an error `Data too long for column 'nsxt_network_profile'`.
- Updates the supported NCP version to NCP v2.5.1. Refer to the NCP Release Notes for more information.
- Support for NSX-T v2.5.1.

Product Snapshot

Element	Details
Version	v1.6.1
Release date	January 13, 2020
Compatible Ops Manager versions	See Pivotal Network
Xenial stemcell version	See Pivotal Network
Windows stemcell version	v2019.7
Kubernetes version	v1.15.5
On-Demand Broker version	v0.29.0
Compatible NSX-T versions	v2.5.1, v2.5.0, v2.4.3
NCP version	v2.5.1
Docker version	v18.09.9
Backup and Restore SDK version	v1.17.0
UAA	v73.4.16

vSphere Version Requirements

For Enterprise PKS installations on vSphere or on vSphere with NSX-T Data Center, refer to the VMware Product Interoperability Matrices.

Upgrade Path

The supported upgrade paths to Enterprise PKS v1.6.1 are from Enterprise PKS v1.5.0 and later.

Breaking Changes

All breaking changes in Enterprise PKS v1.6.1 are also in Enterprise PKS v1.6.0. See Breaking Changes in Enterprise PKS v1.6.0.

Known Issues

All known issues in Enterprise PKS v1.6.1 are also in Enterprise PKS v1.6.0. See Known Issues in Enterprise PKS v1.6.0.

v1.6.0

Release Date: November 14, 2019

Features

This section describes new features and changes in this release.

PKS Control Plane and API

Enterprise PKS v1.6.0 updates include:

- Enables operators to upgrade multiple Kubernetes clusters simultaneously and to designate specific upgrade clusters as canary

clusters. For more information about multiple cluster upgrades, see Upgrade Clusters in *Upgrading Clusters*.

- Adds a new UAA scope, `pks.clusters.admin.read`, for Enterprise PKS users. For information about UAA scopes, see UAA Scopes for Enterprise PKS Users and Managing Enterprise PKS Users with UAA.
- Provides experimental integration with Tanzu Mission Control. For more information, see Tanzu Mission Control Integration.
- Enables operators to limit the total number of clusters a user can provision in Enterprise PKS. For more information about quotas, see Managing Resource Usage with Quotas and Viewing Usage Quotas.
- Enables operators to configure a single Kubernetes cluster with a specific Docker Registry CA certificate. For more information about configuring a cluster with a Docker Registry CA certificate, see Configuring Enterprise PKS Clusters with Private Docker Registry CA Certificates (Beta).
- Updates the `pks delete-cluster` PKS CLI command so that all cluster objects, including NSX-T networking objects, are deleted without the need to use the `bosh delete deployment` command to remove failed cluster deletions.

Kubernetes Control Plane

Enterprise PKS v1.6.0 updates include:

- Increases the **Worker VM Max in Flight** default value from `1` to `4` in the **PKS API** configuration pane, which accelerates cluster creation by allowing up to four new nodes to be provisioned simultaneously. The updated default value is only applied during new Enterprise PKS installation and is not applied during an Enterprise PKS upgrade. If you are upgrading Enterprise PKS from a previous version and want to accelerate multi-cluster provisioning, you can increase the value of **Worker VM Max in Flight** manually.

PKS Monitoring and Logging

Enterprise PKS v1.6.0 updates include:

- Redesigns the **Logging** and **Monitoring** panes of the Enterprise PKS tile and renames them to **Host Monitoring** and **In-Cluster Monitoring**. For information about configuring these panes, see the *Installing Enterprise PKS* topic for your IaaS.
- Adds the **Max Message Size** field in the **Host Monitoring** pane. This allows you to configure the maximum number of characters of a log message that is forwarded to a syslog endpoint. This feature helps ensure that log messages are not truncated at the syslog endpoint. By default, the **Max Message Size** field is 10,000 characters. For more information, see Host Monitoring in the *Installing Enterprise PKS* topic for your IaaS.
- Adds the **Include kubelet metrics** setting. This enables operators to collect workload metrics across all Kubernetes clusters. For more information, see Host Monitoring in the *Installing Enterprise PKS* topic for your IaaS.
- Adds support for Fluent Bit output plugins to log sinks. For information about configuring Fluent Bit output plugins, see Create a ClusterLogSink or LogSink Resource with a Fluent Bit Output Plugin in *Creating and Managing Sink Resources*.
- Adds support for filtering logs and events from a `ClusterLogSink` or `LogSink` resource. For more information, see Filter Sinks in *Creating and Managing Sink Resources*.

Windows on PKS

Enterprise PKS v1.6.0 updates include:

- Adds support for floating Windows stemcells on vSphere. For information about Kubernetes clusters with Windows workers in Enterprise PKS, see Configuring Windows Worker-Based Kubernetes Clusters (Beta).
- Enables operators to configure the location of the Windows pause image. For information about configuring **Kubelet customization - Windows pause image location**, see Plans in *Configuring Windows Worker-Based Kubernetes Clusters (Beta)*.

PKS with NSX-T Networking

Enterprise PKS v1.6.0 updates include:

- NSX Error CRD lets cluster managers and users view NSX errors in Kubernetes resource annotations, and use the command `kubectl get nsxerror` to view the health status of NSX-T cluster networking objects (NCP v2.5.0+). For more information, see Viewing the Health Status of Cluster Networking Objects (NSX-T only).
- DFW log control for dropped traffic lets cluster administrators define network profile to turn on logging and log any dropped or rejected packet by NSX-T distributed firewall rules (NCP v2.5.0+). For more information, see Defining Network Profiles for NCP Logging.
- Load balancer and ingress resource capacity observability using the NSXLoadBalancerMonitor CRD lets cluster managers and users use the command `kubectl get nsxLoadBalancerMonitors` to view a health score that reflects the current performance of the NSX-T load balancer service, including usage, traffic, and current status (NCP v2.5.1+). For more information, see Ingress Scaling (NSX-T only).
- Ingress scale out using the LoadBalancer CRD lets cluster managers scale out the NSX-T load balancer for ingress routing (NCP v2.5.1+). For more information, see Ingress Scaling (NSX-T only).
- Support for Ingress URL Rewrite. For more information, see Using Ingress URL Rewrite.
- Support for Active–Active Tier-0 router configuration when using a Shared-Tier-1 topology.
- Ability to place the load balancer and Tier-1 Active/Standby routers on different failure domains. See Multisite Deployment of NSX-T Data Center for more information.

PKS on AWS Networking

Enterprise PKS v1.6.0 updates include:

- Support for HTTP/HTTPS Proxy on AWS. For more information see, Using Proxies with Enterprise PKS on AWS.

Customer Experience Improvement Program

Enterprise PKS v1.6.0 updates include:

- Administrators can name Enterprise PKS installations so they are more easily recognizable in reports. For more information, see Sample Reports.

Component Updates

Enterprise PKS v1.6.0 updates include:

- Bumps Kubernetes to v1.15.5.
- Bumps UAA to v73.4.8.
- Bumps Jackson dependencies in the PKS API.

Bug Fixes

Enterprise PKS v1.6.0 includes the following bug fixes:

- Fixes an issue where enabling the Availability Sets mode at the BOSH Director > Azure Config resulted in the kubelet failing to start on provisioning of a Kubernetes cluster.

- Fixes an issue where persistent volume attachment failed on vSphere in a scenario where an AZ defined in Ops Manager does not contain a resource pool.
- Increases `network_profile` column size.
- Fixes a Telemetry event generation issue where the `upgrade_cluster_end` event is not sent for completed cluster upgrades.
- Fixes an issue where networking changes did not propagate when upgrading from Enterprise PKS v1.5 or later.
- Fixes an issue where the Ingress IP address was excluded from the Enterprise PKS floating IP pool.
- Fixes an issue where the PKS OSB Proxy start was delayed by scanning all NSX-T firewall rules.
- Fixes an issue with the PKS clusters upgrade errand not pushing the latest NSX-T certificate to Kubernetes Master nodes.
- Fixes an issue with the PKS OSB Proxy taking a long time to start due to scanning all NSX-T firewall rules.
- Fixes an issue with PKS releasing floating IP addresses incompletely while deleting clusters under active/active mode.
- Fixes an issue with the DNS Lookup Feature: INGRESS IP not kept out of PKS Floating IP pool.
- Fixes an issue with the command `pks cluster details` does not display NS Group ID of master VMs.
- Checks the high availability mode of the Tier-0 router before creating PKS a cluster.

Product Snapshot

Element	Details
Version	v1.6.0
Release date	November 14, 2019
Compatible Ops Manager versions	See Pivotal Network
Xenial stemcell version	See Pivotal Network
Windows stemcell version	v2019.7
Kubernetes version	v1.15.5
On-Demand Broker version	v0.29.0
Compatible NSX-T versions	v2.5.0, v2.4.3
NCP version	v2.5.1
Docker version	v18.09.9
Backup and Restore SDK version	v1.17.0
UAA	v73.4.8

vSphere Version Requirements

For Enterprise PKS installations on vSphere or on vSphere with NSX-T Data Center, refer to the VMware Product Interoperability Matrices.

Upgrade Path

The supported upgrade paths to Enterprise PKS v1.6.0 are from Enterprise PKS v1.5.0 and later.

Breaking Changes

Enterprise PKS v1.6.0 has the following breaking changes:

Persistent Volume Data Loss with Worker Reboot

With old versions of Ops Manager, PKS worker nodes with persistent disk volumes may get stuck in a startup state and lose data when they are rebooted manually from the dashboard or automatically by vSphere HA.

This issue is fixed in the following Ops Manager versions:

- v2.8.0+
- v2.7.6+
- v2.6.16+

For all PKS installations that host workers using persistent volumes, Pivotal recommends upgrading to one of the Ops Manager versions above.

Enterprise PKS Removes Sink Commands in the PKS CLI

Enterprise PKS removes the following Enterprise PKS Command Line Interface (PKS CLI) commands:

- `pks create-sink`
- `pks sinks`
- `pks delete-sink`

You can use the following Kubernetes CLI commands instead:

- `kubectl apply -f YOUR-SINK.yml`
- `kubectl get clusterlogsinks`
- `kubectl delete clusterlogsink YOUR-SINK`

For more information about defining and managing sink resources, see Creating and Managing Sink Resources.

Changes to PKS API Endpoints

This release moves the `clusters`, `compute-profiles`, `quotas`, and `usages` PKS API endpoints from `v1beta1` to `v1`. `v1beta1` is no longer supported for these endpoints. You must use `v1`. For example, instead of `https://YOUR-PKS-API-FQDN:9021/v1beta1/quotas`, use `https://YOUR-PKS-API-FQDN:9021/v1/quotas`.

Known Issues

Enterprise PKS v1.6.0 has the following known issues.

Your Kubernetes API Server CA Certificate Expires Unless You Regenerate It

Symptom

Your Kubernetes API server's `tls-kubernetes-2018` certificate is a one-year certificate instead of a four-year certificate.

Explanation

When you upgraded from PKS v1.2.7 to PKS v1.3.1, the upgrade process extended the lifespan of all PKS CA certificates to four years, except for the Kubernetes API server's `tls-kubernetes-2018` certificate. The `tls-kubernetes-2018` certificate remained a one-year certificate.

Unless you regenerate the `tls-kubernetes-2018` certificate it retains its one-year lifespan, even through subsequent Enterprise PKS upgrades.

Workaround

If you have not already done so, you should replace the Kubernetes API server's one-year `tls-kubernetes-2018` certificate before it expires. For information about generating and applying a new four-year `tls-kubernetes-2018` certificate, see How to regenerate tls-kubernetes-2018 certificate when it is not regenerated in the upgrade to PKS v1.3.x in the Pivotal Knowledge Base.

Cluster Upgrade Does Not Upgrade Kubernetes Version on Windows Workers

When PKS clusters are upgraded, Windows worker nodes in the cluster do not upgrade their Kubernetes version. The master and Linux worker nodes in the cluster do upgrade their Kubernetes version as expected.

When the Kubernetes version of a Windows worker does not exactly match the version of the master node, the cluster still functions. `kube-apiserver` has no restriction on lagging patch bumps.

PKS clusters upgrade manually with the `pks upgrade-cluster` command, or automatically with PKS upgrades when the **Upgrade all clusters** errand is set to **Default (On)** in the PKS tile **Errands** pane.

Network Profile for "pks update-cluster" Does Not Use the Defaults from the Original Cluster Manifest

Note: This issue is resolved in Enterprise PKS v1.6.1.

Symptom

The Network profile for `pks update-cluster` uses contents that are being updated and not using the defaults from the original cluster manifest.

Explanation

The `pks update-cluster` operation sets the `subnet_prefix` to 0 in the ncp.ini file when the network-profile has `pod_ip_block_ids` set but it does not have `pod_subnet_prefix`.

Workaround

When creating the network profile to be used for update, include all the below fields. Then update-cluster with the network profile should work.

```
{
  "name": "np",
  "parameters": {
    "t0_router_id": "c501f114-870b-4eda-99ac-966adf464452",
    "fip_pool_ids": ["b7acbda8-46de-4195-add2-5fb11ca46cbf"],
    "pod_ip_block_ids": ["b03bff60-854b-4ccb-9b2b-016867b319c9","234c3652-69e7-4365-9627-8e0d8d4a6b86"],
    "pod_subnet_prefix": 24,
    "single_tier_topology": false
  }
}
```

Azure Default Security Group Is Not Automatically Assigned to Cluster VMs

Symptom

You experience issues when configuring a load balancer for a multi-master Kubernetes cluster or creating a service of type `LoadBalancer`. Additionally, in the Azure portal, the **VM** > **Networking** page does not display any inbound and outbound traffic rules for your cluster VMs.

Explanation

As part of configuring the Enterprise PKS tile for Azure, you enter **Default Security Group** in the **Kubernetes Cloud Provider** pane. When you create a Kubernetes cluster, Enterprise PKS automatically assigns this security group to each VM in the cluster. However, on Azure the automatic assignment may not occur.

As a result, your inbound and outbound traffic rules defined in the security group are not applied to the cluster VMs.

Workaround

If you experience this issue, manually assign the default security group to each VM NIC in your cluster.

Cluster Creation Fails When First AZ Runs Out of Resources

Symptom

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 Enterprise PKS deployment using a round-robin algorithm, creating the first VM in the first AZ that your plan uses. If the AZ runs out of resources, cluster creation fails because BOSH cannot create the cluster VM.

For example, if you have three AZs and you create two clusters with four worker VMs each, BOSH deploys VMs in the following AZs:

	AZ1	AZ2	AZ3
Cluster 1	Worker VM 1	Worker VM 2	Worker VM 3
	Worker VM 4		
Cluster 2	Worker VM 1	Worker VM 2	Worker VM 3
	Worker VM 4		

In this scenario, AZ1 has twice as many VMs as AZ2 or AZ3.

Cluster Creation Fails with Long Network Profile

Note: This issue is resolved in Enterprise PKS v1.6.1.

Creating a cluster with a long network profile, such as with multiple `pod_ip_block_ids` values, causes an error `Data too long for column 'nsxt_network_profile'`.

Azure Worker Node Communication Fails after Upgrade

Symptom

Outbound communication from a worker node VM fails after upgrading Enterprise PKS.

Explanation

Enterprise PKS uses Azure Availability Sets to improve the uptime of workloads and worker nodes in the event of Azure platform failures. Worker node VMs are distributed evenly across Availability Sets.

Azure Standard SKU Load Balancers are recommended for the Kubernetes control plane and Kubernetes ingress and egress. This load balancer type provides an IP address for outbound communication using SNAT.

During an upgrade, when BOSH rebuilds a given worker instance in an Availability Set, Azure can time out while re-attaching the worker node network interface to the back-end pool of the Standard SKU Load Balancer.

For more information, see Outbound connections in Azure in the Azure documentation.

Workaround

You can manually re-attach the worker instance to the back-end pool of the Azure Standard SKU Load Balancer in your Azure console.

Error During Individual Cluster Upgrades

Symptom

While submitting a large number of cluster upgrade requests using the `pks upgrade-cluster` command, some of your Kubernetes clusters are marked as failed.

Explanation

BOSH upgrades Kubernetes clusters in parallel with a limit of up to four concurrent cluster upgrades by default. If you schedule more than four cluster upgrades, Enterprise PKS queues the upgrades and waits for BOSH to finish the last upgrade. When BOSH finishes the last upgrade, it starts working on the next upgrade request.

If you submit too many cluster upgrades to BOSH, an error may occur, where some of your clusters are marked as `FAILED` because BOSH can start the upgrade only within the specified timeout. The timeout is set to 168 hours by default. However, BOSH does not remove the task from the queue or stop working on the upgrade if it has been picked up.

Solution

If you expect that upgrading all of your Kubernetes clusters takes more than 168 hours, do not use a script that submits upgrade requests for all of your clusters at once. For information about upgrading Kubernetes clusters provisioned by Enterprise PKS, see Upgrading Clusters.

Kubectl CLI Commands Do Not Work after Changing an Existing Plan to a Different AZ

Symptom

After you update the AZ of an existing plan, kubectl CLI commands do not work for your clusters associated with the plan.

Explanation

This issue occurs in IaaS environments that do not support attaching a disk across multiple AZs.

When the plan of an existing cluster changes to a different AZ, BOSH migrates the cluster by creating VMs for the cluster in the new AZ and removing your cluster VMs from the original AZ.

On an IaaS that does not support attaching VM disks across AZs, the disks BOSH attaches to the new VMs do not have the original content.

Workaround

If you cannot run kubectl CLI commands after reconfiguring the AZ of an existing cluster, contact Support for assistance.

Applying Changes Fails If Plan 8 Is Enabled

Note: This issue is resolved in Enterprise PKS v1.6.1.

Symptom

After you click **Apply Changes** on the Ops Manager Installation Dashboard, the following error occurs:

```
Cannot generate manifest for product Enterprise PKS
```

.

Explanation

This error occurs if Plan 8 is enabled in your Enterprise PKS v1.6.0 tile.

Workaround

Disable Plan 8 in the Enterprise PKS tile and move your plan settings to a plan that is available for configuration, for example, Plan 9 or 10.

To disable Plan 8:

1. In **Plan 8**, select **Plan > Inactive**.
2. Click **Save**.

One Plan ID Longer than Other Plan IDs

Symptom

One of your plan IDs is one character longer than your other plan IDs.

Explanation

In Enterprise PKS, each plan has a unique plan ID. A plan ID is normally a UUID consisting of 32 alphanumeric characters and 4 hyphens. However, the **Plan 4** ID consists of 33 alphanumeric characters and 4 hyphens.

Solution

You can safely configure and use **Plan 4**. The length of the **Plan 4** ID does not affect the functionality of **Plan 4** clusters.

If you require all plan IDs to have identical length, do not activate or use **Plan 4**.

Kubernetes Cluster Name Limitation for Tanzu Mission Control Integration

Tanzu Mission Control integration cannot attach Tanzu Mission Control to Kubernetes clusters that have uppercase letters in their names.

Symptom

Clusters that you create with `pks create-cluster` do not appear in the Tanzu Mission Control, even though you configured Tanzu Mission Control integration as described in Integrate Tanzu Mission Control.

Explanation

The regex pattern that parses cluster names in Tanzu Mission Control integration fails with names that contain uppercase letters.

Solution

When running `pks create-cluster` to create clusters that you want to track in Tanzu Mission Control, pass in names that contain only lowercase letters and numbers.

Enterprise PKS Creates a Linux Cluster When You Expect a Windows Cluster

Note: This issue is resolved in Enterprise PKS v1.6.1.

Symptom

When you create an Enterprise PKS cluster using either Plan 11, 12 or 13 the cluster is created as a Linux cluster instead of a Windows cluster.

Explanation

When you create an Enterprise PKS cluster using either Plan 11, 12 or 13 a Windows cluster should be created. If you are using Enterprise PKS v1.6 with Operations Manager v2.7 a Linux cluster is created instead.

Saving UAA Tab Settings Fails With Error: 'InvalidURIError bad URI'

Symptom

When you save your UAA tab with **LDAP Server** selected and multiple LDAP servers specified, you receive the error: `URI::InvalidURIError bad URI(is not URI?):LDAP URLs`.

Explanation

When you configure the UAA tab with multiple LDAP servers your settings will fail to validate when using the following Ops Manager releases:

Ops Manager Version	Affected Releases
Ops Manager v2.6	Ops Manager v2.6.18 and earlier patch releases.
Ops Manager v2.7	All patch releases.
Ops Manager v2.8	All patch releases.

Workaround

To resolve this issue see the following:

Ops Manager Version	Workaround
Ops Manager v2.6	Perform one of the following: • Upgrade to Ops Manager v2.6.19 or later v2.6 patch release. • Complete the procedures in UAA authentication tab in PKS 1.6 fails to save with error "URI::InvalidURIError bad URI(is not URI?):LDAP URLs" (76495) in the Pivotal Support Knowledge Base.
Ops Manager v2.7	Complete the procedures in UAA authentication tab in PKS 1.6 fails to save with error "URI::InvalidURIError bad URI(is not URI?):LDAP URLs" (76495) in the Pivotal Support Knowledge Base.
Ops Manager v2.8	Complete the procedures in UAA authentication tab in PKS 1.6 fails to save with error "URI::InvalidURIError bad URI(is not URI?):LDAP URLs" (76495) in the Pivotal Support Knowledge Base.

Windows Worker Clusters Fail to Upgrade to v1.6

Symptoms

During your upgrade from Enterprise PKS v1.5 to Enterprise PKS v1.6 a Windows worker VM fails to upgrade, as evidenced by:

- The command line outputs an error `Failed jobs: docker-windows`.
- The Windows worker VM disappears from the output of `kubectl get nodes`.
- The command line shows the status `failed` and the action `UPGRADE` for the cluster that contains the worker.
- The log shows an entry `\docker\dockerd.exe: Access is denied`.

Explanation

Between PKS v1.5 and v1.6, the name of the Docker service changed from `docker` to `docker-windows`, but your environment continues to use the old Docker service name and paths. The incompatible service name and pathing causes a Windows worker upgrade to fail.

If your cluster has multiple Windows workers, this issue does not incur downtime . Before BOSH attempts to upgrade a Windows worker, it moves the worker's apps to other Windows workers in the cluster. When the upgrade fails, BOSH stops the cluster upgrade process and the other Windows workers continue running at the earlier version.

Workaround

After upgrading to Enterprise PKS v1.6 and your Windows worker clusters have failed to upgrade, complete the following steps:

1. Upload a vSphere stemcell v2019.8 or later for Windows Server version 2019 to your Enterprise PKS tile.
2. To upgrade your Windows worker clusters, perform one of the following:
   - Enable the **Upgrade all clusters errand** setting and deploy the PKS tile. For more information about configuring the **Upgrade all clusters errand** and deploying the Enterprise PKS tile, see Modify Errand Configuration in the Enterprise PKS Tile in *Upgrading Clusters*.

- Run `pks upgrade-cluster` or `pks upgrade-clusters` on your failed Windows worker cluster(s). For more information about upgrading specific Enterprise PKS clusters, see Upgrade Clusters in *Upgrading Clusters*.

502 Bad Gateway After OIDC Login

Symptom

You experience a "502 Bad Gateway" error from the NSX load balancer after you log in to OIDC.

Explanation

A large response header has exceeded your NSX-T load balancer maximum response header size. The default maximum response header size is 10,240 characters and should be resized to 50,000.

Workaround

If you experience this issue, manually reconfigure your NSX-T `request_header_size` and `response_header_size` to 50,000 characters. For information about configuring NSX-T default header sizes, see OIDC Response Header Overflow in the Pivotal Knowledge Base.

NSX-T Pre-Check Errand Fails Due to Edge Node Configuration

Symptom

You have configured your NSX-T Edge Node VM as `medium` size, and the NSX-T Pre-Check Errand fails with the following error: "*ERROR: NSX-T Precheck failed due to Edge Node … no of cpu cores is less than 8*".

Explanation

The NSX-T Pre-Check Errand is erroneously returning the "*cpu cores is less than 8*" error.

Solution

You can safely configure your NSX-T Edge Node VMs as `medium` size and ignore the error.

Character Limitations in HTTP Proxy Password

For vSphere with NSX-T, the HTTP Proxy password field does not support the following special characters: `&` or `;`.

Enterprise PKS Management Console 1.6.2

Release Date: April 29, 2020

Features

Other than support for Enterprise PKS v1.6.2, Enterprise PKS Management Console 1.6.2 has no new features.

Bug Fixes

Enterprise PKS Management Console 1.6.2 includes no bug fixes.

Product Snapshot

Note: Enterprise PKS Management Console provides an opinionated installation of Enterprise PKS. The supported versions may differ from or be more limited than what is generally supported by Enterprise PKS.

Element	Details
Version	v1.6.2
Release date	April 29, 2020
Installed Enterprise PKS version	v1.6.2
Installed Ops Manager version	v2.8.5
Installed Kubernetes version	v1.15.10
Compatible NSX-T versions	v2.5.0, v2.4.3
Installed Harbor Registry version	v1.9.4

Known Issues

The Enterprise PKS Management Console v1.6.2 appliance and user interface have the same known issues as v1.6.1.

Enterprise PKS Management Console 1.6.1

Release Date: January 23, 2020

Features

Other than support for Enterprise PKS v1.6.1, Enterprise PKS Management Console 1.6.1 has no new features.

Bug Fixes

Enterprise PKS Management Console 1.6.1 includes no bug fixes.

Product Snapshot

Note: Enterprise PKS Management Console provides an opinionated installation of Enterprise PKS. The supported versions may differ from or be more limited than what is generally supported by Enterprise PKS.

Element	Details
Version	v1.6.1
Release date	January 23, 2020
Installed Enterprise PKS version	v1.6.1
Installed Ops Manager version	v2.8.0
Installed Kubernetes version	v1.15.5
Compatible NSX-T versions	v2.5.0, v2.4.3

Installed Harbor Registry version	v1.9.3

Known Issues

The Enterprise PKS Management Console v1.6.1 appliance and user interface have the same known issues as v1.6.0-rev.3 and v1.6.0-rev.2.

Enterprise PKS Management Console 1.6.0-rev.3

Release Date: December 19, 2019

> **important:** The Enterprise PKS Management Console 1.6.0-rev.3 offline patch can only be applied in an air-gapped environment. It can only be applied to 1.6.0-rev.2 and not to any other version. For information about how to apply the patch, see Patch Enterprise PKS Management Console Components.

Features

Enterprise PKS Management Console 1.6.0-rev.3 has no new features.

Bug Fixes

Enterprise PKS Management Console 1.6.0-rev.3 includes the following bug fixes:

- Fixes UI failure caused by multiple datacenters being present in vCenter Server.
- Adds support for both FQDN and IP addresses in LDAP/LDAPS configuration for identity management.
- Fixes UI freezing after entering unconventionally formatted URLs for SAML provider metadata.
- Adds support for UAA role `pks.clusters.admin.read` in identity Management configuration.
- Adds validation for Harbor FQDN in lower case.
- Fixes misconfigured Wavefront HTTP Proxy when field is left empty.

Product Snapshot

> **Note**: Enterprise PKS Management Console provides an opinionated installation of Enterprise PKS. The supported versions may differ from or be more limited than what is generally supported by Enterprise PKS.

Element	Details
Version	v1.6.0-rev.3
Release date	December 19, 2019
Installed Enterprise PKS version	v1.6.0
Installed Ops Manager version	v2.7.3
Installed Kubernetes version	v1.15.5
Compatible NSX-T versions	v2.5.0, v2.4.3
Installed Harbor Registry version	v1.9.3

Known Issues

With the exception of the Bug Fixes listed above, the Enterprise PKS Management Console v1.6.0-rev.3 appliance and user interface have the same known issues as v1.6.0-rev.2.

Enterprise PKS Management Console v1.6.0-rev.2

Release Date: November 26, 2019

Features

Enterprise PKS Management Console v1.6.0-rev.2 updates include:

- Provides experimental integration with VMware Tanzu Mission Control. For more information, see Tanzu Mission Control Integration.
- Provides experimental support for plans that use Windows worker nodes. For information, see Configure Plans.
- Deploys Harbor registry v1.9. For information, see Configure Harbor.
- Adds support for active-active mode on the tier 0 router in automated-NAT deployments and No-NAT configurations in Bring Your Own Topology deployments. For information, see Configure Networking.
- Adds the ability to configure proxies for the integration with Wavefront. For information, see Configure a Connection to Wavefront.
- Adds the ability to configure the size of the PKS API VM. For information, see Configure Resources and Storage.
- Allows you to use the management console to upgrade to v1.6.0-rev.2. For information, see Upgrade Enterprise PKS Management Console.

Product Snapshot

Note: Enterprise PKS Management Console provides an opinionated installation of Enterprise PKS. The supported versions may differ from or be more limited than what is generally supported by Enterprise PKS.

Element	Details
Version	v1.6.0-rev.2
Release date	November 26, 2019
Installed Enterprise PKS version	v1.6.0
Installed Ops Manager version	v2.7.3
Installed Kubernetes version	v1.15.5
Compatible NSX-T versions	v2.5.0, v2.4.3
Installed Harbor Registry version	v1.9.3

Known Issues

The following known issues are specific to the Enterprise PKS Management Console v1.6.0-rev.2 appliance and user interface.

YAML Validation Errors Not Cleared

Symptom

If you attempt to upload a YAML configuration file and the deployment fails because of an invalid manifest, Enterprise PKS Management Console displays an error notification with the validation error. If subsequent attempts also fail because of validation issues, the validation errors are appended to each other.

Explanation

The validation errors are not cleared when you resubmit the YAML configuration file.

Workaround

None

Enterprise PKS Management Console Notifications Persist

Symptom

In the **Enterprise PKS** view of Enterprise PKS Management Console, error notifications sometimes persist in memory on the **Clusters** and **Nodes** pages after you clear those notifications.

Explanation

After clicking the **X** button to clear a notification it is removed, but when you navigate back to those pages the notification might show again.

Workaround

Use shift+refresh to reload the page.

Cannot Delete Enterprise PKS Deployment from Management Console

Symptom

In the **Enterprise PKS** view of Enterprise PKS Management Console, you cannot use the **Delete Enterprise PKS Deployment** option even after you have removed all clusters.

Explanation

The option to delete the deployment is only activated in the management console a short period after the clusters are deleted.

Workaround

After removing clusters, wait for a few minutes before attempting to use the **Delete Enterprise PKS Deployment** option again.

Configuring Enterprise PKS Management Console Integration with VMware vRealize Log Insight

Symptom

Enterprise PKS Management Console appliance sends logs to VMware vRealize Log Insight over HTTP, not HTTPS.

Explanation

When you deploy the Enterprise PKS Management Console appliance from the OVA, if you require log forwarding to vRealize Log Insight, you must provide the port on the vRealize Log Insight server on which it listens for HTTP traffic. Do not provide the HTTPS port.

Workaround

Set the vRealize Log Insight port to the HTTP port. This is typically port `9000`.

Deploying Enterprise PKS to an Unprepared NSX-T Data Center Environment Results in Flannel Error

Symptom

When using the management console to deploy Enterprise PKS in **NSX-T Data Center (Not prepared for PKS)**mode, if an error occurs during the network configuration, the message `Unable to set flannel environment` is displayed in the deployment progress page.

Explanation

The network configuration has failed, but the error message is incorrect.

Workaround

To see the correct reason for the failure, see the server logs. For instructions about how to obtain the server logs, see Troubleshooting Enterprise PKS Management Console.

Using BOSH CLI from Operations Manager VM

Symptom

The BOSH CLI client bash command that you obtain from the **Deployment Metadata** view does not work when logged in to the Operations Manager VM.

Explanation

The BOSH CLI client bash command from the **Deployment Metadata** view is intended to be used from within the Enterprise PKS Management Console appliance.

Workaround

To use the BOSH CLI from within the Operations Manager VM, see Connect to Operations Manager.

From the Ops Manager VM, use the BOSH CLI client bash command from the **Deployment Metadata** page, with the following modifications:

- Remove the clause `BOSH_ALL_PROXY=xxx`
- Replace the `BOSH_CA_CERT` section with `BOSH_CA_CERT=/var/tempest/workspaces/default/root_ca_certificate`

Run `pks` Commands against the PKS API Server

Explanation

The PKS CLI is available in the Enterprise PKS Management Console appliance.

Workaround

To be able to run `pks` commands against the PKS API Server, you must first log to PKS using the following command syntax `pks login -a fqdn_of_pks ...`.

To do this, you must ensure either of the following:

- The FQDN configured for the PKS Server is resolvable by the DNS server configured for the Enterprise PKS Management Console appliance, or
- An entry that maps the Floating IP assigned to the PKS Server to the FQDN exists on /etc/hosts in the appliance. For example: `192.168.160.102 api.pks.local`.

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

Enterprise PKS Concepts

Page last updated:

This topic describes VMware Enterprise PKS concepts. See the following sections:

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

Enterprise PKS Cluster Management

In this topic

Page last updated:

This topic describes how VMware Enterprise PKS manages the deployment of Kubernetes clusters.

Overview

Users interact with Enterprise PKS and Enterprise 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, Enterprise 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 Enterprise 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

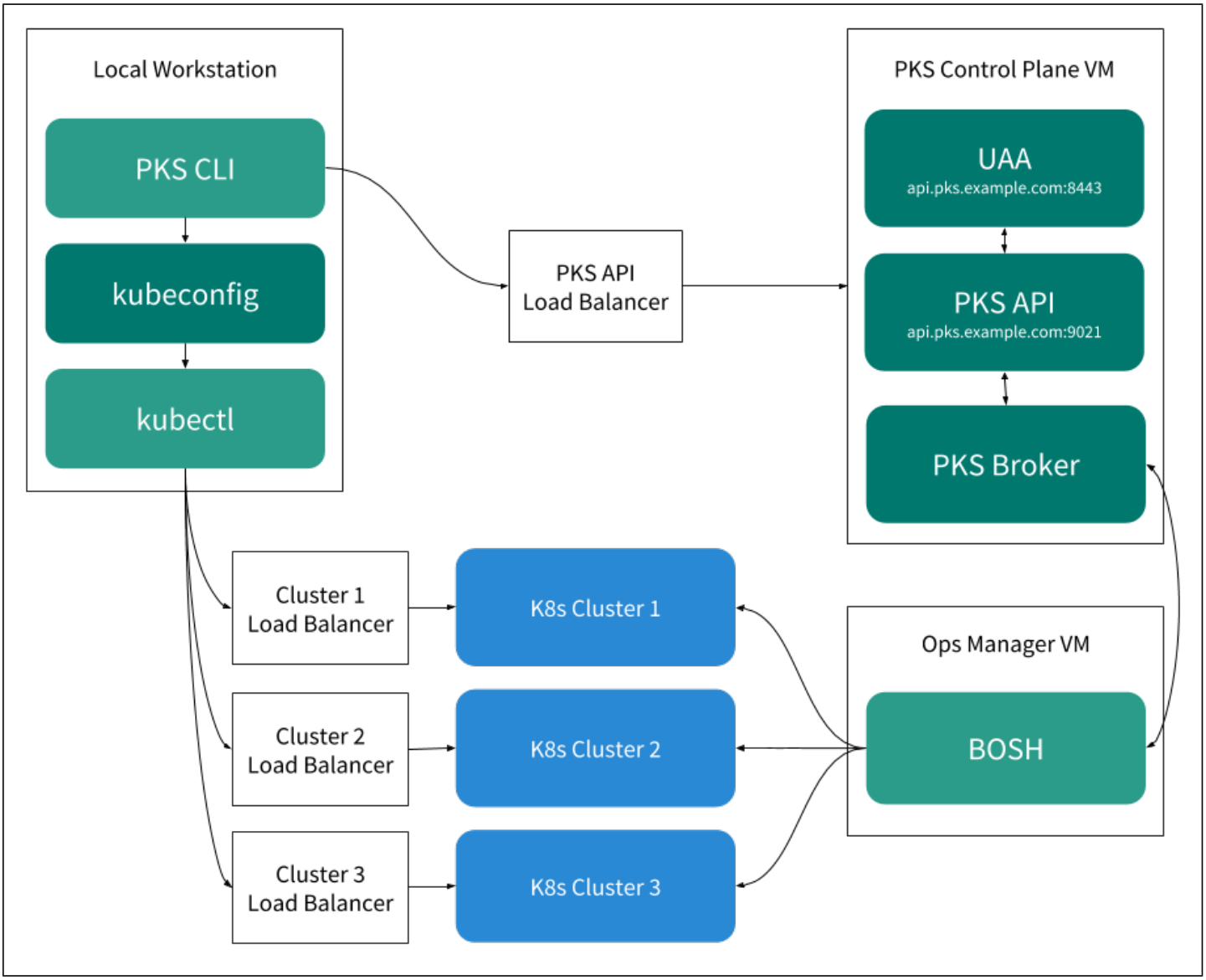
In addition, the PKS control plane can upgrade all existing clusters using the **Upgrade all clusters** BOSH errand. For more information, see Upgrade Kubernetes Clusters in *Upgrading Enterprise PKS*.

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 Enterprise 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 Enterprise 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 Enterprise PKS Users 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 Enterprise PKS deployments on vSphere with NSX-T, there is an additional component, the Enterprise 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

Enterprise 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.

PKS API Authentication

In this topic

Page last updated:

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

Authentication of 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 Managing Enterprise PKS Users with UAA and Logging in to Enterprise PKS.

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.

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** > **Enterprise PKS tile** > **PKS API** > **API Hostname (FQDN)** for your PKS API domain.

Load balancer implementations differ by deployment environment. For Enterprise PKS deployments on GCP, AWS, or vSphere without NSX-T, you configure a load balancer to access the PKS API when you install the Enterprise PKS tile. For example, see Configuring PKS API Load Balancer.

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

UAA Scopes for Enterprise PKS Users

In this topic

Page last updated:

This topic describes User Account and Authentication (UAA) scopes that a UAA admin can assign to VMware Enterprise PKS users.

Overview

UAA is the identity management service for Enterprise PKS.

By assigning UAA scopes, you grant users the ability to create, manage, and audit Kubernetes clusters in Enterprise PKS.

A UAA admin user can assign the following UAA scopes to Enterprise PKS users:

- `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.
- `pks.clusters.admin.read` : Accounts with this scope can access any information about all clusters except for cluster credentials.

You can assign these scopes to individual users, external identity provider groups, or clients for automation purposes.

UAA Scopes

Each UAA scope grants Enterprise PKS users a set of permissions for creating, managing, and auditing Enterprise PKS-provisioned Kubernetes clusters. For information about the permissions, see the table below.

Operation	`pks.clusters.manage`	`pks.clusters.admin`	`pks.clusters.admin.read`
Create, update, resize, and delete a cluster	**Yes.** Users with this scope can create, modify, and delete only their own clusters.	**Yes.** Users with this scope can create, modify, and delete all clusters.	**No.** Users with this scope cannot create, modify, and delete clusters.
Get cluster credentials	**Yes.** Users with this scope can retrieve cluster credentials only for their own clusters.	**Yes.** Users with this scope can retrieve cluster credentials for all clusters.	**No.** Users with this scope cannot retrieve cluster credentials.
Upgrade clusters	**Yes.** Users with this scope can upgrade only their own clusters.	**Yes.** Users with this scope can upgrade all clusters.	**No.** Users with this scope cannot upgrade clusters.
List clusters	**Yes.** Users with this scope can list only their own clusters.	**Yes.** Users with this scope can list all clusters.	**Yes.** Users with this scope can list all clusters.
View cluster details	**Yes.** Users with this scope can view cluster details only for their own clusters.	**Yes.** Users with this scope can view cluster details for all clusters.	**Yes.** Users with this scope can view cluster details for all clusters.

Create and delete a compute profile	**No.** Users with this scope cannot create and delete compute profiles.	**Yes.** Users with this scope can create and delete compute profiles.	**No.** Users with this scope cannot create and delete compute profiles.
Create and delete a network profile	**No.** Users with this scope cannot create and delete network profiles.	**Yes.** Users with this scope can create and delete network profiles.	**No.** Users with this scope cannot create and delete network profiles.
Create, update, and delete a quota	**No.** Users with this scope cannot create, update, and delete quotas.	**Yes.** Users with this scope can create, update, and delete quotas.	**No.** Users with this scope cannot create, update, and delete quotas.
List Enterprise PKS plans	**Yes.** Users with this scope can list all available plans.	**Yes.** Users with this scope can list all available plans.	**Yes.** Users with this scope can list all available plans.

To assign UAA scopes in Enterprise PKS, follow the instructions in Managing Enterprise PKS Users with UAA.

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

Load Balancers in Enterprise PKS

In this topic

Page last updated:

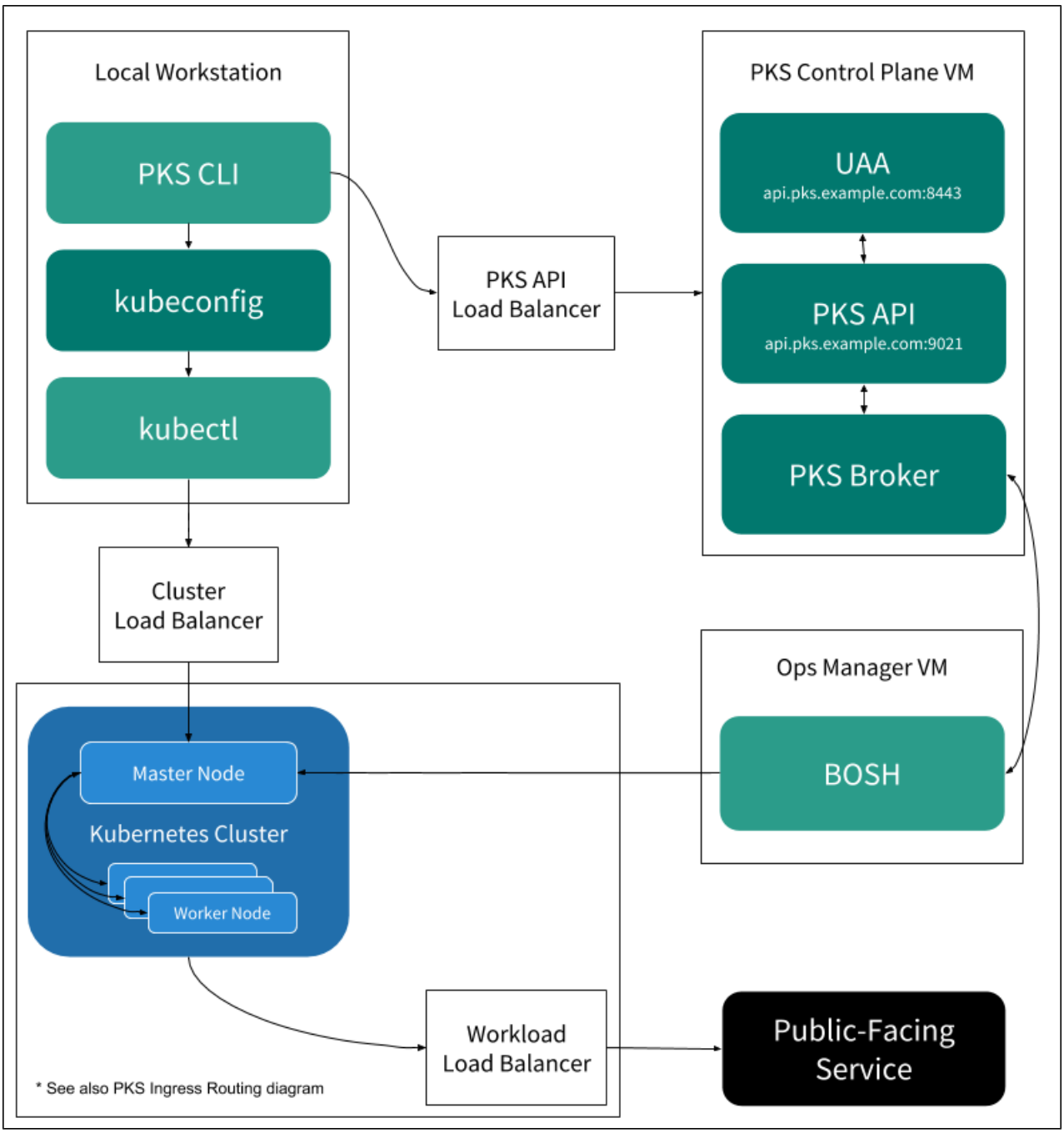
This topic describes the types of load balancers that are used in VMware Enterprise PKS deployments. Load balancers differ by the type of deployment.

Load Balancers in Enterprise PKS Deployments without NSX−T

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

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

The following diagram, applicable to GCP, AWS, and vSphere without NSX-T, shows where each of the above load balancers can be used within your Enterprise PKS deployment.



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

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

For information about configuring the PKS API load balancer on vSphere without NSX-T, see Configuring PKS API Load Balancer.

About Kubernetes Cluster Load Balancers

When you create an Enterprise PKS cluster on GCP, AWS, and on vSphere without NSX-T, you must configure external access to the

cluster by creating an external TCP or HTTPS load balancer. The load balancer enables 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 enable 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 Enterprise PKS Clusters
- Creating and Configuring an AWS Load Balancer for Enterprise PKS Clusters
- Creating and Configuring an Azure Load Balancer for Enterprise PKS Clusters

About Workload Load Balancers

To enable external access to your Enterprise PKS app on GCP, AWS, and on vSphere without NSX-T, 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 Linux 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.

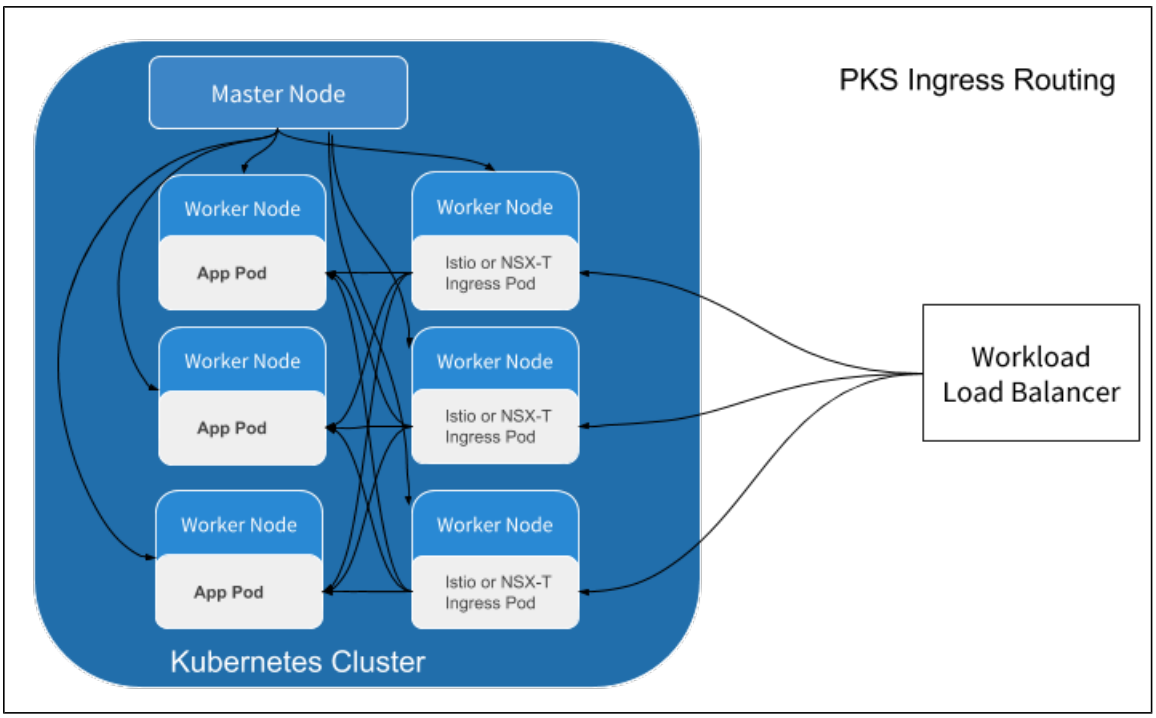
See the AWS Prerequisites section of *Deploying and Exposing Basic Linux Workloads* before you create a workload load balancer.

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 Enterprise PKS deployment, traffic routing is controlled by the ingress resource rules you define. Pivotal recommends configuring Enterprise 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 Enterprise PKS deployment.



The load balancer on Enterprise 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 Enterprise PKS Deployments on vSphere with NSX–T

Enterprise 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 Enterprise PKS Endpoint in *Installing Enterprise 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 Logical Load Balancer 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 enables requests to be load balanced across multiple master nodes.
- **Ingress controller**. This enables the virtual server instance to dispatch HTTP and HTTPS requests to services associated with Ingress rules.
- `type:loadbalancer` **services**. This enables 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 Enterprise 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 Enterprise PKS Clusters

In this topic

Page last updated:

This topic describes how VMware Enterprise PKS recommends you approach the sizing of VMs for cluster components.

Overview

When you configure plans in the Enterprise 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 Enterprise 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 Enterprise 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

Number of Workers	CPU	RAM (GB)
251-500	16	60
500+	32	120

Do not overload your master node VMs by exceeding the recommended maximum number of worker node VMs or by downsizing from the recommended VM sizings listed above. These recommendations support both a typical workload managed by a VM and the higher than usual workload managed by the VM while other VM's in the cluster are upgrading.

warning: Upgrading an overloaded Kubernetes cluster master node VM can result in downtime.

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 an Enterprise PKS-provisioned Kubernetes cluster is set by the **Maximum number of workers on a cluster** field in the **Plans** pane of the Enterprise 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 Enterprise 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 Enterprise PKS installation topic for your IaaS. For example, Installing Enterprise 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 Knowledge Base.

warning: Do not reduce the size of your Kubernetes master node VMs below the recommended sizes listed in Master Node VM Size, above. Upgrading an overloaded Kubernetes cluster master node VM can result in downtime.

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

Telemetry

In this topic

Page last updated:

This topic describes the VMware Customer Experience Improvement Program (CEIP) and the Pivotal Telemetry Program (Telemetry) used in the Enterprise PKS tile.

Overview

The CEIP and Telemetry program allow VMware and Pivotal to collect data from customer installations to improve your Enterprise PKS experience. Collecting data at scale enables us to identify patterns and alert you to warning signals in your Enterprise PKS installation.

Participation Levels

You can configure Enterprise PKS to use one of the following CEIP and Telemetry participation levels:

- **None:** This level disables data collection.
- **Standard:** (Default) This level collects data anonymously. Your data is used to inform the ongoing development of Enterprise PKS.
- **Enhanced:** This level enables VMware and Pivotal to warn you about security vulnerabilities and potential issues with your software configurations. For more information, see Benefits of the Enhanced Participation Levelbelow.

> **Note:** Enterprise PKS does not collect any personally identifiable information (PII) at either participation level. For a list of the data Enterprise PKS collects, see Data Dictionary.

Configure CEIP and Telemetry

Video: For information about configuring CEIP and Telemetry participation, see the CEIP Opt-In Walkthrough video on YouTube.

For information about configuring CEIP and Telemetry participation, see the *CEIP and Telemetry* section of the installation topic for your IaaS:

- Installing Enterprise PKS on vSphere
- Installing Enterprise PKS on vSphere with NSX-T
- Installing Enterprise PKS on AWS
- Installing Enterprise PKS on Azure

- Installing Enterprise PKS on GCP

Benefits of the Enhanced Participation Level

Benefits you receive with the Enhanced participation level include but are not limited to the following:

- **Usage data:** This gives you access to data about Kubernetes pod and cluster usage in your Enterprise PKS installation. See sample reports below for more details.
- **Access to your telemetry data:** This gives you access to configuration and usage data about your Enterprise PKS installation. See sample reports below for more details.
- **Proactive support:** This enables VMware and Pivotal to proactively warn you about unhealthy patterns.
- **Benchmarks:** This is your usage relative to the rest of the Enterprise PKS user base.

The table below compares the Standard and Enhanced participation levels.

Benefit	Standard Level	Enhanced Level
Usage data	Raw data	Reports and trend analysis
Access to your telemetry data	No	Yes
Proactive support	No	Yes
Benchmarks	No	Yes

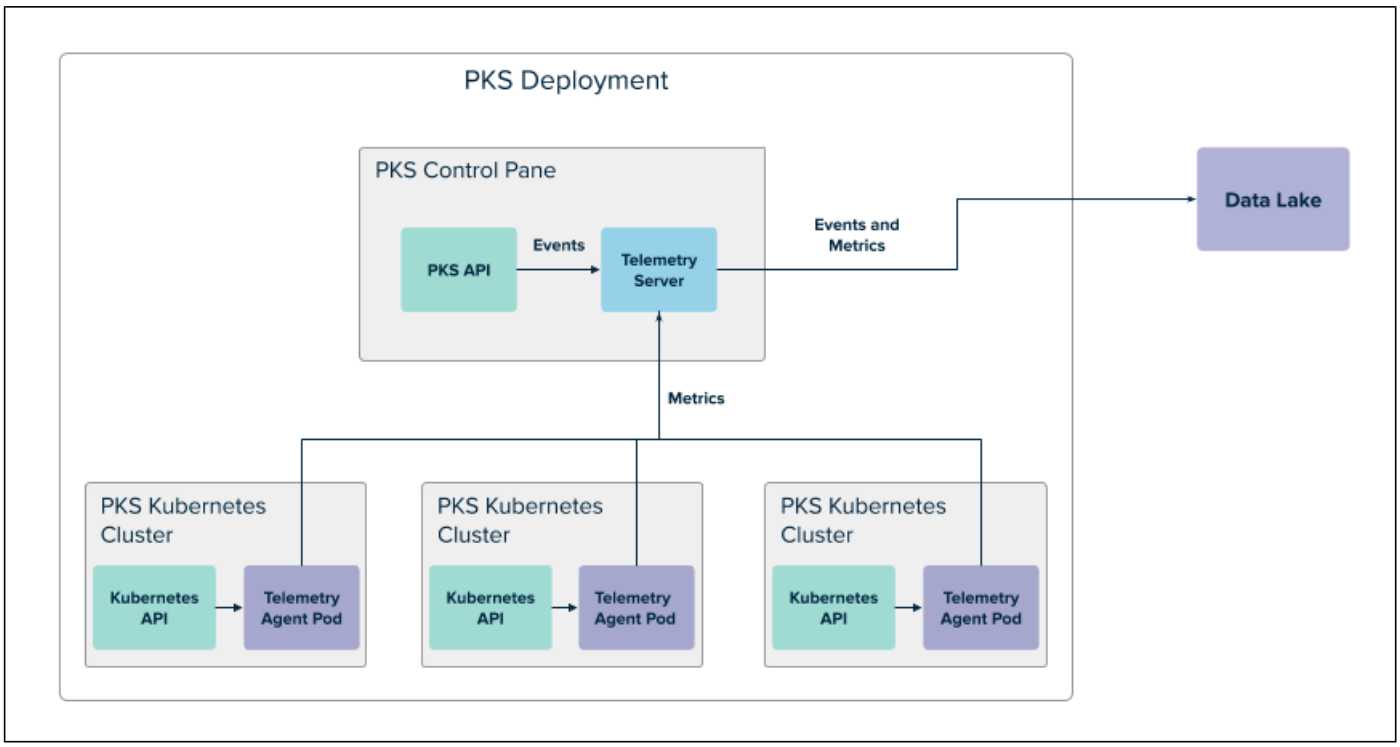
Note: VMware reserves the right to change the benefits associated with the Enhanced participation level at any time.

System Components

The CEIP and Telemetry programs use the following components to collect data:

- **Telemetry Server:** This component runs on the PKS control plane. The server receives telemetry events from the PKS API and metrics from Telemetry agent pods. The server sends events and metrics to a data lake for archiving and analysis.
- **Telemetry Agent Pod:** This component runs in each Kubernetes cluster as a deployment with one replica. Agent pods periodically poll the Kubernetes API for cluster metrics and send the metrics to the Telemetry server.

The following diagram shows how telemetry data flows through the system components:



Data Dictionary

For information about PKS Telemetry collection and reporting, see the PKS Telemetry Data spreadsheet, hosted on Google Drive.

Sample Reports

Video: See the Sample Report: Create Cluster Duration video on YouTube.

You can view the interactive version of the Sample Workbook with Tableau Reader (free to use). Click on the links below to see static screenshots of the reports.

1. Consumption: As an Operator of PKS, I need a way to monitor pod consumption across my PKS environments over time, so I can:
    - See which environments and clusters get the heaviest use
    - See temporal patterns in pod consumption
    - Scale capacity accordingly
    - Show and charge back users of PKS within my organization
2. API heartbeats + Cluster heartbeats: As an Operator of PKS I need a way to see the version of PKS each of my environments was running over time, so I can:
    - Keep track of all my PKS environments and clusters
    - Identify environments and clusters in need of upgrading
3. Cluster creation events: As an Operator of PKS I want to see how often cluster creation succeeds across my PKS environments, so I can:
    - Identify environments that encounter repeated failures and debug or intervene as appropriate to avoid frustration for cluster admins and users

4. Cluster creation duration: As an Operator of PKS I want to see how long it takes to create clusters, so I can:
    - Intervene when cluster creation significantly more time than expected, and adjust my plan and network configuration as appropriate
5. Cluster creation errors: As an Operator of PKS, I want to see what errors are being encountered most frequently during cluster creation so I can:
    - Quickly identify widespread problems and remediate (e.g. NSX errors)
6. Container images: As an Operator of PKS, I want to see which container images are in use across my PKS installations so I can:
    - Conduct an audit of container images and identify prohibited or problematic images
    - Infer which workloads are running on PKS, to inform my planning, resourcing, and outreach

---

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

Installing Enterprise PKS

Page last updated:

Enterprise PKS Management Console (vSphere Only)

See the following documentation for the Enterprise PKS Management Console, which deploys Enterprise PKS as a virtual appliance on vSphere without Pivotal Platform:

- Enterprise PKS Management Console (vSphere Only)

VMware Enterprise PKS on Pivotal Platform

See the following documentation for how to install VMware Enterprise PKS on Pivotal Platform:

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

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

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

vSphere

Page last updated:

This topic lists the procedures to follow to install VMware Enterprise PKS on vSphere.

Install Enterprise PKS on vSphere

To install Enterprise PKS on vSphere without NSX-T, follow the instructions below:

- Prerequisites and Resource Requirements
- Firewall Ports and Protocols Requirements for vSphere without NSX-T
- Creating Dedicated Users and Roles for vSphere (Optional)
- Installing and Configuring Ops Manager on vSphere
- Installing Enterprise PKS on vSphere
- Configuring PKS API Load Balancer
- Setting Up Enterprise PKS Admin Users on vSphere
- (Optional) Integrating VMware Harbor with Enterprise PKS

Note: VMware Harbor is an enterprise-class registry server for container images. For more information, see VMware Harbor Registry in the *Pivotal Partner documentation*.

Install the PKS and Kubernetes CLIs

The PKS CLI and Kubernetes CLI help you interact with your Enterprise 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

In this topic

Page last updated:

This topic describes the prerequisites and resource requirements for installing VMware Enterprise PKS on vSphere.

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

Prerequisites

Before installing Enterprise PKS:

1. Review the sections below and the instructions in Creating Dedicated Users and Roles for vSphere (Optional).
2. Install and configure Ops Manager. To install Ops Manager, follow the instructions in Installing and Configuring Ops Manager on vSphere.

vSphere Version Requirements

For Enterprise PKS on vSphere version requirements, refer to the VMware Product Interoperability Matrices.

Resource Requirements

Installing Ops Manager and Enterprise 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 Pivotal Container Service VM as follows:

Number of Pods	Storage (Persistent Disk) Requirements
1,000 pods	20 GB

Number of Pods	Storage (Persistent Disk) Requirements
5,000 pods	100 GB
10,000 pods	200 GB
50,000 pods	1,000 GB

Ephemeral VM Resources

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

Network Communication Requirements

For a complete list of network communication requirements for vSphere without NSX-T, see Firewall Ports and Protocols Requirements for vSphere without NSX-T.

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

Firewall Ports and Protocols Requirements for vSphere without NSX-T

Page last updated:

In this topic

This topic describes the firewall ports and protocols requirements for using VMware Enterprise 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 Enterprise PKS, Pivotal recommends that you disable security policies that filter traffic between the networks supporting the system. With Enterprise 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 Enterprise 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.

Enterprise PKS Ports and Protocols

The following tables list ports and protocols required for network communications between Enterprise PKS v1.5.0 and later, and vSphere 6.7 and later.

Enterprise PKS Users Ports and Protocols

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

Source Component	Destination Component	Destination Protocol	Destination Port	Service
Admin/Operator Console	All System Components	TCP	22	ssh

Source Component	Destination Component	Destination Protocol	Destination Port	Service
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
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

Enterprise PKS Core Ports and Protocols

The following table lists ports and protocols used for network communication between core Enterprise 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 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

Source Component	Destination Component	Destination Protocol	Destination Port	Service
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	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

Source Component	Destination Component	Destination Protocol	Destination Port	Service
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

Source Component	Destination Component	Destination Protocol	Destination Port	Service
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
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	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.

Creating Dedicated Users and Roles for vSphere (Optional)

In this topic

Page last updated:

This topic describes how to create dedicated users and roles for your vSphere environment before deploying VMware Enterprise PKS.

> **Note**: This topic provides security considerations for defining dedicated vSphere user accounts for use with Kubernetes cluster VMs provisioned by Enterprise PKS. The information in this topic is only relevant if you **do not** want to use the vSphere administrator account for the Enterprise PKS and Kubernetes cluster VMs. If you are comfortable using the vSphere administrator account for the PKS and Kubernetes cluster VMs, skip this topic.

Overview

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

You can create the following service accounts in vSphere:

- **Master Node User Account** for the Kubernetes master node VMs.
- **BOSH/Ops Manager User 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 User 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 Enterprise PKS installation.
- **BOSH/Ops Manager User 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 user 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 user 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

Note: The following procedures document configuring vSphere Web Services API v6.7 permissions. For configuring vSphere Web Services API v6.5 permissions, see Creating Dedicated Users and Roles for vSphere (Optional) in the Enterprise PKS v1.5 documentation.

Prerequisites

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

Create the Master Node User Account

Virtual Machine Configuration privileges control the ability to configure virtual machine options and devices.

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

Privilege (UI)	Privilege (API)
Virtual Machine > Advanced configuration	VirtualMachine.Config.AdvancedConfig
Virtual Machine > Change Settings	VirtualMachine.Config.Settings

Grant Storage Permissions

Kubernetes master node VM user 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 Enterprise PKS is deployed

Grant these permissions to the master node user 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)

The procedures in this topic use the following vCenter permissions objects:

- **Virtual Machine Configuration** privileges control the ability to configure virtual machine options and devices. For information about **Virtual Machine Configuration** see Virtual Machine Configuration Privileges in the VMware vSphere documentation.
- **Datastore** privileges control the ability to browse, manage, and allocate space on datastores. For information about **Datastore** see Datastore Privileges in the VMware vSphere documentation.
- **Resource** privileges control the creation and management of resource pools, as well as the migration of virtual machines. For information about **Resource** see Resource Privileges in the VMware vSphere documentation.
- **Storage Views** privileges control privileges for Storage Monitoring Service APIs. **Starting with vSphere 6.0, storage views are deprecated and these privileges no longer apply to them.** For information about **Storage Views** see Storage Views Privileges in the VMware vSphere documentation. For more information about vSphere storage configurations, see vSphere Storage for Kubernetes in the VMware vSphere documentation.

For information about the vSphere virtual machine permissions API, see ReconfigVM_Task(reconfigure) in the *vSphere Web Services API* documentation.

Static Only Persistent Volume Provisioning

To configure your Kubernetes master node user 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. For more information about custom roles in vCenter, see Create a Custom Role in the VMware vSphere documentation.

   a. Give this role a name. For example, `manage-k8s-node-vms`.
   b. Grant the following privileges at the **VM Folder** level using either the vCenter UI or API:

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

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

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

   c. 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: 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 User Account.

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

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

1. Create a custom role that allows the user account to manage Kubernetes node VMs. For more information about custom roles in vCenter, see Create a Custom Role in the VMware vSphere documentation.

   a. Give this role a name. For example, `manage-k8s-node-vms`.
   b. Grant the following privileges at the **Cluster, Hosts,** and **VM Folder** levels using either the vCenter UI or API:

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

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

2. Create a custom role that allows the user account to manage Kubernetes volumes.

   a. Give this role a name. For example, `manage-k8s-volumes`.
   b. Grant the following privileges using either the vCenter UI or API:

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

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

3. Create a custom role that allows the user account to read the Kubernetes storage profile.

   a. Give this role a name. For example, `k8s-system-read-and-spbm-profile-view`.
   b. 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

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

4. Grant the user 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 user account using dynamic PV provisioning **without** storage policy-based placement, do the following:

1. Create a custom role that allows the user account to manage Kubernetes node VMs. For more information about custom roles in vCenter, see Create a Custom Role in the VMware vSphere documentation.

   a. Give this role a name. For example, `manage-k8s-node-vms`.
   b. 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 > Add existing disk	VirtualMachine.Config.AddExistingDisk
Virtual Machine > Add new disk	VirtualMachine.Config.AddNewDisk
Virtual Machine > Add or remove device	VirtualMachine.Config.AddRemoveDevice
Virtual Machine > Remove disk	VirtualMachine.Config.RemoveDisk

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

2. Create a custom role that allows the user account to manage Kubernetes volumes.

   a. Give this role a name. For example, `manage-k8s-volumes`.
   b. Grant the following privileges using either the vCenter UI or API:

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

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

3. Grant the user 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 User Account

1. From the vCenter console, create the BOSH/Ops Manager User 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 User Account

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

- Grant minimal permissions. Grant each BOSH/Ops Manager User 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 Enterprise PKS tile, enter this FQDN in the **PKS API** pane.

After you deploy Enterprise 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 Installation Step

To install and configure Ops Manager, follow the instructions in Installing and Configuring Ops Manager on vSphere.

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

Installing and Configuring Ops Manager on vSphere

In this topic

Page last updated:

This topic describes how to install and configure Ops Manager before deploying VMware Enterprise PKS on vSphere.

Prerequisites

You use Ops Manager to install and configure Enterprise PKS. Before you install Ops Manager, review the following prerequisites:

- vSphere Prerequisites and Resource Requirements
- Firewall Ports and Protocols Requirements for vSphere without NSX-T
- Creating Dedicated Users and Roles for vSphere (Optional)

Install and Configure Ops Manager

Multiple versions of Ops Manager are compatible with each version of Enterprise PKS. To confirm that your Ops Manager version is compatible with the version of Enterprise PKS that you want to install, see the Enterprise PKS Release Notes.

To install and configure Ops Manager, follow the instructions in the table below:

Version	Instructions
Ops Manager v2.6.x	1. Deploying Ops Manager on vSphere 2. Configuring BOSH Director on vSphere
Ops Manager v2.7.x	1. Deploying Ops Manager on vSphere 2. Configuring BOSH Director on vSphere

Next Installation Step

To install and configure Enterprise PKS, follow the instructions in Installing Enterprise PKS on vSphere.

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

Installing Enterprise PKS on vSphere

In this topic

Page last updated:

This topic describes how to install and configure VMware Enterprise PKS on vSphere with Flannel.

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 Enterprise 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 Enterprise PKS

To install Enterprise 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 **Enterprise PKS** in the left column, click the plus sign to add this product to your staging area.

Step 2: Configure Enterprise PKS

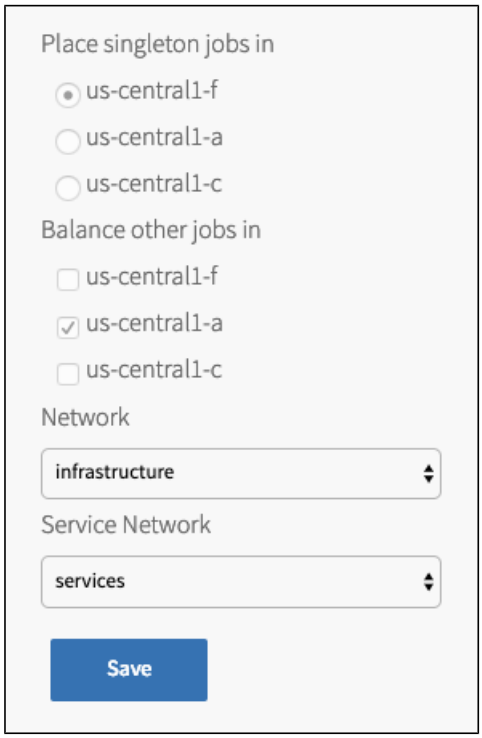
Click the orange **Enterprise PKS** tile to start the configuration process.

warning: When you configure the Enterprise 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 Enterprise PKS fails.

Assign AZs and Networks

To configure the availability zones (AZs) and networks used by the Enterprise PKS control plane:

1. Click **Assign AZs and Networks**.
2. Under **Place singleton jobs in**, select the availability zone (AZ) where you want to deploy the PKS API VM as a singleton job.



3. Under **Balance other jobs in**, select the AZ for balancing other Enterprise PKS control plane jobs.

> **Note**: You must specify the **Balance other jobs in** AZ, but the selection has no effect in the current version of Enterprise PKS.

4. Under **Network**, select the infrastructure subnet that you created for the PKS API VM.

5. Under **Service Network**, select the services subnet that you created for Kubernetes cluster VMs.

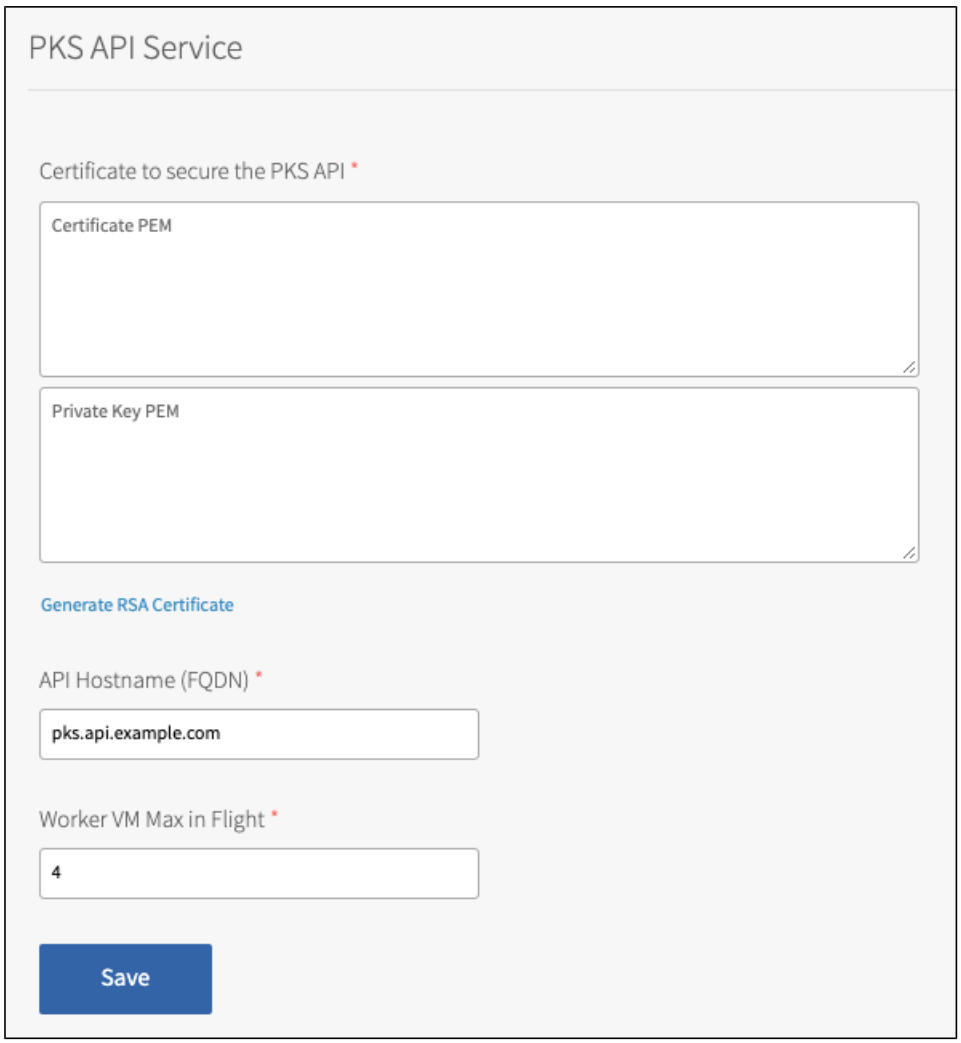
6. Click **Save**.

PKS API

Perform the following steps:

1. Click **PKS API**.

2. Under **Certificate to secure the PKS API**, provide a certificate and private key pair.

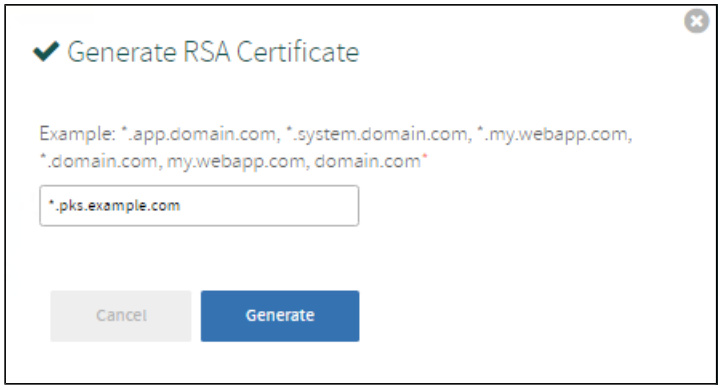


The certificate that you supply should cover the specific subdomain that routes to the PKS API VM with TLS termination on the ingress.

> **warning:** TLS certificates generated for wildcard DNS records only work for a single domain level. For example, a certificate generated for `*.pks.EXAMPLE.com` does not permit communication to `*.api.pks.EXAMPLE.com`. If the certificate does not contain the correct FQDN for the PKS API, calls to the API will fail.

You can enter your own certificate and private key pair, or have Ops Manager generate one for you.
To generate a certificate using Ops Manager:

a. If the **Generate RSA Certificate** option is not available, click **Change**.
b. Click **Generate RSA Certificate**.
c. Enter the domain for your API hostname. This can be a standard FQDN or a wildcard domain.
d. 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 value. When you create or resize a cluster, the `max_in_flight` value limits the number of component instances that can be created or started simultaneously. By default, the `max_in_flight` value is set to `4`, which means that up to four component instances are simultaneously created or started at a time.

5. Click **Save**.

Plans

A plan defines a set of resource types used for deploying a cluster.

Activate a Plan

You must first activate and configure **Plan 1**,

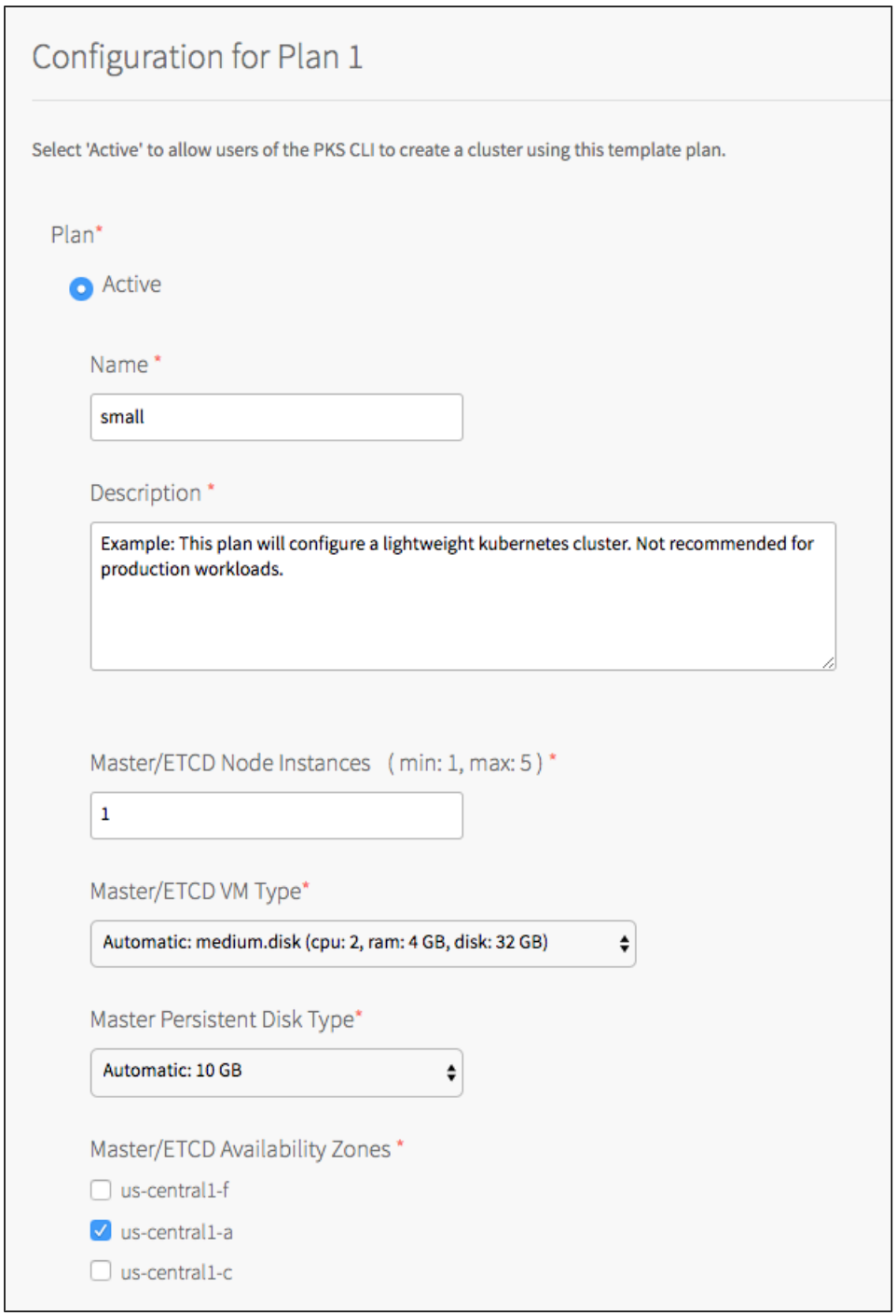
and afterwards you can activate up to twelve additional, optional, plans.

To activate and configure a plan, perform the following steps:

1. Click the plan that you want to activate.

   > **Note**: Plans 11, 12 and 13 support only Windows worker-based Kubernetes clusters, on vSphere with Flannel. To configure a Windows worker plan see Plans in *Configuring Windows Worker-Based Kubernetes Clusters (Beta)* for more information.

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 the 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 `1`, `3`, or `5`.

> **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.

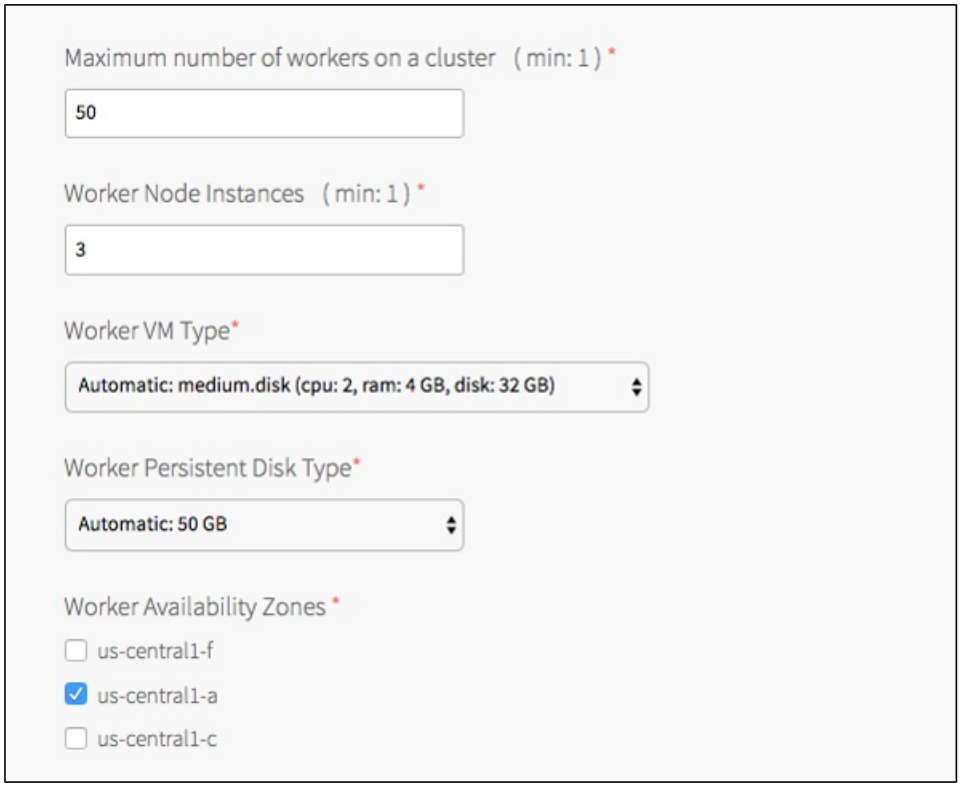
warning: To change the number of master/etcd nodes for a plan, you must ensure that no existing clusters use the plan. Enterprise PKS does not support changing the number of master/etcd nodes for plans with existing clusters.

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 Enterprise 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 Enterprise PKS. If you select more than one AZ, Enterprise PKS deploys the master VM in the first AZ and the worker VMs across the remaining AZs. If you are using multiple masters, Enterprise PKS deploys the master and worker VMs across the AZs in round-robin fashion.

9. Under **Maximum number of workers on a cluster**, set the maximum number of Kubernetes worker node VMs that Enterprise 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.

For more information about creating clusters, see Creating Clusters.

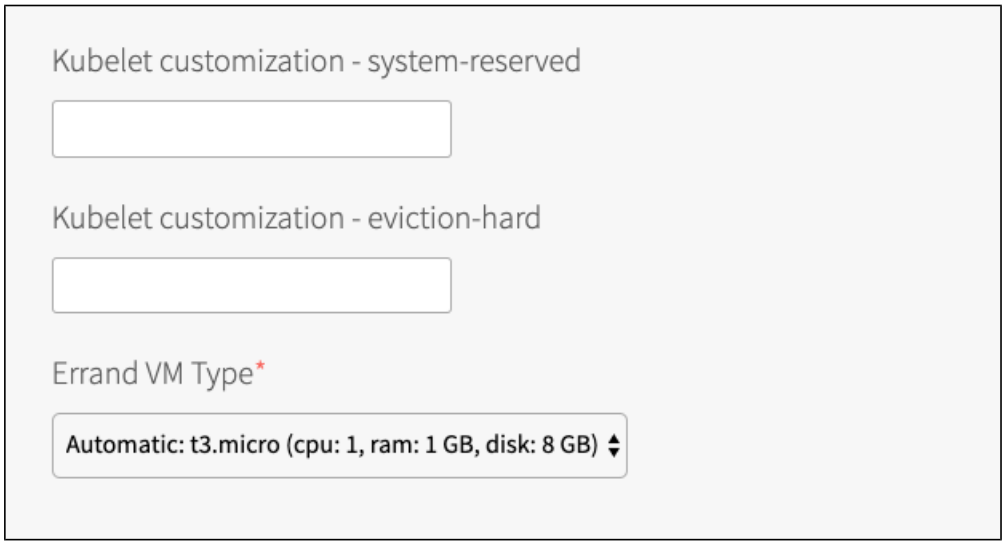
> **Note**: Changing a plan's **Worker Node Instances** setting does not alter the number of worker nodes on existing clusters. For information about scaling an existing cluster, see Scale Horizontally by Changing the Number of Worker Nodes Using the PKS CLI in *Scaling Existing Clusters*.

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 Enterprise PKS Clusters*.

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. Enterprise 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.

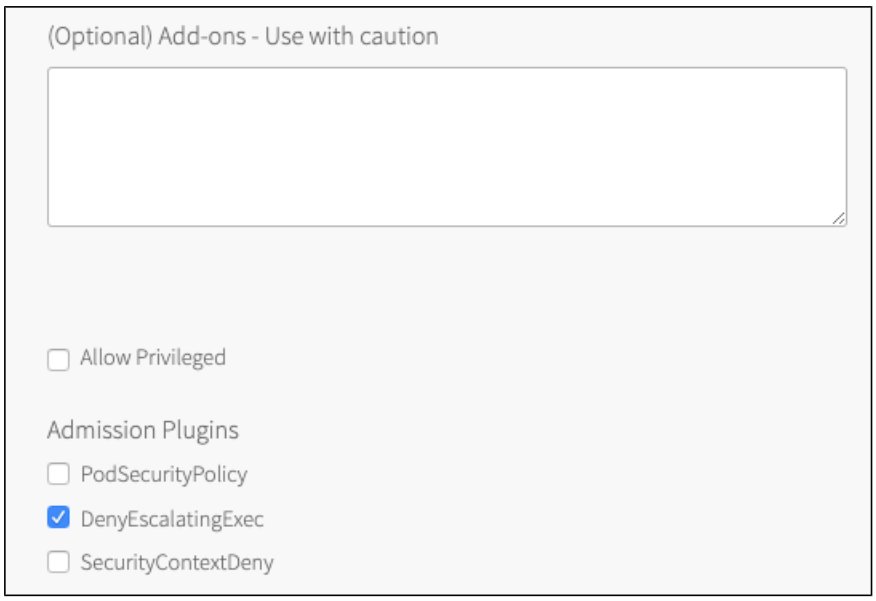


15. Under **Kubelet customization - eviction-hard**, enter threshold limits that Kubelet can use to evict pods when they exceed the limit. Enter limits in the format `EVICTION-SIGNAL=QUANTITY`. For example, `memory.available=100Mi, nodefs.available=10%, nodefs.inodesFree=5%`. For more information about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



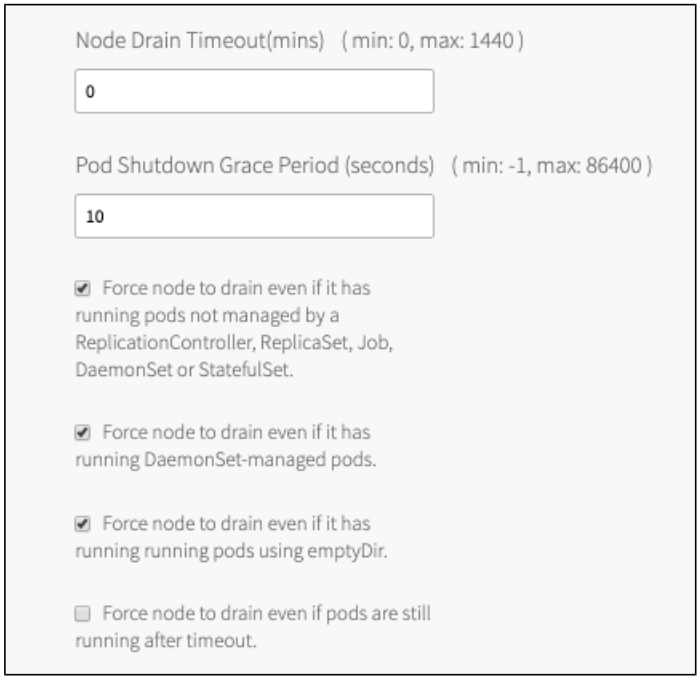
Linux Workloads.

18. (Optional) To allow users to create pods with privileged containers, select the **Allow Privileged** option. For more information, see Pods in the Kubernetes documentation.

> **Note:** Enabling the `Allow Privileged` option means that all containers in the cluster will run in privileged mode. Pod Security Policy provides a privileged parameter that can be used to enable or disable Pods running in privileged mode. As a best practice, if you enable `Allow Privileged`, define PSP to limit which Pods run in privileged mode. If you are implementing PSP for privileged pods, you must enable `Allow Privileged` mode.

19. (Optional) Enable or disable one or more admission controller plugins: **PodSecurityPolicy**, **DenyEscalatingExec**, and **SecurityContextDeny**. See Admission Plugins for more information.

20. (Optional) Under **Node Drain Timeout(mins)**, enter the timeout in minutes for the node to drain pods. If you set this value to



`0`, the node drain does not terminate.

21. (Optional) Under **Pod Shutdown Grace Period (seconds)**, enter a timeout in seconds for the node to wait before it forces the pod to terminate. If you set this value to `-1`, the default timeout is set to the one specified by the pod.

22. (Optional) To configure when the node drains, enable the following:
    - **Force node to drain even if it has running pods not managed by a ReplicationController, ReplicaSet, Job, DaemonSet or StatefulSet.**
    - **Force node to drain even if it has running DaemonSet-managed pods.**
    - **Force node to drain even if it has running running pods using emptyDir.**
    - **Force node to drain even if pods are still running after timeout.**

    > **warning:** If you select **Force node to drain even if pods are still running after timeout**, the node kills all running workloads on pods. Before enabling this configuration, set **Node Drain Timeout** to a value greater than `0`.

    For more information about configuring default node drain behavior, see Worker Node Hangs Indefinitely in *Troubleshooting*.

23. Click **Save**.

Deactivate a Plan

To deactivate a plan, perform the following steps:

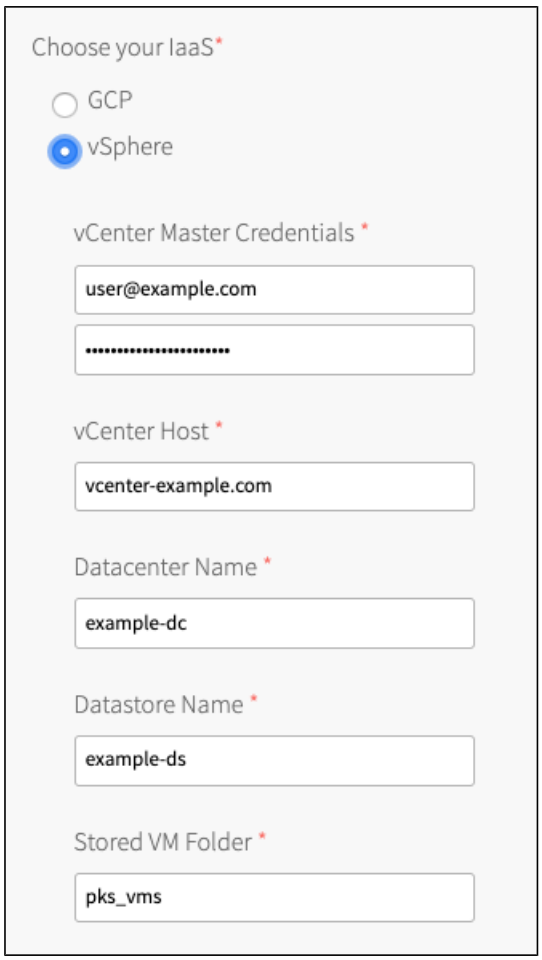
1. Click the plan that you want to deactivate.
2. Select **Inactive**.
3. Click **Save**.

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 Enterprise 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 vSphere Before Deploying Enterprise PKS.
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`. Populate **Datastore Name** with the Persistent Datastore name configured in your **BOSH Director** tile under **vCenter Config** > **Persistent Datastore Names**. The **Datastore Name** field should contain a single Persistent datastore.

> **Note**: The vSphere datastore type must be Datastore. Enterprise PKS does not support the use of vSphere Datastore Clusters with or without Storage DRS. For more information, see Datastores and Datastore Clusters in the vSphere documentation.

> **Note**: The **Datastore Name** is the default datastore used if the Kubernetes cluster `StorageClass` does not define a `StoragePolicy`. Do not enter a datastore that is a list of BOSH Job/VMDK datastores. For more information, see PersistentVolume Storage Options on vSphere.

> **Note**: For multi-AZ and multi-cluster environments, your **Datastore Name** should be a shared Persistent datastore available to each vSphere cluster. Do not enter a datastore that is local to a single cluster. For more information, see PersistentVolume Storage Options on vSphere.

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

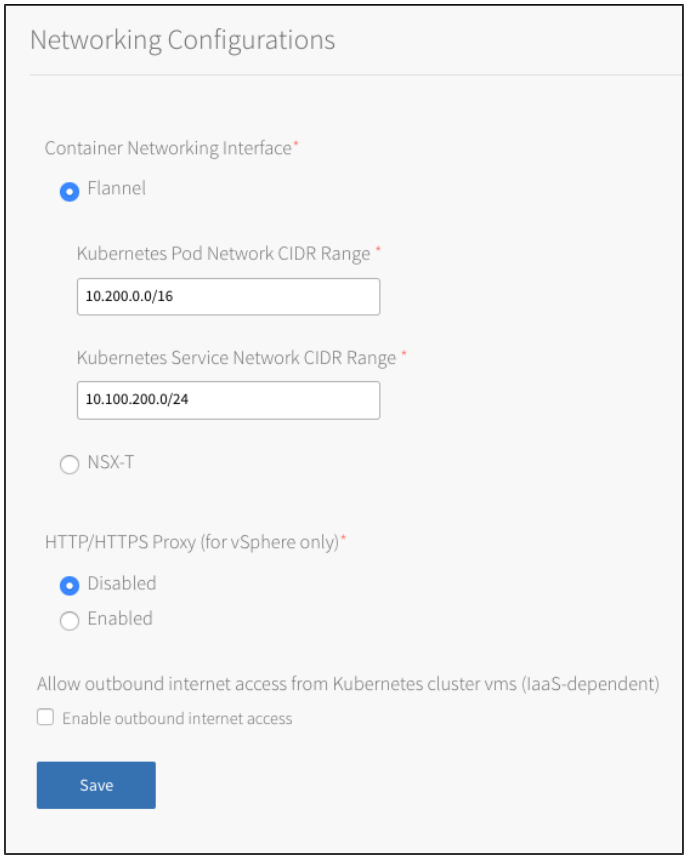
`pks_vms`.

f. Click **Save**.

Networking

To configure networking, do the following:

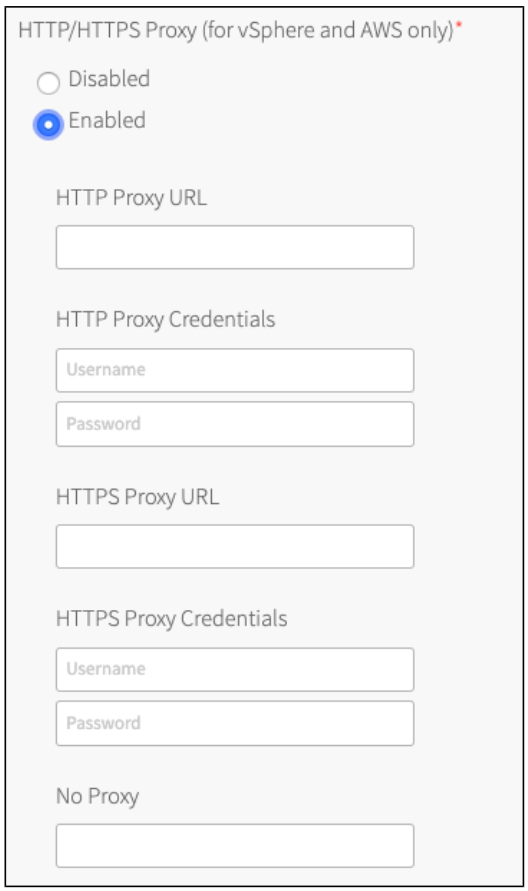
1. Click **Networking**.



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.



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

Configuring Enterprise PKS to use these proxies allows Enterprise PKS-deployed Kubernetes nodes to access public Internet services and other internal services.

> **Note**: This setting will not set the proxy for running Kubernetes workloads or pods.

To complete your global proxy configuration for all outgoing HTTP/HTTPS traffic from your Kubernetes clusters, perform the following steps:

- To proxy outgoing HTTP traffic, enter the URL of your HTTP proxy endpoint under **HTTP Proxy URL**. For example, `http://myproxy.com:1234`.
- (Optional) If your outgoing HTTP proxy uses basic authentication, enter the username and password in the **HTTP Proxy Credentials** fields.
- To proxy outgoing HTTPS traffic, enter the URL of your HTTP proxy endpoint under **HTTPS Proxy URL**. For example, `http://myproxy.com:1234`.

> **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.

- (Optional) If your HTTPS proxy uses basic authentication, enter the username and password in the **HTTPS Proxy Credentials** fields.
- Under **No Proxy**, enter the comma-separated list of IP addresses that must bypass the proxy to allow for internal

Enterprise PKS communication.

The list should include `127.0.0.1` and `localhost`.

Also include:

- Your Enterprise PKS environment's CIDRs, such as the service network CIDR where your Enterprise PKS cluster is deployed, the deployment network CIDR, the node network IP block CIDR, and the pod network IP block CIDR.
- If a component is communicating with Enterprise PKS or a registry using a hostname, such as the Harbor API FQDN, instead of an IP address you will need to add the corresponding FQDN to the **No Proxy** list.
- 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 Enterprise PKS to IaaS traffic from going through the proxy: NSX Manager, vCenter Server, and all ESXi hosts.
- 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:

  ```
  127.0.0.1,localhost,
  *.example1.com,
  .example2.com,
  example3.com,
  198.51.100.0/24,
  203.0.113.0/24,
  192.0.2.0/24
  ```

  Note: By default the `10.100.0.0/8` and `10.200.0.0/8` IP address ranges, `.internal`, `.svc`, `.svc.cluster.local`, `.svc.cluster`, and your Enterprise PKS FQDN are not proxied. This allows internal Enterprise 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.

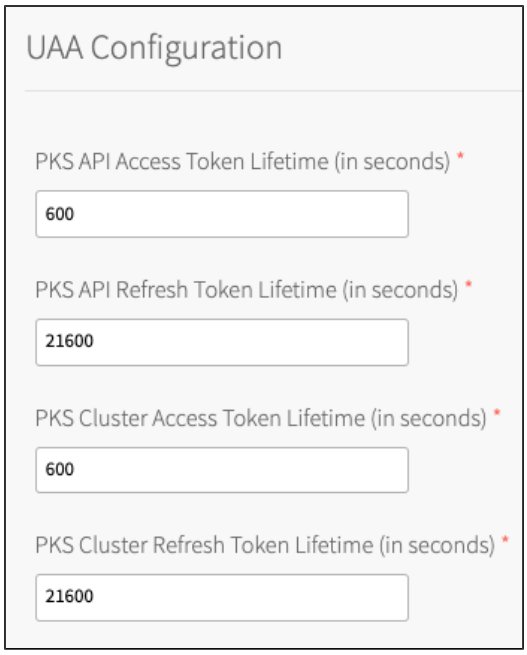
4. Under **Allow outbound internet access from Kubernetes cluster vms (IaaS-dependent)**, ignore the **Enable outbound internet access** checkbox.
5. Click **Save**.

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. This field defaults to `600`.



3. Under **PKS API Refresh Token Lifetime**, enter a time in seconds for the PKS API refresh token lifetime. This field defaults to `21600`.

4. Under **PKS Cluster Access Token Lifetime**, enter a time in seconds for the cluster access token lifetime. This field defaults to `600`.

5. Under **PKS Cluster Refresh Token Lifetime**, enter a time in seconds for the cluster refresh token lifetime. This field defaults to `21600`.

> **Note**: Pivotal recommends using the default UAA token timeout values. By default, access tokens expire after ten minutes and refresh tokens expire after six hours. If you want to customize your token timeout values, see Token Management in *UAA Overview*.

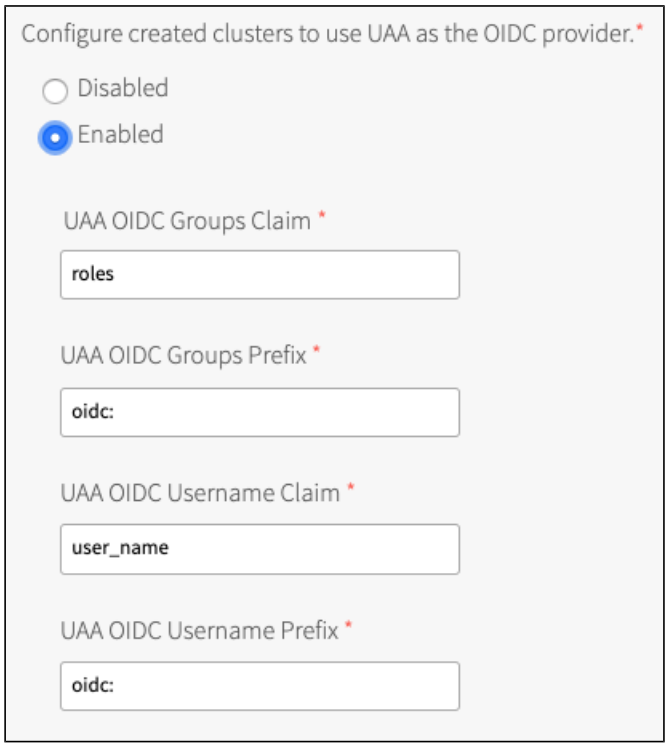
6. Under **Configure created clusters to use UAA as the OIDC provider**, select **Enabled** or **Disabled**. If you click **Enabled**, Kubernetes verifies end-user identities based on authentication executed by UAA. For more information, see the table below.

Option	Description
Disabled	If you do not enable UAA as the OpenID Connect (OIDC) provider, Kubernetes authenticates users against its internal user management system.
Enabled	If you enable UAA as the OIDC provider, Kubernetes authenticates users as follows: ◦ If you select **Internal UAA** in the next step, Kubernetes authenticates users against the internal UAA authentication mechanism. ◦ If you select **LDAP Server** in the next step, Kubernetes authenticates users against the LDAP server. ◦ If you select **SAML Identity Provider** in the next step, Kubernetes authenticates users against the SAML identity provider.

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

To configure Enterprise PKS to use UAA as the OIDC provider, do the following:

a. Under **Configure created clusters to use UAA as the OIDC provider**, select **Enabled**.



b. For **UAA OIDC Groups Claim**, enter the name of your groups claim. This is used to set a user's group in the JSON Web Token (JWT) claim. The default value is `roles`.
c. For **UAA OIDC Groups Prefix**, enter a prefix for your groups claim. This prevents conflicts with existing names. For example, if you enter the prefix `oidc:`, UAA creates a group name like `oidc:developers`. If you are configuring a new Enterprise PKS installation, the default is `oidc:`. If you are upgrading to Enterprise PKS v1.5, the default is `-`.
d. For **UAA OIDC Username Claim**, enter the name of your username claim. This is used to set a user's username in the JWT claim. The default value is `user_name`. Depending on your provider, admins can enter claims besides `user_name`, like `email` or `name`.
e. For **UAA OIDC Username Prefix**, enter a prefix for your username claim. This prevents conflicts with existing names. For example, if you enter the prefix `oidc:`, UAA creates a username like `oidc:admin`. If you are configuring a new Enterprise PKS installation, the default is `oidc:`. If you are upgrading to Enterprise PKS v1.5, the default is `-`.

> **Note:** Pivotal recommends adding OIDC prefixes to prevent OIDC users and groups from gaining unintended cluster privileges. When you upgrade to Enterprise PKS v1.5, if you do not change the values for **UAA OIDC Groups Prefix** or **UAA OIDC Username Prefix**, Enterprise PKS does not add prefixes.

> **warning:** If you change the above values for a pre-existing Enterprise PKS installation, you must change any existing role bindings that bind to a username or group. If you do not change your role bindings, developers cannot access Kubernetes clusters. For instructions about creating role bindings, see Managing Cluster Access and Permissions.

7. Select one of the following options:

   - To use an internal user account store for UAA, select **Internal UAA**. Click **Save** and continue to (Optional) Host Monitoring.

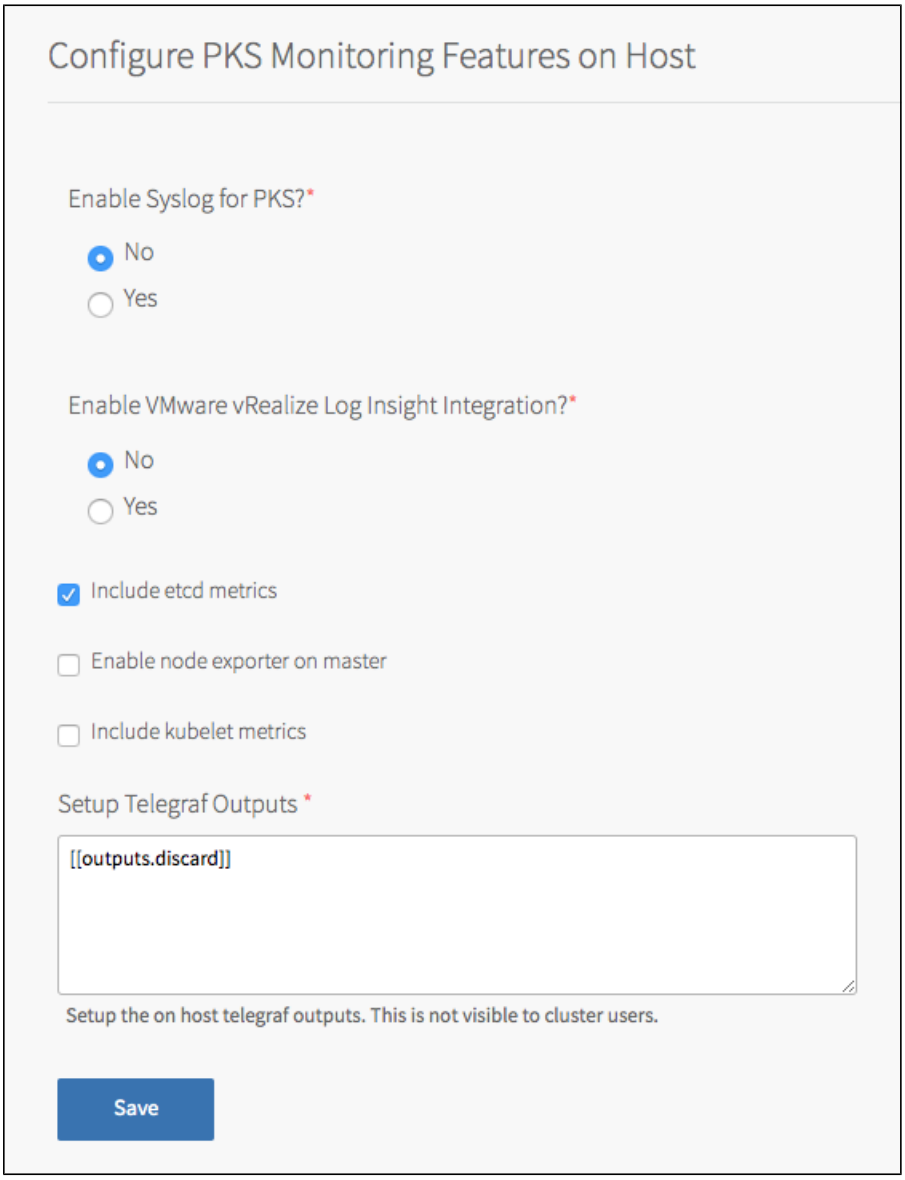
- To use LDAP for UAA, select **LDAP Server** and continue to Connecting Enterprise PKS to an LDAP Server.
- To use SAML for UAA, select **SAML Identity Provider** and continue to Connecting Enterprise PKS to a SAML Identity Provider.

(Optional) Host Monitoring

In **Host Monitoring**, you can configure one or more of the following:

- To configure Syslog for BOSH-deployed VMs, see Syslog. Syslog forwards log messages from all BOSH-deployed VMs to a syslog endpoint.
- To configure VMware vRealize Log Insight (vRLI) Integration, see VMware vRealize Log Insight Integration. The vRLI 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`.
- To configure the Telegraf agent, see Telegraf. The Telegraf agent sends metrics from master node and worker node VMs to a monitoring service, such as Wavefront or Datadog.

For more information about these components, see Monitoring PKS and PKS-Provisioned Clusters.



Syslog

To configure Syslog for all BOSH-deployed VMs in Enterprise PKS:

1. Click **Host Monitoring**.
2. Under **Enable Syslog for PKS**, select **Yes**.
3. Under **Address**, enter the destination syslog endpoint.
4. Under **Port**, enter the destination syslog port.
5. Under **Transport Protocol**, select a transport protocol for log forwarding.
6. (Optional) To enable TLS encryption during log forwarding, complete the following steps:
   a. Ensure **Enable TLS** is selected.

      > **Note**: Logs may contain sensitive information, such as cloud provider credentials. Pivotal strongly recommends that you enable TLS encryption for log forwarding.

   b. Under **Permitted Peer**, provide the accepted fingerprint (SHA1) or name of remote peer. For example, `*.YOUR-LOGGING-SYSTEM.com`.
   c. Under **TLS Certificate**, provide a TLS certificate for the destination syslog endpoint.

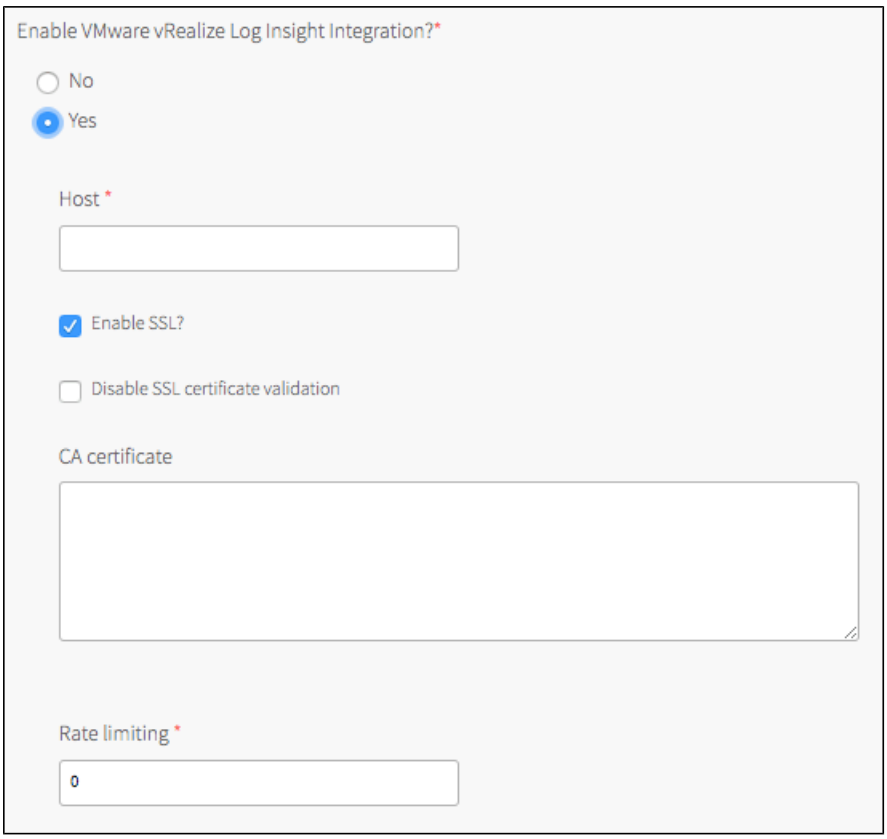
      > **Note**: You do not need to provide a new certificate if the TLS certificate for the destination syslog endpoint is signed by a Certificate Authority (CA) in your BOSH certificate store.

7. (Optional) Under **Max Message Size**, enter a maximum message size for logs that are forwarded to a syslog endpoint. By default, the **Max Message Size** field is 10,000 characters.
8. Click **Save**.

VMware vRealize Log Insight Integration

> **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.

To configure vRLI logging, under **Enable VMware vRealize Log Insight Integration?**, select **Yes** and then perform the following steps:



1. Under **Host**, enter the IP address or FQDN of the vRLI host.
2. (Optional) Select the **Enable SSL?** checkbox to encrypt the logs being sent to vRLI using SSL.
3. 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.
4. (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.
5. 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 that 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 value, such as `10`, then tuning to optimize for your deployment. A large number might result in dropping too many log entries.

6. 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 Enterprise PKS tile does not validate your vRLI configuration settings. To verify your setup, look for log entries in vRLI.

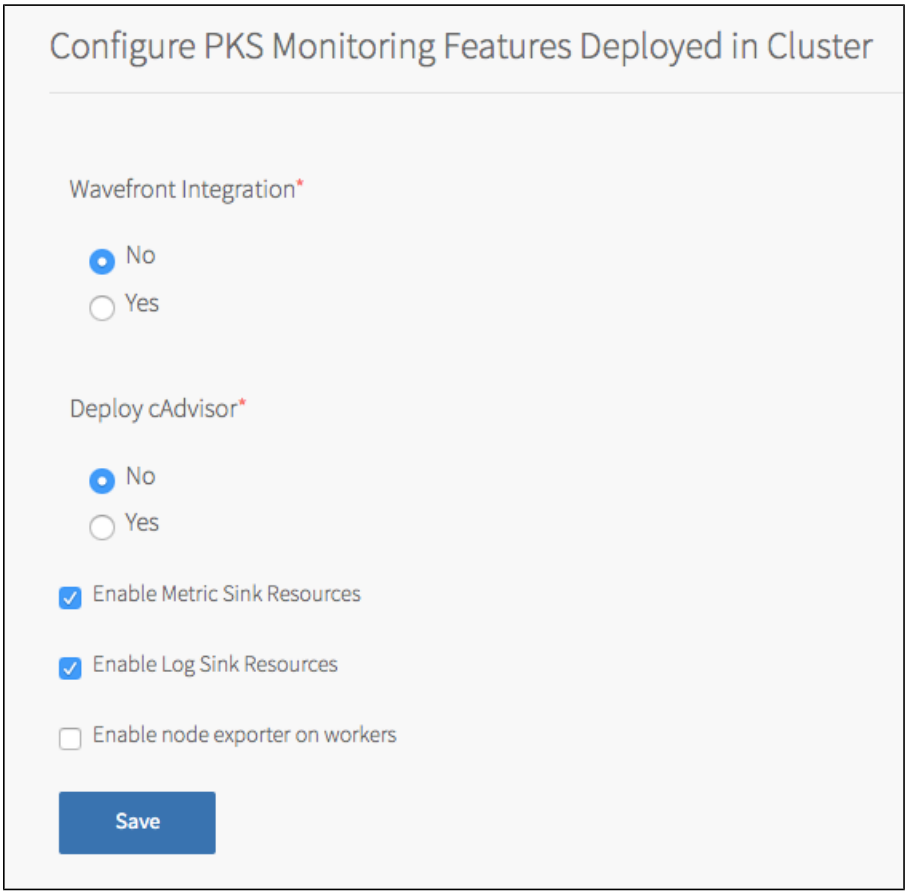
Telegraf

To configure Enterprise PKS to use Telegraf for metric collection:

1. Create a configuration file for your monitoring service. For instructions, see Create a Configuration File.
2. Select one or more of the following checkboxes:
   - **Include etcd metrics**: This includes etcd server and debug metrics.
   - **Enable node exporter on master**: This enables Node Exporter on the *localhost* of each master node VM.
   - **Include kubelet metrics**: This includes all workload metrics across your Kubernetes clusters. Enabling **Include kubelet metrics** generates a high volume of metrics.
3. In **Setup Telegraf Outputs**, replace the default value `[[outputs.discard]]` with the contents of the configuration file that you created above.
4. Click **Save**.

(Optional) In-Cluster Monitoring

In **In-Cluster Monitoring**, you can configure one or more observability components and integrations that run in Kubernetes clusters and capture logs and metrics about your workloads. For more information, see Monitoring Workers and Workloads.



To configure in-cluster monitoring:

- To configure Wavefront, see Wavefront.
- To configure cAdvisor, see VMware vRealize Operations Management Pack for Container Monitoring.
- To configure sink resources, see:
  - Metric Sink Resources
  - Log Sink Resources

  You can enable both log and metric sink resources or only one of them.

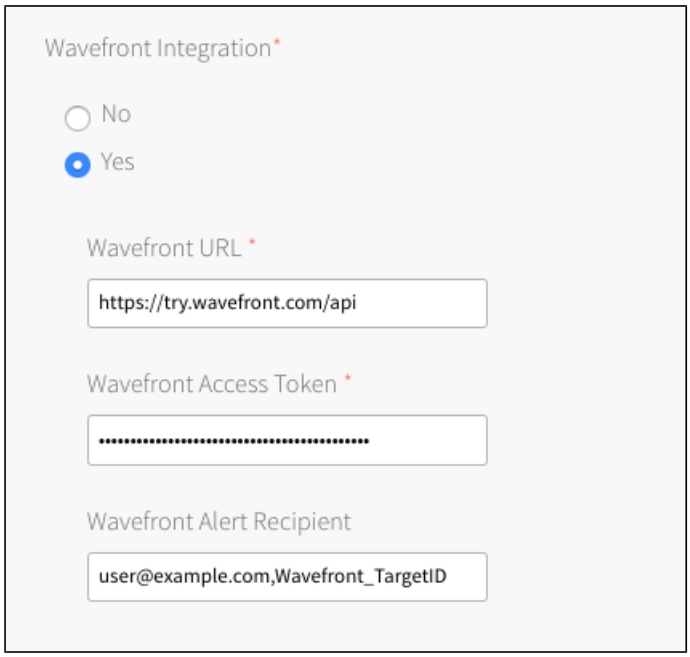
Wavefront

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 the Wavefront documentation.

To enable and configure Wavefront monitoring:

1. In the Enterprise PKS tile, select **In-Cluster Monitoring**.



2. 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**, using the following syntax:

```
USER-EMAIL,WAVEFRONT-TARGETID_001,WAVEFRONT-TARGETID_002
```

Where:

- `USER-EMAIL` is the alert recipient's email address.
- `WAVEFRONT-TARGETID_001` and `WAVEFRONT-TARGETID_002` are your comma-delimited Wavefront Target IDs.

For example:

```
randomuser@example.com,51n6psdj933ozdjf
```

6. Click **Save**.

To create alerts, you must enable errands in Enterprise PKS.

1. In the Enterprise PKS tile, select **Errands**.
2. On the **Errands** pane, enable **Create pre-defined Wavefront alerts errand**.
3. Enable **Delete pre-defined Wavefront alerts errand**.
4. Click **Save**. Your settings apply to any clusters created after you have saved these configuration settings and clicked **Apply Changes**.

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

VMware vRealize Operations Management Pack for Container Monitoring

You can monitor Enterprise PKS Kubernetes clusters with VMware vRealize Operations Management Pack for Container Monitoring.

To integrate Enterprise PKS with VMware vRealize Operations Management Pack for Container Monitoring, you must deploy a container running cAdvisor in your PKS deployment.

cAdvisor is an open source tool that provides monitoring and statistics for Kubernetes clusters.

To deploy a cAdvisor container:

1. Select **In-Cluster Monitoring**.
2. Under **Deploy cAdvisor**, select **Yes**.
3. Click **Save**.

For more information about integrating this type of monitoring with PKS, see the VMware vRealize Operations Management Pack for Container Monitoring User Guide and Release Notes in the VMware documentation.

Metric Sink Resources

You can configure PKS-provisioned clusters to send Kubernetes node metrics and pod metrics to metric sinks. For more information about metric sink resources and what to do after you enable them in the tile, see Sink Resources in *Monitoring Workers and Workloads*.

To enable clusters to send Kubernetes node metrics and pod metrics to metric sinks:

1. In **In-Cluster Monitoring**, select **Enable Metric Sink Resources**. If you enable this checkbox, Enterprise PKS deploys Telegraf as a `DaemonSet`, a pod that runs on each worker node in all your Kubernetes clusters.

2. (Optional) To enable Node Exporter to send worker node metrics to metric sinks of kind `ClusterMetricSink`, select **Enable node exporter on workers**. If you enable this checkbox, Enterprise PKS deploys Node Exporter as a `DaemonSet`, a pod that runs on each worker node in all your Kubernetes clusters.
   For instructions on how to create a metric sink of kind `ClusterMetricSink` for Node Exporter metrics, see Create a ClusterMetricSink Resource for Node Exporter Metrics in *Creating and Managing Sink Resources*.

3. Click **Save**.

Log Sink Resources

You can configure PKS-provisioned clusters to send Kubernetes API events and pod logs to log sinks. For more information about log sink resources and what to do after you enable them in the tile, see Sink Resources in *Monitoring Workers and Workloads*.

To enable clusters to send Kubernetes API events and pod logs to log sinks:

1. Select **Enable Log Sink Resources**. If you enable this checkbox, Enterprise PKS deploys Fluent Bit as a `DaemonSet`, a pod that runs on each worker node in all your Kubernetes clusters.

2. Click **Save**.

Tanzu Mission Control (Experimental)

Participants in the VMware Tanzu Mission Control beta program can use the **Tanzu Mission Control (Experimental)** pane of the Enterprise PKS tile to integrate their Enterprise PKS deployment with Tanzu Mission Control.

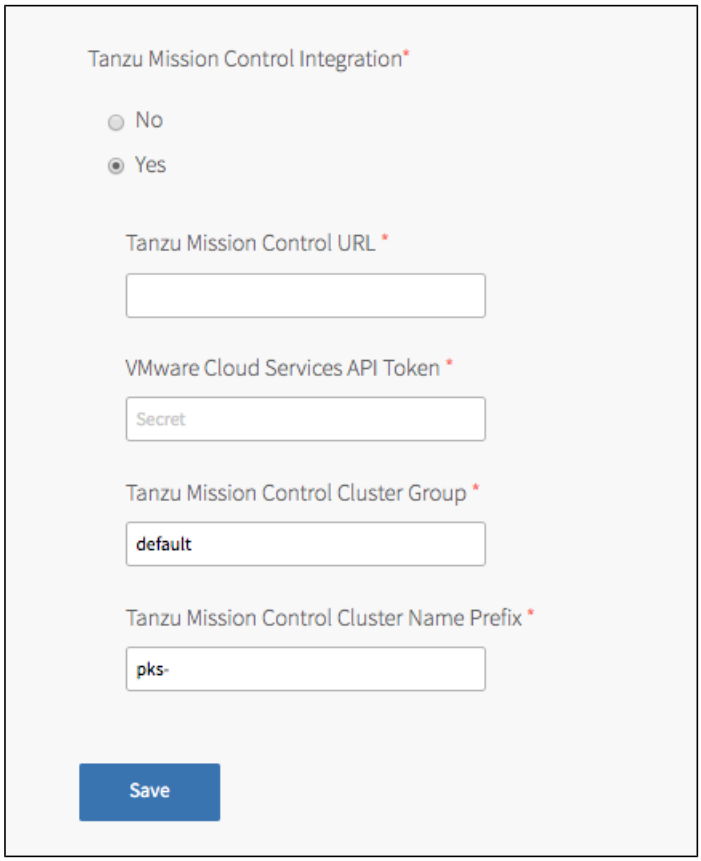
Tanzu Mission Control integration lets you monitor and manage Enterprise PKS clusters from the Tanzu Mission Control console, which makes the Tanzu Mission Control console a single point of control for all Kubernetes clusters.

> **warning:** VMware Tanzu Mission Control is currently experimental beta software and is intended for evaluation and test purposes only. For more information about Tanzu Mission Control, see the VMware Tanzu Mission Control home page.

To integrate Enterprise PKS with Tanzu Mission Control:

1. Confirm that the PKS Control Plane VM has internet access and can connect to `*.tmc.cloud.vmware.com` and the other outbound URLs listed in the What Happens When You Attach a Cluster section of the Tanzu Mission Control documentation.

2. Navigate to the **Enterprise PKS** tile > the **Tanzu Mission Control (Experimental)** pane and select **Yes** under **Tanzu Mission Control Integration**.



3. Configure the fields below:

   - **Tanzu Mission Control URL**: Enter the Org URL of your Tanzu Mission Control subscription, without a trailing slash (`/`). For example, `YOUR-ORG.tmc.cloud.vmware.com`.
   - **VMware Cloud Services API token**: Enter your API token to authenticate with VMware Cloud Services APIs. You can retrieve this token by logging in to VMware Cloud Services and viewing your account information.
   - **Tanzu Mission Control Cluster Group**: Enter the name of a Tanzu Mission Control cluster group.

     The name can be `default` or another value, depending on your role and access policy:

     - `Org Member` users in VMware cloud services have a `service.admin` role in Tanzu Mission Control. These users:
       - By default, can create and attach clusters only in the `default` cluster group.
       - Can create and attach clusters to other cluster groups after an `organization.admin` user grants them the `clustergroup.admin` or `clustergroup.edit` role for those groups.
     - `Org Owner` users in VMware cloud services have `organization.admin` permissions in Tanzu Mission Control. These users:
       - Can create cluster groups.
       - Can grant `clustergroup` roles to `service.admin` users through the Tanzu Mission Control Access Policy view.

     For more information about role and access policy, see Access Control in the *VMware Tanzu Mission Control Product Documentation*.

   - **Tanzu Mission Control Cluster Name Prefix**: Enter a name prefix for identifying the Enterprise PKS clusters in Tanzu Mission Control.

4. Click **Save**.

warning: After the Enterprise PKS tile is deployed with a configured cluster group, the cluster group cannot be updated.

Note: When you upgrade your Kubernetes clusters and have Tanzu Mission Control integration enabled, existing clusters will be attached to Tanzu Mission Control.

CEIP and Telemetry

To configure VMware's Customer Experience Improvement Program (CEIP) and the Pivotal Telemetry Program (Telemetry), do the following:

1. Click **CEIP and Telemetry**.
2. Review the information about the CEIP and Telemetry.

CEIP and Telemetry

About the CEIP and Telemetry Program

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.

Customers who participate (at the enhanced tier) are eligible for several benefits, including Proactive Support.

Additional information regarding the data collected through CEIP or Telemetry, and the purposes for which it is used by VMware is set forth in the Trust & Assurance Center and for Pivotal on the Pivotal Telemetry pages. If you prefer not to participate in CEIP and Telemetry for PKS, you should not join below. You may join or leave CEIP and Telemetry for PKS at any time.

Levels of Participation

No personally identifiable information (PII) is collected at either level of participation. Please refer to the data dictionary for more information on the data we collect.

Standard: This participation tier is anonymous. Your data will be used to improve PKS, but is not identifiable to your organization

Enhanced: This participation tier allows us to identify your organization so we may provide proactive support and other benefits. See sample reports for more details.

Please Note

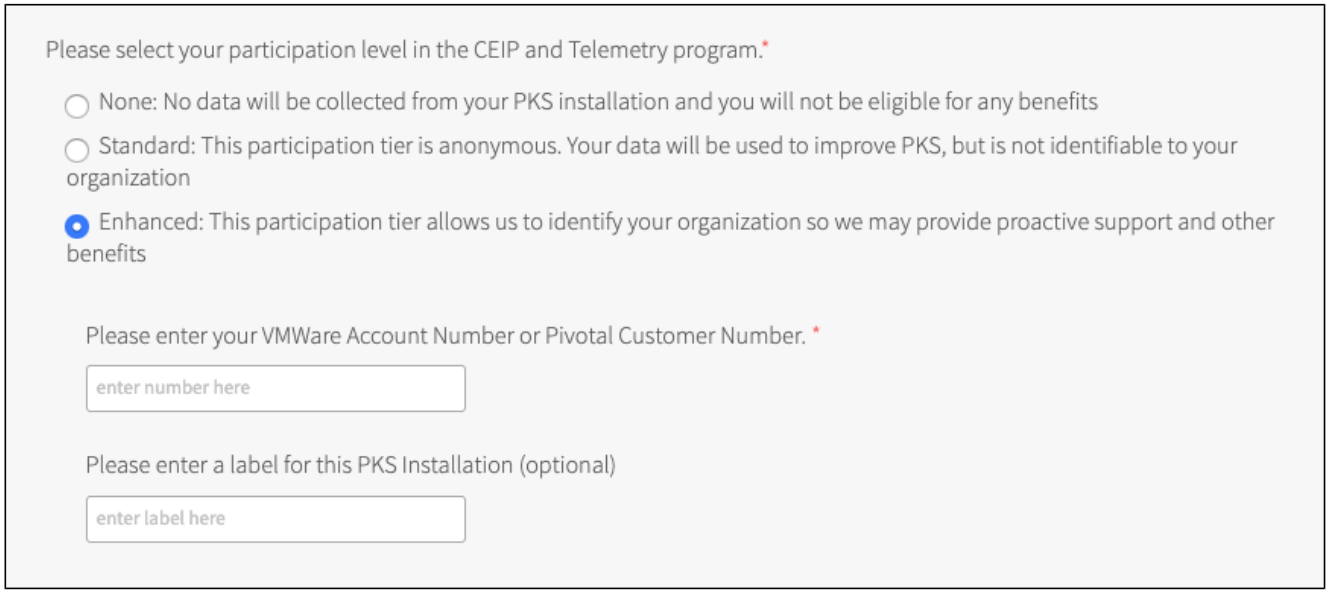
- If you are opting in on behalf of an organization (and not for you as an individual), you represent and warrant that you have legal authority to bind that organization, and you hereby join CEIP/Telemetry on behalf of your organization.
- If you are opting in to the enhanced tier, in the event a term or condition of CEIP or the Telemetry program conflicts with a term or condition of a previously executed license procurement agreement between you and Licensor (Pivotal or VMware), the CEIP or Telemetry program terms supersede solely for purposes of CEIP and Telemetry
- If you are running PKS on a private network, you will need to enable outgoing internet access by opening your firewall to allow traffic to `https://vcsa.vmware.com/ph` on port 443

Resources

- Data Dictionary
- Participation Benefits
- Sample Reports
- Trust and Assurance Center
- Pivotal Telemetry

View a larger version of this image.

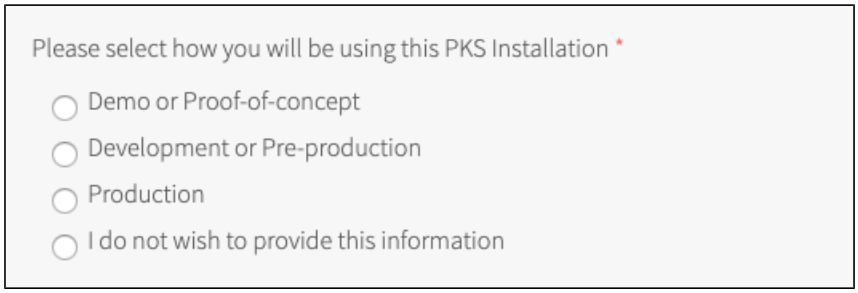
3. To specify your level of participation in the CEIP and Telemetry program, select one of the **Participation Level** options:
   - **None:** If you select this option, data is not collected from your Enterprise PKS installation.
   - (Default) **Standard:** If you select this option, data is collected from your Enterprise PKS installation to improve Enterprise PKS. This participation level is anonymous and does not permit the CEIP and Telemetry to identify your organization.
   - **Enhanced:** If you select this option, data is collected from your Enterprise PKS installation to provide you proactive support and other benefits. This participation level permits the CEIP and Telemetry to identify your organization.

   For more information about the CEIP and Telemetry participation levels, see Participation Levels in *Telemetry*.

4. If you selected the **Enhanced** participation level, complete the following:
   - Enter your VMware account number or Pivotal customer number in the **VMware Account Number or Pivotal Customer Number** field. If you are a VMware customer, you can find your VMware Account Number in your **Account Summary** on my.vmware.com. If you are a Pivotal customer, you can find your Pivotal Customer Number in your Pivotal Order Confirmation email.
   - (Optional) Enter a descriptive name for your PKS installation in the **PKS Installation Label** field. The label you assign to this installation will be used in telemetry reports to identify the environment.

5. To provide information about the purpose for this installation, select an option in the **PKS Installation Type** list.

6. Click **Save**.

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

Note: Even if you select **None**, Enterprise PKS-provisioned clusters send usage data to the PKS control plane. However, this

data is not sent to VMware or Pivotal and remains on your Enterprise PKS installation.

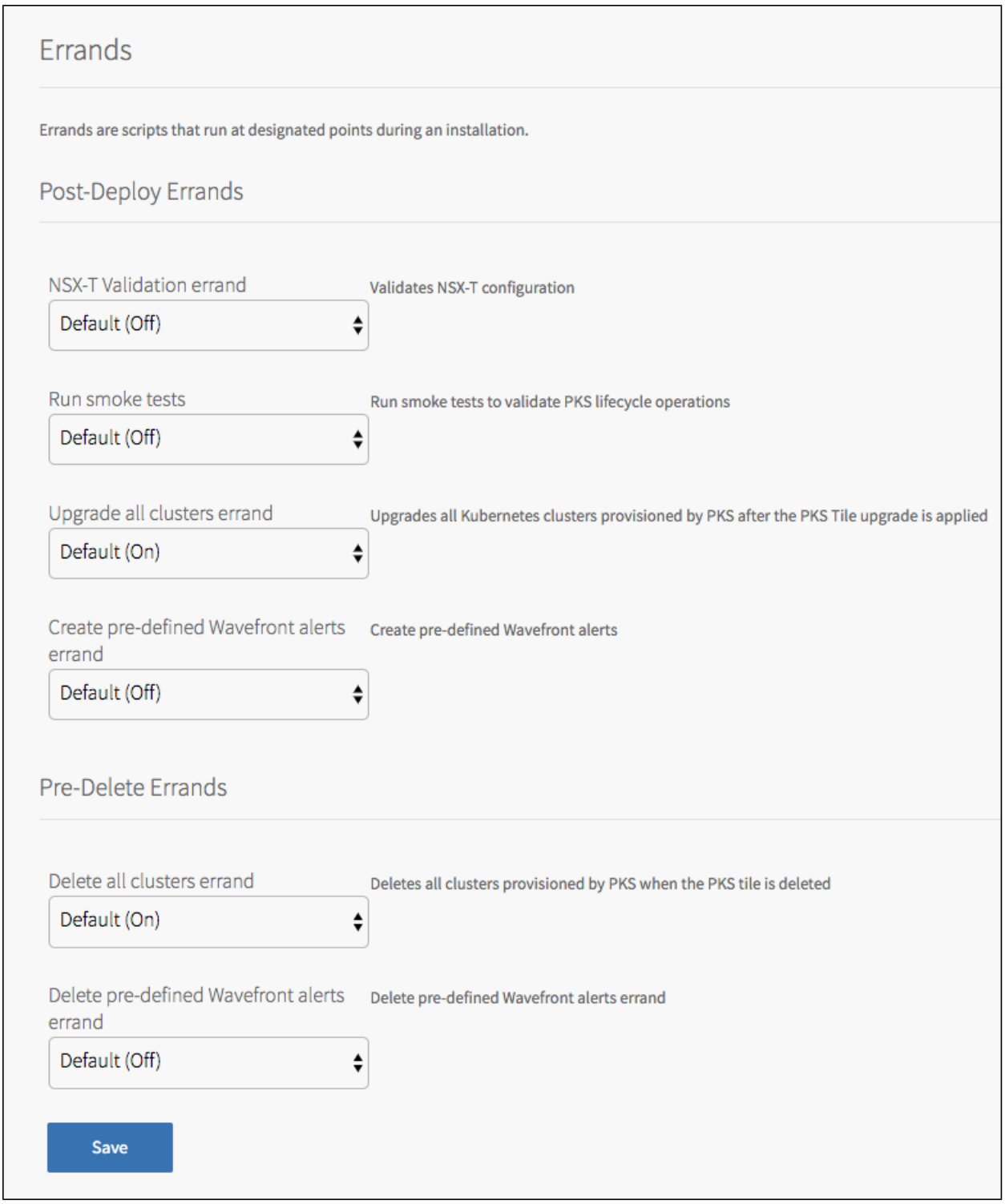
Errands

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

To configure when post-deploy and pre-delete errands for Enterprise 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 CLI to create a Kubernetes cluster and then delete it. If the creation or deletion fails, the errand fails and the installation of the Enterprise PKS tile is aborted.

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



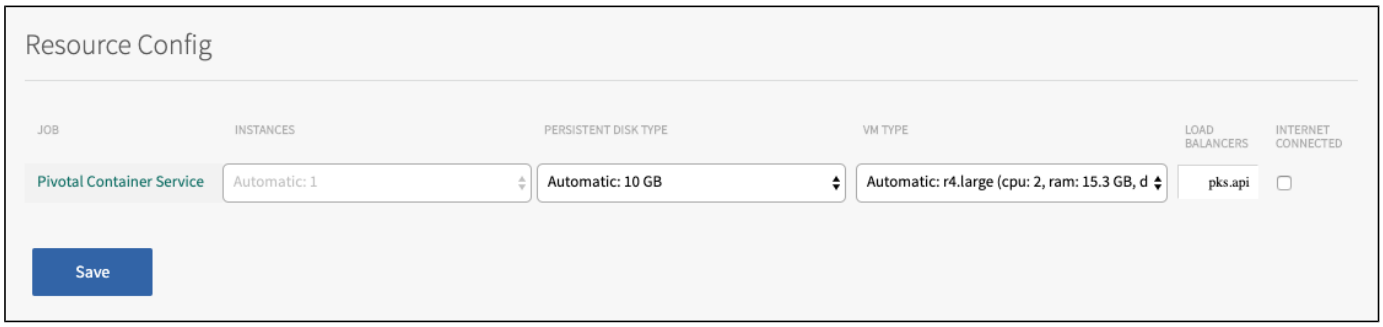
For more information about errands and their configuration state, see Managing Errands in Ops Manager.

warning: If **Upgrade all clusters errand** is enabled, updating the Enterprise PKS tile with a new Linux stemcell triggers the rolling of every Linux VM in each Kubernetes cluster. Similarly, updating the Enterprise PKS tile with a new Windows stemcell triggers the rolling of every Windows VM in your Kubernetes clusters. This automatic rolling ensures that all your VMs are patched. To avoid workload downtime, use the resource configuration recommended in What Happens During Enterprise PKS Upgrades and Maintaining Workload Uptime.

(Optional) Resource Config

To modify the resource usage of Enterprise PKS and specify your PKS API load balancer, follow the steps below:

1. Select **Resource Config**.



2. (Optional) Edit other resources used by the **Pivotal Container Service** job. The **Pivotal Container Service** job requires a VM with the following minimum resources:

CPU	Memory	Disk
2	8 GB	29 GB

> **Note**: The automatic **VM Type** value matches the minimum recommended size for the **Pivotal Container Service** job. If you experience timeouts or slowness when interacting with the PKS API, select a **VM Type** with greater CPU and memory resources.

3. (Optional) In the **Load Balancers** column, enter the name of your PKS API load balancer.

> **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.

4. (Optional) If you do not use a NAT instance, select **Internet Connected** to allow component instances direct access to the internet.

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**.

Next Installation Step

To configure the PKS API load balancer, follow the instructions in Configure PKS API Load Balancer.

Configuring PKS API Load Balancer

In this topic

Page last updated:

This topic describes how to configure an external load balancer for the PKS API.

Overview

You must configure an external load balancer to make the PKS API accessible from outside the network. This external load balancer forwards traffic to the PKS API endpoint on ports 8443 and 9021. You can use any external load balancer for the PKS API.

To set up an external load balancer for the PKS API, do the following after you install the Enterprise PKS tile:

1. Retrieve the PKS API Endpoint
2. Configure an External Load Balancer

Prerequisites

Before configuring an external load balancer for the PKS API, you must have the following:

- The PKS API certificate that you provided in the **Enterprise PKS** tile > **PKS API** > **Certificate to secure the PKS API**.
- The PKS API hostname that you entered in the **Enterprise PKS** tile > **PKS API** > **API Hostname (FQDN)**.

Step 1: Retrieve the PKS API Endpoint

You need to retrieve the PKS API endpoint to allow your organization to use the API to create, update, and delete Kubernetes clusters.

To retrieve the PKS API endpoint, do the following:

1. Navigate to the Ops Manager **Installation Dashboard**.
2. Click the **Enterprise PKS** 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 2: Configure an External Load Balancer

To set up an external load balancer for the PKS API, configure the external load balancer to resolve to the domain name you entered in the **Enterprise PKS** tile > **PKS API** > **API Hostname (FQDN)** using the following information:

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

Next Installation Step

To set up Enterprise PKS admin users who can create and manage Kubernetes clusters, follow the instructions in Setting Up Enterprise PKS Admin Users on vSphere.

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

Setting Up Enterprise PKS Admin Users on vSphere

In this topic

Page last updated:

This topic describes how to create admin users in VMware Enterprise PKS with User Account and Authentication (UAA). Creating at least one admin user is a necessary step during the initial set up of Enterprise PKS.

Overview

UAA is the identity management service for Enterprise PKS. Enterprise PKS includes a UAA server, which is hosted on the PKS control plane VM.

To interact with the UAA server, you can use the UAA Command Line Interface (UAAC). You can either run UAAC commands from the Ops Manager VM or install UAAC on your local workstation.

Prerequisites

Before setting up admin users for Enterprise PKS, you must have one of the following:

- SSH access to the Ops Manager VM
- A machine that can connect to your PKS control plane VM

Step 1: Connect to the PKS Control Plane VM

You can connect to the PKS control plane VM from the Ops Manager VM or from a different machine such as your local workstation.

Option 1: Connect through the Ops Manager VM

You can connect to PKS control plane VM by logging in to the Ops Manager VM through SSH. To SSH into the Ops Manager VM on vSphere, do the following:

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 your credentials in the Prepare vSphere section of *Deploying Ops Manager on vSphere* for your Ops Manager version.

> **Note**: If you lose your password, 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.

> **warning:** If you deployed Ops Manager v2.6 on vSphere, you can only SSH into the Ops Manager VM with a private SSH key. If you do not have a private SSH key, you must add a public key to your `.ova` or `.ovf` file and then use the private key to SSH onto the Ops Manager VM. If you do not add a key, Ops Manager shuts down automatically because it cannot find a key and may enter a reboot loop.

To SSH into the Ops Manager VM on vSphere, do one of the following:

- **If you set a password when you installed Ops Manager:**

  1. SSH into the Ops Manager VM by running the following command:

     ```
     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 Adminsection to manage users with UAAC.

- **If you set a SSH key when you installed Ops Manager:**

  1. Change the permissions for your private SSH key by running the following command:

     ```
     chmod 600 PRIVATE-KEY
     ```

     Where `PRIVATE-KEY` is the name of your private SSH key.

  2. SSH into the Ops Manager VM by running the following command:

     ```
     ssh -i PRIVATE-KEY ubuntu@OPS-MANAGER-FQDN
     ```

     Where `OPS-MANAGER-FQDN` is the fully qualified domain name (FQDN) of Ops Manager.
     For example:

     ```
     $ ssh -i id_rsa ubuntu@my-opsmanager-fqdn.example.com
     ```

  3. Proceed to the Log In as a UAA Adminsection to create admin users with UAAC.

Option 2: Connect through a Non-Ops Manager Machine

To connect to the PKS control plane VM and run UAA commands, do the following:

1. Install UAAC on your machine. For example:

   ```
   gem install cf-uaac
   ```

2. Download a copy of your Ops Manager root CA certificate to the machine. To download the certificate, do the following:

   a. In a web browser, navigate to the FQDN of Ops Manager and log in.

b. In Ops Manager, navigate to **Settings** in the drop-down menu under your username.
c. Click **Advanced Options**.
d. On the **Advanced Options** configuration page, click **Download Root CA Cert**
e. Move the certificate to a secure location on your machine and record the path.

3. Proceed to the Log In as a UAA Admin section to create admin users with UAAC.

Step 2: Log In as a UAA Admin

Before creating PKS users, you must log in to the UAA server as a UAA admin. To log in to the UAA server, do the following:

1. Retrieve the UAA management admin client secret:

   a. In a web browser, navigate to the Ops Manager **Installation Dashboard** and click the **Enterprise PKS** tile.
   b. Click the **Credentials** tab.
   c. Click **Link to Credential** next to **Pks Uaa Management Admin Client** and copy the value of `secret`.

2. Target your UAA server by running the following command:

   ```
   uaac target https://PKS-API:8443 --ca-cert CERTIFICATE-PATH
   ```

   Where:

   - `PKS-API` is the domain name of your PKS API server. You entered this domain name in the **Enterprise PKS** tile > **PKS API** > **API Hostname (FQDN)**.
   - `CERTIFICATE-PATH` is the path to your Ops Manager root CA certificate. Provide this certificate to validate the PKS API certificate with SSL.
     - If you are logged in to the Ops Manager VM, specify `/var/tempest/workspaces/default/root_ca_certificate` as the path. This is the default location of the root certificate on the Ops Manager VM.
     - If you downloaded the Ops Manager root CA certificate to your machine, specify the path where you stored the 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 ignore it because it has no impact.

3. Authenticate with UAA by running the following command:

   ```
   uaac token client get admin -s ADMIN-CLIENT-SECRET
   ```

   Where `ADMIN-CLIENT-SECRET` is your UAA management admin client secret that you retrieved in a previous step. The client username is `admin`.

Step 3: Assign Enterprise PKS Cluster Scopes

The `pks.clusters.manage` and `pks.clusters.admin` UAA scopes grant users the ability to create and manage Kubernetes clusters in Enterprise

PKS. For information about UAA scopes in Enterprise PKS, see UAA Scopes for Enterprise PKS Users.

To create Enterprise PKS users with the `pks.clusters.manage` or `pks.clusters.admin` UAA scope, perform one or more of the following procedures based on the needs of your deployment:

- To assign PKS cluster scopes to an individual user, see Grant Enterprise PKS Access to an Individual User. Follow this procedure if you selected **Internal UAA** when you configured **UAA** in the Enterprise PKS tile. For more information, see Installing Enterprise PKS on vSphere.
- To assign PKS cluster scopes to an LDAP group, see Grant Enterprise PKS Access to an External LDAP Group. Follow this procedure if you selected **LDAP Server** when you configured **UAA** in the Enterprise PKS tile. For more information, see Installing Enterprise PKS PKS on vSphere.
- To assign PKS cluster scopes to a SAML group, see Grant Enterprise PKS Access to an External SAML Group. Follow this procedure if you selected **SAML Identity Provider** when you configured **UAA** in the Enterprise PKS tile. For more information, see Installing Enterprise PKS PKS on vSphere.
- To assign PKS cluster scopes to a client, see Grant Enterprise PKS Access to a Client.

Next Step

After you create admin users in Enterprise PKS, the admin users can create and manage Kubernetes clusters in Enterprise PKS. For more information, see Managing Kubernetes Clusters and Workloads.

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

Installing Enterprise PKS on vSphere with NSX-T Data Center

In this topic

Page last updated:

This topic lists the procedures to follow to install VMware Enterprise PKS on vSphere with NSX-T Data Center.

> **note:** This topic explains how to install Enterprise PKS on vSphere with NSX-T on Pivotal Platform. As an alternative, you can use the VMware Enterprise PKS Management Console to install Enterprise PKS on vSphere with NSX-T. Refer to Enterprise PKS Management Console (vSphere Only) for details.

Step 1: Prepare to Install Enterprise PKS on vSphere with NSX-T

In preparation for installing Enterprise PKS on vSphere with NSX-T, review all of the topics in the subsection Preparing to Install Enterprise PKS on vSphere with NSX-T Data Center.

Step 2: Install and Configure NSX-T Data Center for Enterprise PKS

NSX-T Data Center must be installed and configured before you install Enterprise PKS.

For instructions, see Installing and Configuring NSX-T for Enterprise PKS.

Step 3: Create the Management Plane for Enterprise PKS

Prepare the vSphere and NSX-T infrastructure for the Enterprise PKS Management Plane where the PKS API, Ops Manager, BOSH Director, and Harbor Registry VMs are deployed. This includes creating a vSphere resource pool for Enterprise PKS management components, an NSX Tier-1 (T1) Logical Switch, an NSX Tier-1 Logical Router and Port, and NAT rules (if you are using NAT mode).

For instructions, see Creating the Enterprise PKS Management Plane.

Step 4: Create the Compute Plane for Enterprise PKS

Create vSphere Resource Pools for the Availability Zones where you will deploy Kubernetes clusters. These resource pools map to the AZs you will create when you configure BOSH Director and reference when you install the Enterprise PKS tile.

Create IP blocks for the node networks and the pod networks. Typically the initial subnets for both nodes and pods will have a size of 256 (/16).

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 Enterprise PKS. The network also provides IP addresses for Kubernetes API access and Kubernetes exposed services.

For instructions, see Creating the Enterprise PKS Compute Plane.

Step 5: Deploy Ops Manager for Enterprise PKS with NSX-T

Deploy a supported version of Ops Manager on the NSX-T Management Plane network.

For instructions, see Deploying Ops Manager with NSX-T for Enterprise PKS.

Step 6: Generate the NSX-T Management Cluster Root CA Certificate and Key

Generate the CA Cert for the NSX Manager and import the certificate to NSX Manager.

For instructions, see Generating and Registering the NSX-T Management Root CA Certificate for Enterprise PKS.

Step 7: 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 Configuring BOSH Director with NSX-T for Enterprise PKS.

Step 8: Generate and Register the NSX-T Management Cluster Super User Principal Identity Certificate and Key

Generate the NSX Manager Super User Principal Identity Certificate and register it with the NSX Manager using the NSX API.

For instructions, see Generating and Registering the NSX Manager Superuser Principal Identity Certificate and Key.

Step 9: Install Enterprise PKS on vSphere with NSX-T

At this point your NSX-T environment is prepared for Enterprise PKS installation using the Enterprise PKS tile in Ops Manager.

For instructions, see Installing Enterprise PKS on vSphere with NSX-T.

Step 10: Install Harbor Harbor Registry for Enterprise PKS

The VMware Harbor Registry is recommended for Enterprise PKS. Install Harbor in the NSX Management Plane with other Enterprise PKS components (PKS API, Ops Manager, and BOSH).

If you are using the NAT deployment topology, 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.

For instructions, see Installing VMware Harbor Registry.

Step 11: Install the PKS and Kubectl CLIs

See Installing the PKS CLI and Installing the Kubernetes CLI.

Step 12: Create Admin Users for Enterprise PKS

See Setting Up Enterprise PKS Admin Users on vSphere

Step 13: Verify the Installation of Enterprise PKS

Create a Kubernetes cluster using the PKS CLI. For instructions, see Create a Kubernetes Cluster.

Deploy a simple workload to the Kubernetes cluster. For instructions, see Deploy Workloads on vSphere with NSX-T.

Step 14: Perform Desired Post-Installation Configurations

After you have installed Enterprise PKS on vSphere with NSX-T, refer to the following subsection for topics describing additional NSX-T configuration options: Advanced Configurations for Enterprise PKS on vSphere with NSX-T Data Center

Step 15: Create Network Profiles to Customize Cluster Deployments

Network profiles let you provide customized deployment templates for Kubernetes clusters. See Network Profiles (NSX-T Only) for details.

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

Preparing to Install Enterprise PKS on vSphere with NSX-T Data Center

Page last updated:

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

Preparing to Install Enterprise PKS on vSphere with NSX-T

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

- Version Requirements for Enterprise PKS
- Hardware Requirements for Enterprise PKS
- Firewall Ports and Protocols Requirements
- NSX-T Deployment Topologies for Enterprise PKS
- NSX-T Cluster Objects Created for Enterprise PKS
- Network Planning Enterprise PKS

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

vSphere with NSX-T Version Requirements

In this topic

Page last updated:

This topic describes the version requirements for installing VMware Enterprise PKS on vSphere with NSX-T integration.

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

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

Enterprise 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

For Enterprise PKS on vSphere version requirements, refer to the VMware Product Interoperability Matrices.

NSX-T Integration Component Version Requirements

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

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

Hardware Requirements for Enterprise PKS on vSphere with NSX-T

In this topic

Page last updated:

This topic provides hardware requirements for production deployments of VMware Enterprise PKS on vSphere with NSX-T.

vSphere Clusters for Enterprise PKS

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

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

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

Enterprise PKS Management Cluster

The Enterprise PKS Management Cluster on vSphere comprises the following components:

- vCenter Server
- NSX-T Manager v2.4 or later (quantity 3)

For more information, see Installing and Configuring NSX-T for Enterprise PKS.

Enterprise PKS Edge Cluster

The Enterprise 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. Enterprise PKS supports running Edge Node pairs in active/standby mode only.

For more information, see Installing and Configuring NSX-T for Enterprise PKS.

Enterprise PKS Compute Cluster

The Enterprise PKS Compute Cluster on vSphere comprises the following components:

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

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

Enterprise PKS Management Plane Placement Considerations

The Enterprise 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 Enterprise PKS Management Cluster on the standard vSphere network or in the Enterprise PKS Compute Cluster on the NSX-T-defined virtual network. For more information, see NSX-T Deployment Topologies for Enterprise PKS.

Configuration Requirements for vSphere Clusters for Enterprise PKS

For each vSphere cluster defined for Enterprise 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

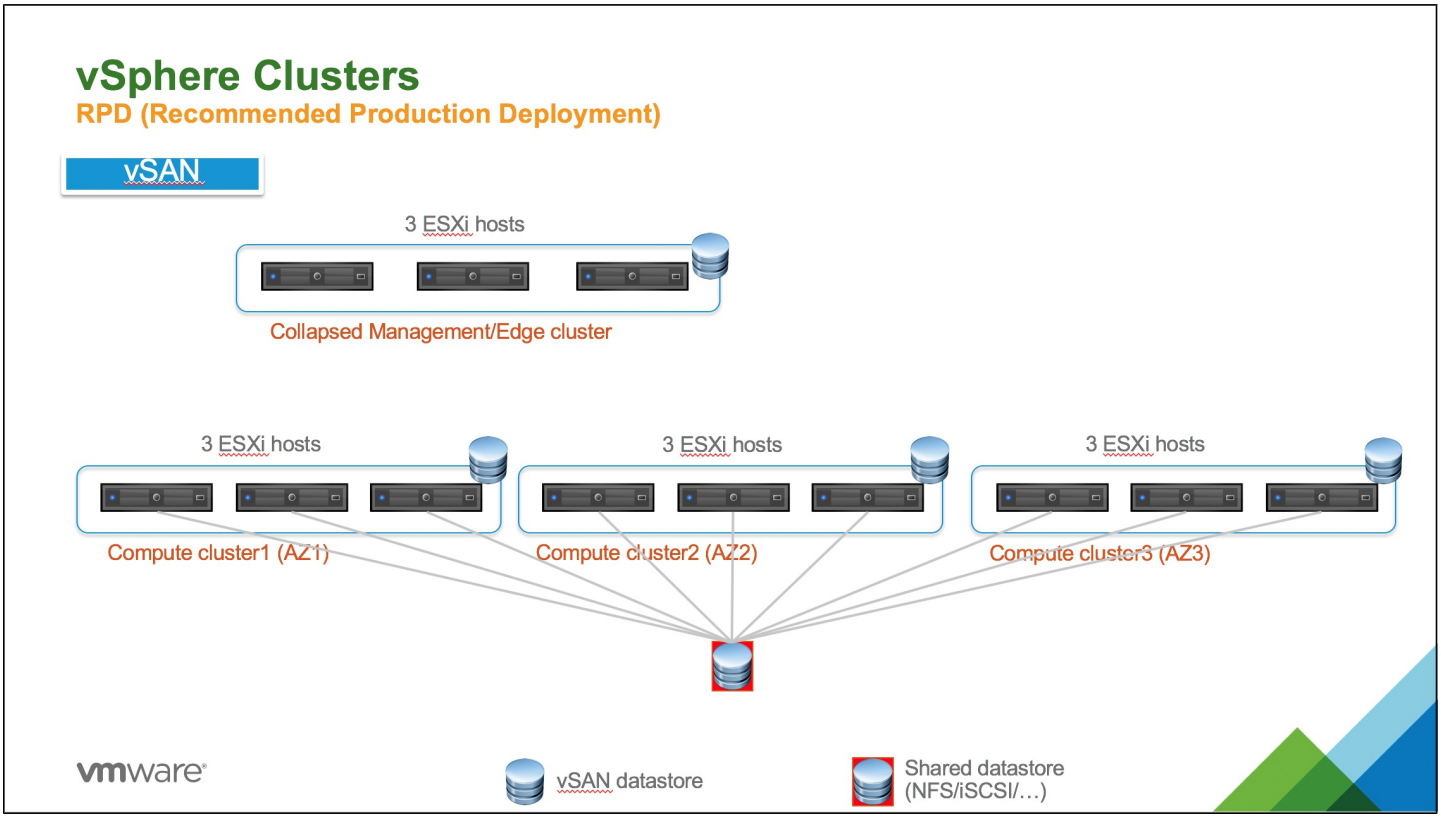
RPD for Enterprise PKS on vSphere with NSX-T

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

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

RPD for Enterprise PKS with vSAN

The RPD for Enterprise 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 Manager. The NSX-T Control Plane has three NSX-T Managers 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 an Enterprise 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 Enterprise PKS, see PersistentVolume Storage Options on vSphere.

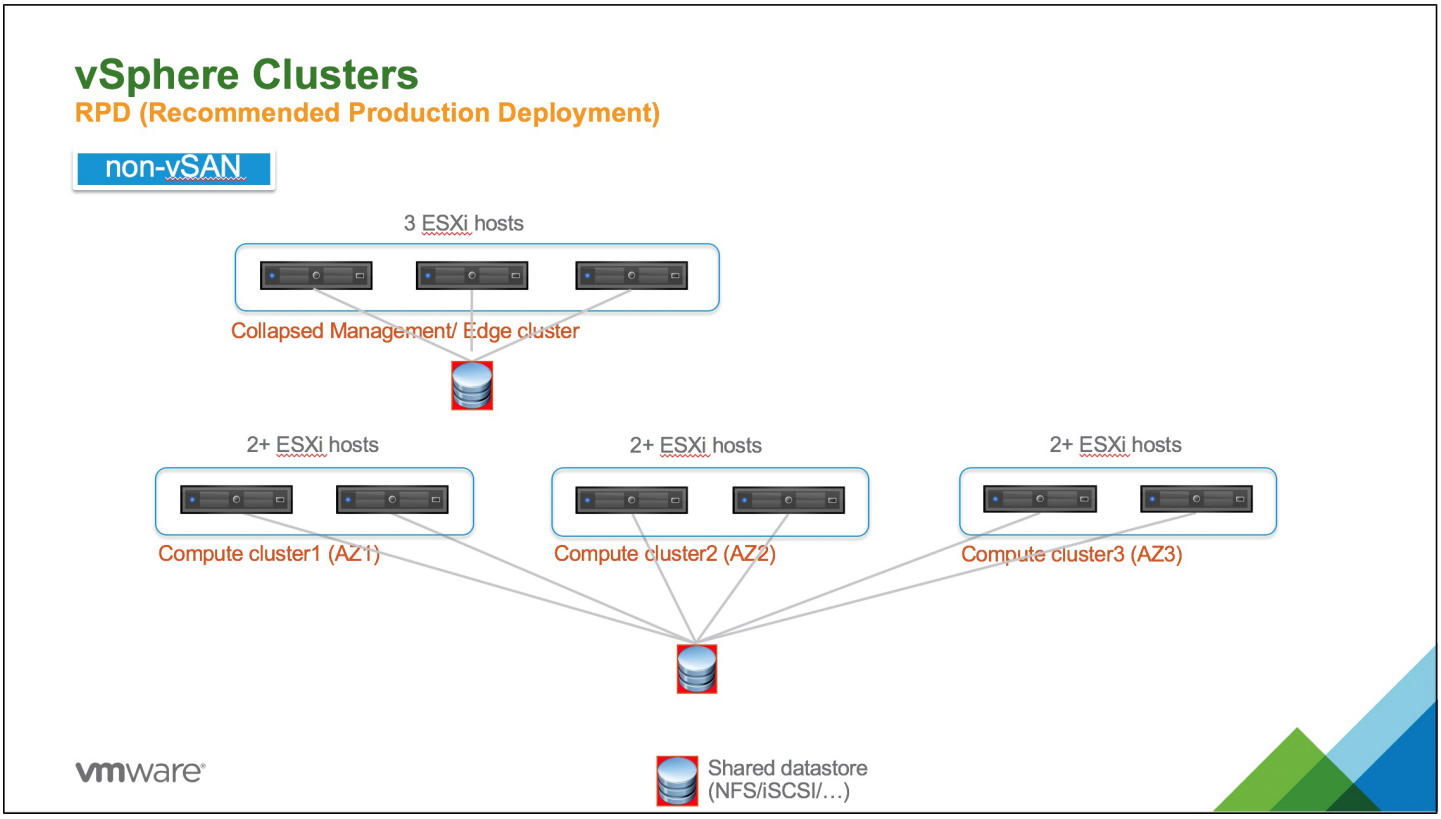
Future Growth

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 Enterprise PKS without vSAN

The RPD for Enterprise 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 Enterprise 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 Manager. The NSX-T Control Plane has three NSX-T Managers 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 Enterprise 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 Enterprise 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:

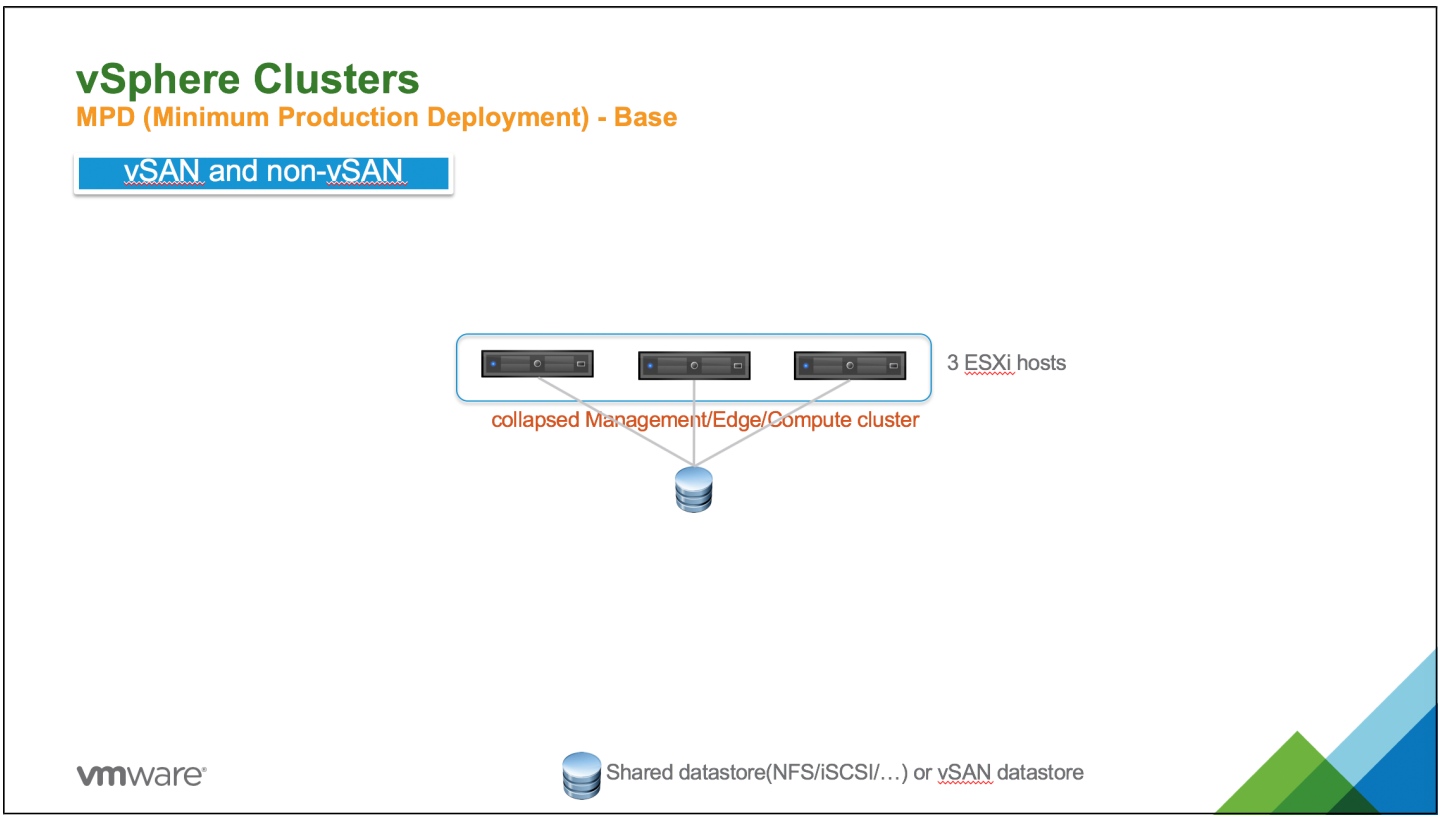
- The collapsed Management/Edge Cluster can be expanded to include up to 64 ESXi hosts.
- Each Compute Cluster can be expanded to include up to 64 ESXi hosts.

MPD for Enterprise PKS on vSphere with NSX-T

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

> **Note:** The MPD topology for Enterprise 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 Enterprise PKS.

MPD Topology

The MPD topology for Enterprise 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 Manager. The NSX-T Control Plane has three NSX-T Managers 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 Enterprise 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 Enterprise PKS deployment requires a minimum of three nodes, Enterprise 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 Enterprise 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 Enterprise PKS, keep in mind the following:

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

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

VM Inventory and Sizes

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

Control Plane VMs and Sizes

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

VM	CPU	Memory (GB)	Disk Space (GB)
vCenter Appliance	4	16	290
NSX-T Manager 1	6	24	200
NSX-T Manager 2	6	24	200
NSX-T Manager 3	6	24	200
Ops Manager	1	8	160
BOSH Director	2	8	103
PKS Control Plane	2	8	29
Harbor Registry	2	8	167
TOTAL	**29**	**120**	**1.35 TB**

Note: The NSX Manager resource requirements are based on the medium size VM, which is the minimum recommended form factor for NSX-T v2.4 and later. For more information, see NSX Manager VM System Requirements.

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 Pivotal Container Service VM as follows:

Number of Pods	Storage (Persistent Disk) Requirements
1,000 pods	20 GB

Number of Pods	Storage (Persistent Disk) Requirements
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	32	120
NSX-T Edge Node 2	8	32	120
TOTAL	**16**	**64**	**.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 Enterprise 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 Enterprise 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

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	4x 10GbE	5.9 TB

Adding Hardware Capacity

To add hardware capacity to your Enterprise 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.
2. 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 Enterprise PKS Compute Cluster.

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

Firewall Ports and Protocols Requirements for vSphere with NSX-T

Page last updated:

In this topic

This topic describes the firewall ports and protocols requirements for using VMware Enterprise 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 Enterprise 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 Enterprise 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 Enterprise 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.

Enterprise PKS Ports and Protocols

The following tables list ports and protocols required for network communications between Enterprise PKS v1.5.0 and later, and vSphere 6.7 and NSX-T 2.4.0.1 and later.

Enterprise PKS Users Ports and Protocols

The following table lists ports and protocols used for network communication between Enterprise 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
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 Enterprise PKS deployments on vSphere with NSX-T. Only `type:LoadBalancer` and Services associated with Ingress rules are supported on vSphere with NSX-T.

Enterprise PKS Core Ports and Protocols

The following table lists ports and protocols used for network communication between core Enterprise 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 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

Source Component	Destination Component	Destination Protocol	Destination Port	Service
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 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

Source Component	Destination Component	Destination Protocol	Destination Port	Service
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	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

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	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

Source Component	Destination Component	Destination Protocol	Destination Port	Service
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
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

Source Component	Destination Component	Destination Protocol	Destination Port	Service
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
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

Source Component	Destination Component	Destination Protocol	Destination Port	Service
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 Enterprise PKS

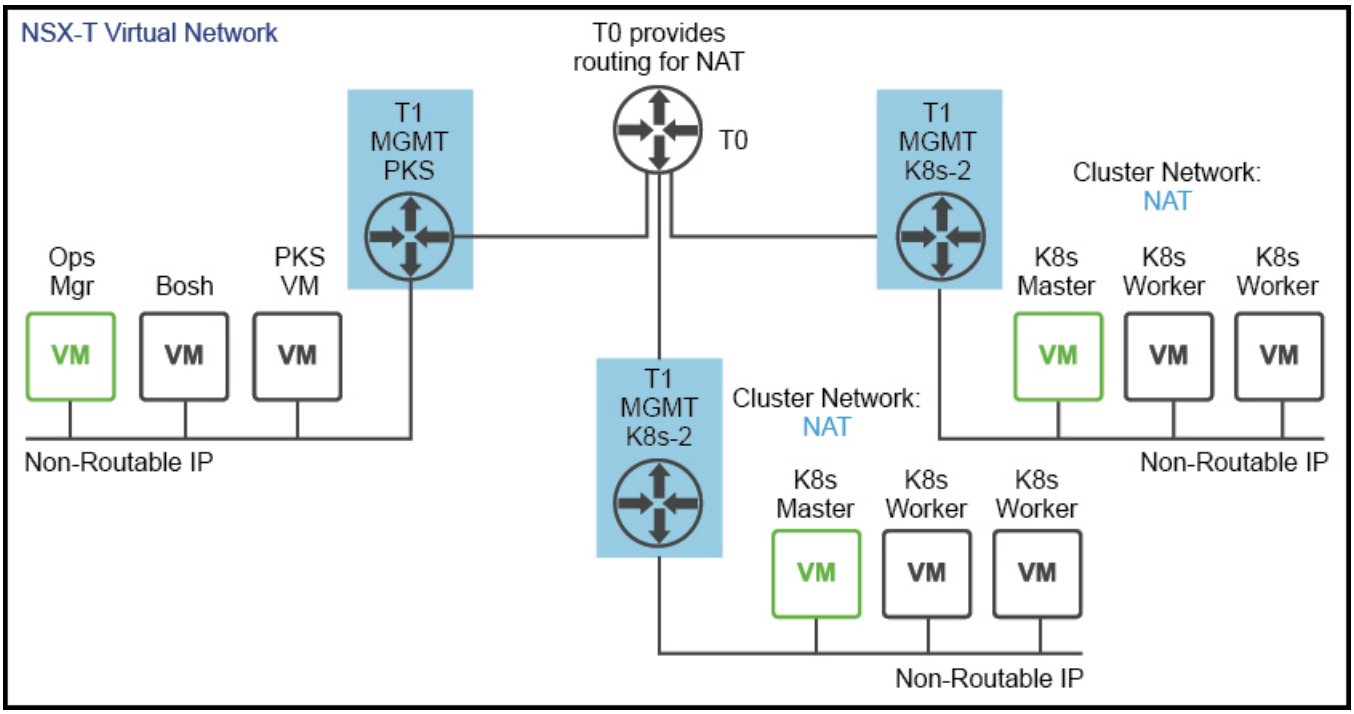
In this topic

Page last updated:

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

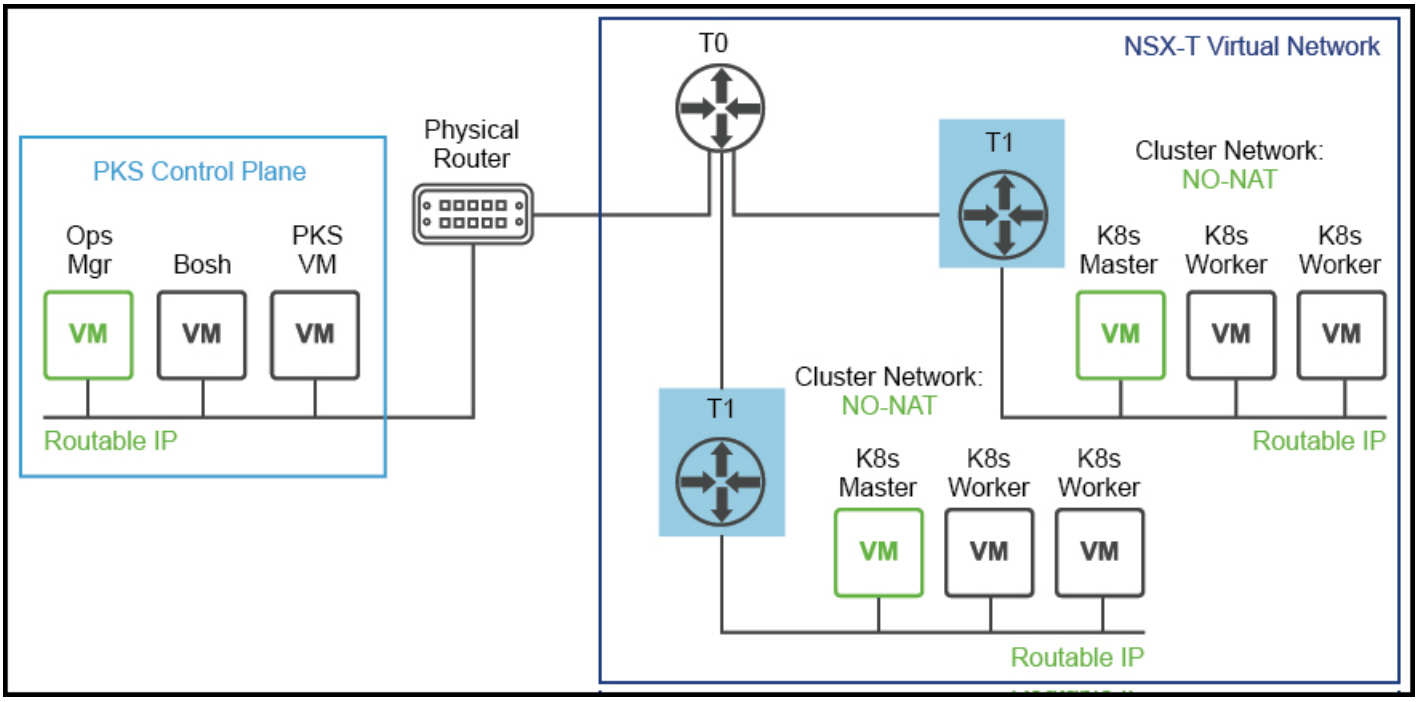
No-NAT Topology

A No-NAT topology uses a routable IP subnet for the PKS Management network and for Kubernetes nodes.

There are two flavors of No-NAT topology: No-NAT with Virtual Switch or No-NAT with Logical Switch.

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

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



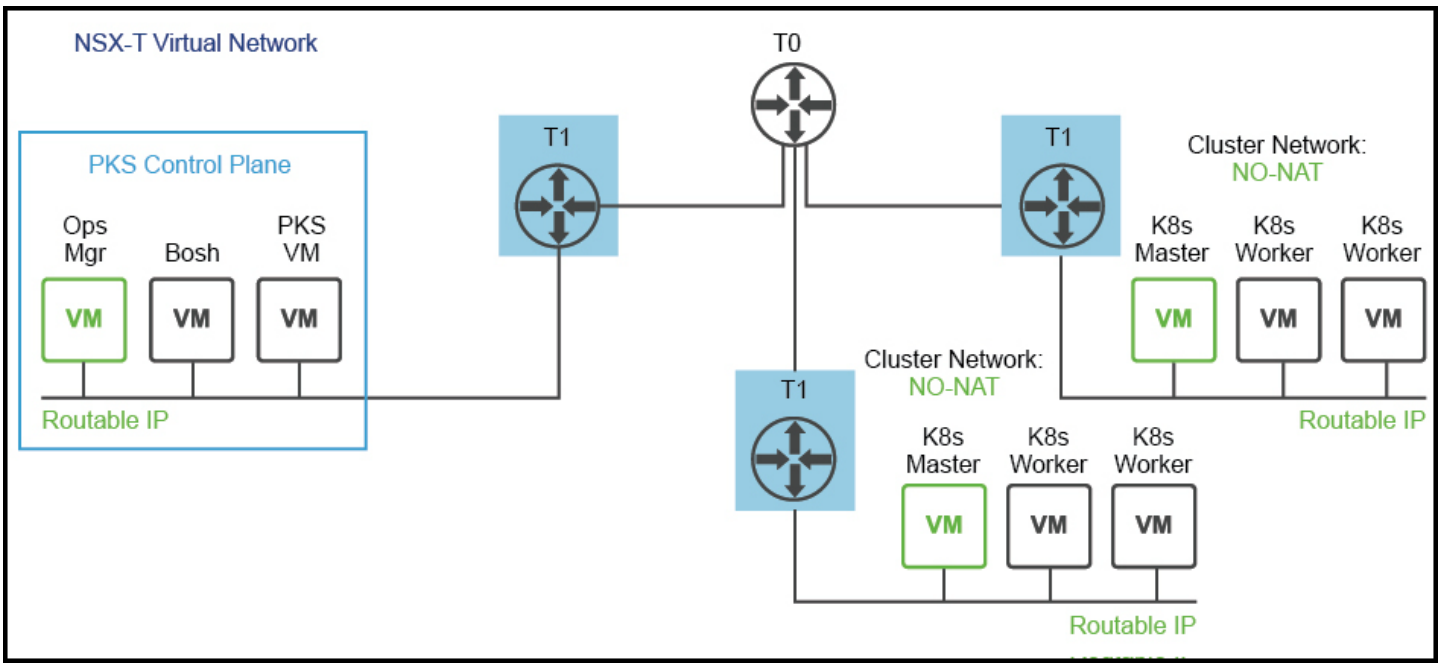
View a larger version of this image.

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.

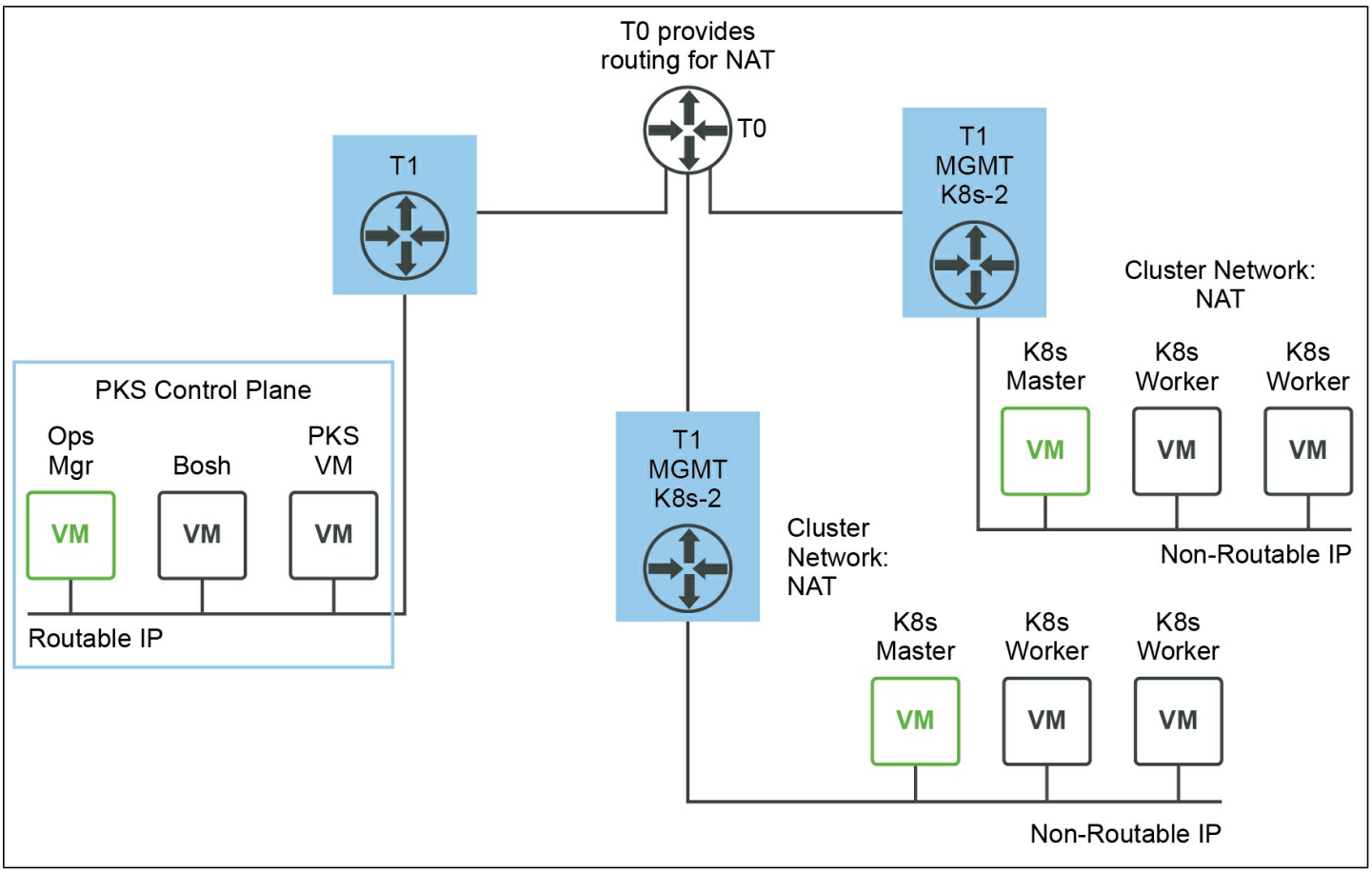
This topology has the following characteristics:

- 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.

Hybrid Topology

With a hybrid topology, the PKS Management Network is on a routable subnet, while the Kubernetes Nodes Network uses a non-routable subnet (NAT mode is checked in the PKS tile).

The following figure shows a hybrid topology 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 located on a logical switch that has undergone Network Address Translation on a T0. This requires DNAT rules to allow access to Kubernetes APIs.

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

vSphere with NSX-T Cluster Objects

In this topic

Page last updated:

This topic lists and describes the vSphere VMs and NSX-T objects that VMware Enterprise PKS creates when you create a Kubernetes cluster. When you delete a Kubernetes cluster, Enterprise PKS removes these objects.

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

vSphere Virtual Machines

When a new Kubernetes cluster is created, Enterprise 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 Enterprise PKS on vSphere with NSX-T for more information.

NSX-T Logical Switches

When a new Kubernetes cluster is created, Enterprise 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, Enterprise 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, Enterprise 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 Enterprise 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, Enterprise PKS extracts and allocates the following NSX-T subnets from the IP blocks created in preparation for installing Enterprise 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, Enterprise 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, Enterprise PKS defines the following NSX-T distributed firewall rules:

Object Amount	Object Description
1	DFW rule for `kube-dns`, applied to CoreDNS pod logical port: `Source=Kubernetes worker node (hosting the DNS Pod); Destination=Any; Port: TCP/8080; Action: allow`
1	DFW rule for Validator in `pks-system` namespace, applied to Validator pod logical port: `Source=Kubernetes worker node (hosting the DNS Pod) IP Address; Destination=Any; Port: TCP/9000; Action: allow`
1	DFW rule for `kubernetes-dashboard`: `Source=Kubernetes worker node (hosting the Dashboard Pod); Destination=Dashboard Pod IP; Port: TCP/8443; Action: allow`

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

Network Planning for Installing Enterprise PKS with NSX-T

In this topic

Page last updated:

Before you install VMware Enterprise PKS on vSphere with NSX-T integration, you must plan the environment as described in this topic.

Prerequisites

Familiarize yourself with the following VMware documentation:

- vSphere, vCenter, vSAN, and ESXi documentation
- NSX-T Data Center documentation
- NSX Container Plugin (NCP) documentation

Familiarize yourself with the following related documentation:

- Ops Manager documentation
- BOSH documentation
- Kubernetes documentation
- Docker documentation

Review the following Enterprise PKS documentation:

- vSphere with NSX-T Version Requirements
- Hardware Requirements for Enterprise PKS on vSphere with NSX-T
- Firewall Ports and Protocols Requirements for vSphere with NSX-T
- Network Objects Created by NSX-T for Enterprise PKS

Understand Component Interactions

Enterprise PKS on vSphere with NSX-T requires the following component interactions:

- vCenter, NSX-T Manager Nodes, NSX-T Edge Nodes, 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-T Management Cluster.
- The BOSH Director VM must be able to communicate with all nodes in all Kubernetes clusters.
- Each Enterprise PKS-provisioned Kubernetes cluster deploys the NSX-T Node Agent and the Kube Proxy that run as BOSH-managed processes on each worker node.
- NCP runs as a BOSH-managed process on the Kubernetes master node. In a multi-master deployment, the NCP process runs on all master nodes, but is active only on one master node. If the NCP process on an active master is unresponsive, BOSH activates another NCP process.

Plan Deployment Topology

Review the Deployment Topologies for Enterprise PKS on vSphere with NSX-T. The most common deployment topology is the NAT topology. Decide which deployment topology you will implement, and plan accordingly.

Plan Network CIDRs

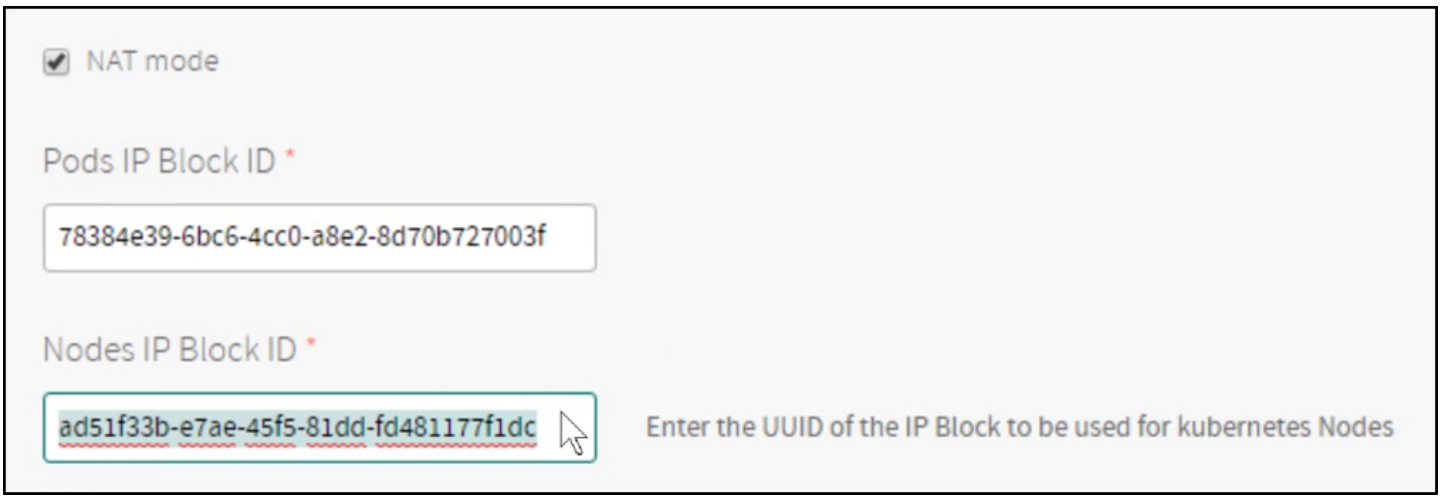
Before you install Enterprise PKS on vSphere with NSX-T, you should plan for the CIDRs and IP blocks that you are using in your deployment.

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 Enterprise 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 Enterprise PKS management components.
- **PKS LB CIDR**: This network provides your load balancing address space for each Kubernetes cluster created by Enterprise 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 Enterprise PKS, or when the services are deployed. You enter this network in the **Floating IP Pool ID** field in the **Networking** pane of the Enterprise PKS tile.

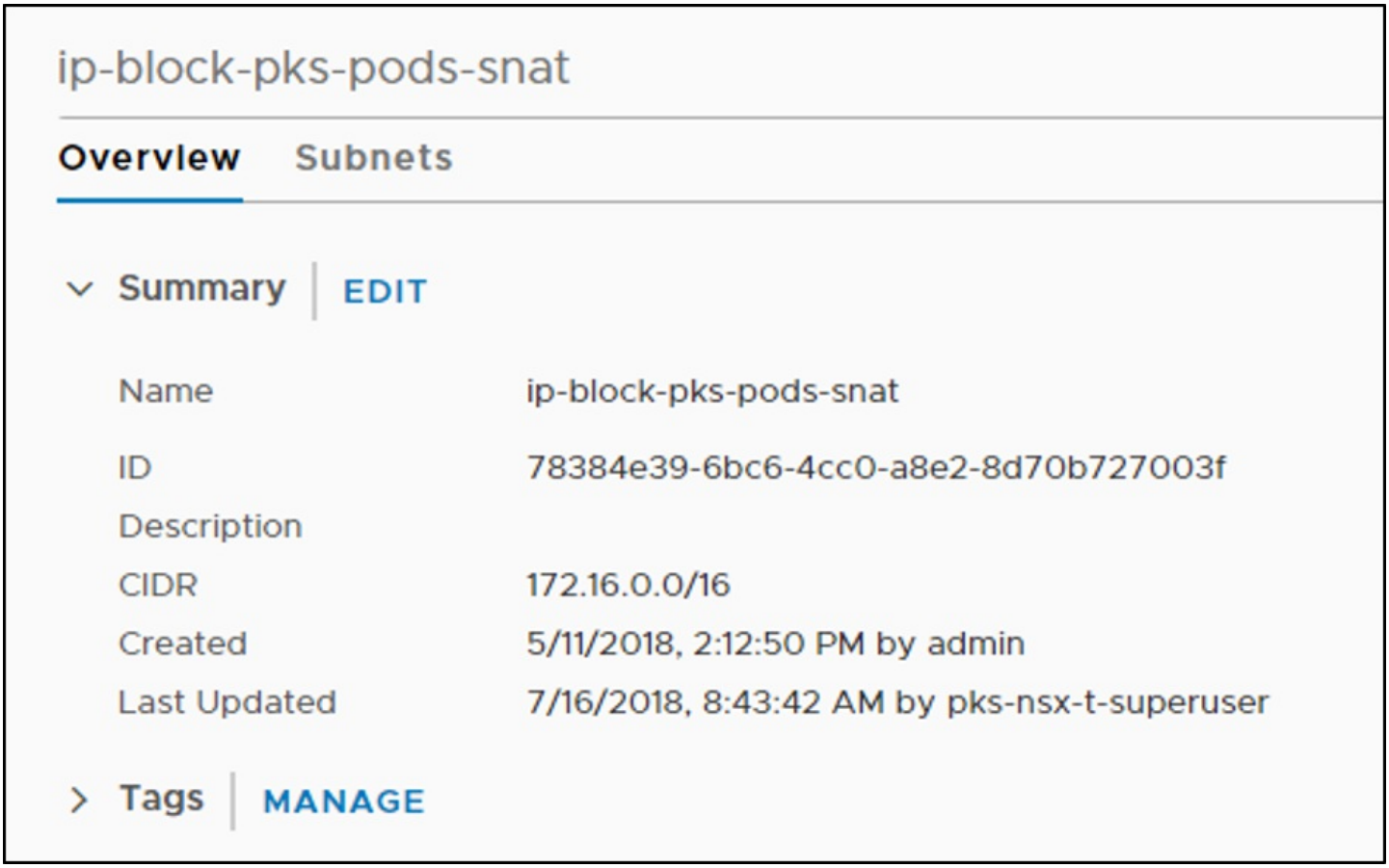
Plan IP Blocks

When you install Enterprise 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 Enterprise 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 Enterprise 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 Enterprise 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 Enterprise PKS.

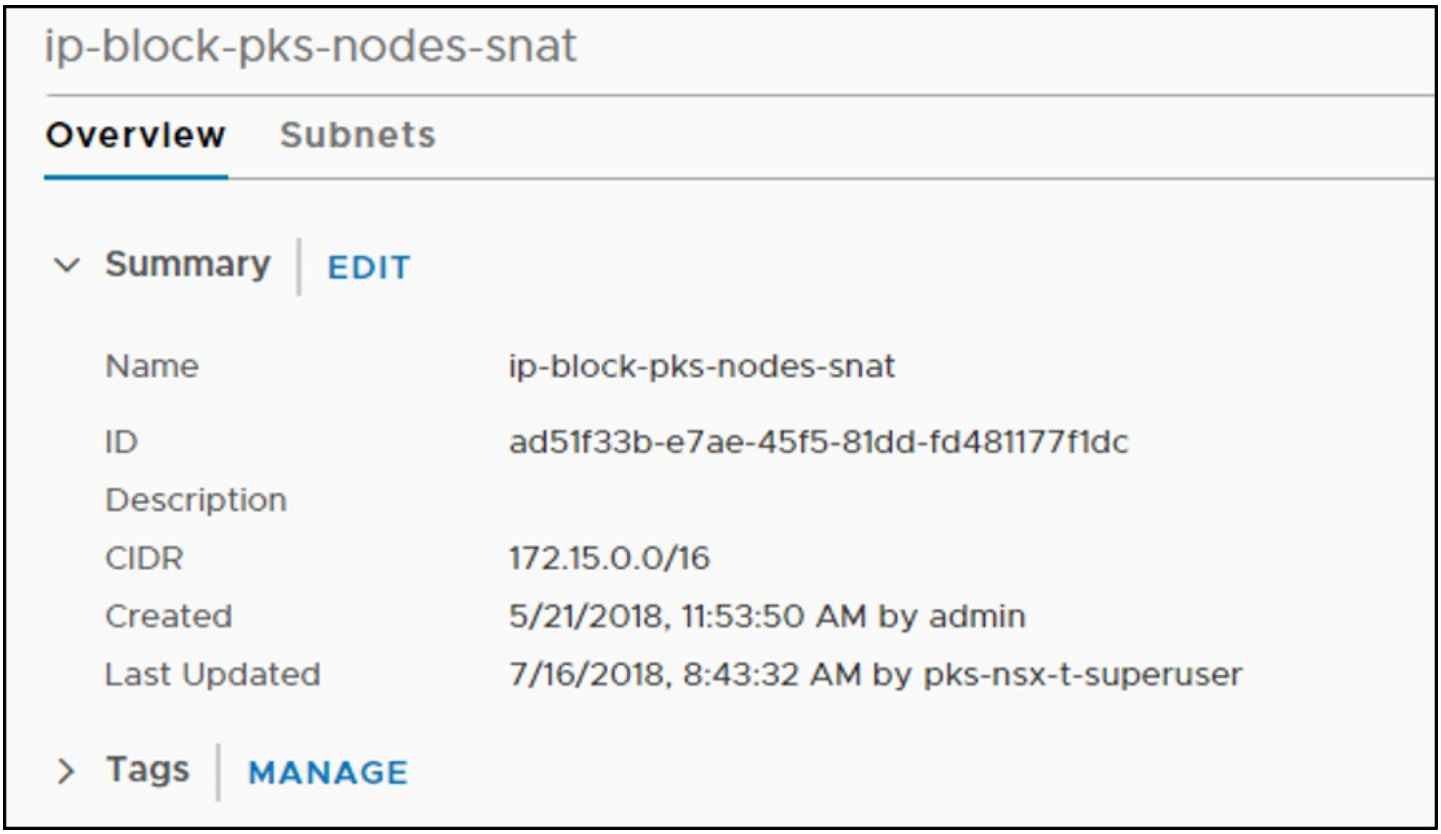


Note: By default, **Pods IP Block** is a block of non-routable, private IP addresses. After you deploy Enterprise PKS, you can define a network profile that specifies a routable IP block for your pods. The routable IP block overrides the default non-routable **Pods IP Block** when a Kubernetes cluster 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 Enterprise PKS owns a /24 subnet. To deploy multiple Kubernetes clusters, set the **Nodes IP Block ID** in the **Networking** pane of the Enterprise PKS tile to larger than /24. The recommended size is /16. For more information, see Creating NSX-T Objects for Enterprise 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 Enterprise 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 Enterprise 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 Enterprise PKS is deployed with Harbor v1.9.3 or v1.9.4, also do not use IP addresses in the following CIDR ranges. Harbor v1.9.3 and v1.9.4 use these for internal Docker bridges:

- 172.23.0.1/16
- 172.24.0.1/16
- 172.25.0.1/16
- 172.26.0.1/16
- 172.27.0.1/16

Starting with Harbor v1.10.0, the default container network pool is set to `172.20.0.0/16`. If Enterprise PKS is deployed with Harbor v1.10.0 or later, the reserved IP range is `172.20.0.0/16`.

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

Gather Other Required IP Addresses

To install Enterprise PKS on vSphere with NSX-T, you will need to know the following:

- Subnet name where you will install Enterprise PKS
- VLAN ID for the subnet
- CIDR for the subnet
- Netmask for the subnet
- Gateway for the subnet
- DNS server for the subnet
- NTP server for the subnet
- IP address and CIDR you plan to use for the NSX-T Tier-0 Router

Installing and Configuring NSX-T Data Center v2.5 for Enterprise PKS

In this topic

Page last updated:

This topic provides the workflow for installing and configuring NSX-T Data Center v2.5 for use with VMware Enterprise PKS on vSphere.

Prerequisites for Installing NSX-T Data Center v2.5 for Enterprise PKS

To perform a new installation of NSX-T Data Center for Enterprise PKS, complete the following steps in the order presented.

1. Read the Release Notes for the target PKS version you are installing and verify NSX-T 2.5 support.
2. Read the topics in the Preparing to Install Enterprise PKS on vSphere with NSX-T Data Center section of the documentation.

Workflow for Installing and Configuring NSX-T Data Center v2.5 for Enterprise PKS

1. Deploy the NSX-T Manager Appliance Using an OVA Template.
2. Add vCenter as the Compute Manager.
3. Add Additional NSX-T Manager Nodes using the NSX Manager web interface.
4. Configure a VIP Address or Provision a Load Balancer for the NSX-T Management Cluster.

   > **Note:** A VIP provides high availability (HA) for the NSX-T Management Cluster. If you need to scale the NSX Manager API, configure a load balancer.

5. Create Overlay and VLAN Transport Zones.
6. Create DVS Port Group for Edge Nodes.
7. Create Uplink Logical Switch for the Tier-0 Router.
8. Create a Tunnel Endpoint IP Pool.
9. Deploying NSX Edge Nodes for Enterprise PKS using the NSX Manager web interface.
10. Create Edge Cluster and Add Edge Nodes.
11. Prepare ESXi Hosts as Transport Nodes.
12. Verify the Installation of NSX-T.

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

Install the NSX Manager Unified Appliance for Enterprise PKS

In this topic

Page last updated:

This topic provides instructions for installing the NSX Manager VM on vSphere for use with VMware Enterprise PKS.

Prerequisites

Make sure you have completed the Prerequisites for Installing NSX-T for Enterprise PKS.

About the NSX Manager Appliance

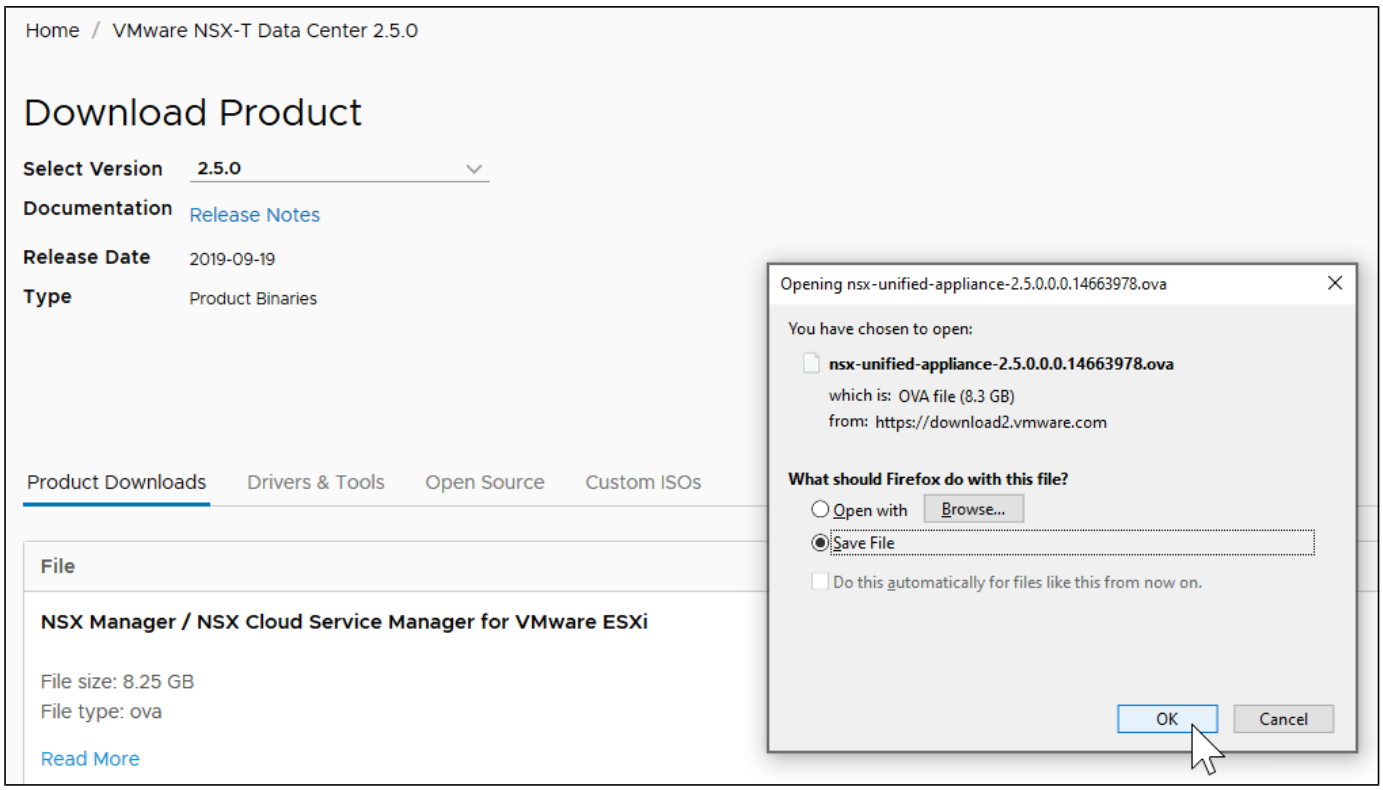
The NSX-T Manager VM is provided as an OVA file named the **NSX Unified Appliance** that you import into your vSphere environment and configure.

For more information, see Install the NSX Manager Unified Appliance and NSX Manager VM System Requirements in the NSX-T Data Center documentation.

Install and Configure the NSX Manager Appliance

Complete the following procedure to deploy the NSX Manager appliance:

1. Download the NSX Manager v2.5.0 OVA file to your local machine.

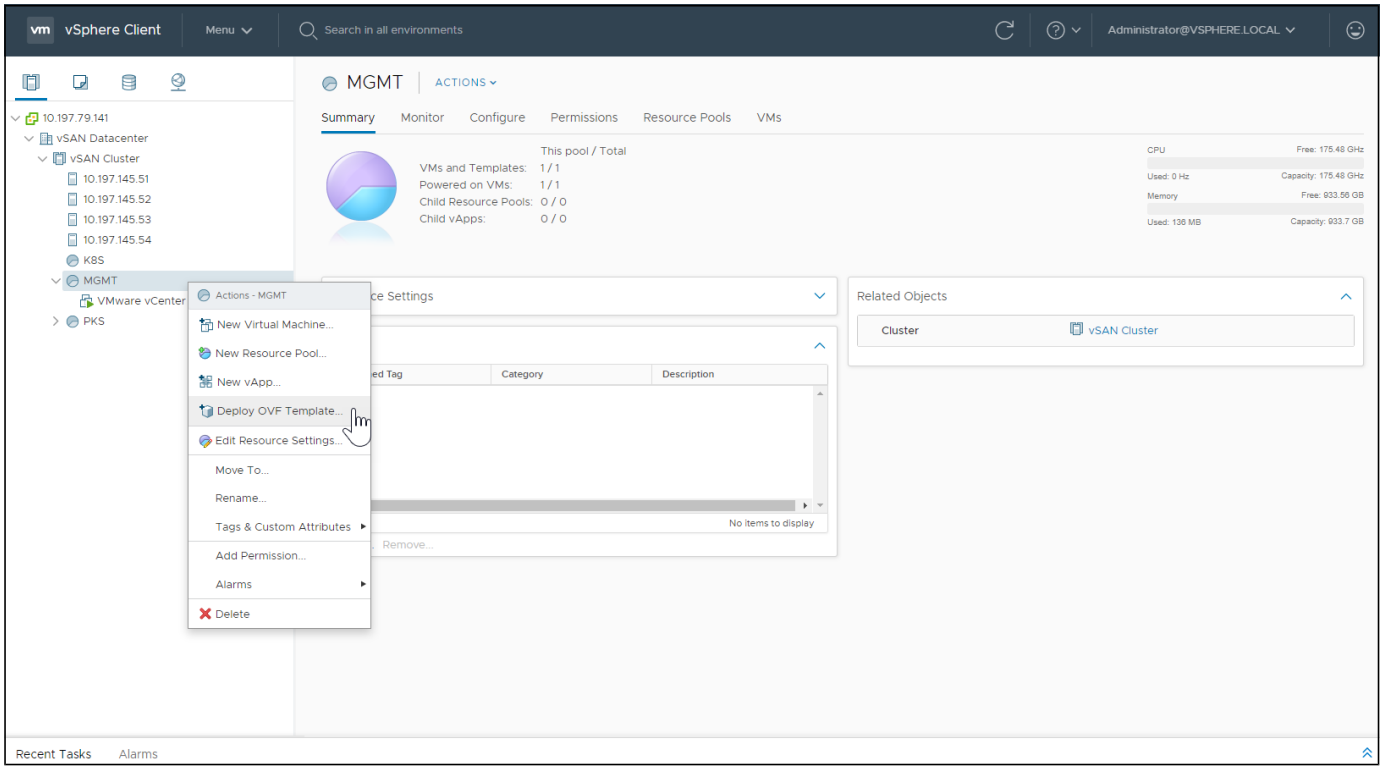


2. Log in to vCenter using the vSphere Client.

3. Create a **Resource Pool** for NSX-T infrastructure, such as `infra`.

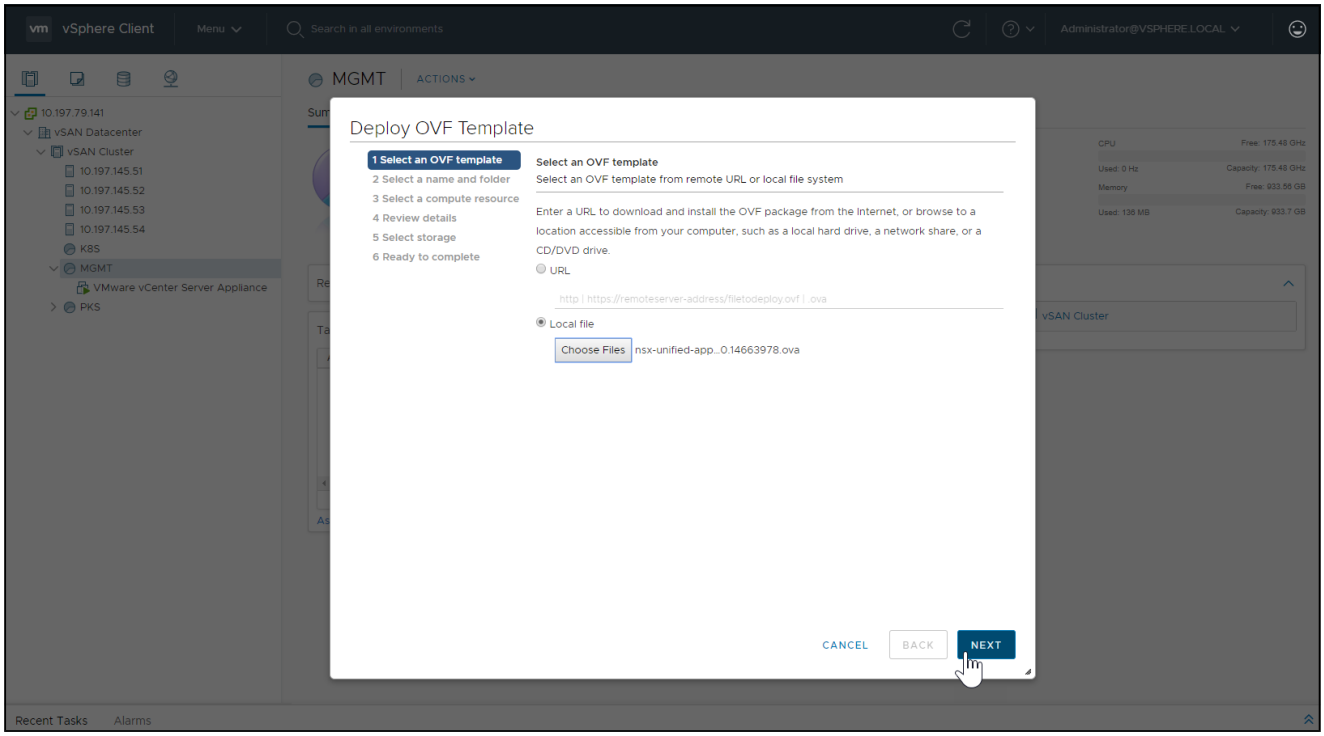
4. In the vSphere Client, select the **Resource Pool** where you want to install NSX-T Data Center.

5. Right-click and select **Deploy OVF Template** to start the installation wizard.



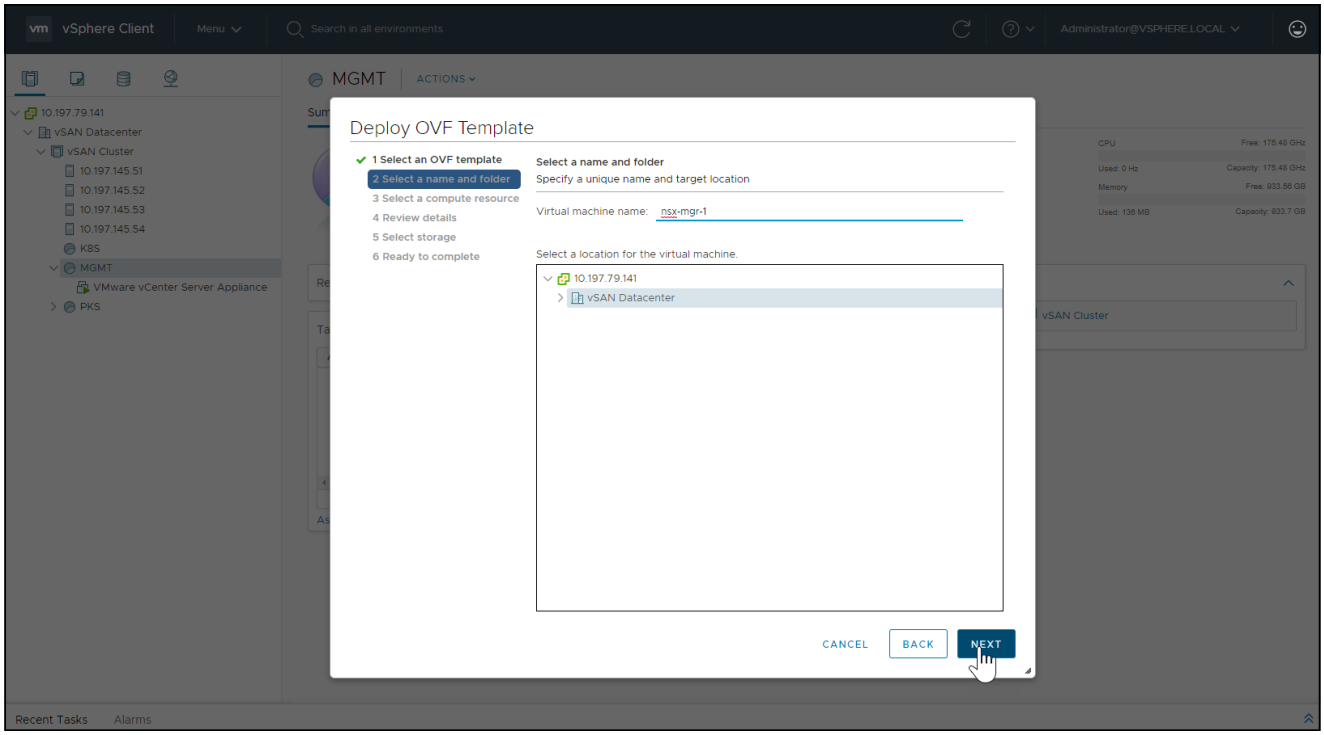
6. On the **Select an OVF template** screen:

a. Select the **Local file** option.
b. Click **Choose Files**.
c. Navigate to where you downloaded the OVA file and select it.
d. Click **Next**.



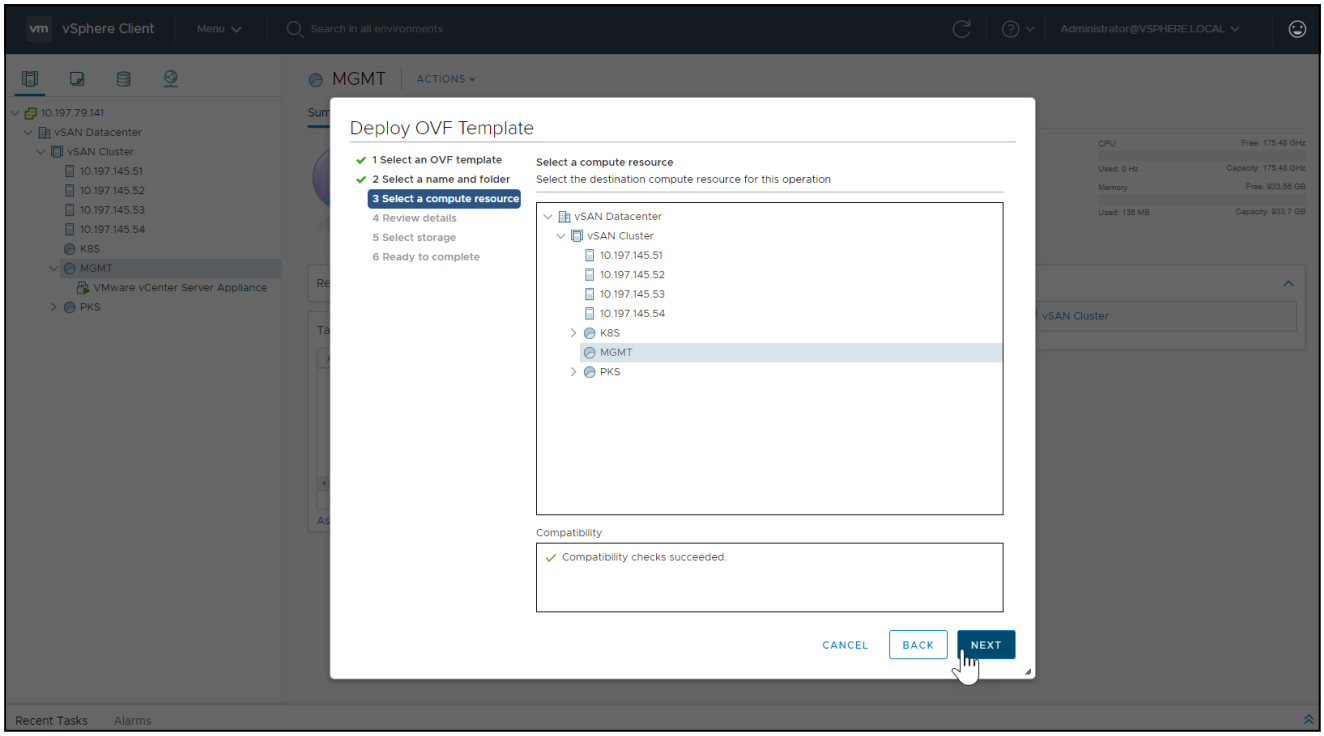
7. On the **Select a name and folder** screen:

a. Enter a name for the NSX Manager VM, such as `nsx-manager-1`.
b. Select the **Datacenter** for the VM deployment.
c. Click **Next**.



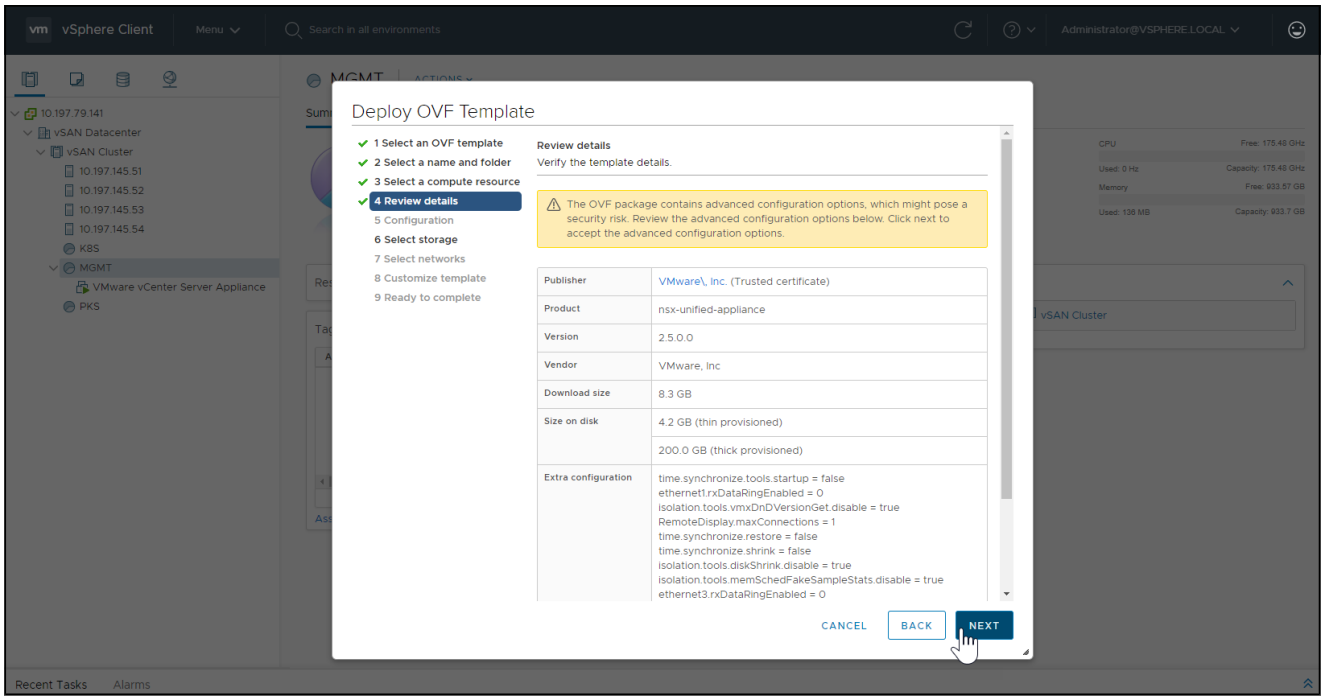
8. On the **Select a compute resource** screen:

   a. Select the **Resource Pool** where the NSX Manager VM will be deployed.
   b. Click **Next**.



9. On the **Review details** screen:

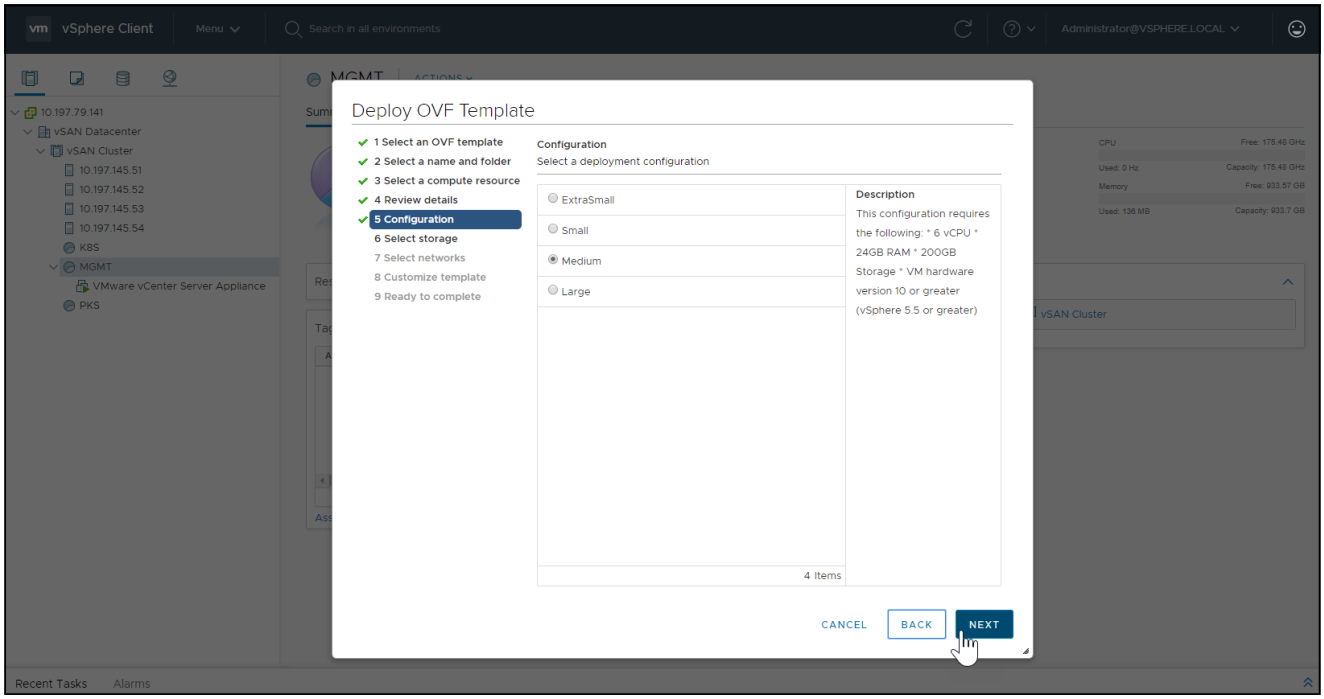
   a. Verify the OVF template details.
   b. Click **Next**.



10. On the **Configuration** screen:

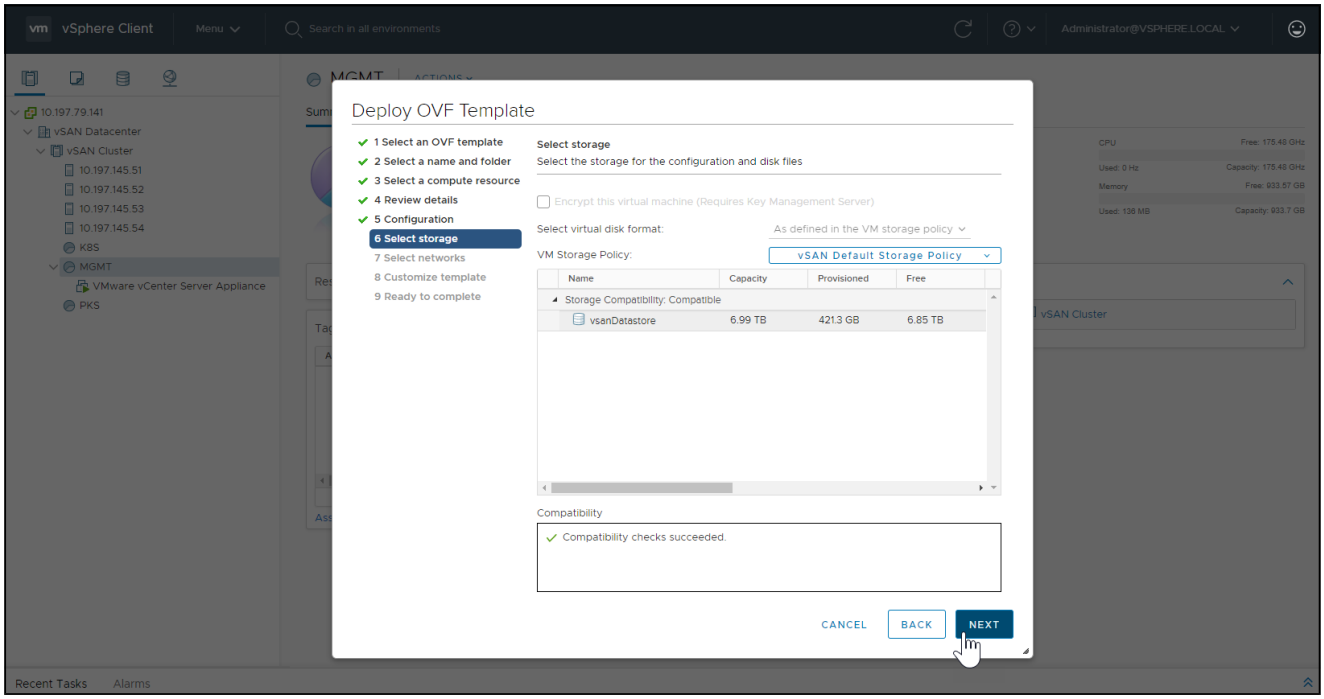
   a. Select either the **Medium** size VM or the **Large** size VM.

b. Click **Next**.



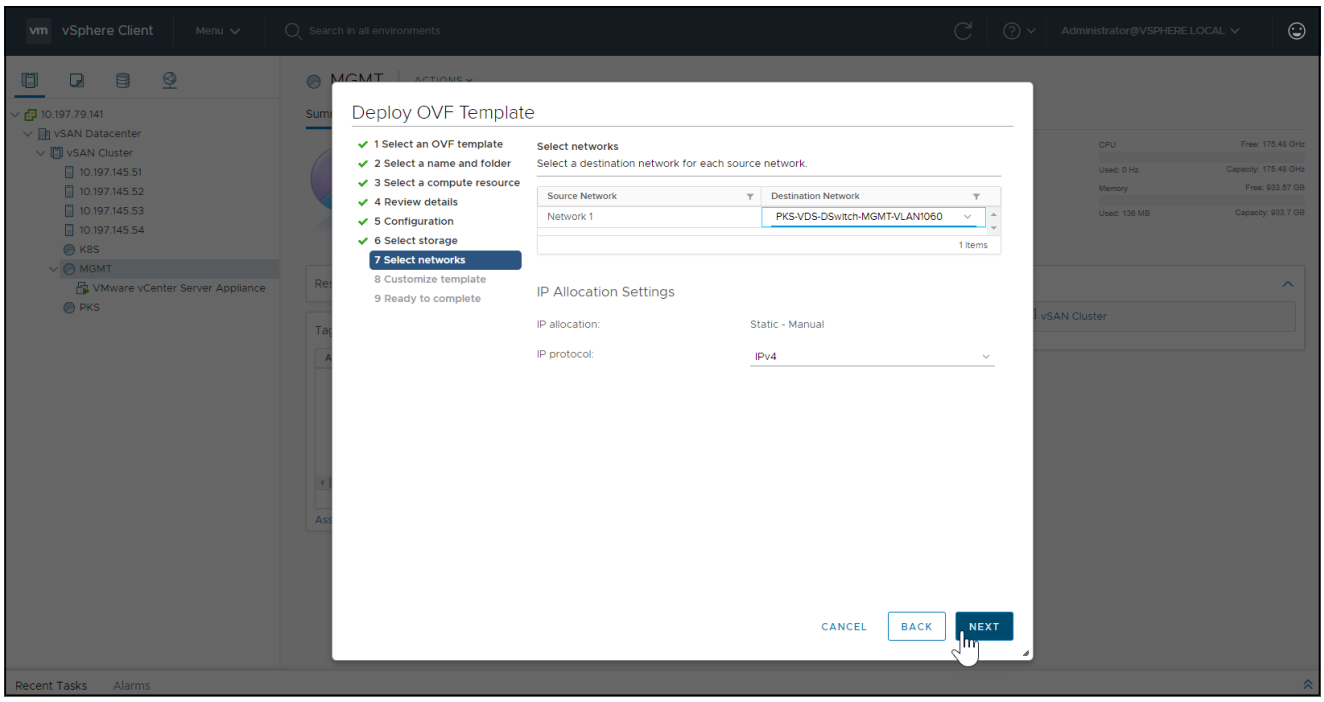
11. On the **Select storage** screen:

    a. Select the **vsanDatastore** if you are using vSAN, or a dedicated datastore if you are not.
    b. Click **Next**.



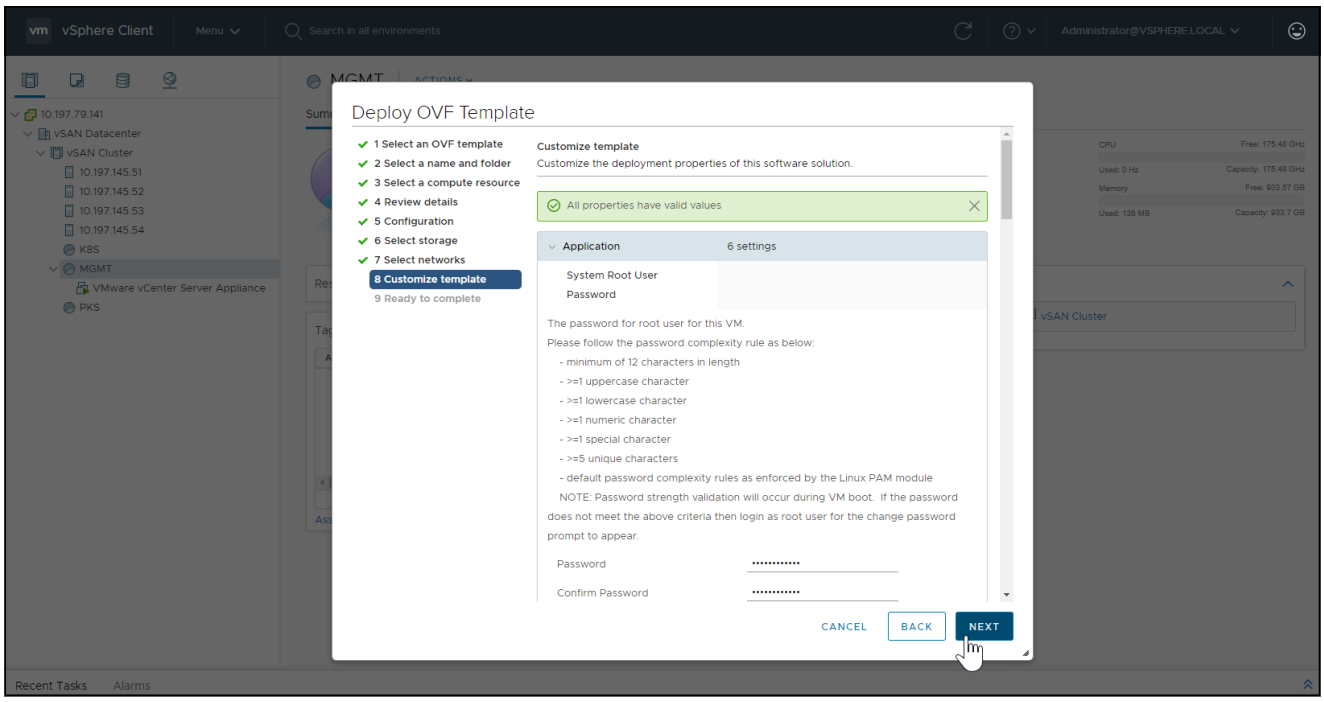
12. On the **Select networks** screen:

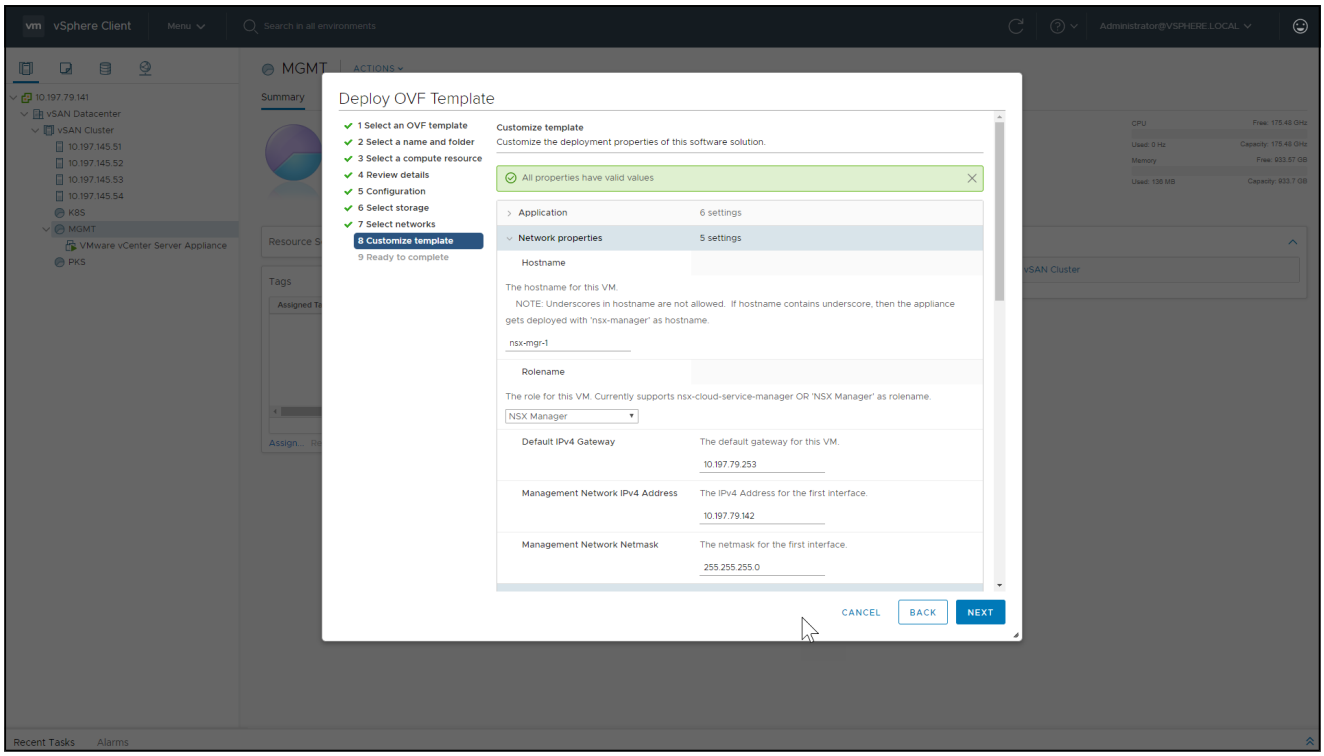
    a. Select **Destination Network** for the NSX Manager VM.
    b. Click **Next**.

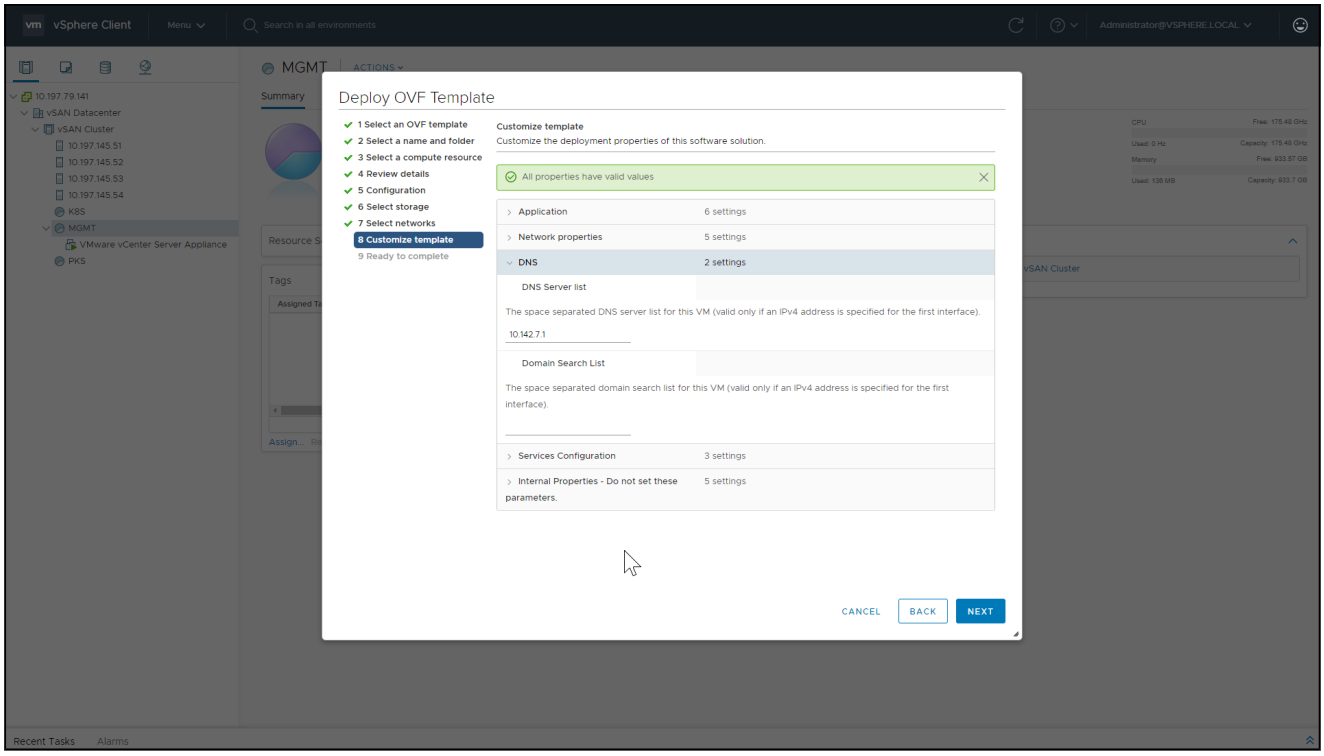


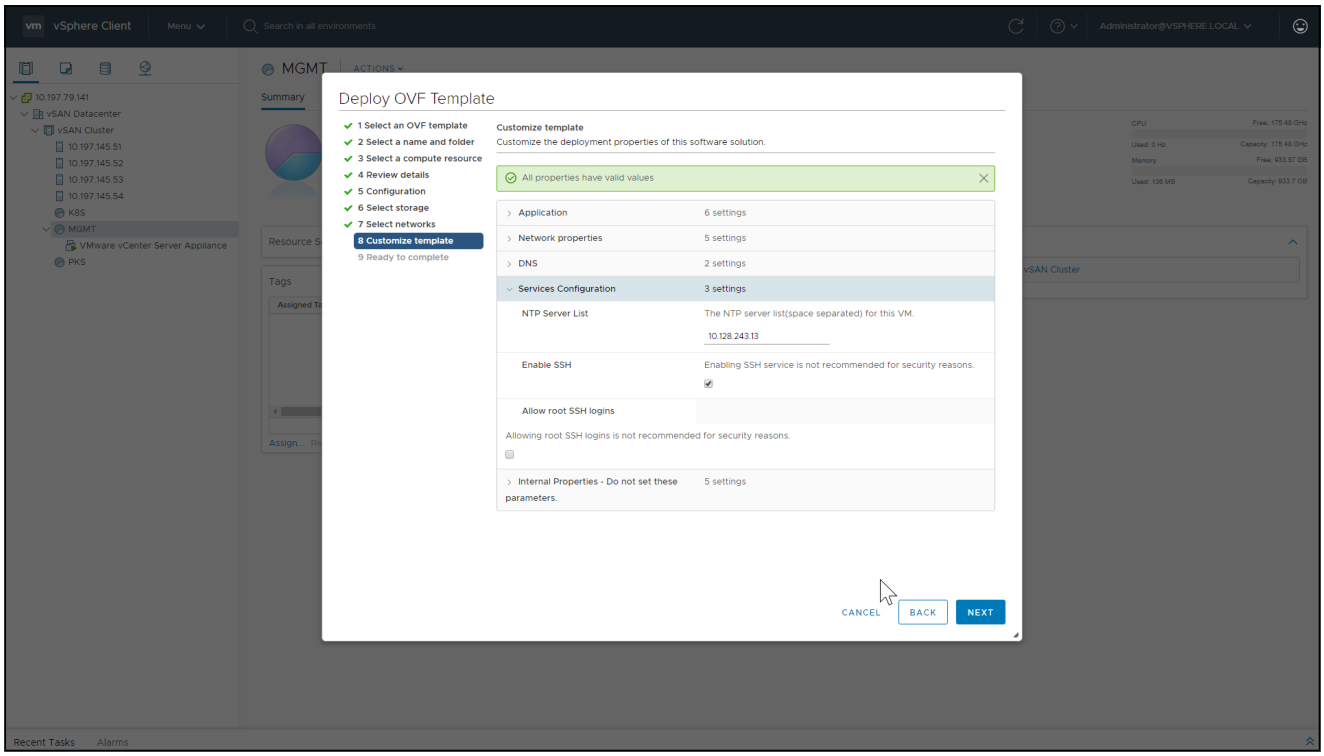
13. On the **Customize Template** screen, configure the following settings:

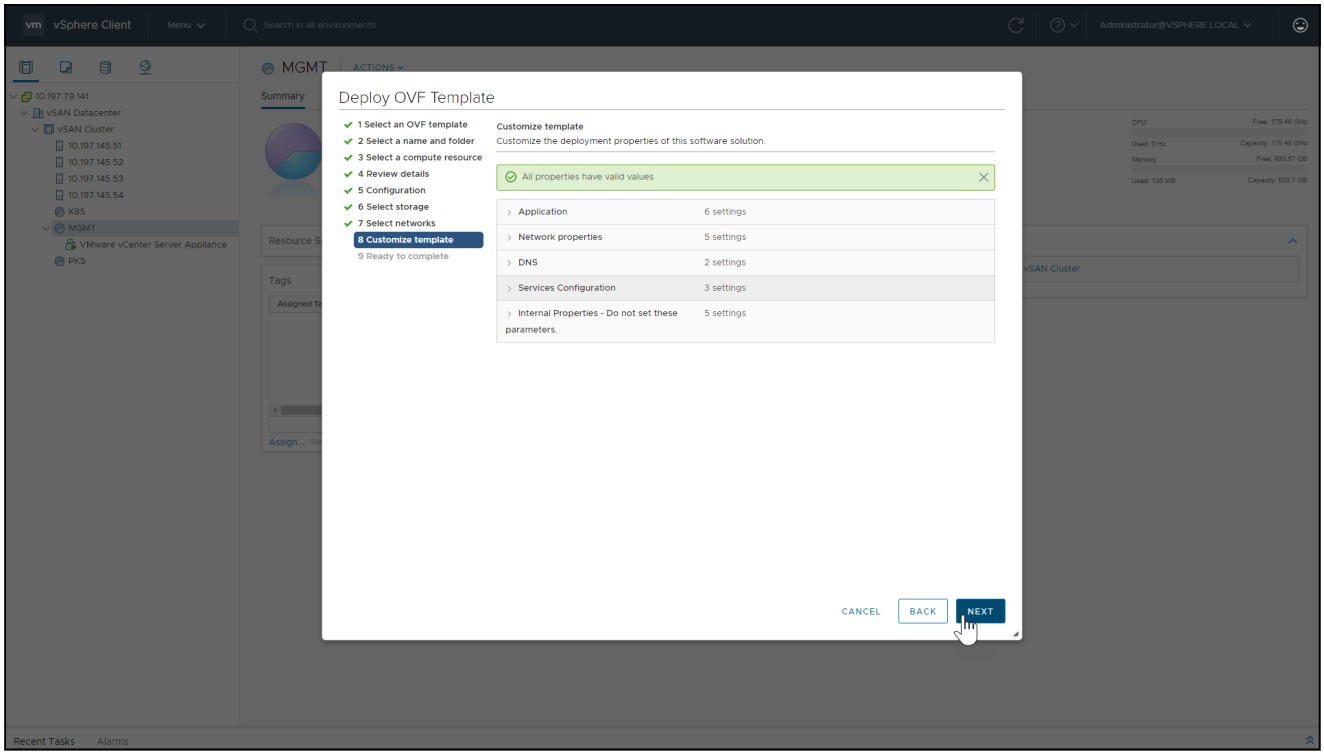
    a. **System Root User Password**: This password must comply with password strength restrictions.
    b. **CLI "admin" User Password**: This password must comply with password strength restrictions.
    c. **CLI "audit" User Password**: This password must comply with password strength restrictions.
    d. **Hostname**: Enter the hostname for the NSX Manager VM, such as `nsx-manager-1`.
    e. **Default IPv4 Gateway**: Enter the default gateway IPv4 address for the NSX Manager VM.
    f. **Management Network IPv4 Address**: Enter the IPv4 address for the first network interface.
    g. **Management Network Netmask**: Enter the netmask for the first network interface.
    h. **DNS Server List**: Enter one or more DNS servers (space-separated if multiple).
    i. **NTP Server List**: Enter one or more NTP servers (space-separated if multiple).
    j. **Enable SSH**: Select **Enable SSH**. By default, this setting is disabled for security reasons.
    k. **Allow root SSH logins**: Enable **Allow root SSH logins**. By default, this setting is disabled for security reasons.
    l. Click **Next**.





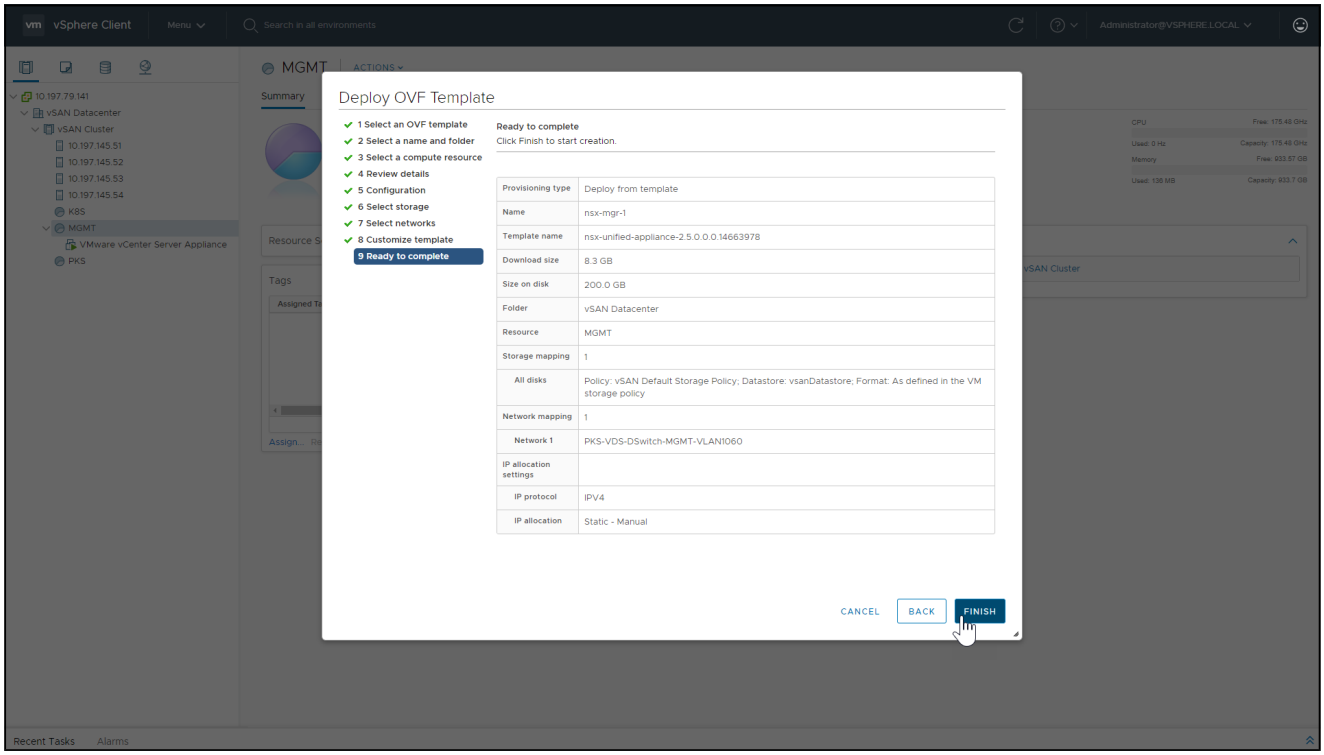






14. On the **Ready to complete** screen:

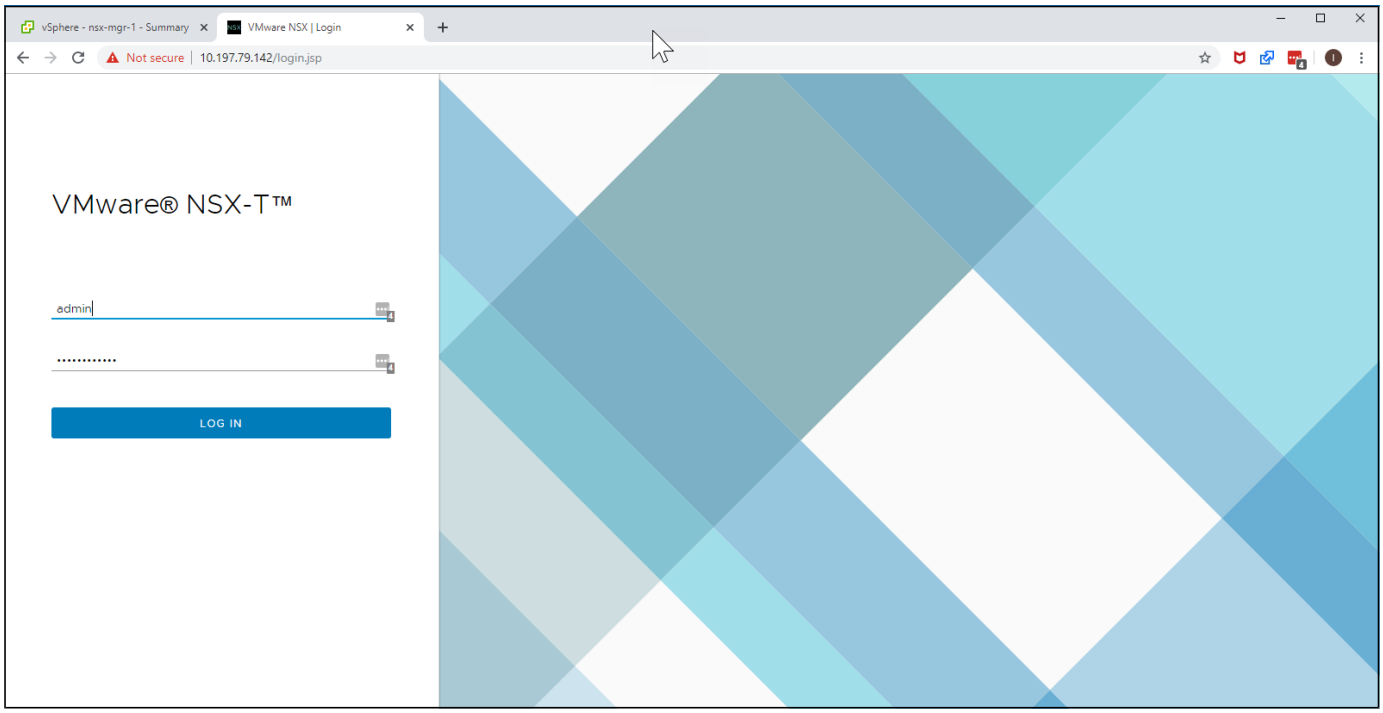
    a. Verify that the OVF template specification is accurate.
    b. Click **Finish** to begin the installation. It takes approximately 10 minutes to complete.



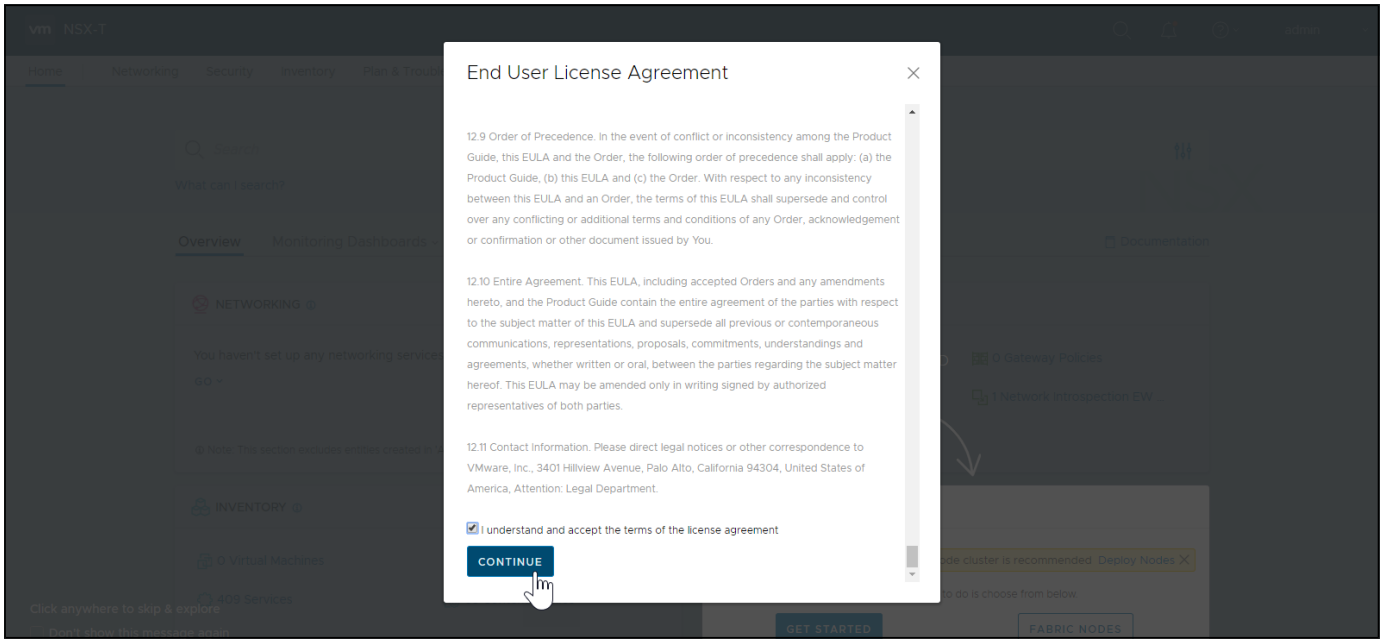
15. Expand the **Recent Tasks** panel at the bottom of the vCenter screen to view the progress of the OVA deployment.

Verify NSX-T Manager Installation

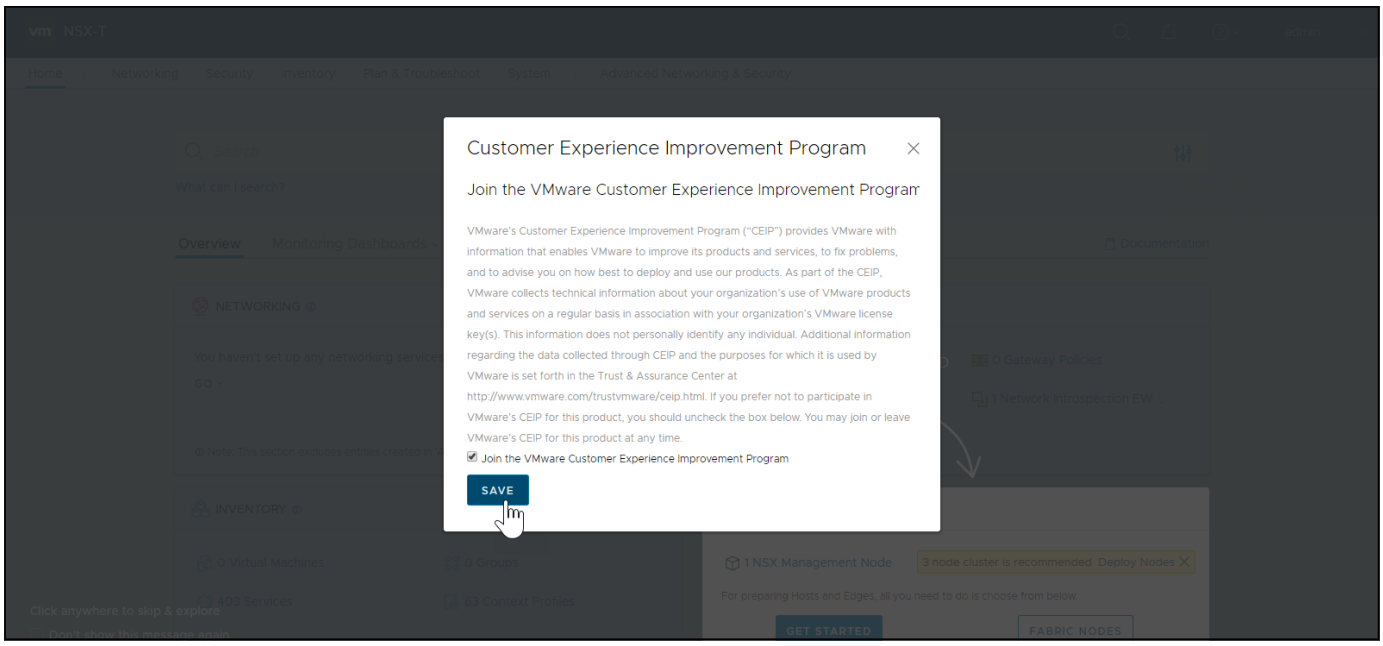
1. From your browser, log in with admin privileges to the NSX Manager user interface at https://NSX-MANAGER-IP-ADDRESS .



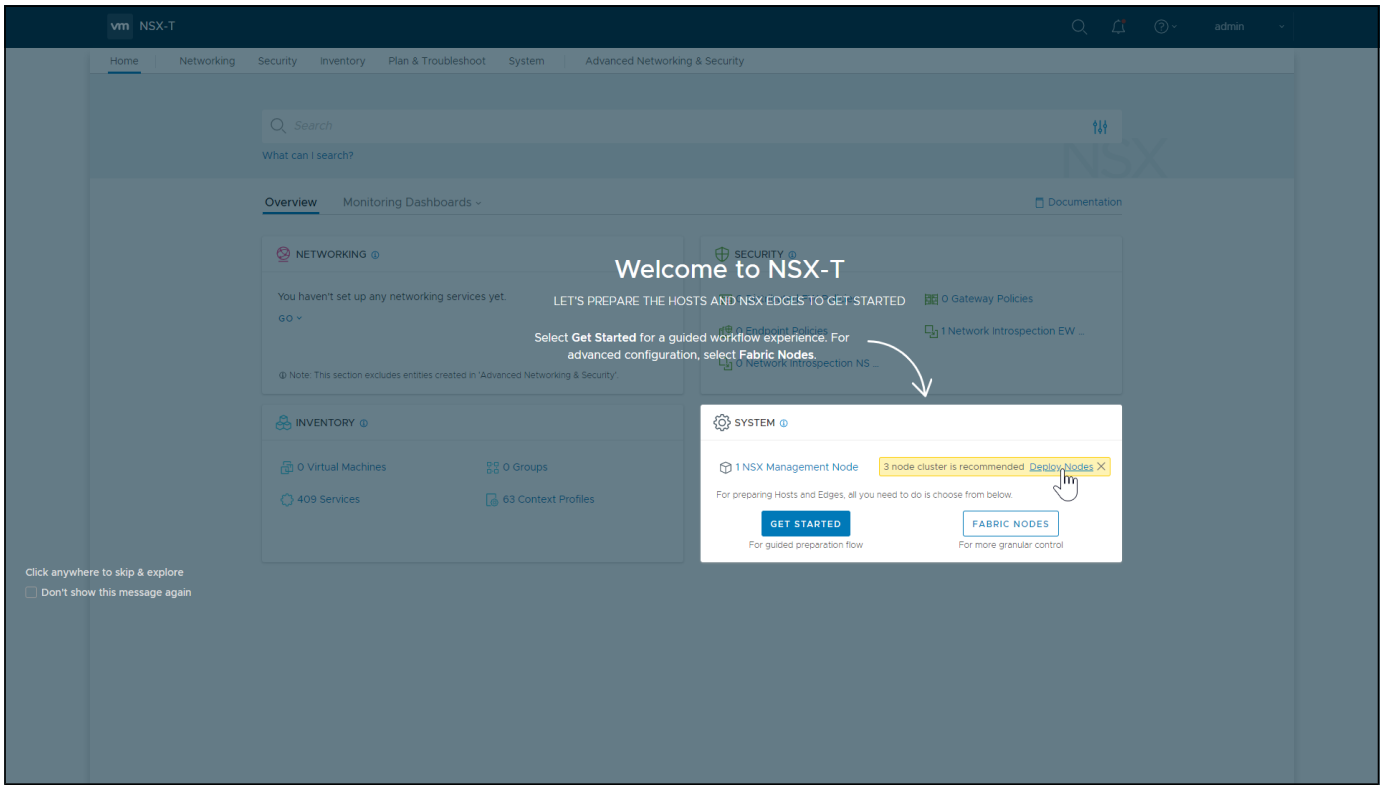
2. Read and accept the EULA terms.



3. Select whether to join the VMware's Customer Experience Improvement Program (CEIP) and click **Save**.



4. Notice at the **Welcome to NSX-T** screen, you are alerted that 1 NSX Manager Node is installed and a "3 node cluster is recommended." Before you can add additional NSX-T Manager nodes, you must add a Compute Manager.



5. Navigate to the **Advanced Networking & Security** tab. This is the tab to use for creating and working with NSX-T objects for Enterprise PKS.

> **Note:** For Enterprise PKS, you must use the `Advanced Networking and Security` tab to create, read, update, and delete required network objects. For NSX-T host preparation and configuration, such as deploying NSX-T Managers and Edge Nodes, use the `System` tab. Enterprise PKS does not support the `Simplified UI/API` tab.

Next Step

Add Compute Manager.

Installation Instructions Home

See Installing and Configuring NSX-T v2.5 for Enterprise PKS.

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

Add vCenter as the Compute Manager for NSX-T

In this topic

Page last updated:

This topic provides instructions for adding vCenter as the Compute Manager for NSX-T Data Center.

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

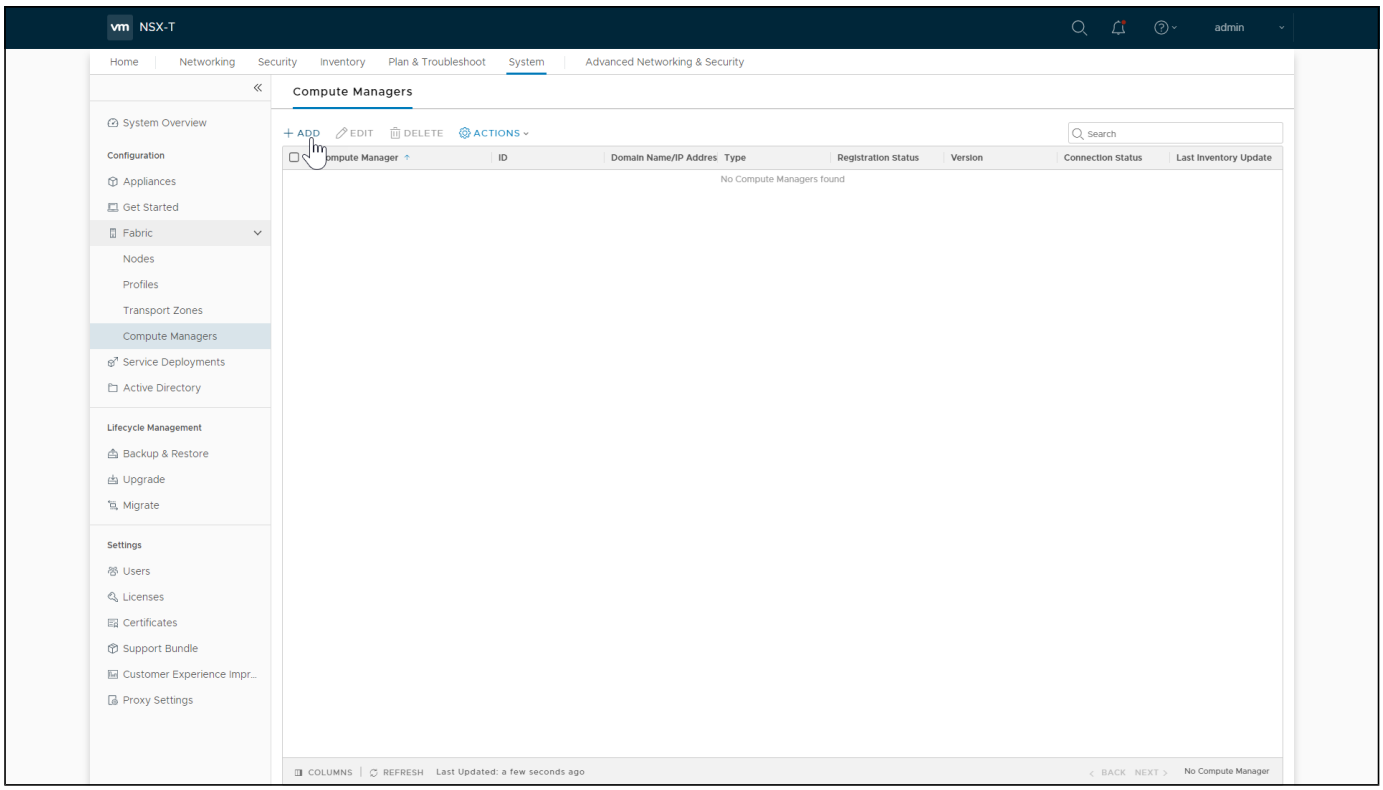
About the Compute Manager

A Compute Manager is an application that manages resources such as hosts and VMs. VMware Enterprise PKS deployments on vSphere with NSX-T use vCenter as the compute manager.

Add vCenter as the Compute Manager

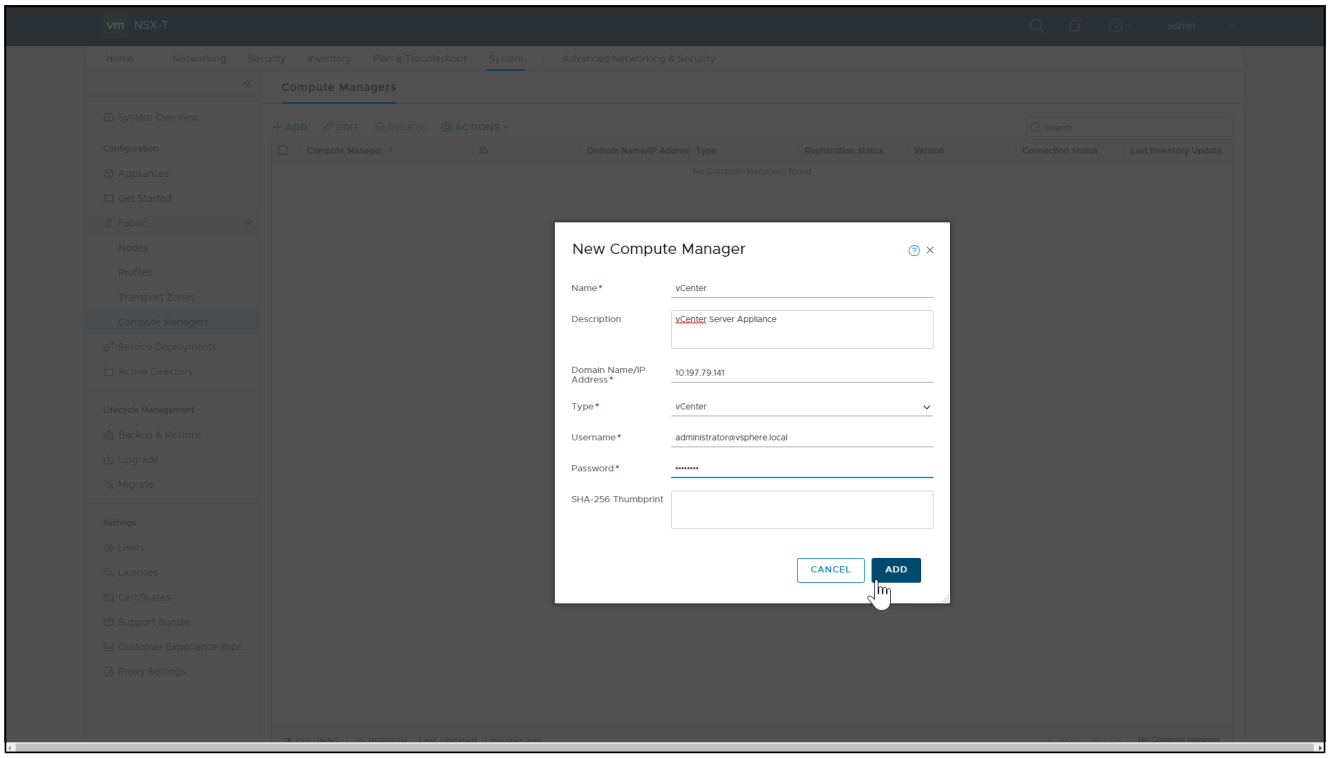
Complete the following steps to add vCenter as the Compute Manager:

1. From your browser, log in with admin privileges to NSX Manager at `https://NSX-MANAGER-IP-ADDRESS`.
2. In NSX-T Manager, select **System** > **Fabric** > **Compute Managers** > **Add**.



3. Configure the Compute Manager as follows:

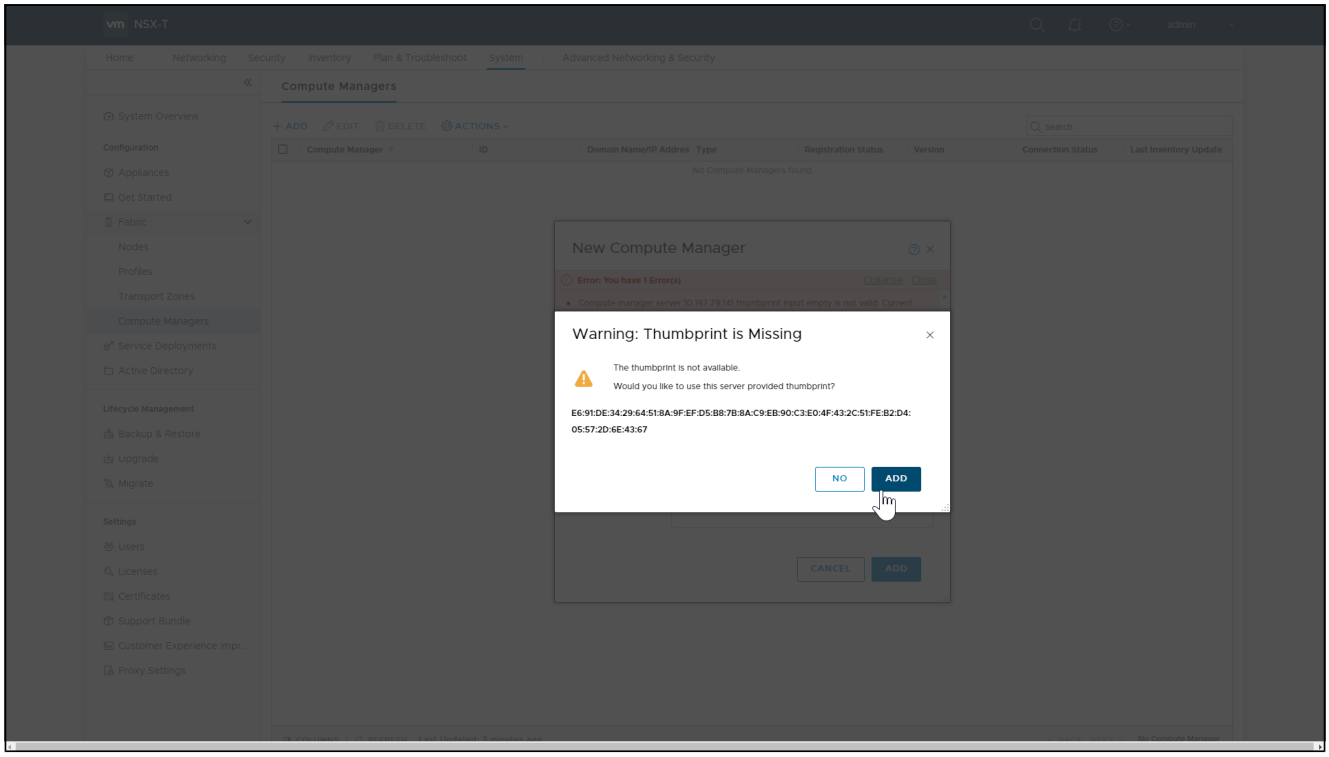
   - **Name**: Enter the name to identify the vCenter Server.
   - **Description**: Optionally add details such as the number of clusters in the vCenter Server.
   - **Domain Name/IP Address**: Enter the IP address of the vCenter Server.
   - **Type**: Keep the default option, vCenter.
   - **Username** and **Password**: Enter the vCenter Server login credentials.
   - **SHA-256 Thumbprint**: Leave the thumbprint value blank.



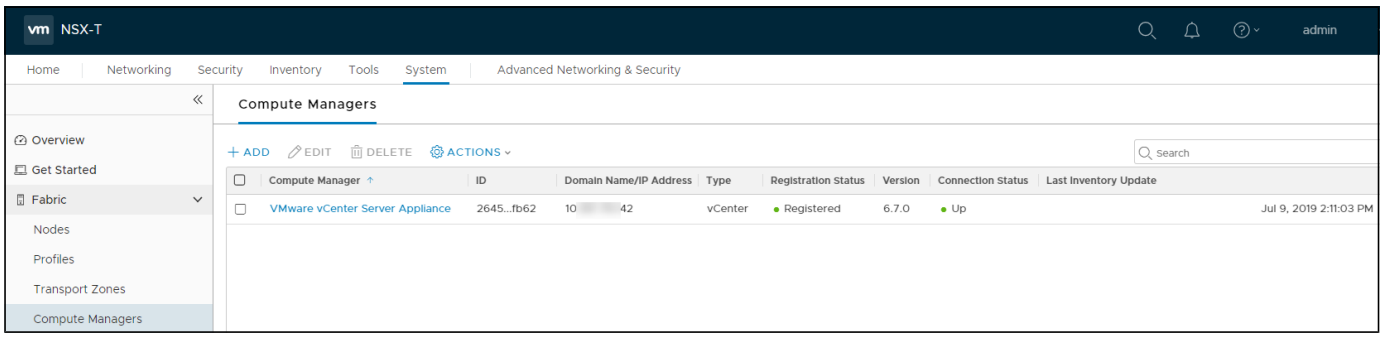
4. Click **Add**.

5. Click **Add** again.

   - Since you left the thumbprint value blank, you are prompted to accept the server-provided thumbprint.
   - After you accept the thumbprint, it takes a few seconds for NSX-T Data Center to discover and register the vCenter Server resources.



6. Verify that the vCenter Compute Manager is successfully **Registered** and its connection status is **Up**.



7. If the Compute Manager is not registered, troubleshoot as necessary.

Next Step

Add NSX-T Manager Nodes.

Installation Instructions

See Installing and Configuring NSX-T v2.5 for Enterprise PKS.

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

Add NSX Manager Nodes

In this topic

Page last updated:

This topic provides instructions for deploying additional NSX Manager nodes and forming the NSX Management Cluster for use with VMware Enterprise PKS.

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

About the NSX-T Management Cluster

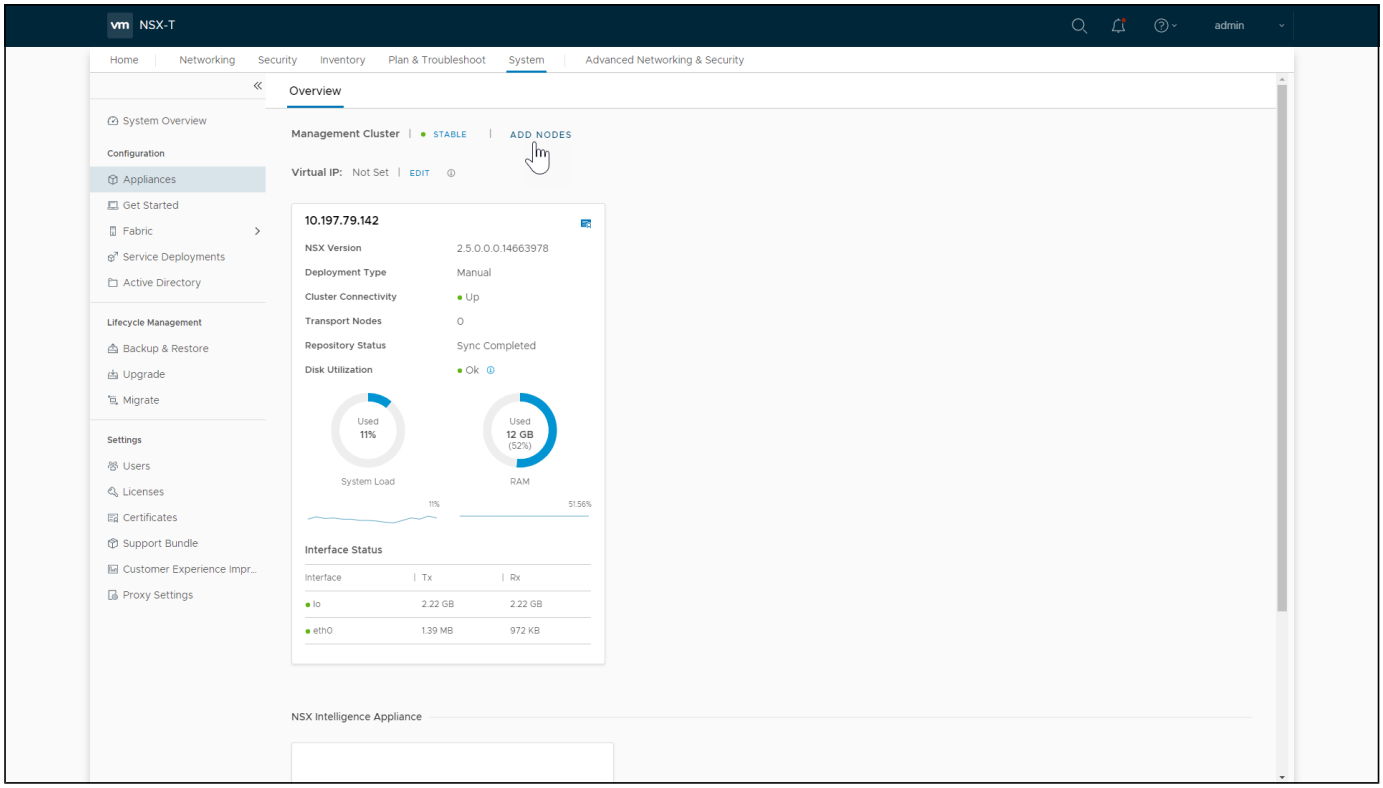
The NSX-T Management Cluster is a collection of three NSX Manager nodes for high availability. You can deploy additional NSX-T Manager nodes using the NSX-T Manager web interface. All repository details and user credentials are synchronized across NSX-T Manager Nodes by the management cluster database.

For more information, see Deploy NSX Manager Nodes to Form a Cluster from UI in the NSX-T Data Center documentation.

Deploy Additional NSX Manager Nodes

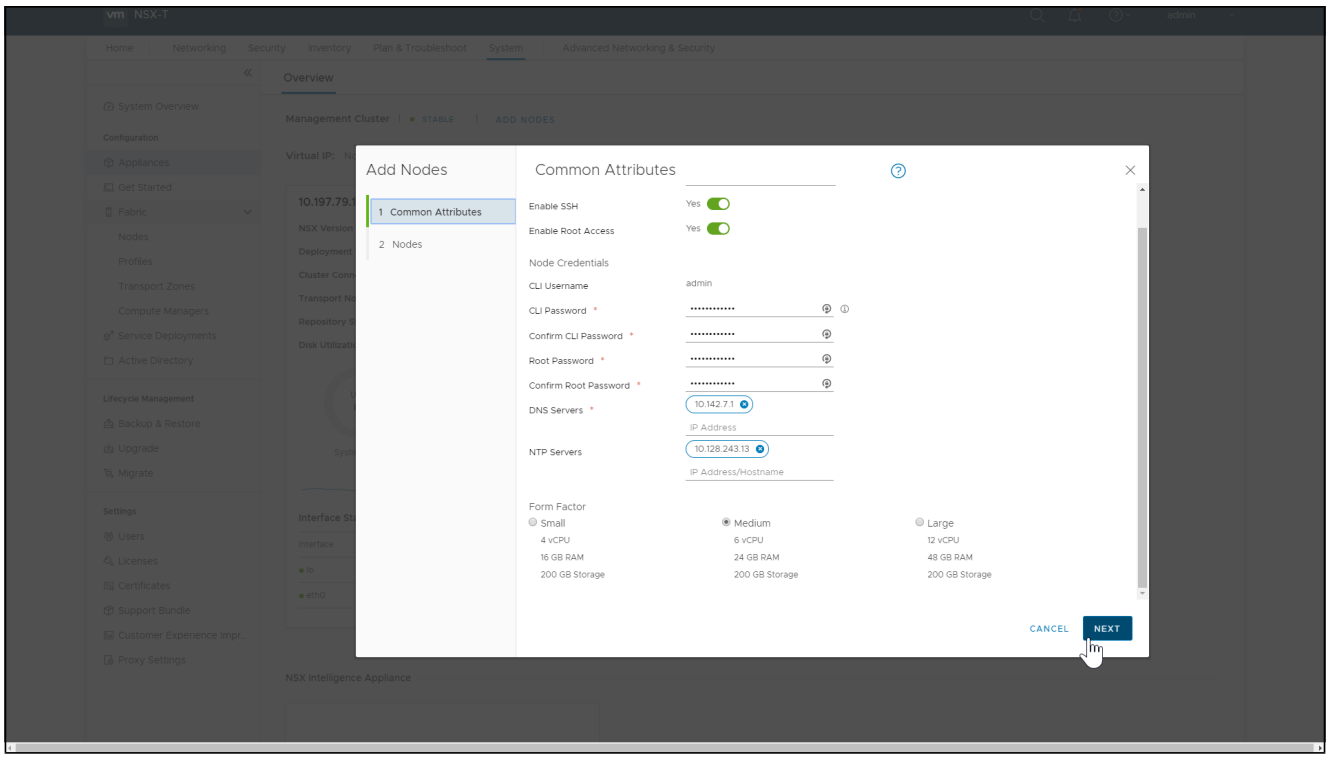
Complete the following procedure to deploy additional manager nodes and form the NSX-T Management Cluster:

1. From your browser, log in with admin privileges to NSX Manager at `https://NSX-MANAGER-IP-ADDRESS`.
2. In NSX-T Manager, select **System** > **Overview** > **Add Nodes**.



3. Enter the NSX-T Manager common attribute details as follows:

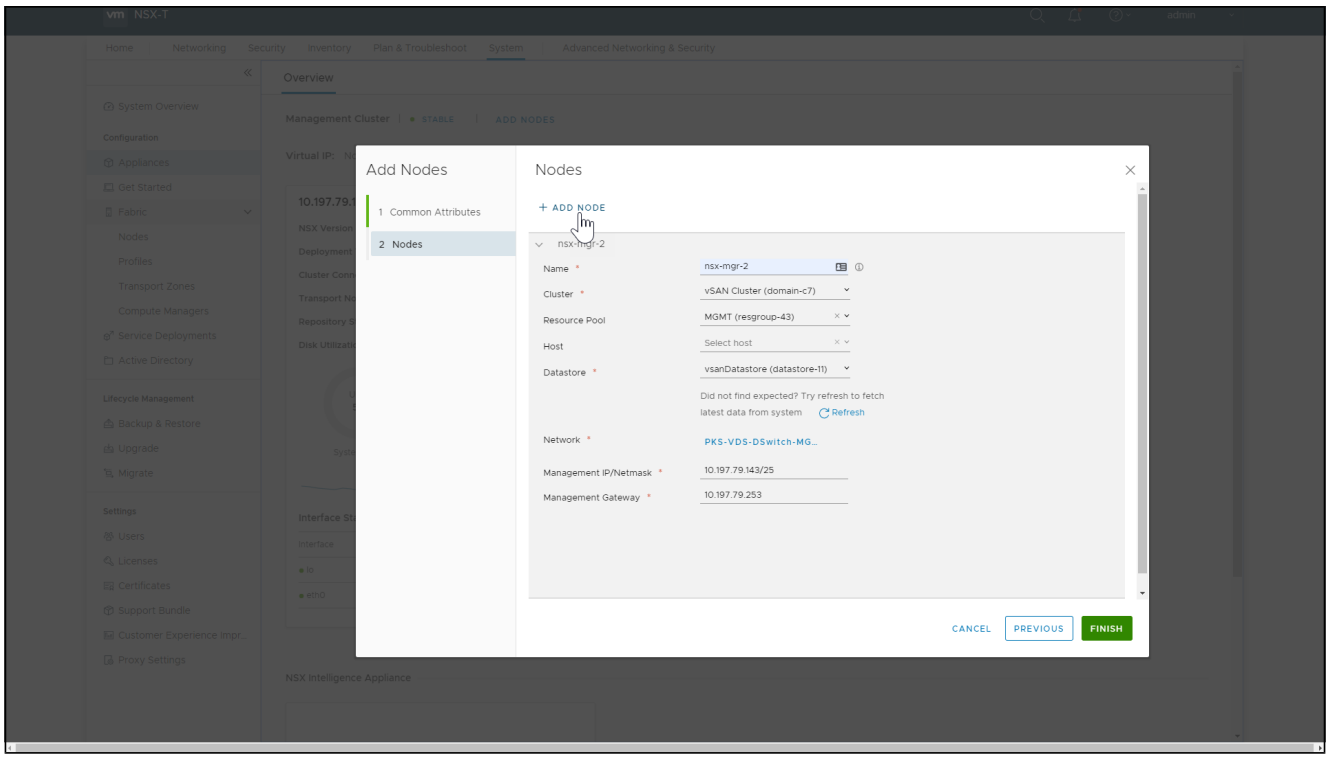
   - **Compute Manager**: Registered resource compute manager is populated.
   - **Enable SSH**: Toggle the button to allow an SSH login to the new NSX Manager node.
   - **Enable Root Access**: Toggle the button to allow the root access to the new NSX Manager node.
   - **CLI Username and Password Confirmation**: The CLI username is set to `admin`. The password must comply with the password strength restrictions.
   - **Root Password and Password Confirmation**: Set the root password for the new node. The password must comply with the password strength restrictions.
   - **DNS Servers**: Enter the DNS server IP address available in the vCenter Server.
   - **NTP Servers**: Enter the NTP server IP address.
   - **Form Factor**: Select **Medium** (default) or **Large**. Do not select **Small**.



4. Click **Next**.

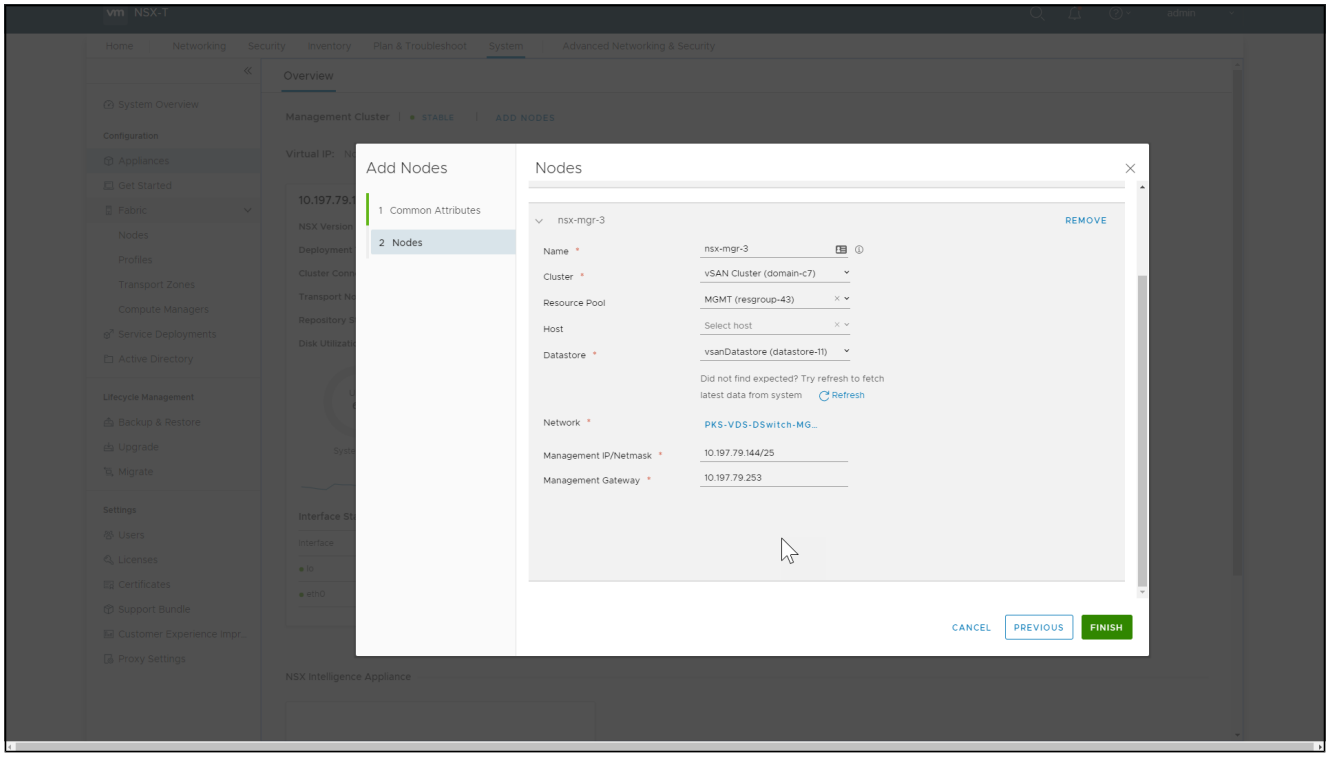
5. At the **Nodes** tab, enter the configuration details for the second NSX-T Manager node:

   - **Name**: Enter a name for the NSX-T Manager node, such as `nsx-manager-2`.
   - **Cluster**: Designate the cluster the node is going to join from the drop-down menu.
   - **Resource Pool or Host**: Select the **infra** resource pool from the drop-down menu.
   - **Datastore**: Select a datastore for the node files from the drop-down menu.
   - **Network**: Assign the network from the drop-down menu.
   - **Management IP/Netmask**: Enter the IP address and netmask.
   - **Management Gateway**: Enter the gateway IP address.

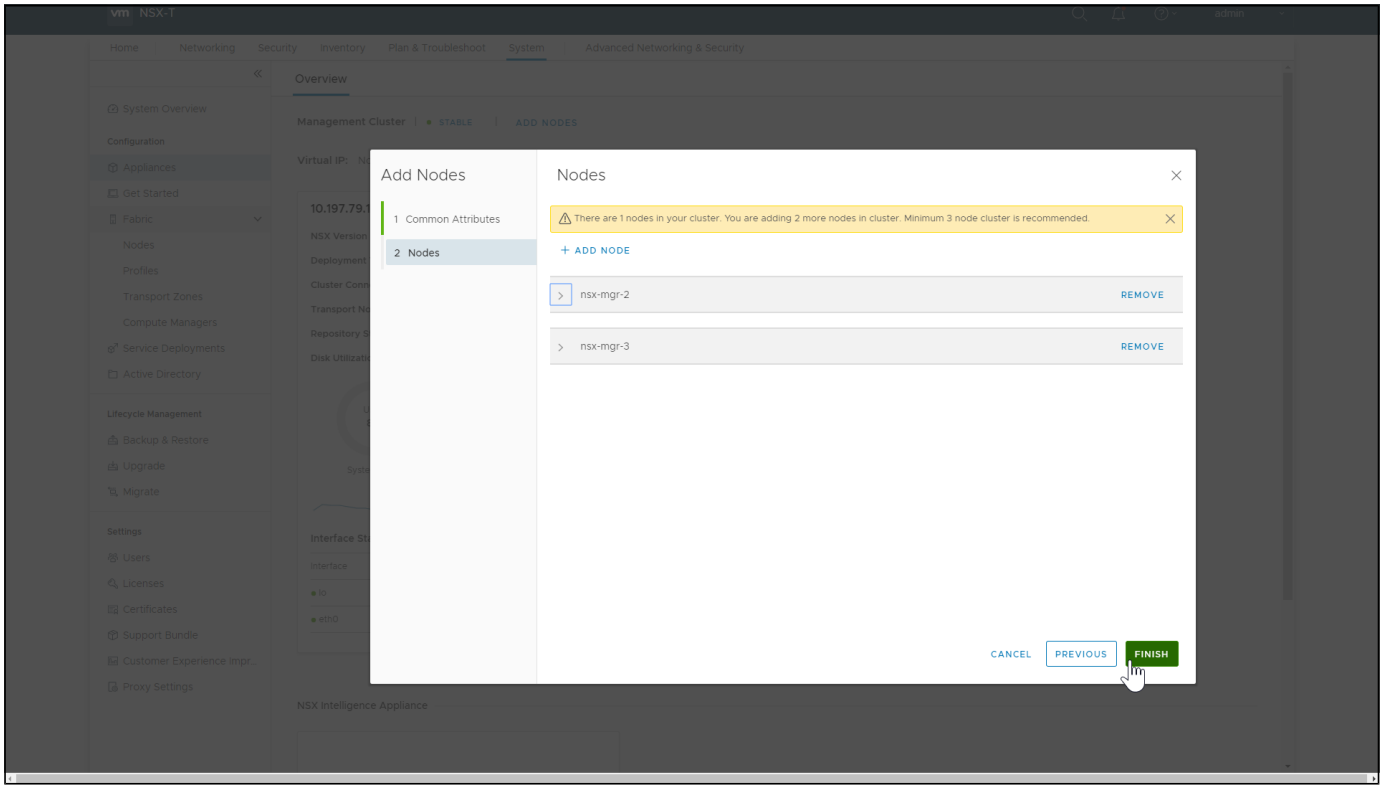


6. Click **Add Node** and configure a third NSX Manager node:

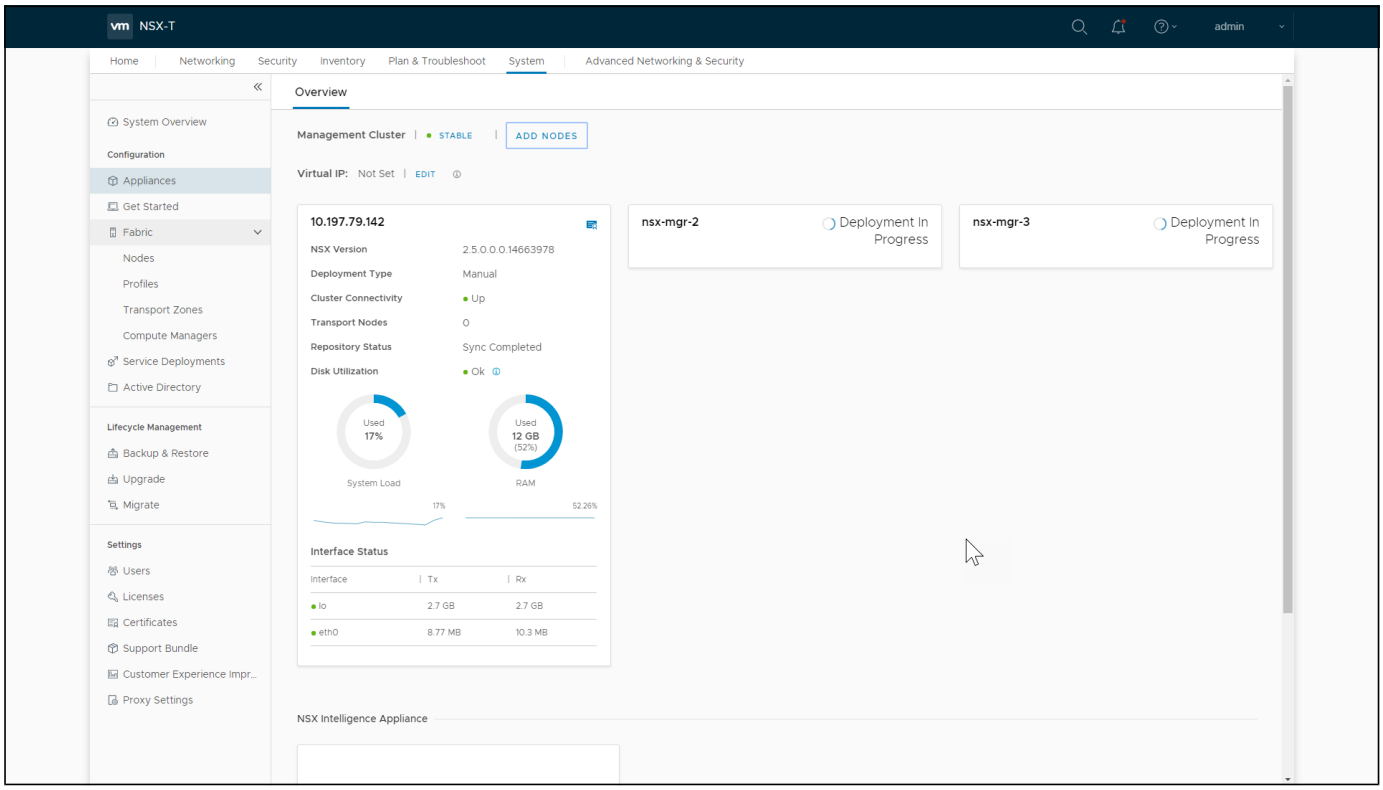
   - **Name**: Enter a name for the NSX-T Manager node, such as `nsx-manager-3`.
   - **Cluster**: Designate the cluster the node is going to join from the drop-down menu.
   - **Resource Pool or Host**: Select the **infra** resource pool from the drop-down menu.
   - **Datastore**: Select a datastore for the node files from the drop-down menu.
   - **Network**: Assign the network from the drop-down menu.
   - **Management IP/Netmask**: Enter the IP address and netmask.
   - **Management Gateway**: Enter the gateway IP address.



7. Click **Finish**.



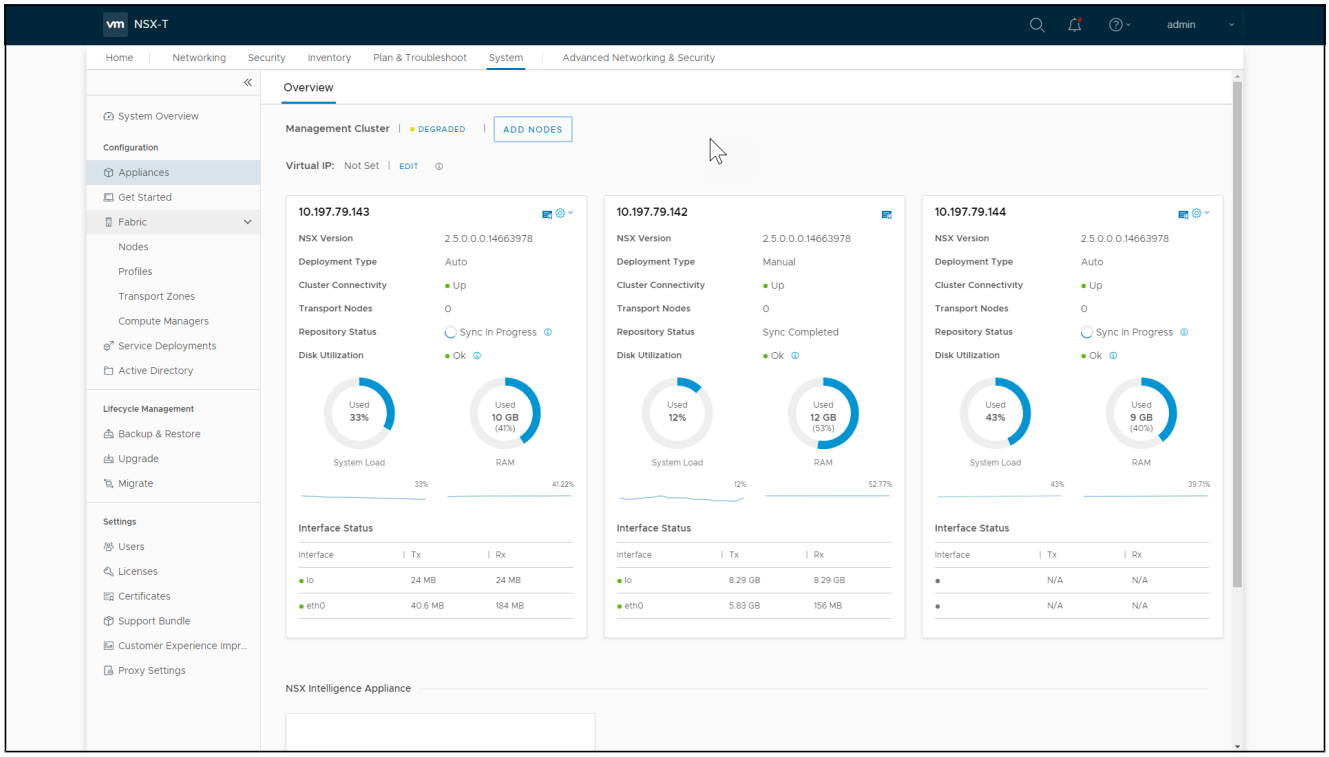
8. The additional NSX-T Manager nodes are deployed.



Verify the NSX Management Cluster

1. Verify the deployment of the additional NSX Manager nodes.

- At the **System** > **Overview** > **Management Cluster**, you should see the deployment of the additional nodes.
- Wait for 10-15 minutes for the deployment, cluster formation, and repository synchronization to complete.
- Track the deployment process at the **System** > **Overview** screen in vCenter Server.



Next Step

Configure VIP Address for the NSX-T Management Cluster.

Installation Instructions

See Installing and Configuring NSX-T v2.5 for Enterprise PKS.

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

Add VIP Address for the NSX-T Management Cluster

In this topic

Page last updated:

This topic describes how to configure a Virtual IP address (VIP) for the NSX-T Management Cluster for VMware Enterprise PKS.

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

About the NSX Management Cluster VIP

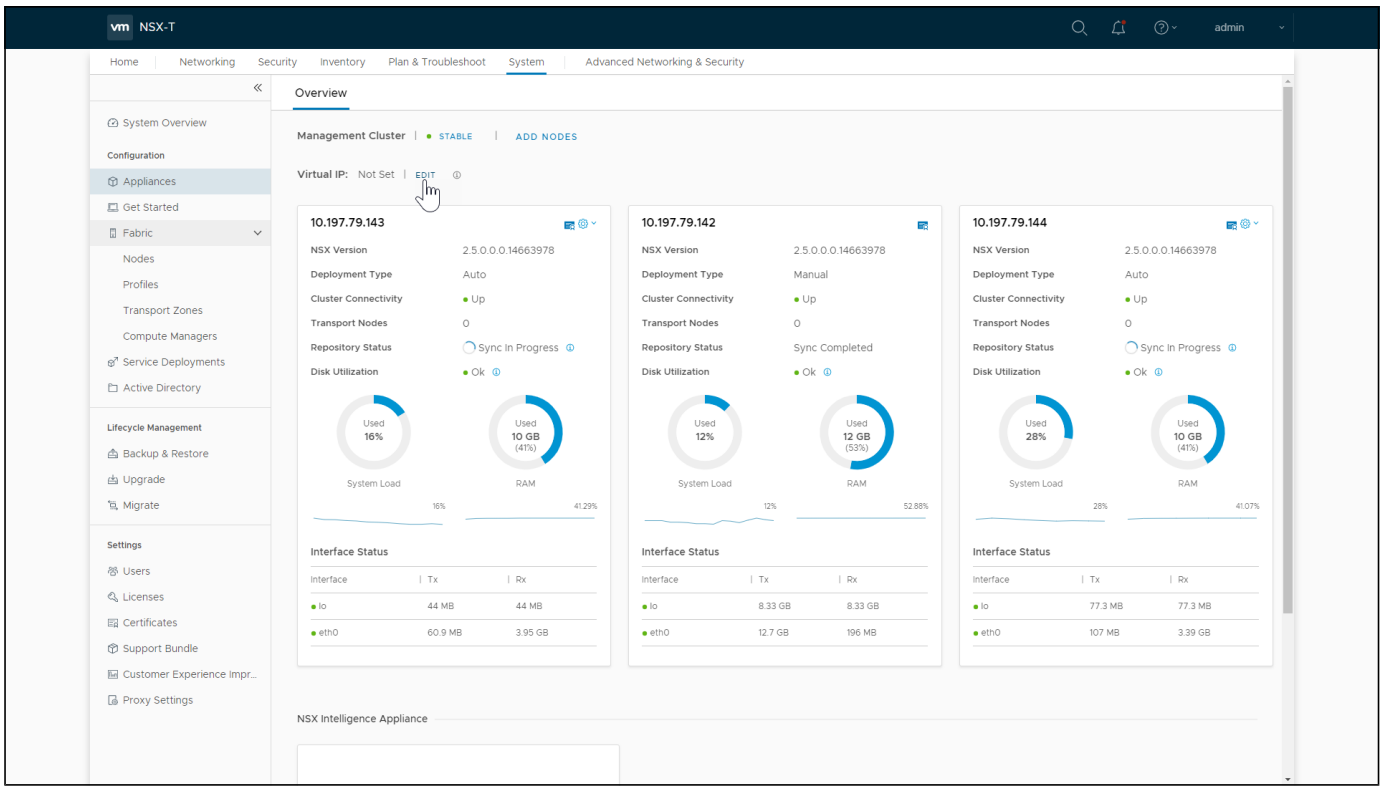
The NSX-T Management Cluster comprises three NSX-T Manager nodes. To support a single access point for the NSX Manager web interface and API, assign a virtual IP address (VIP) to the NSX-T Management Cluster. Once the VIP is assigned, any UI and API requests to NSX-T are redirected to the VIP address, which is owned by the leader node. The leader node then routes the request to the other cluster nodes.

Using a VIP makes the NSX Management Cluster highly available. If you need to scale, you can provision a load balancer for the NSX-T Management Cluster. Provisioning a load balancer requires that NSX-T be fully installed and configured. It is recommended that you configure the VIP now, then if needed, install a load balancer after NSX-T is installed and configured. For more information about configuring a load balancer for the NSX-T Management Cluster, see Provisioning a Load Balancer for the NSX-T Management Cluster.

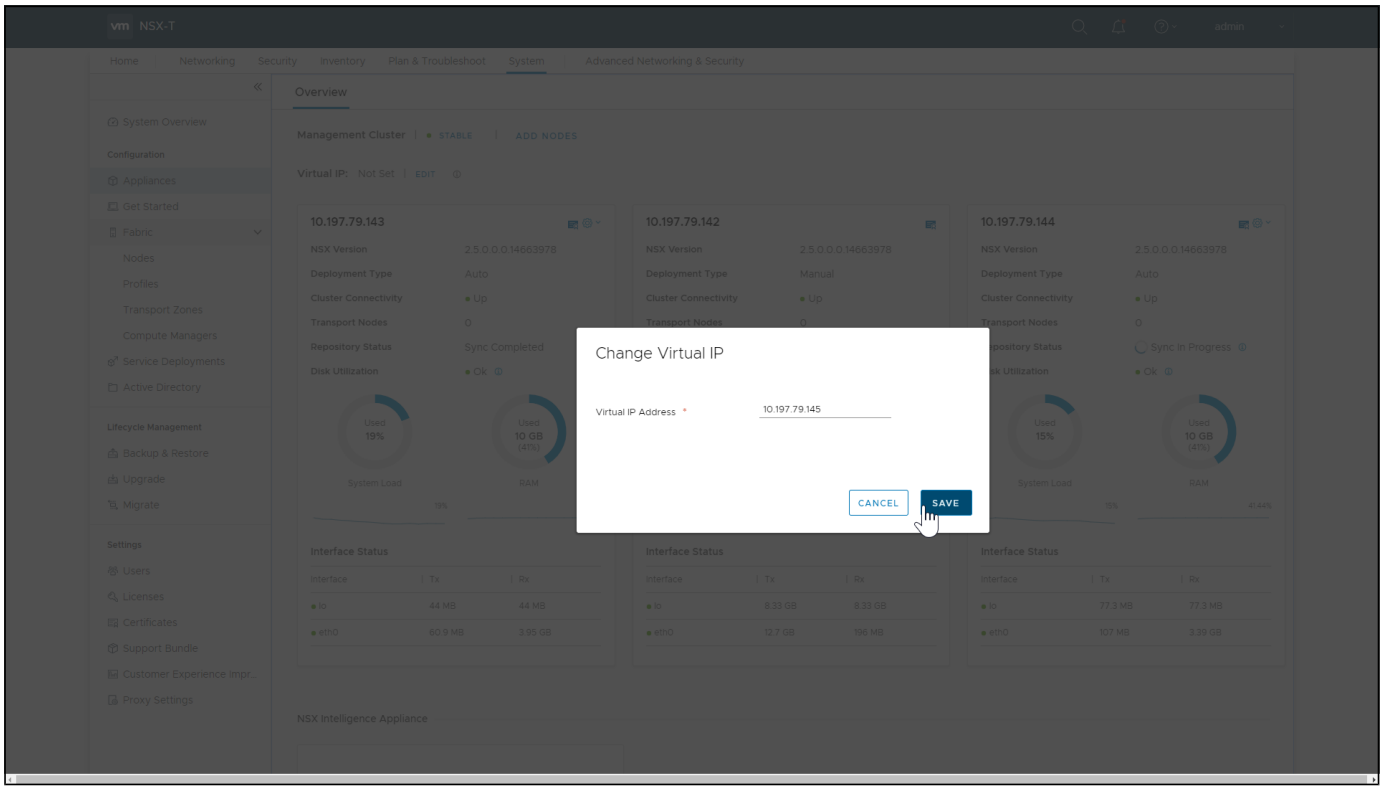
Configure a VIP Address for the NSX-T Management Cluster

Complete the following instructions to create a VIP for the NSX Management Cluster. The IP address you use for the VIP must be part of the same subnet as the NSX-T Management nodes.

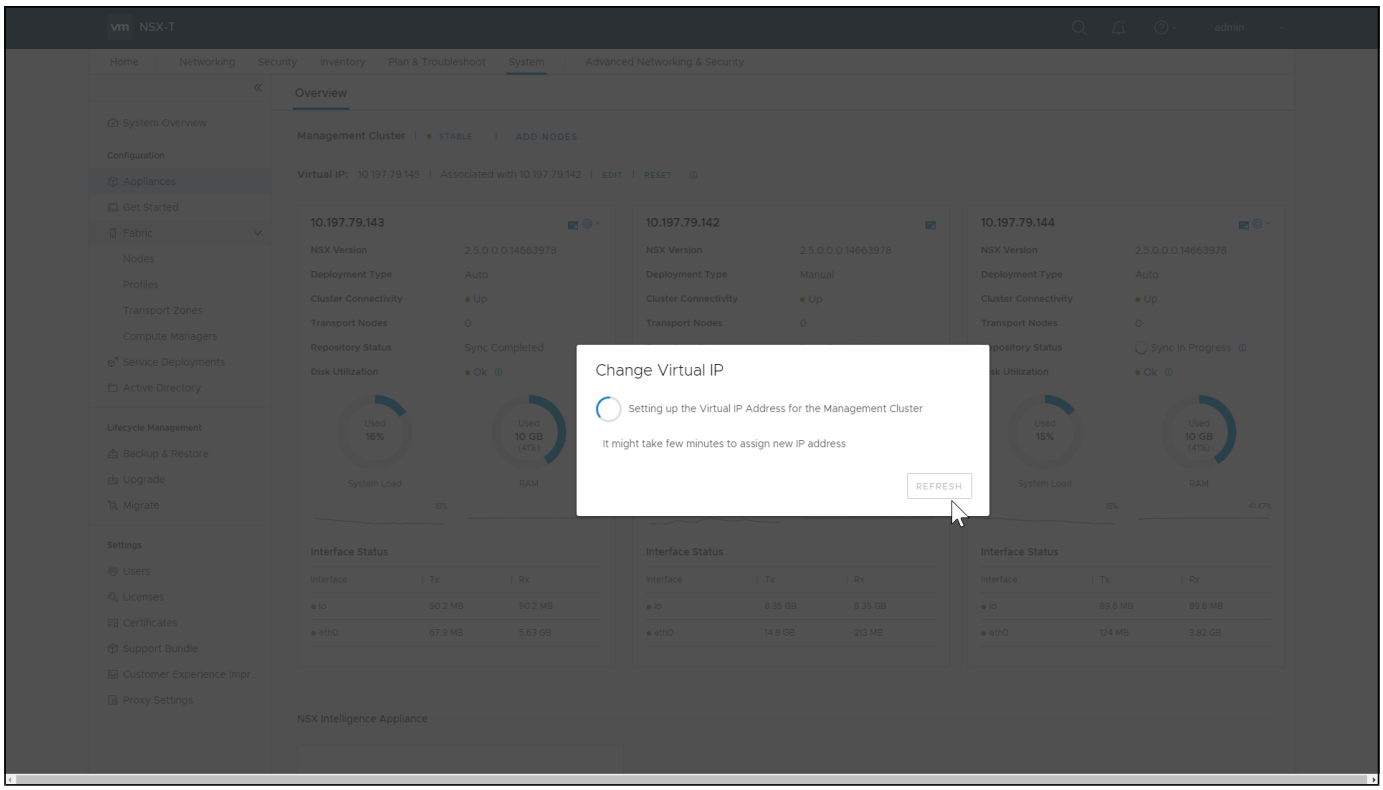
1. From your browser, log in with admin privileges to NSX Manager at `https://NSX-MANAGER-IP-ADDRESS`.
2. Go to **System > Overview**.
3. Click **Edit** next to the Virtual IP field.



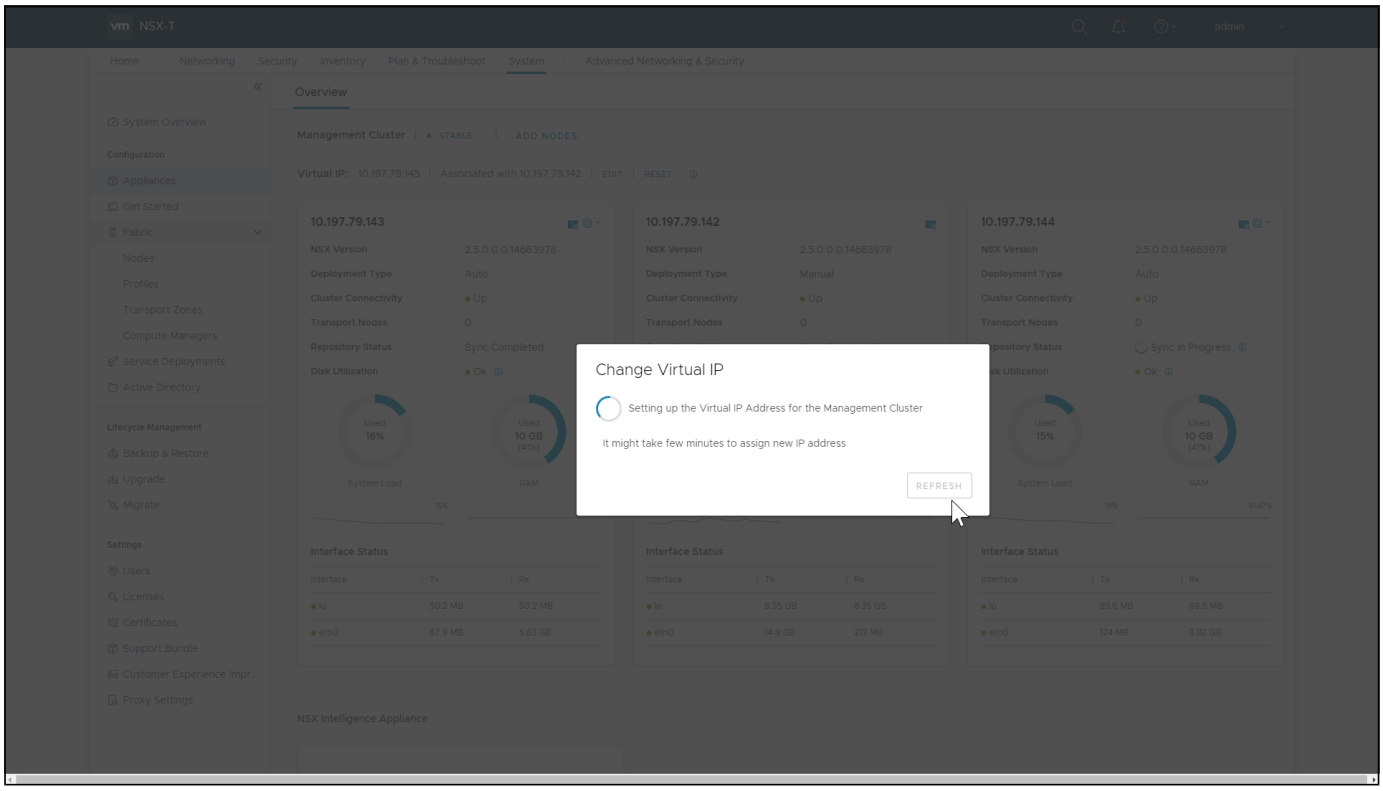
4. Enter a VIP for the cluster, such as `172.16.11.81`. Ensure that the VIP is part of the same subnet as the other NSX Management nodes.



5. Click **Save**.



6. When prompted, click **Refresh**. In the **Virtual IP** field, you should see the VIP address in use as well as the NSX Manager leader node that the VIP is associated with.



After the VIP is configured, you use it to connect to the NSX-T Manager. The VIP is associated with a single NSX-T Manager. If that node goes down, the VIP attaches to another node. If necessary, you can still connect to any NSX-T manager directly using its own IP address.

Verify the VIP Address

To verify the VIP and troubleshoot issues, complete the following steps:

1. From your browser, log in with admin privileges to the NSX-T Management Cluster using the VIP address `https://NSX-MANAGER-VIP-ADDRESS`.
2. To check the API leader of the management cluster, enter the `get cluster status verbose` command in the NSX Manager CLI.
3. To troubleshoot VIP issues, verify the following logs:
   - Reverse Proxy logs at `/var/log/proxy/reverse-proxy.log`
   - Cluster manager logs at `/var/log/cbm/cbm.log`

Next Step

See Create TZ_OVERLAY and TZ_VLAN.

Installation Instructions Home

See Installing and Configuring NSX-T v2.5 for Enterprise PKS.

Create NSX-T Transport Zones

In this topic

Page last updated:

This topic provides instructions for creating transport zones for use with VMware Enterprise PKS.

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

About Transport Zones

In NSX-T Data Center, a transport zone (TZ) is a logical construct that controls which hosts a logical switch can reach. A transport zone defines a collection of hosts that can communicate with each other across the network. This communication happens over one or more interfaces defined as Virtual Tunnel Endpoints (VTEPs).

There are two types of transport zones: **Overlay** and **VLAN**. For Enterprise PKS, the instructions in this topic use both.

- The Overlay transport zone is used by ESXi host transport nodes and NSX-T Edge Nodes. When an ESXi host or NSX-T Edge transport node is added to an Overlay transport zone, an N-VDS is installed on the ESXi host or NSX Edge Node.
- The VLAN transport zone is used by NSX-T Edge Nodes and ESXi host transport nodes for VLAN uplinks. When an NSX-T Edge Node is added to a VLAN transport zone, a VLAN N-VDS is installed on the NSX-T Edge Node.

For more information, see Transport Zones in the NSX-T Data Center documentation.

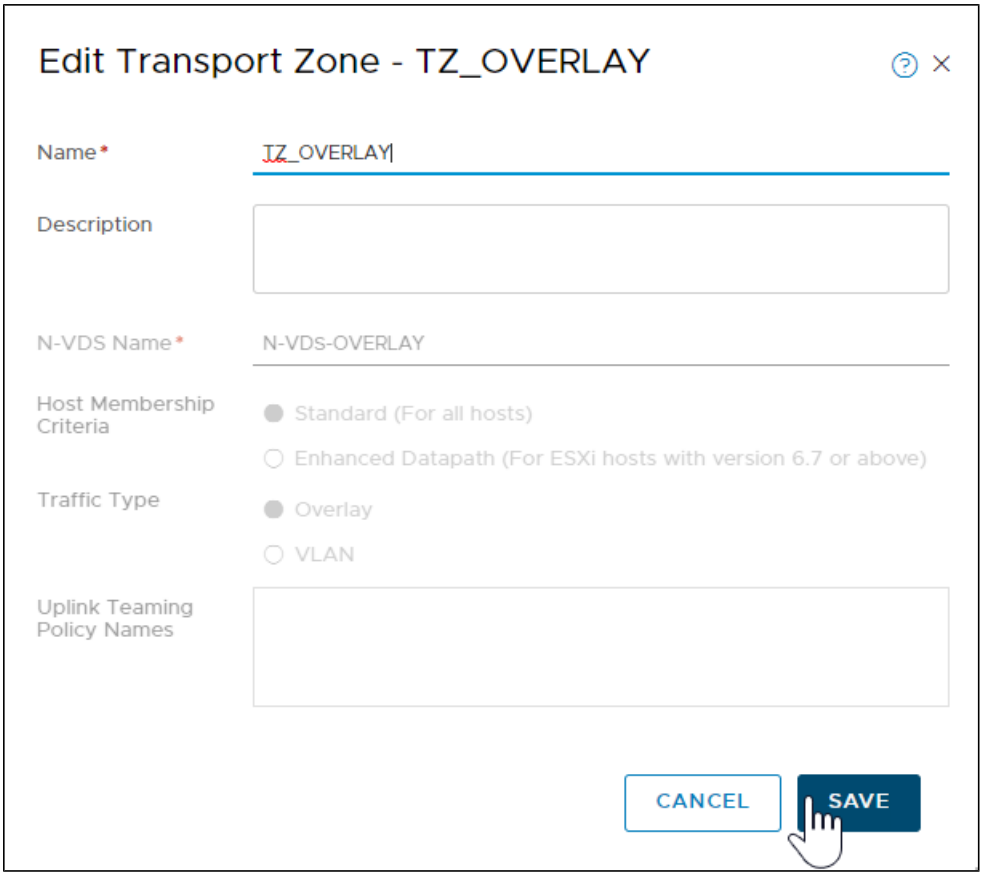
Create Overlay Transport Zone

Create an Overlay Transport Zone (`TZ-OVERLAY`) for PKS control plane services and Kubernetes clusters overlay networks associated with VDS `hostswitch1`:

1. In NSX Manager, select **System** > **Fabric** > **Transport Zones** > **Add**.
2. Enter a **Name** for the transport zone, such as `TZ-OVERLAY`.
3. Enter a **N-VDS Name**, such as `N-VDS-OVERLAY`.
4. Select a **Host Membership Criteria** (N-VDS mode): Standard or Enhanced Datapath.

> **Note:** In enhanced mode, only specific NIC configurations are supported (ESXi hosts v6.7+), and you must ensure that you configure the supported NICs. Refer to the NSX-T Data Center documentation for more information.

5. For **Traffic Type**, select **Overlay**.
6. (Optional) Enter one or more uplink teaming policy names.
7. Click **Add**.



Create VLAN Transport Zone

Create the VLAN Transport Zone (`TZ-VLAN`) for NSX Edge Node uplinks (ingress/egress) for Kubernetes clusters associated with VDS `hostswitch2` :

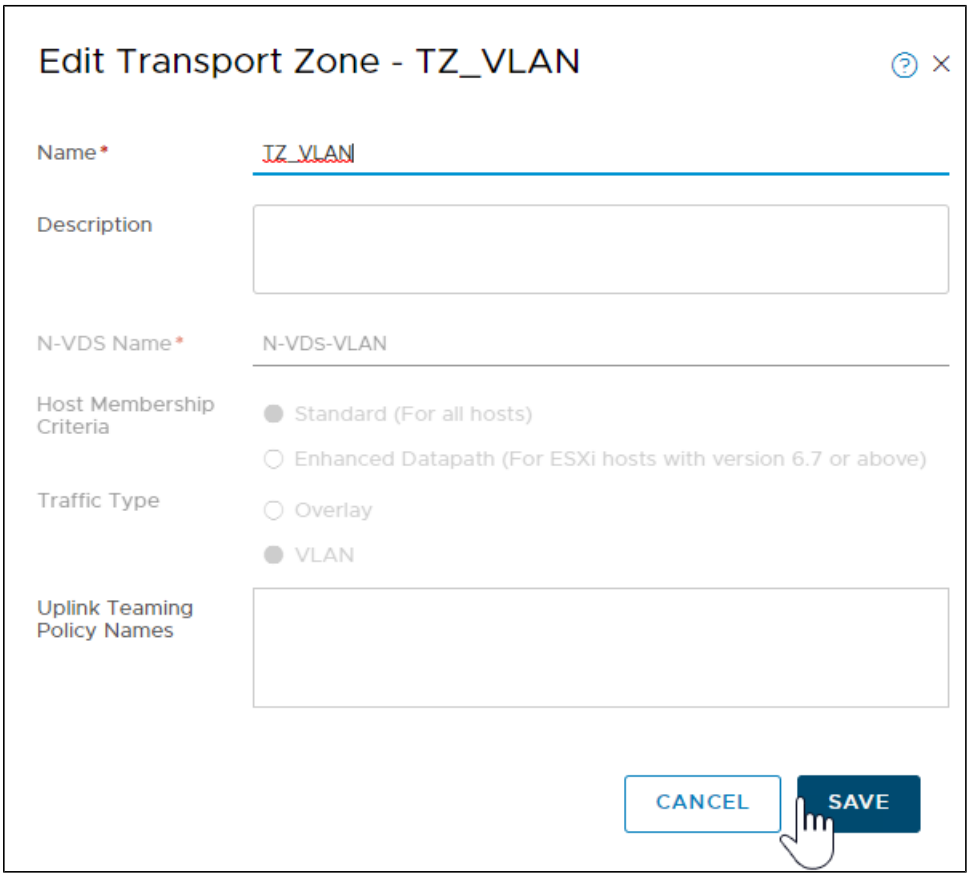
1. In NSX Manager, select **System** > **Fabric** > **Transport Zones** > **Add**.
2. Enter a **Name** for the transport zone, such as `TZ-VLAN` .
3. Enter a **N-VDS Name**, such as `N-VDS-VLAN` .
4. Select a **Host Membership Criteria** (N-VDS mode): Standard or Enhanced Datapath.

> **Note:** In enhanced mode, only specific NIC configurations are supported (ESXi hosts v6.7+), and you must ensure that you configure the supported NICs. Refer to the NSX-T Data Center documentation for more information.

5. For **Traffic Type**, select **VLAN**.

6. (Optional) Enter one or more uplink teaming policy names.

7. Click **Add**.

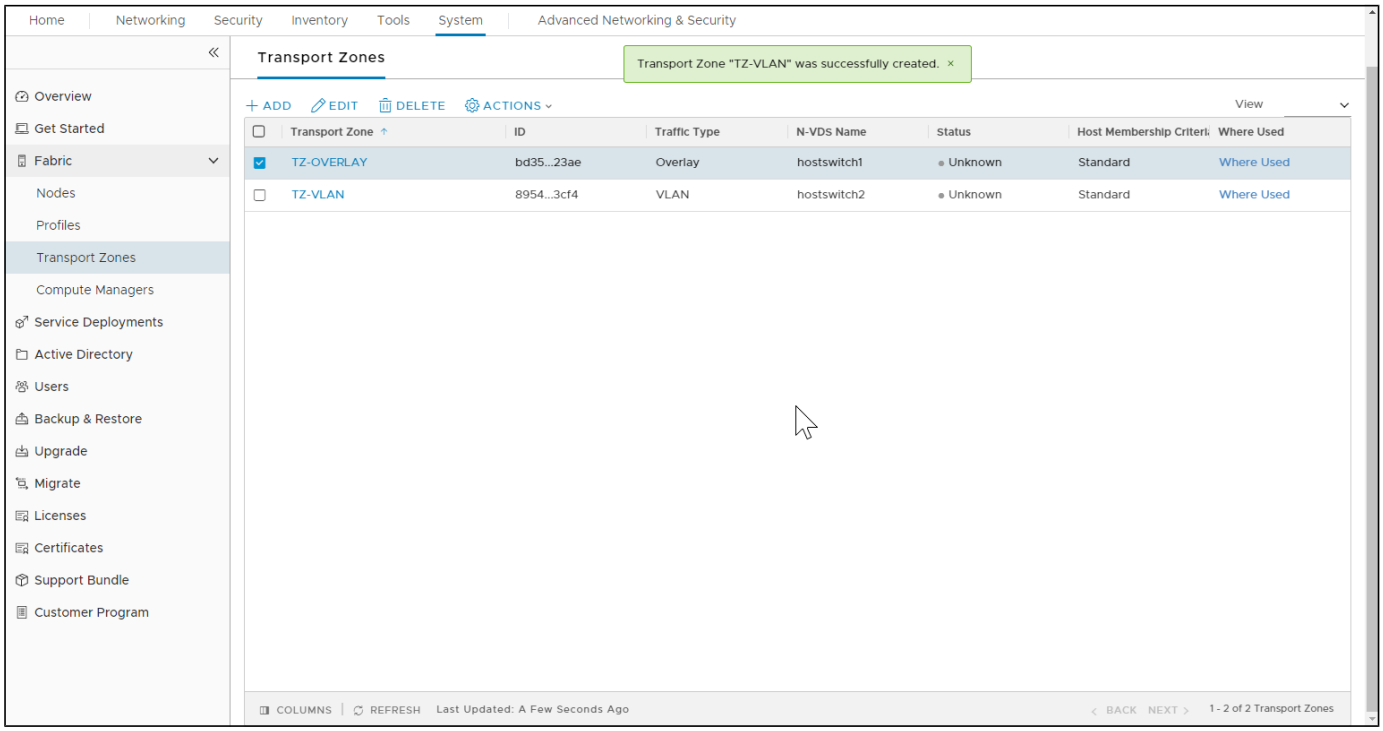


Verify Transport Zone Creation

To verify transport zone creation:

1. In NSX-T Manager, select **System** > **Fabric** > **Transport Zones**.

2. Verify that you see the TZ-OVERLAY and TZ-VLAN transport zones you created:



View a larger version of this image.

Next Step

Create DVS Port Group for Edge Nodes

Installation Instructions Home

See Installing and Configuring NSX-T v2.5 for Enterprise PKS.

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

Create DVS Port Group for NSX-T Edge Nodes

In this topic

Page last updated:

This topic provides instructions for creating a distributed virtual switch (DVS) port group for use with VMware Enterprise PKS.

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

About vSphere Distributed Switch and DVS Port Group

A vSphere distributed switch acts as a single switch across all associated hosts in a data center to provide centralized provisioning, administration, and monitoring of virtual networks. You configure a vSphere distributed switch on the vCenter Server system, and the configuration is propagated to all hosts that are associated with the switch. This lets virtual machines maintain consistent network configuration as they migrate across multiple hosts.

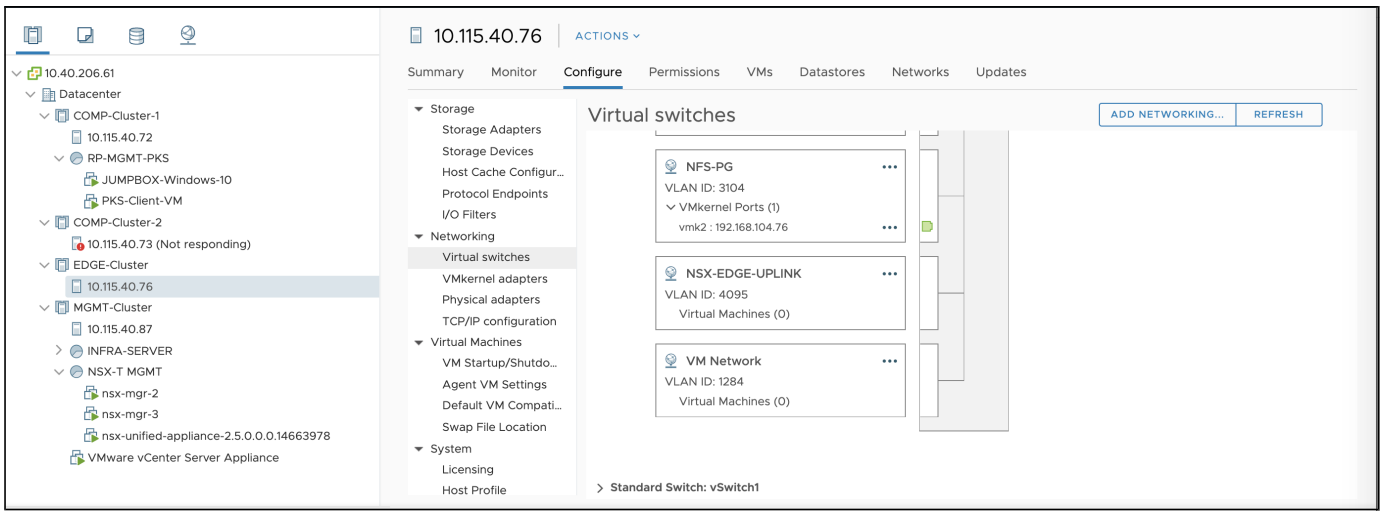
A distributed port group is a port group associated with a vSphere distributed switch that specifies port configuration options for each member port. Distributed port groups define how a connection is made through the vSphere distributed switch to the network.

For more information, see Networking Concepts in the VMware vSphere documentation.

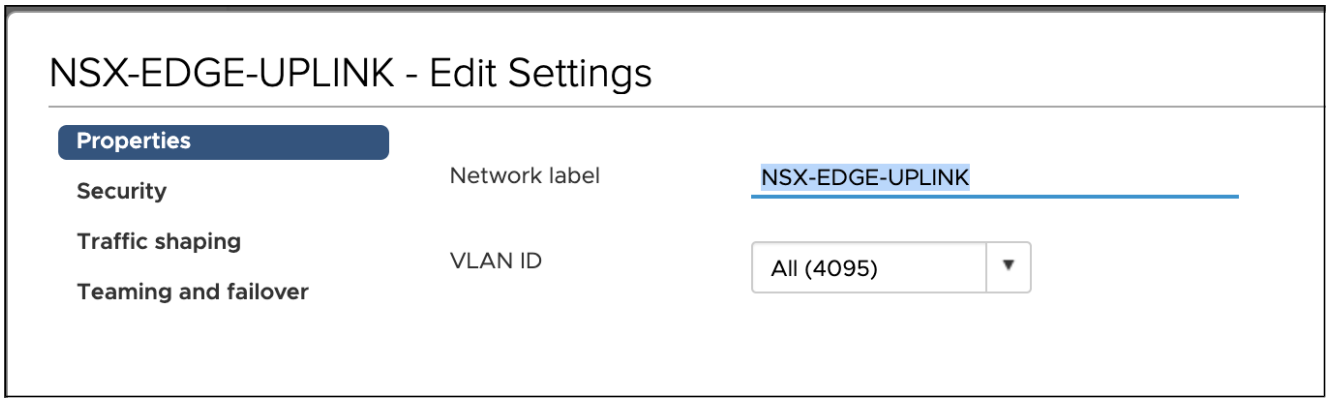
Create DVS Port Groups for Edge Nodes

To create DVS port groups:

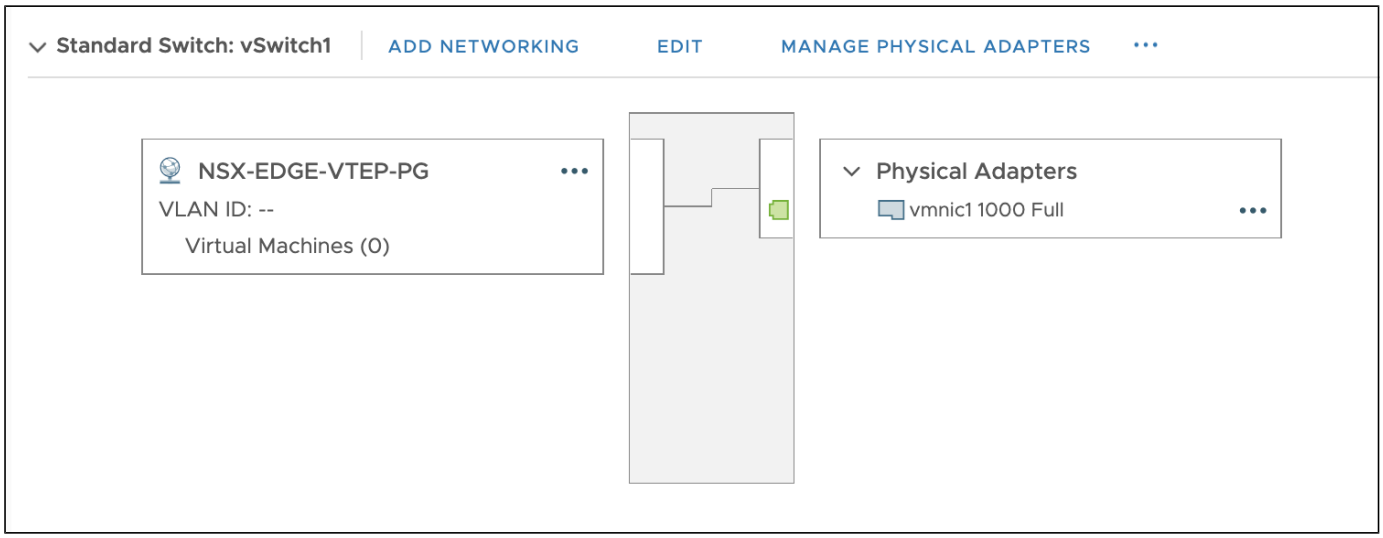
1. Log in to vCenter.
2. Select the vSwitch0 for the Edge Cluster.



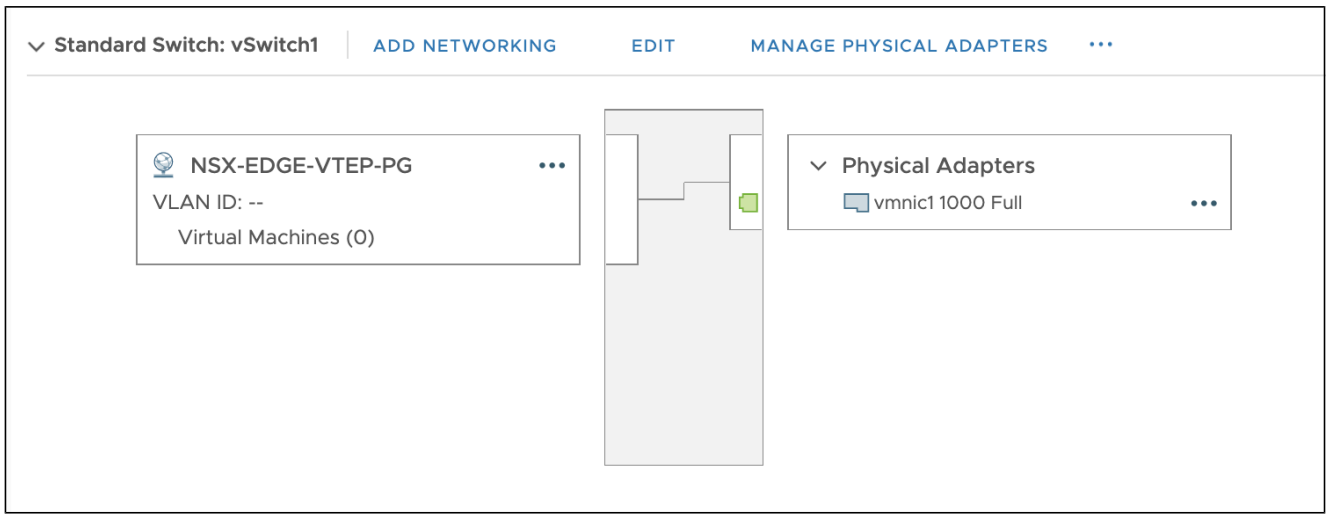
3. For vSwitch0, create an uplink port group.
   - Port Group: **NSX-EDGE-UPLINK**.
   - VLAN ID: **All (4095)**. It is a TRUNK.

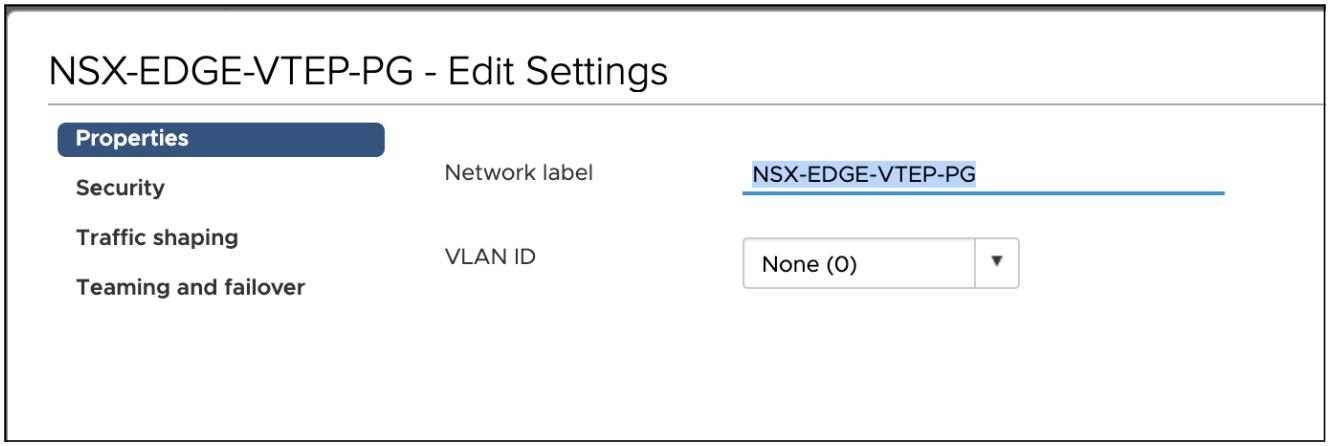


4. Select the vSwitch1 for the Edge Cluster.

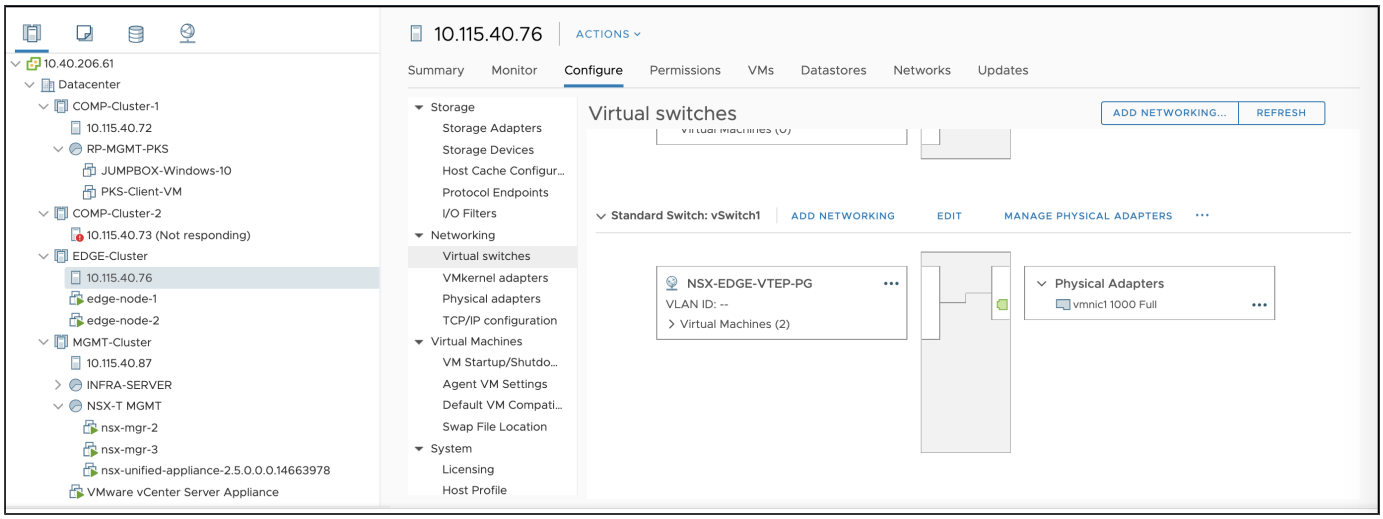


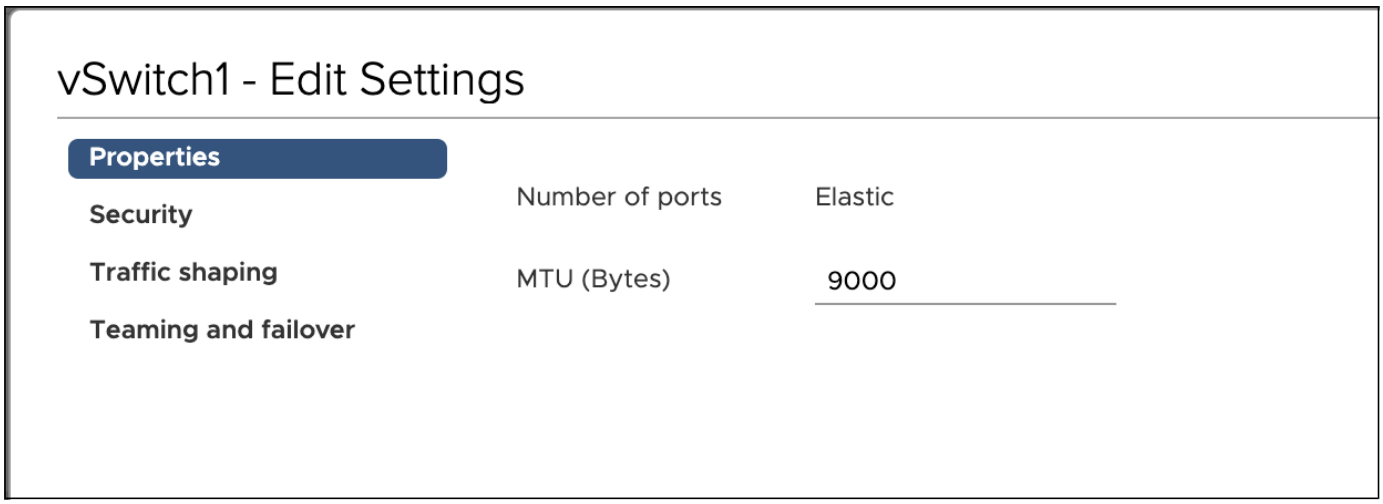
5. For vSwitch1, create a port group for VTEP.
   - Port Group: **NSX-EDGE-VTEP-PG**.
   - VLAN ID: None.





6. Set the MTU to 9000 bytes on the virtual switch that hosts the port group for the Edge Node VTEP. In this example, the VTEP virtual switch is vSwitch1.





Next Step

Create Uplink Logical Switch for the Tier-0 Router.

Installation Instructions Home

See Installing and Configuring NSX-T v2.5 for Enterprise PKS.

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

Create Uplink Logical Switch for the Tier-0 Router

In this topic

Page last updated:

This topic provides instructions for creating an NSX-T Tier-0 Logical Router for use with VMware Enterprise PKS.

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

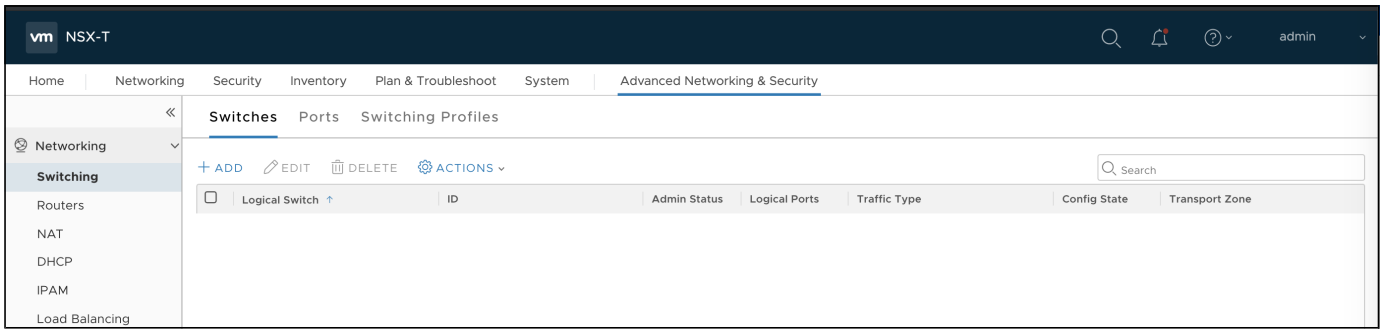
About Tier-0 Logical Routers

In NSX-T Data Center, a Tier-0 logical router provides a gateway service between the logical and physical network. For additional information, see Tier-0 Logical Router in the NSX-T documentation.

Create Uplink Logical Switch (LS)

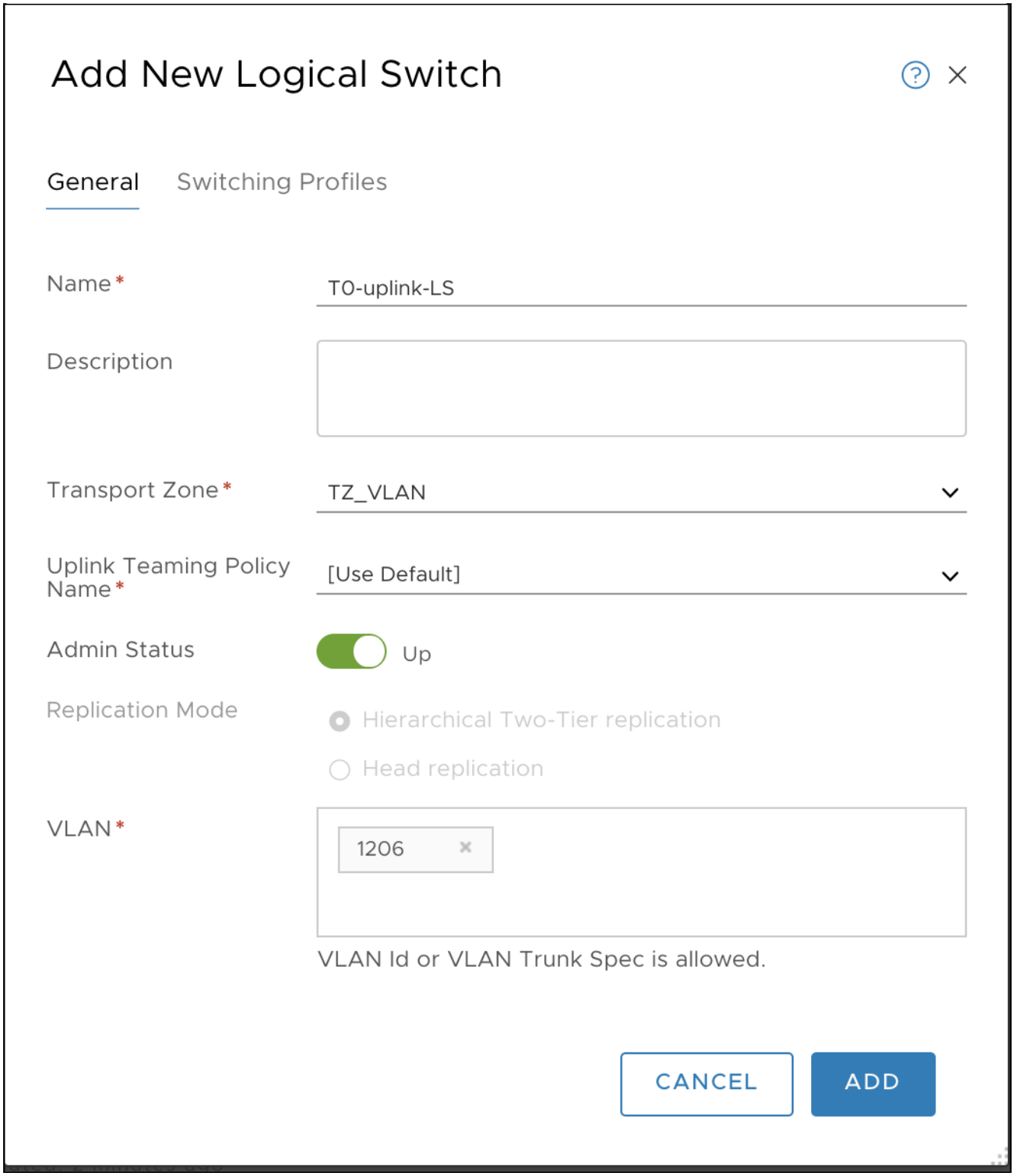
To create a new logical switch:

1. From your browser, log in with admin privileges to NSX Manager at `https://NSX-MANAGER-IP-ADDRESS`.
2. In NSX Manager, go to **Advanced Networking & Security** > **Networking** > **Switching**.

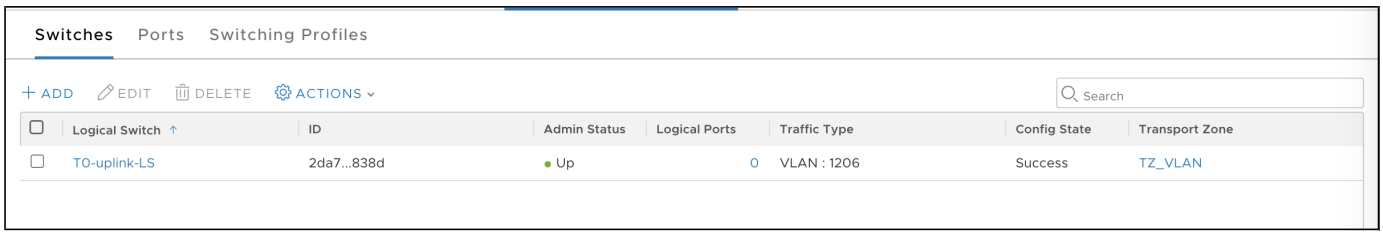


3. Click **Add** to add a new logical switch.
4. Configure the logical switch as follows:
   - **Name**: For example, `T0-uplink-LS`.
   - **Transport Zone**: `TZ-VLAN`.
   - **Uplink Teaming Policy Name**: Use Default.

- **Admin Status**: Up.
- **VLAN**: For example, `1206`. The VLAN ID should match the ID for the PKS Management VLAN.



5. Click **Add** to save the configuration.



Next Step

Create a Tunnel Endpoint IP Pool.

Installation Instructions Home

See Installing and Configuring NSX-T v2.5 for Enterprise PKS.

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

Create TEP IP Pool

In this topic

Page last updated:

This topic provides instructions for creating an IP pool for tunnel endpoints (TEPs).

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

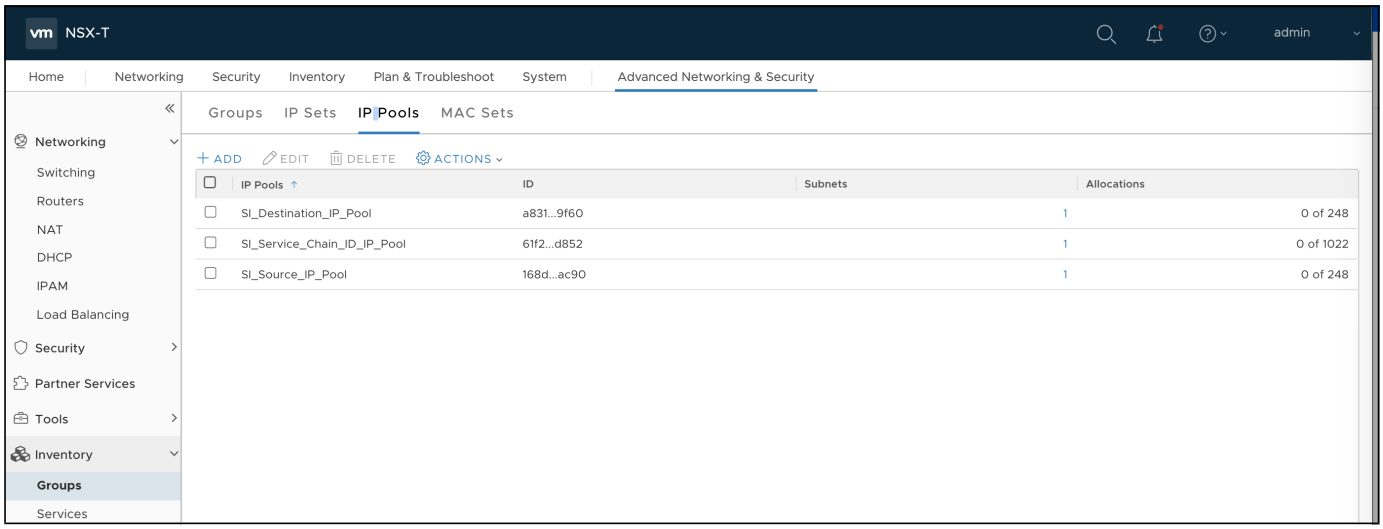
About TEPs

Tunnel endpoints (TEPs) are the source and destination IP addresses used in the external IP header to identify the ESXi hosts that originate and end the NSX-T encapsulation of overlay frames. The TEP addresses do not need to be routable for VMware Enterprise PKS, so you can use any random IP addressing scheme you want. For more information, see Tunnel Endpoint IP Addresses in the NSX-T Data Center documentation.

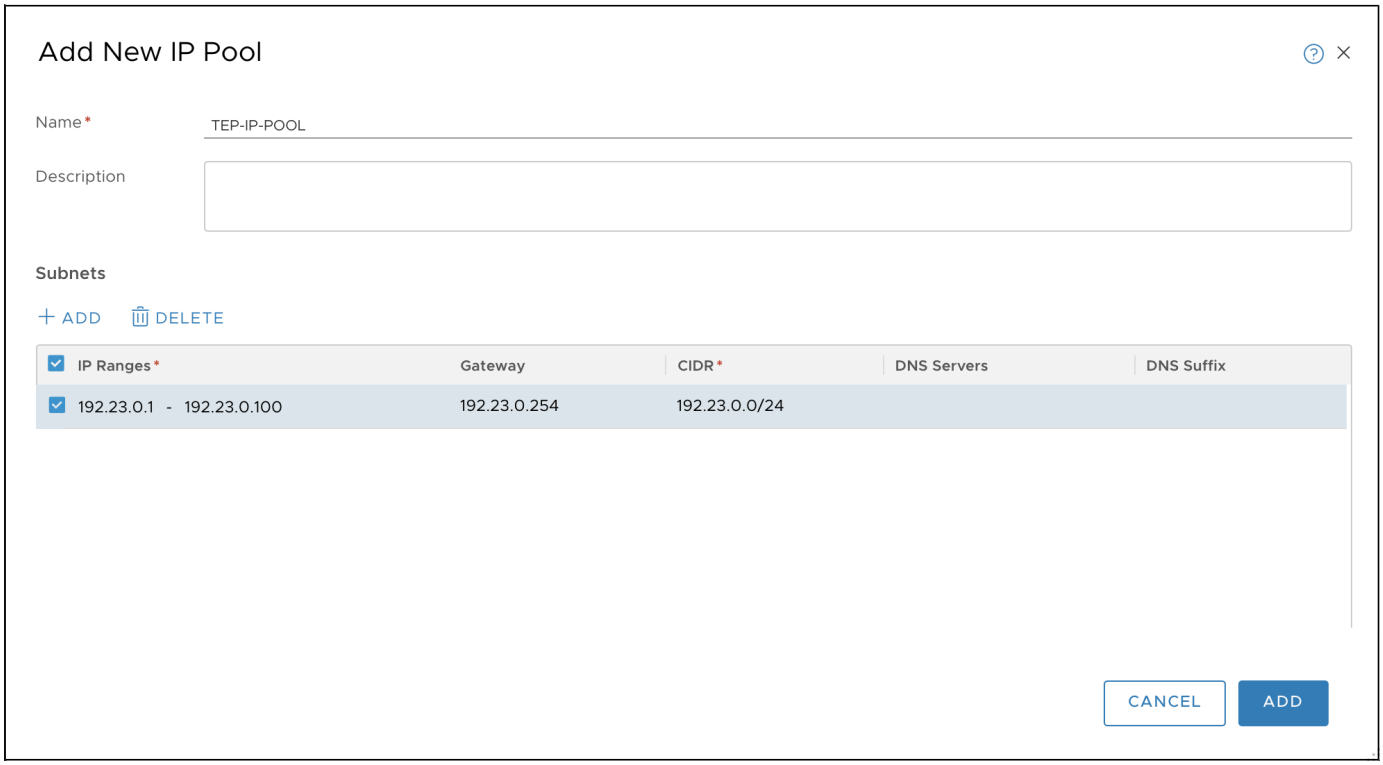
Create TEP IP Pool

To create the TEP IP Pool, complete the following procedure:

1. From your browser, log in with admin privileges to NSX Manager at https://NSX-MANAGER-IP-ADDRESS.
2. In NSX Manager, select **Advanced Networking & Security** > **Inventory** > **Groups** > **IP Pools**.



3. Select **Add**.

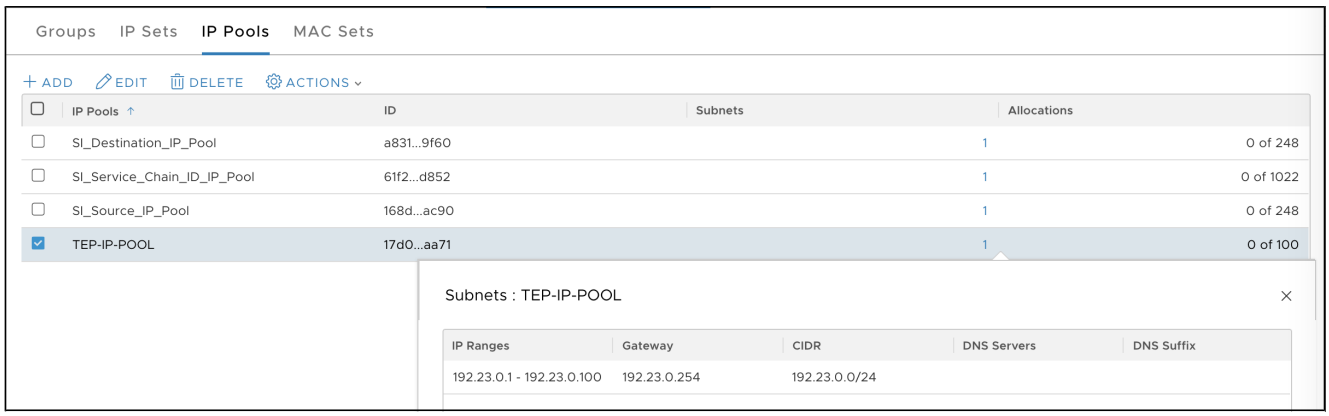


4. Configure the IP pool as follows:

   - **Name**: For example, `TEP-IP-POOL`.
   - **Description**. Optional.
   - **IP Ranges**: Enter the IP allocation ranges. For example, `192.23.0.1 - 192.23.0.100`.

     > **Note**: TEP addresses do not need to be publicly routable.

   - **Gateway**: For example, `192.23.0.254`.
   - **CIDR**: Enter the Network address. For example, `192.23.0.0/24`.
   - **DNS Servers** (optional): Comma-separated list of DNS servers.
   - **DNS Suffix** (optional): Such as `corp.local`.



Verify TEP IP Pool Creation

To verify TEP IP Pool configuration, complete the following steps:

1. In NSX Manager, select **Advanced Networking & Security** > **Inventory** > **Groups** > **IP Pools**.
2. Verify that the TEP IP Pool you created is present.

Next Step

Deploying NSX Edge Nodes for Enterprise PKS.

Installation Instructions Home

See Installing and Configuring NSX-T v2.5 for Enterprise PKS.

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

Deploying NSX Edge Nodes for Enterprise PKS

In this topic

Page last updated:

This topic provides instructions for installing NSX Edge Node VMs on vSphere for use with VMware Enterprise PKS.

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

About Deploying NSX-T Edge Nodes for Enterprise PKS

NSX Edge Nodes provide the bridge between the virtual network environment implemented using NSX-T and the physical network. Edge Nodes for Enterprise PKS run load balancers for PKS API traffic, Kubernetes load balancer services, and ingress controllers. See Load Balancers in Enterprise PKS for more information.

In NSX-T, a load balancer is deployed on the Edge Nodes as a virtual server. The following virtual servers are required for Enterprise PKS:

- 1 TCP Layer 4 virtual server for each Kubernetes service of type: `LoadBalancer`
- 2 Layer 7 global virtual servers for Kubernetes pod ingress resources (HTTP and HTTPS)
- 1 global virtual server for the PKS API

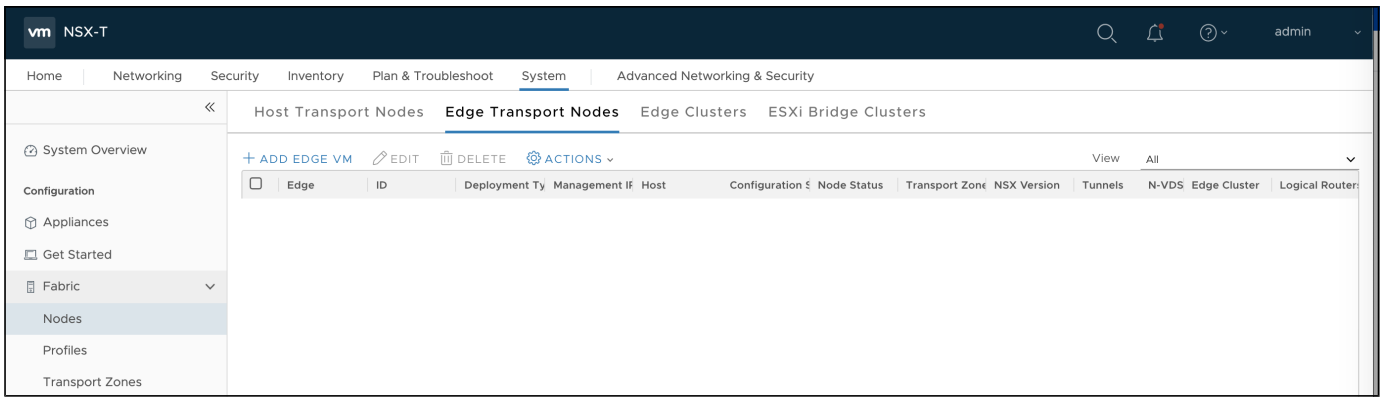
Enterprise PKS supports the `medium` and `large` VM Edge Node form factor, as well as the bare metal Edge Node. The default size of the load balancer deployed by NSX-T for a Kubernetes cluster is `small`. The size of the load balancer can be customized using Network Profiles.

See Scaling Load Balancer Resources in the NSX-T Data Center documentation for more information.

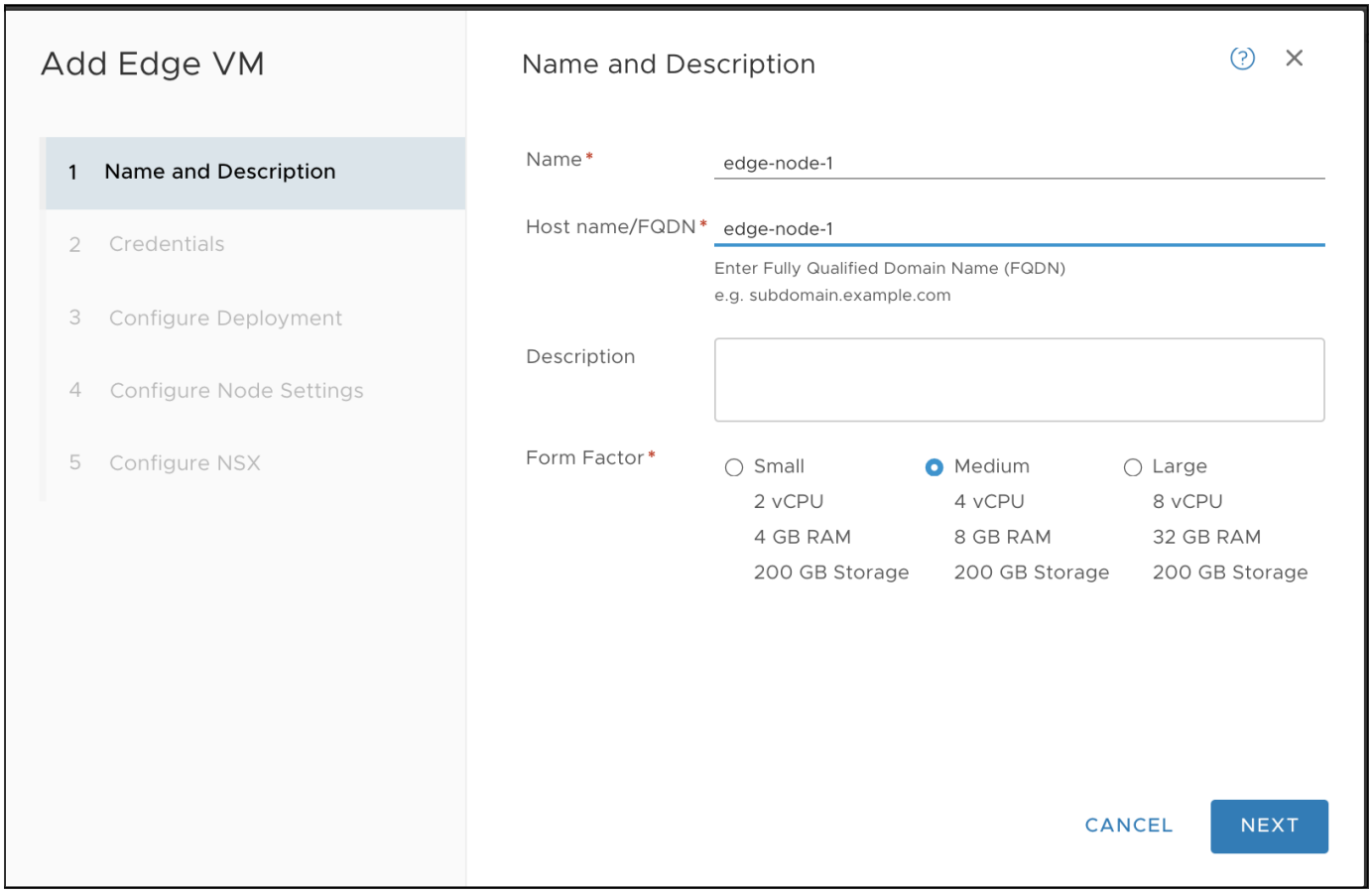
Deploy Edge Node 1

To deploy Edge Node 1 VM using the NSX-T Manager interface:

1. From your browser, log in with admin privileges to NSX Manager at `https://NSX-MANAGER-IP-ADDRESS`.
2. In NSX Manager, go to **System** > **Fabric** > **Nodes** > **Edge Transport Nodes**.

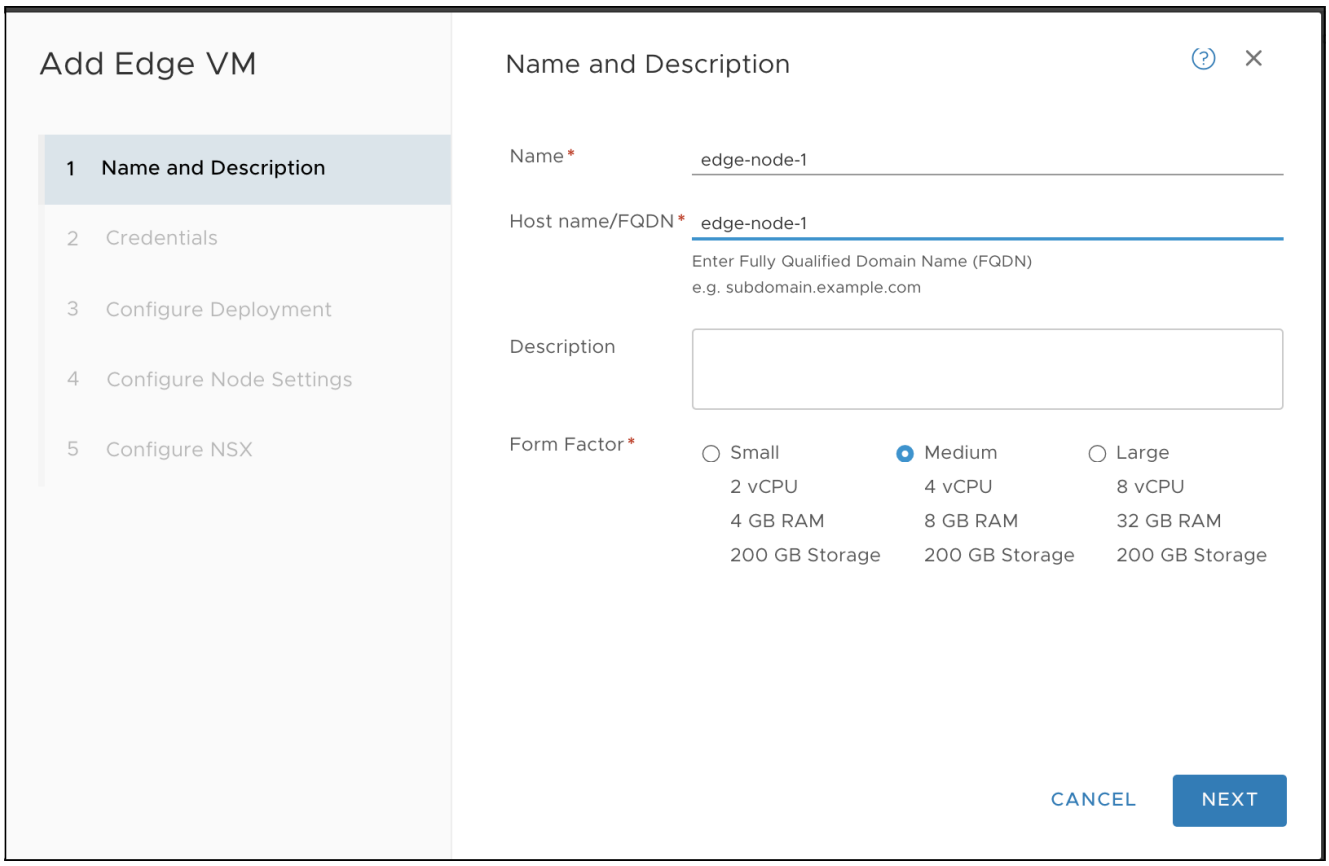


3. Click **Add Edge VM**.



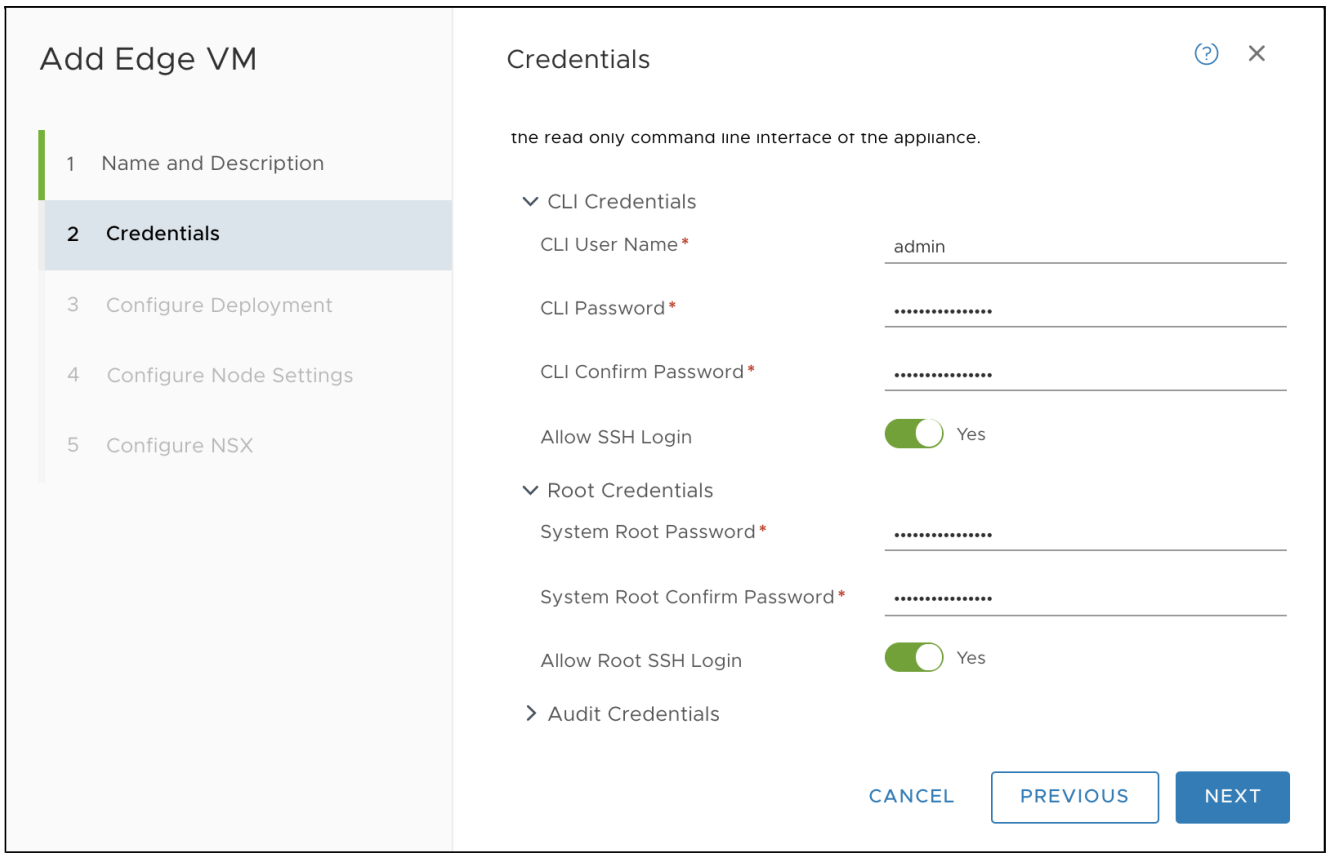
4. Configure the Edge VM as follows:

   - **Name**: `edge-node-1`
   - **FQDN**: `edge-node-1`
   - **Form Factor**: **Medium**

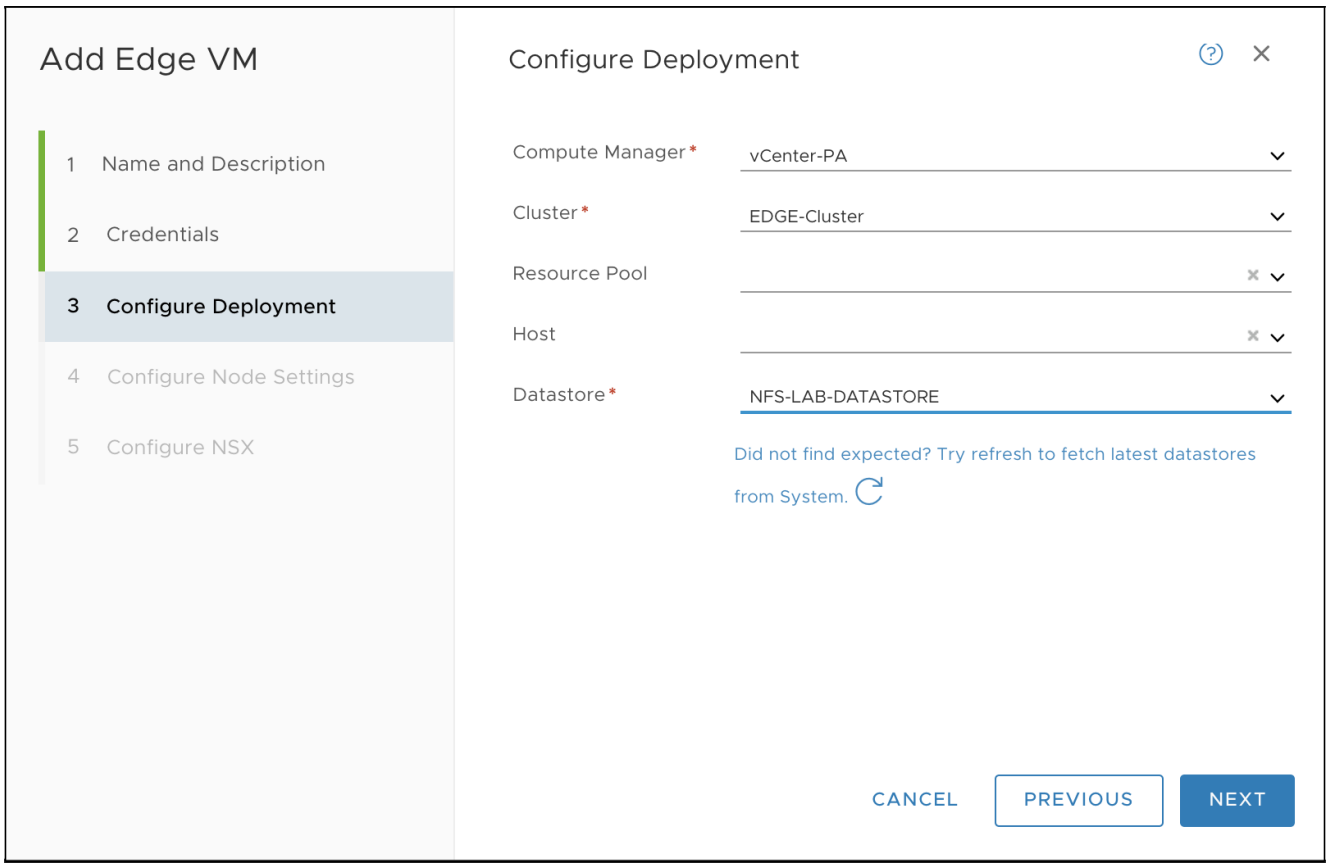


5. Configure **Credentials** as follows:

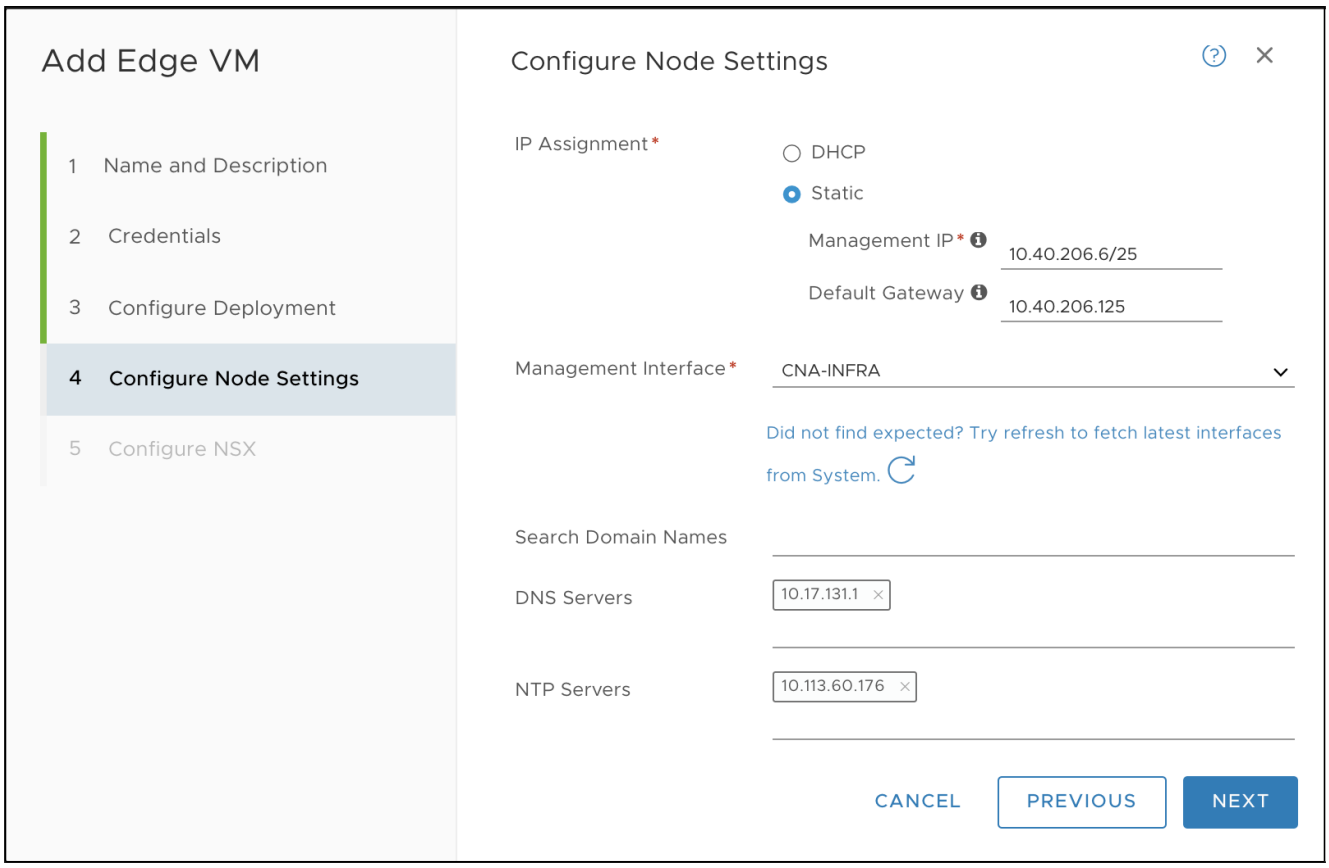
   - **Enable SSH Login**: Yes
   - **Enable Root SSH Login**: Yes
   - **Passwords**: Enter a password



6. Configure the deployment as follows:
   - **Compute Manager**: vCenter
   - **Cluster**: EDGE-Cluster
   - **Datastore**: VSAN Datastore

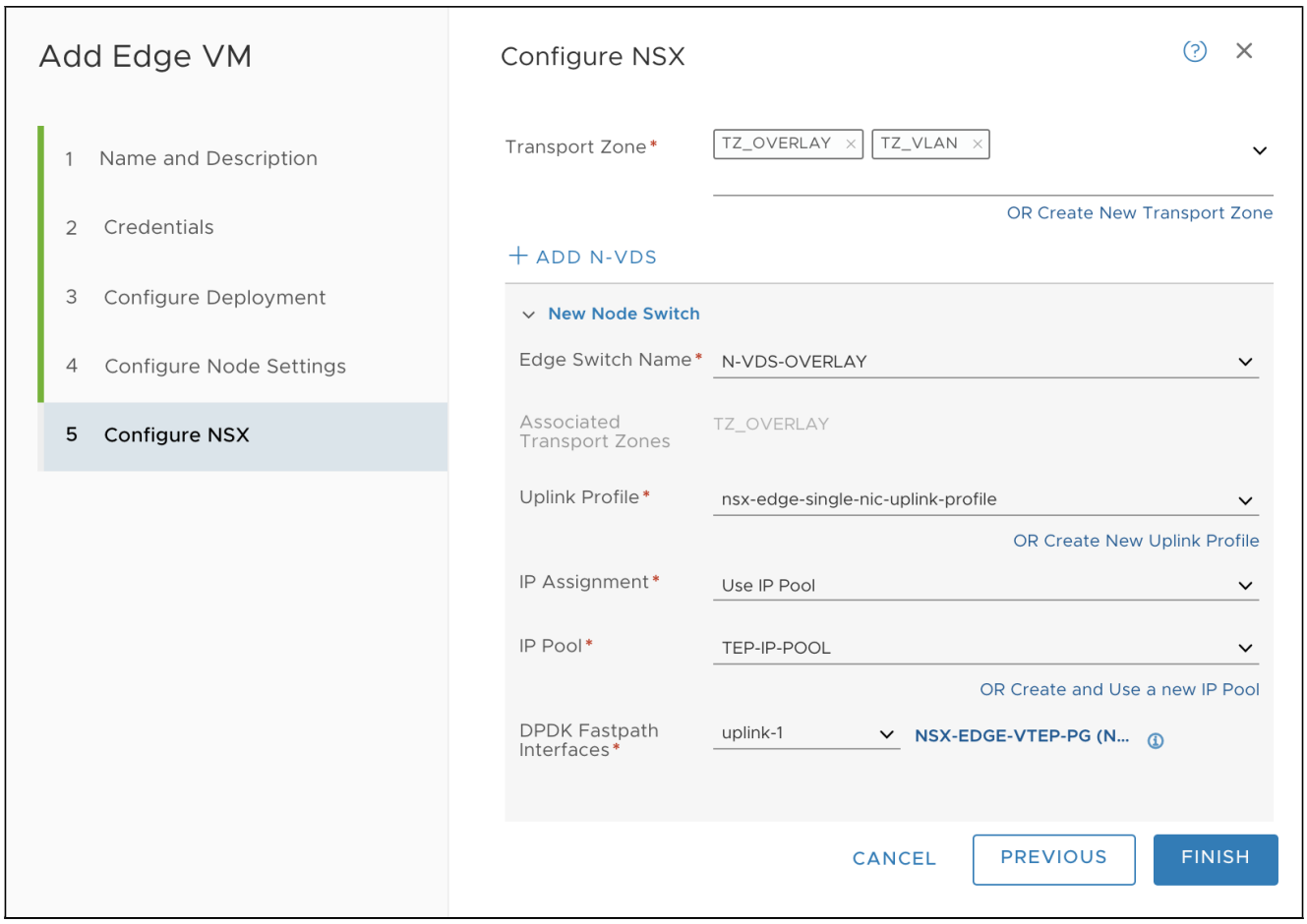


7. Configure the node settings as follows:
   - **Management IP**: 10.40.206.6/25
   - **Default GW**: 10.40.206.125
   - **Management Interface**: CNA-INFRA
   - **DNS Server**: 10.17.131.1
   - **NTP Server**: 10.113.60.176



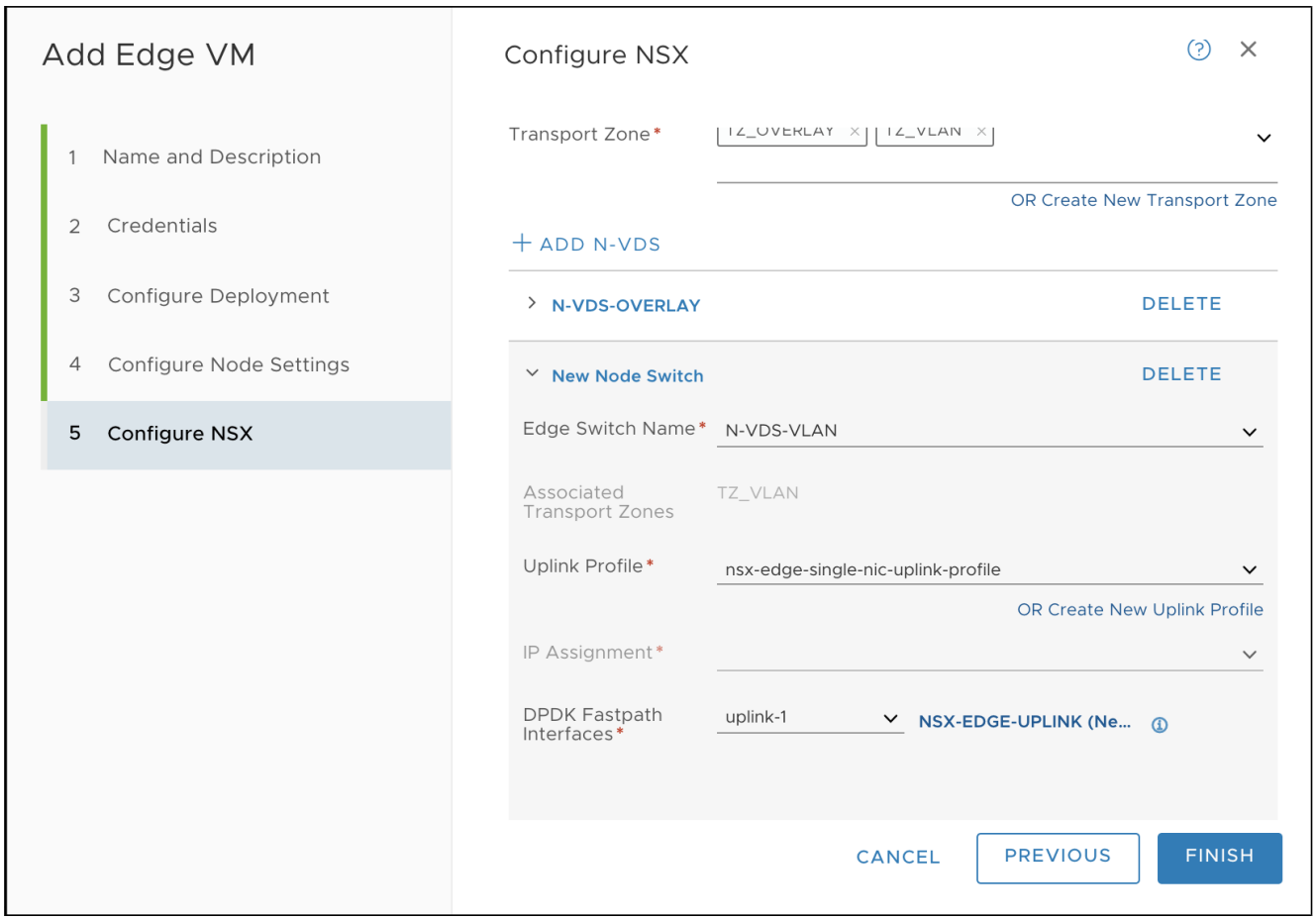
8. Configure NSX as follows:

   - **Transport Zone**: TZ_OVERLAY and TZ_VLAN
   - **Edge Switch Name**: N-VDS-OVERLAY
   - **Uplink Profile**: nsx-edge-single-nic-uplink-profile
   - **IP Assignment**: Use IP Pool
   - **IP Pool**: TEP-IP-POOL
   - **DPDK Fastpath Interfaces**: uplink-1 / NSX-EDGE-VTEP-PG

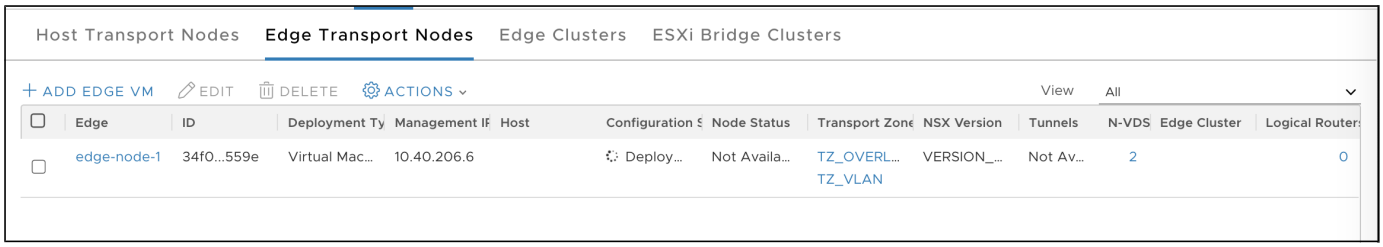


9. Click **Add N-VDS** and configure a second VDS for the Edge Node:

   - **Edge Switch Name**: N-VDS-VLAN
   - **Uplink Profile**: nsx-edge-single-nic-uplink-profile
   - **DPDK Fastpath Interfaces**: uplink-1 / NSX-EDGE-UPLINK

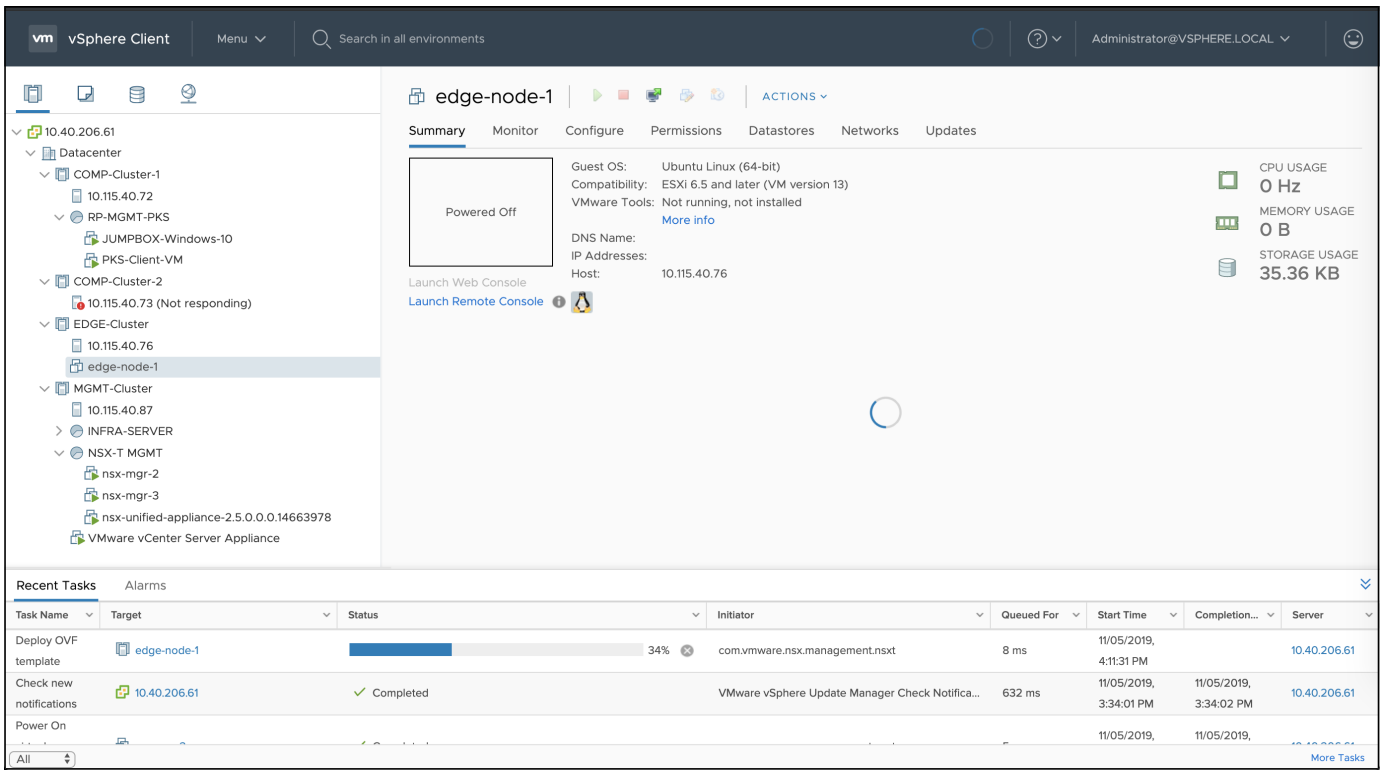


10. Click **Finish** to begin the installation.

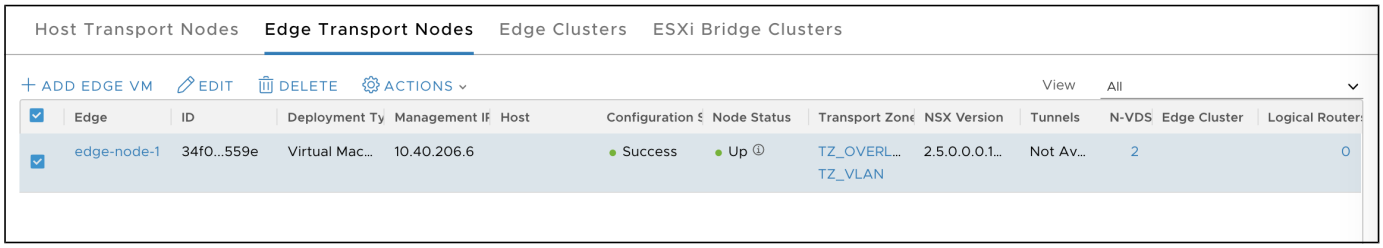


Verify NSX Edge Node 1 Installation

1. In vCenter, use the **Recent Tasks** panel at the bottom of the page to verify that you see the Edge Node 1 VM being deployed.



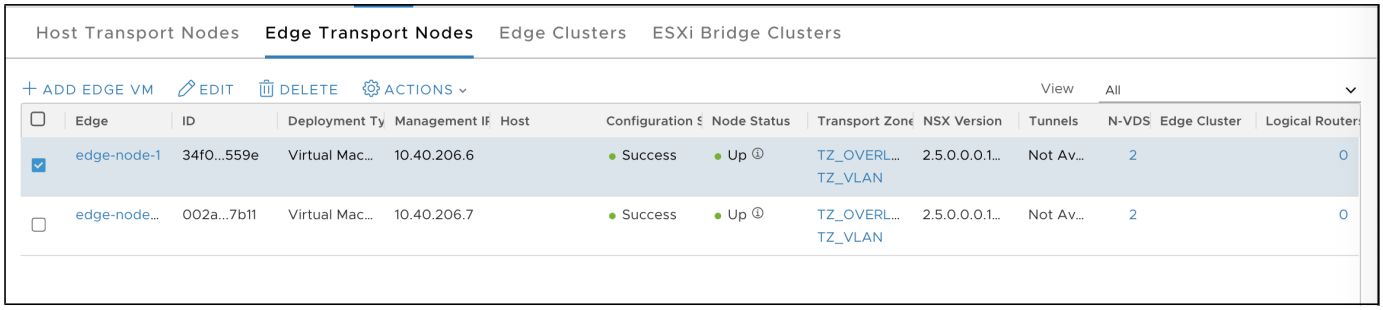
2. Once the process completes, you should see the Edge Node 1 deployed successfully in NSX Manager.



Deploy Edge Node 2

1. Repeat this process for `nsx-edge-2` and for each additional NSX Edge Node you intend to use for Enterprise PKS.

2. Once done, you should be able to see both Edge Nodes in NSX Manager.



Next Step

Create Edge Cluster.

Installation Instructions Home

See Installing and Configuring NSX-T v2.5 for Enterprise PKS.

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

Create Edge Cluster

In this topic

Page last updated:

This topic provides instructions for creating an NSX-T Edge Cluster for use with VMware Enterprise PKS.

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

About Edge Clusters

In NSX-T Data Center, an Edge Cluster is a logical boundary for resourcing. Having a multi-node cluster of NSX-T Edge Nodes helps ensure that at least one NSX Edge is always available. To create a Tier-0 logical router or a Tier-1 router with stateful services such as NAT, a traffic load balancer, and other virtual network objects, you must associate them with an NSX Edge cluster.

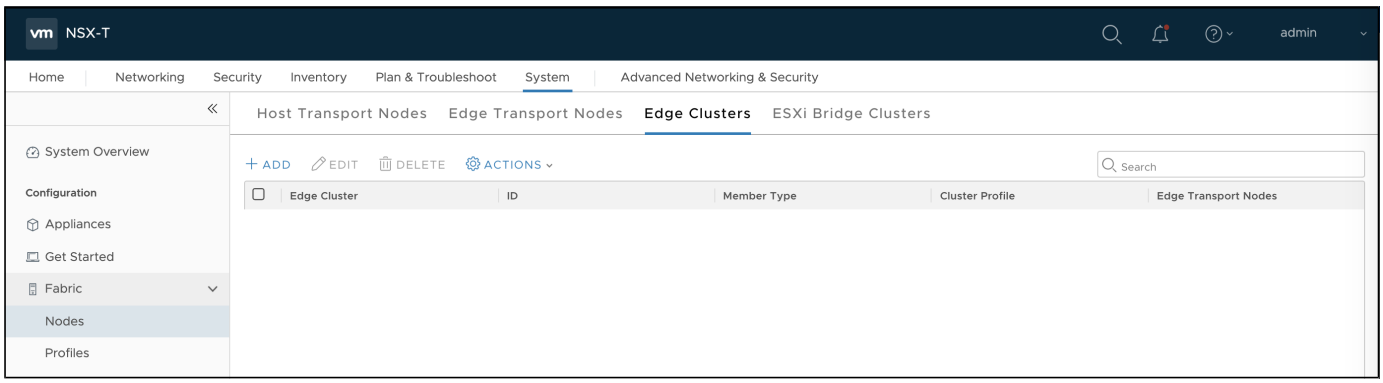
An Edge Cluster can have a maximum of 10 Edge Nodes. If the provisioning requires more Edge Nodes than what a single Edge Cluster can support, multiple Edge Clusters must be deployed. An NSX-T Edge Transport Node can be added to only one NSX-T Edge cluster. After creating the NSX-T Edge cluster, you can later edit it to add additional NSX-T Edge Nodes. An NSX-T Edge cluster can be used to back multiple logical routers.

For more information, see Edge Clusters in the NSX-T Data Center documentation.

Create Edge Cluster

Create an NSX Edge Cluster and add each Edge Transport Node to the Edge Cluster by completing the following procedure:

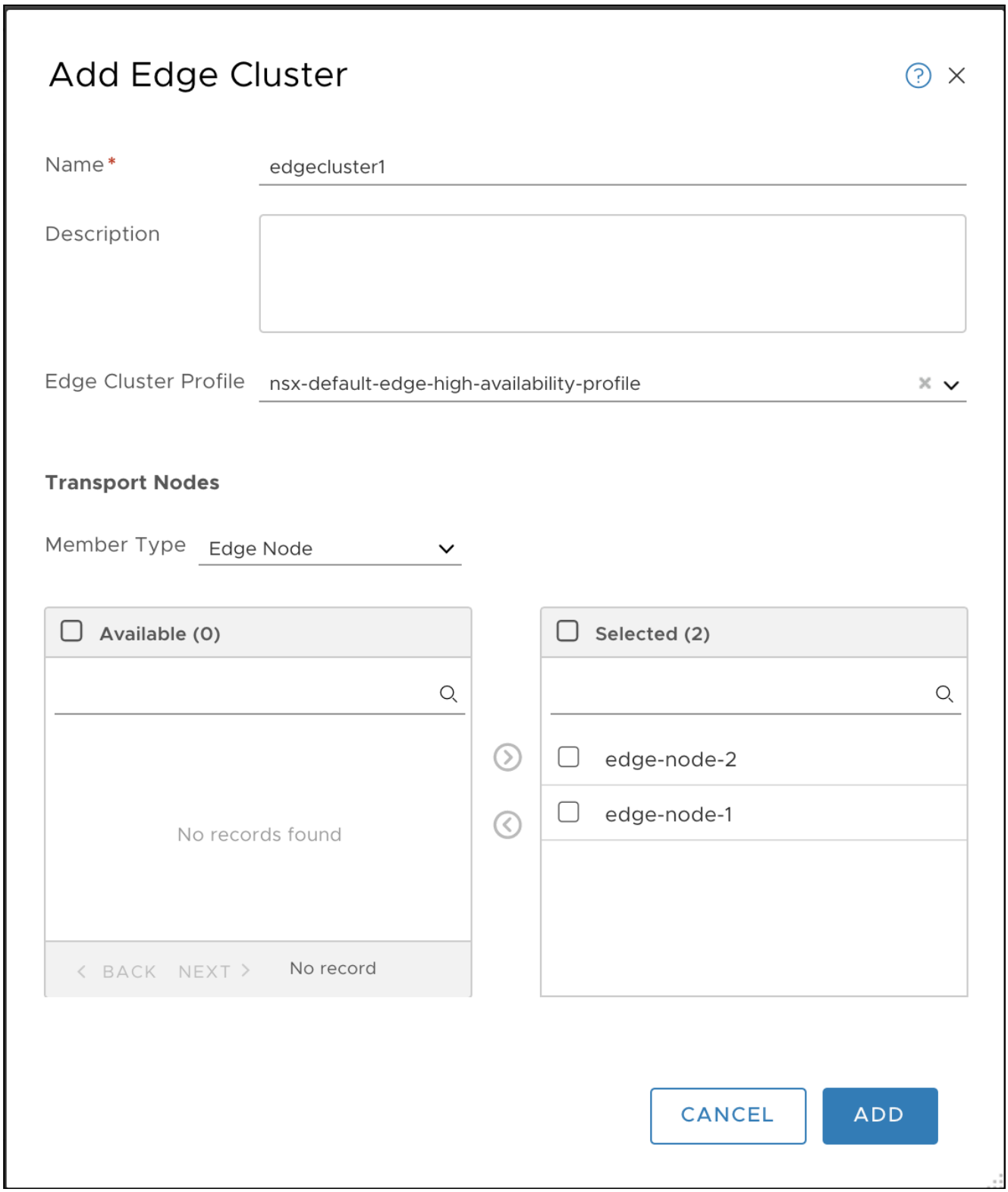
1. In NSX Manager, select **System** > **Fabric** > **Nodes** > **Edge Clusters**.



2. Click **Add**.

3. Configure the NSX Edge Cluster as follows:

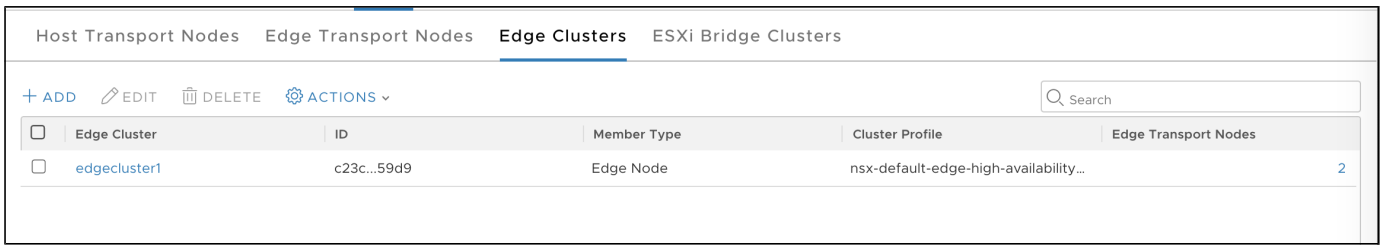
   - **Name**: For example, `edgecluster1`.
   - **Edge Cluster Profile**: Select `nsx-default-edge-high-availability-profile` from the menu.
   - **Transport Nodes > Member Type**: Select `Edge Node` from the menu.
   - From the **Available** column, select both NSX-T Edge Nodes and click the right-arrow to move them to the **Selected** column.
   - Click **Add**.



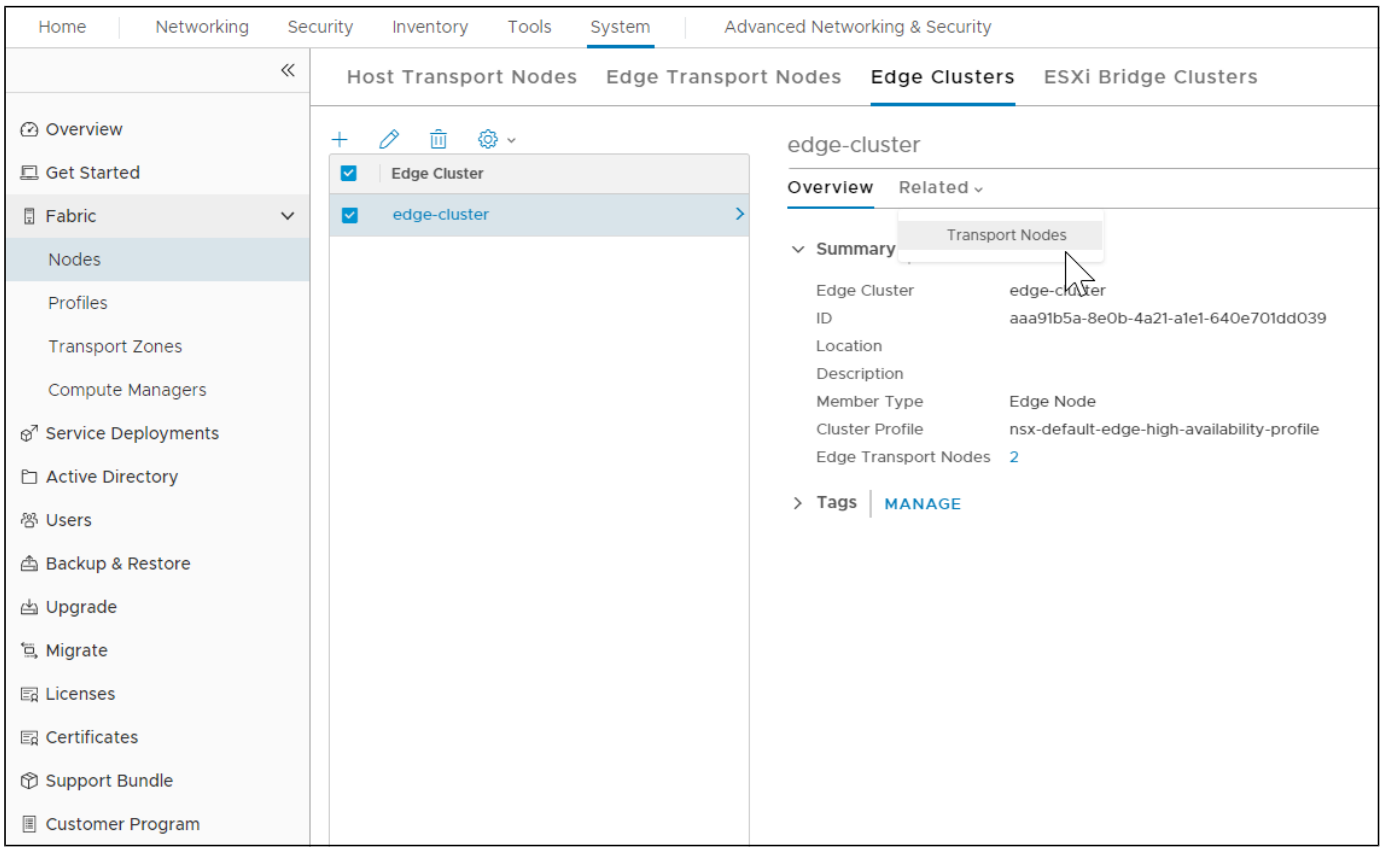
Verify Edge Cluster Creation

To verify Edge Cluster creation, complete the following steps:

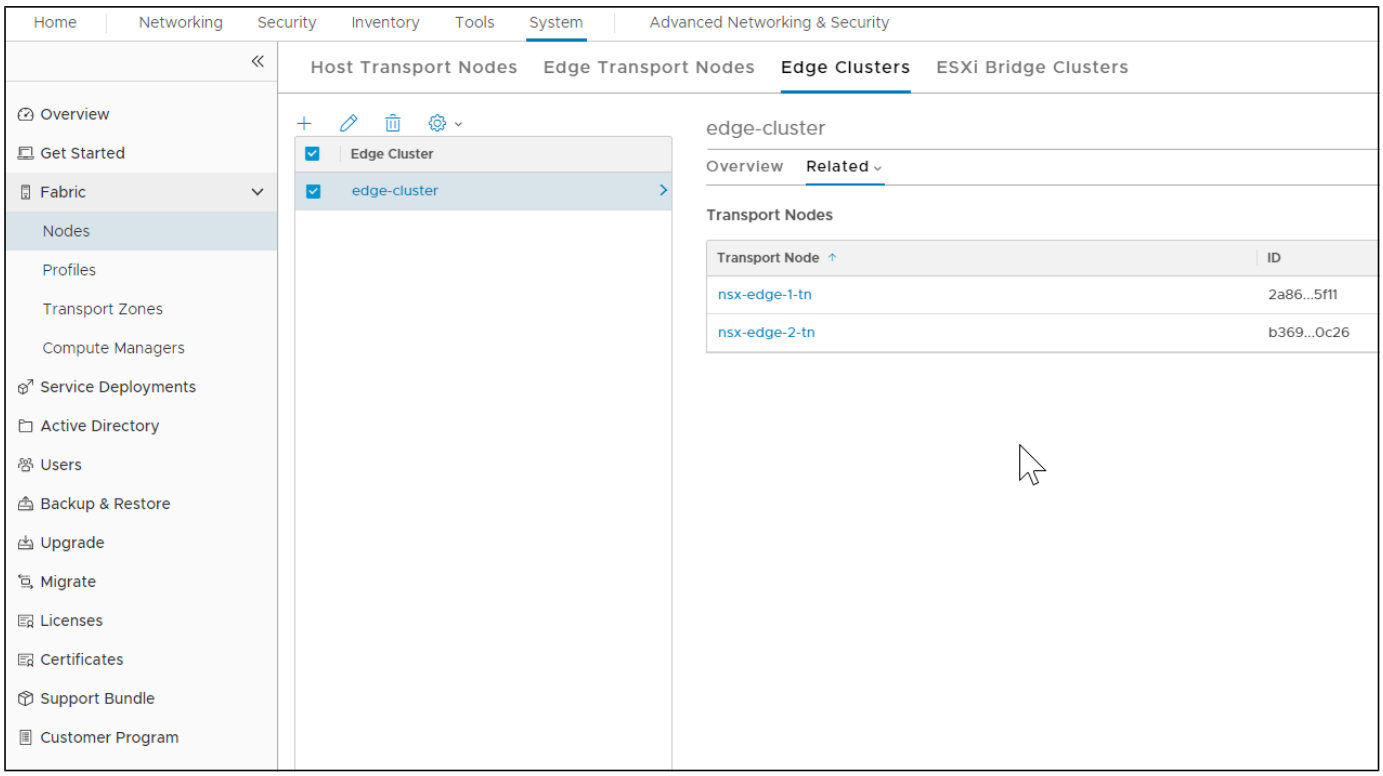
1. In NSX-T Manager, select **System** > **Fabric** > **Nodes** > **Edge Clusters**.
2. Verify that you see the new Edge Cluster.



3. Select **Edge Cluster** > **Related** > **Transport Nodes**.



4. Verify that both 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. SSH to NSX Edge Node 2 and repeat the above commands to verify proper connectivity.

7. Get the TEP IP addresses.

   - Navigate to **System** > **Fabric** > **Nodes** > **Edge Transport Nodes**.
   - Select the Edge Transport Node, such as `nsx-edge-1-tn`.
   - Select the **Monitor tab**.

8. 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 TEP-IP-ADDRESS-EDGE-2
```

You should be able to ping Edge Transport Node 2 using the TEP address.

Next Step

Prepare ESXi Hosts in the Compute Cluster as Transport Nodes

Installation Instructions Home

See Installing and Configuring NSX-T v2.5 for Enterprise PKS.

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

Prepare ESXi Hosts as NSX-T Transport Nodes

In this topic

Page last updated:

This topic provides instructions for preparing ESXi hosts as NSX transport nodes.

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

About ESXi Host Preparation

In NSX-T Data Center, a Transport Node allows nodes to exchange traffic for virtual networks. ESXi hosts dedicated to the Enterprise PKS Compute Cluster must be prepared as transport nodes.

For each ESXi host in the NSX-T Fabric to be used for Enterprise PKS-provisioned Kubernetes clusters, create an associated transport node. For example, if you have three ESXi hosts in the vCenter Cluster reserved for Enterprise PKS use, create three transport nodes. Add the Overlay Transport Zone to each ESXi Host Transport Node.

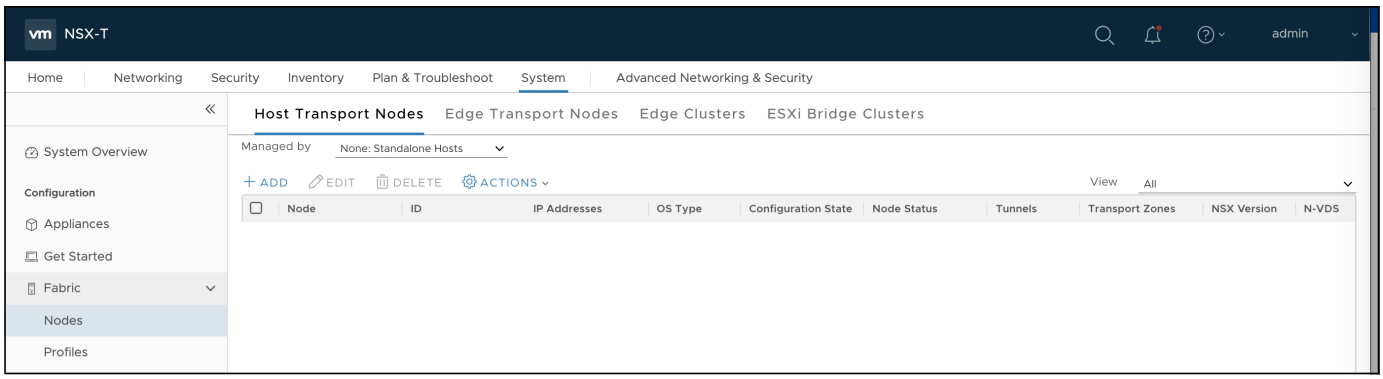
These instructions assume that for each participating ESXi host, the ESXi hypervisor is installed and the `vmk0` is configured. In addition, each ESXi host must have at least one free nic/vmnic for use with NSX-T Host Transport Nodes that is not in use by other vSwitches on the ESXi host. Make sure the `vmnic1` (second physical interface) of the ESXi host is not used. NSX-T will take ownership of it, and opaque NSX vSwitch will use it as uplink.

> **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.

Create Host Transport Nodes

Complete the following operation for each ESXi host to be used by the PKS Compute Cluster:

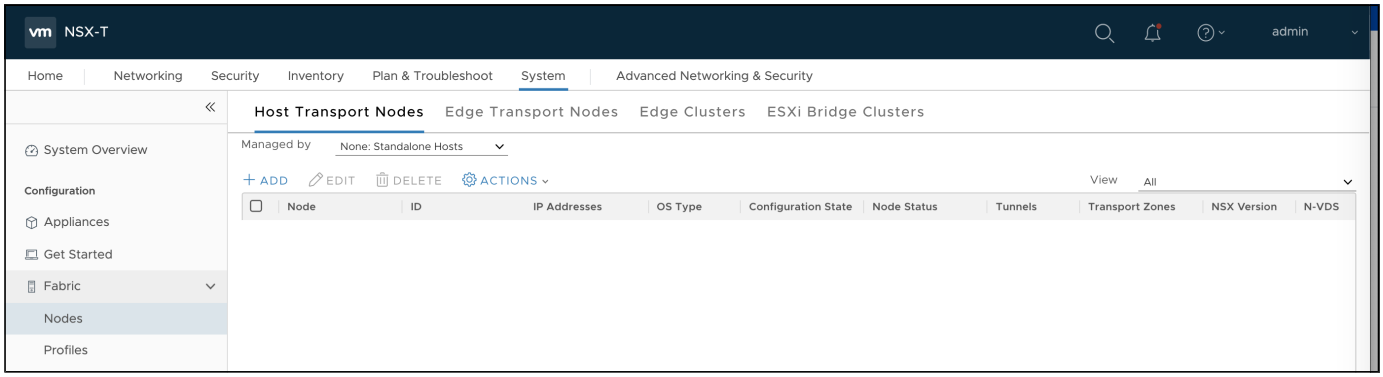
1. In NSX-T Manager, go to **System** > **Fabric** > **Nodes** > **Host Transport Nodes**.



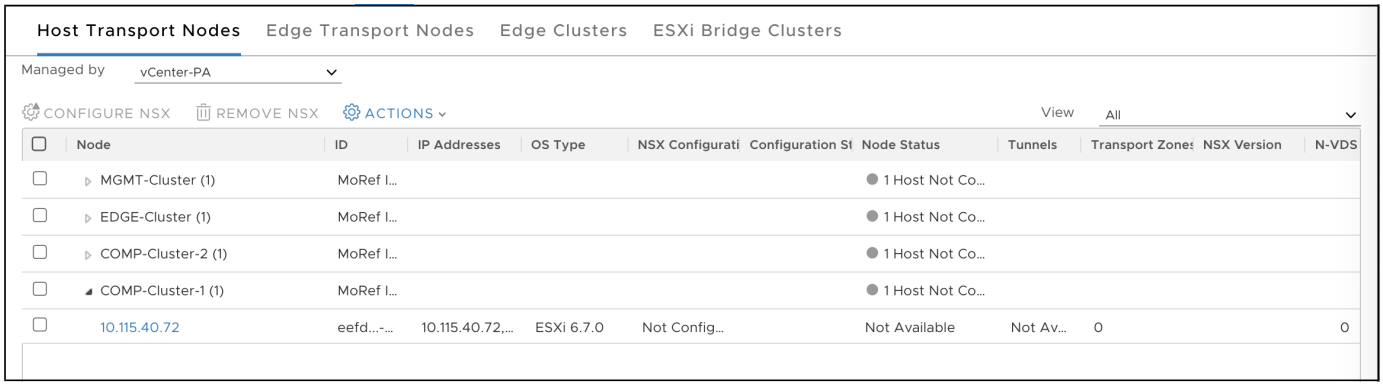
2. For the **Managed by** field, select **VMware vCenter Server App** from the dropdown menu.

> **Note**: This assumes that you configured vCenter as the Compute Manager.

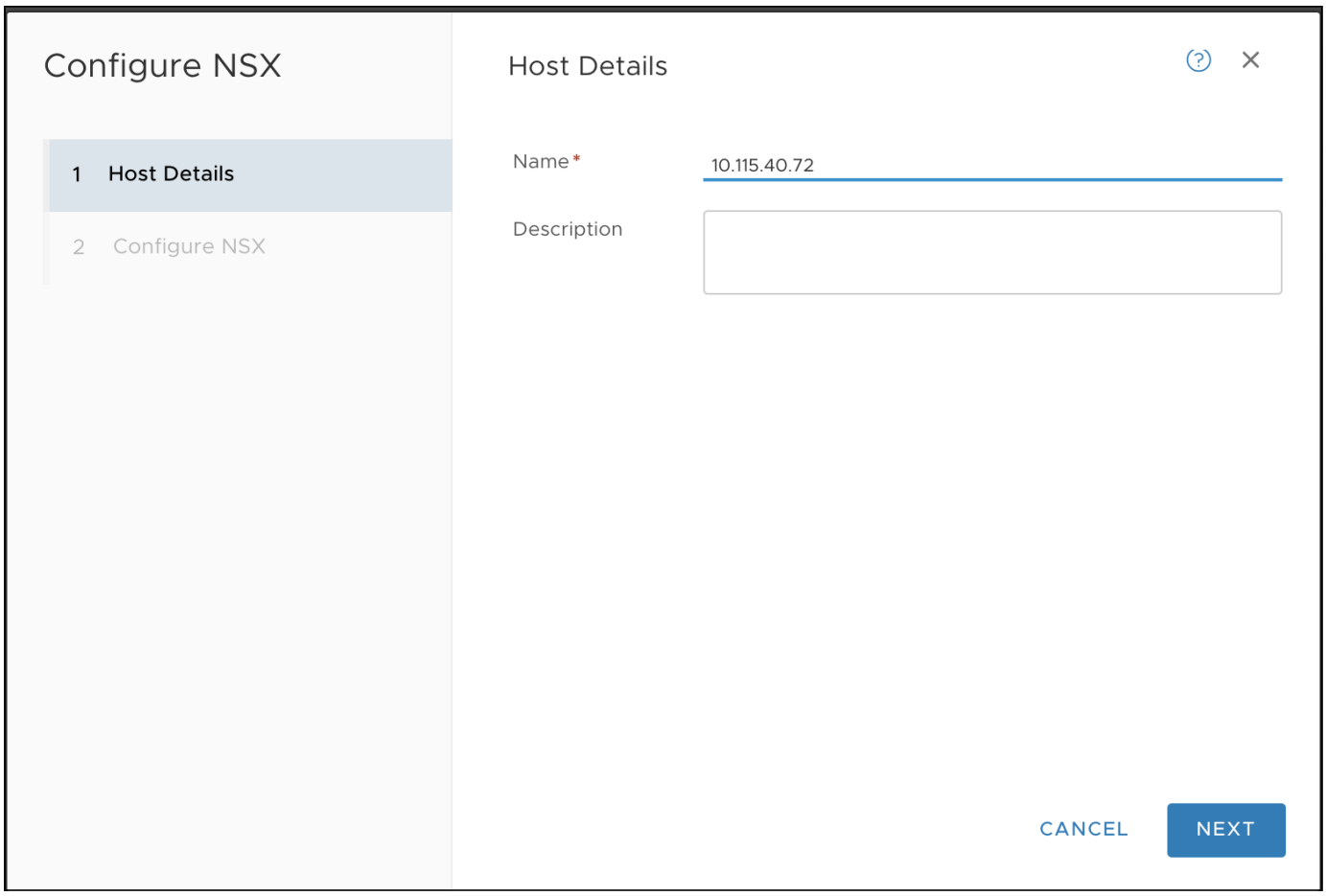
3. Expand the Cluster icon so the ESXi hosts are displayed.



4. Select the ESXi compute host and click **Configure NSX**.

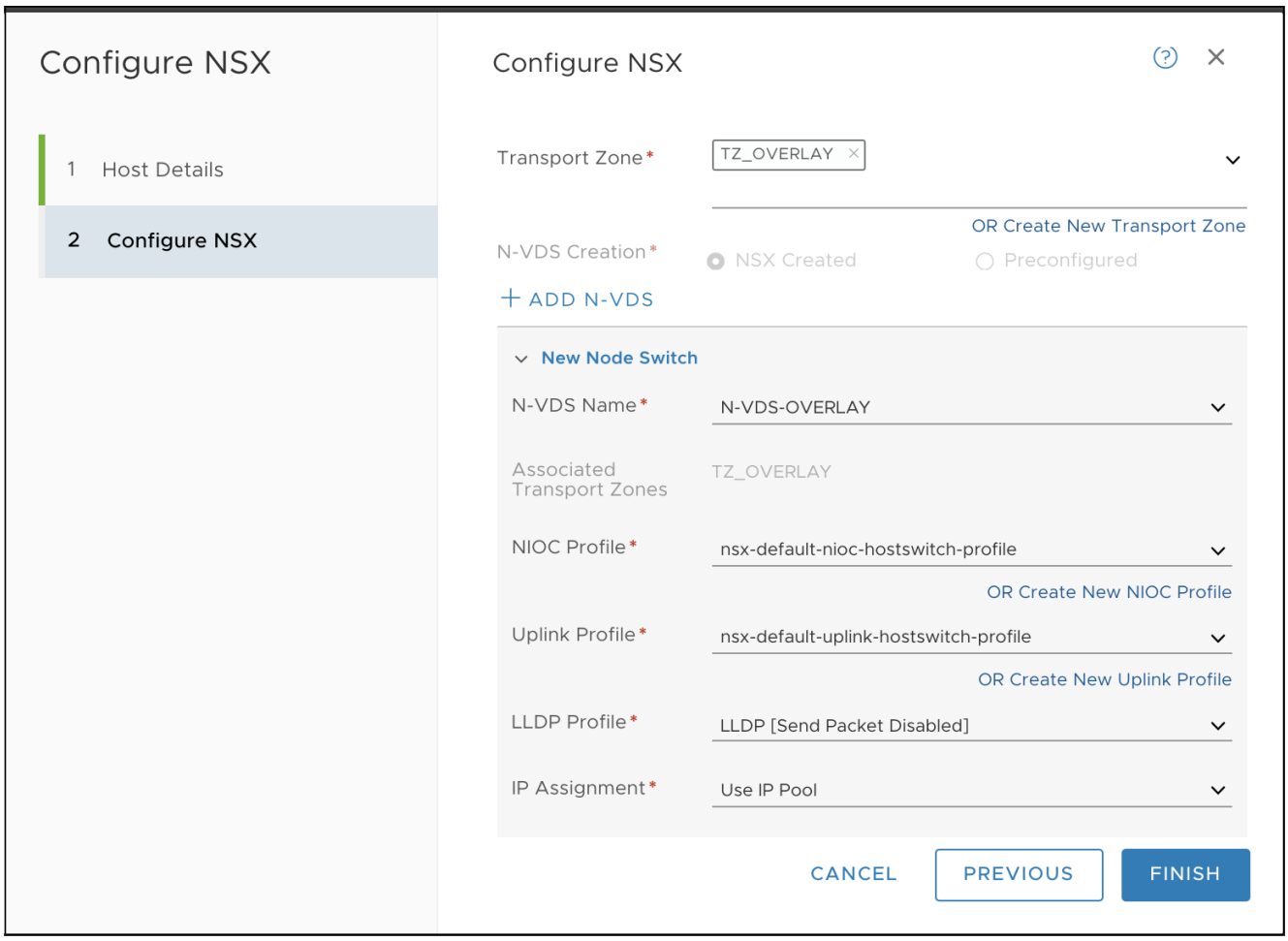


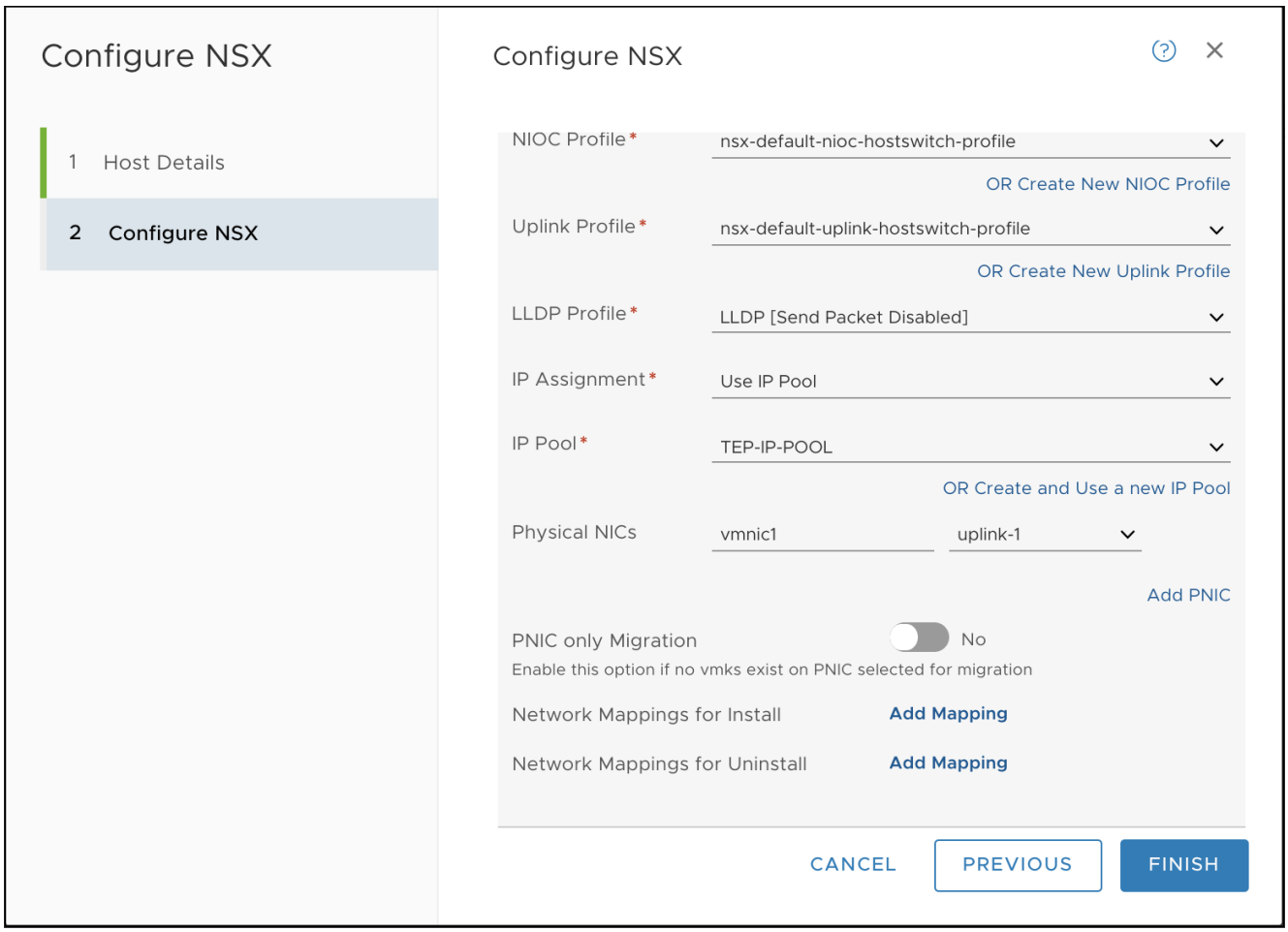
5. At the **Host Details** screen, click **Next** to configure the ESXi host as a transport node.



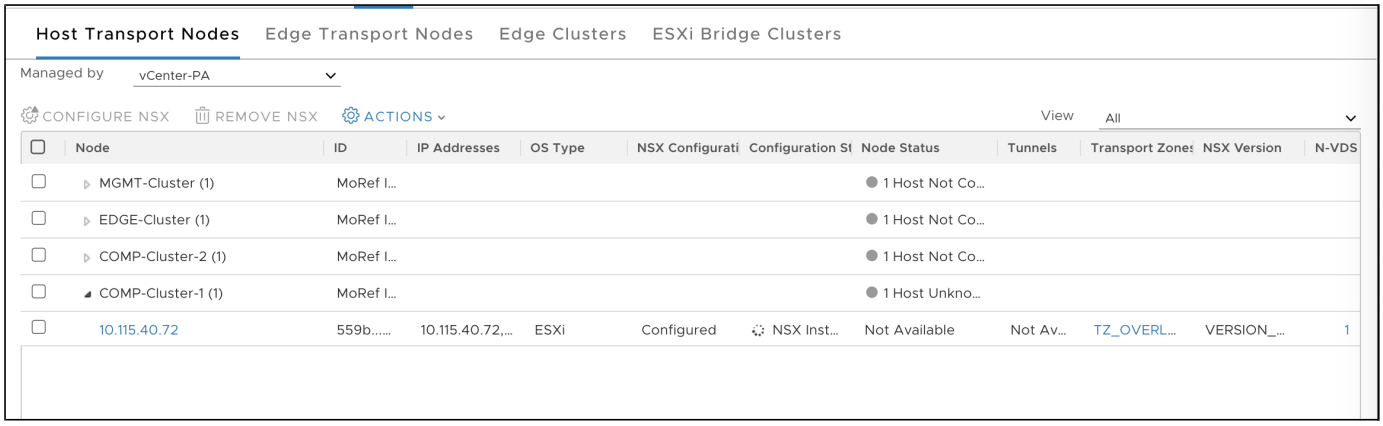
6. Configure the ESXi host as follows:

   - **Transport Zone**: TZ-Overlay
   - **N-VDS Name**: `N-VDS-OVERLAY`
   - **NIOC Profile**: `nsx-default-noic-hostswitch-profile`
   - **Uplink Profile**: `nsx-default-uplink-hostswitch-profile`
   - **LLDP Profile**: LLDP [Send Packet Disabled]
   - **IP Assignment**: `Use IP Pool`
   - **IP POOL**: `TEP-IP-POOL`
   - **Physical NICs**: `vmnic1` and `uplink-1`
   - **PNIC only Migration**: No





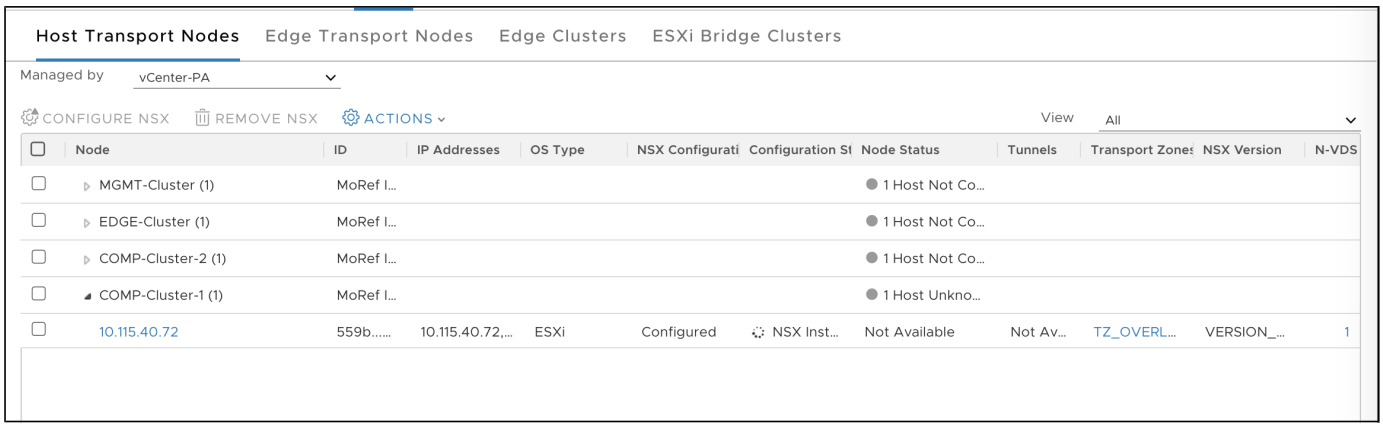
7. Click **Finish** to create the ESXi Transport Node.



8. Repeat the process for each ESXi host in the vSphere Cluster dedicated for Enterprise PKS.

Verify ESXi Host Preparation for Enterprise PKS

1. In NSX Manager, go to **System** > **Fabric** > **Nodes** > **Host Transport Nodes**.
2. For the **Managed by** field, select **VMware vCenter Server App** from the dropdown menu.
3. Verify that the **Node Status** is **Success** for the ESXi host.



4. Select the information icon for the EXSi host in the **Node Status** column. Verify that **Tunnel Status** is **Up**.

5. Verify that 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

Verify the NSX-T Installation.

Installation Instructions Home

See Installing and Configuring NSX-T v2.5 for Enterprise PKS.

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

Verify NSX-T v2.5 Installation

In this topic

Page last updated:

This topic provides instructions for verifying your NSX-T environment after performing an installation of NSX-T v2.5.

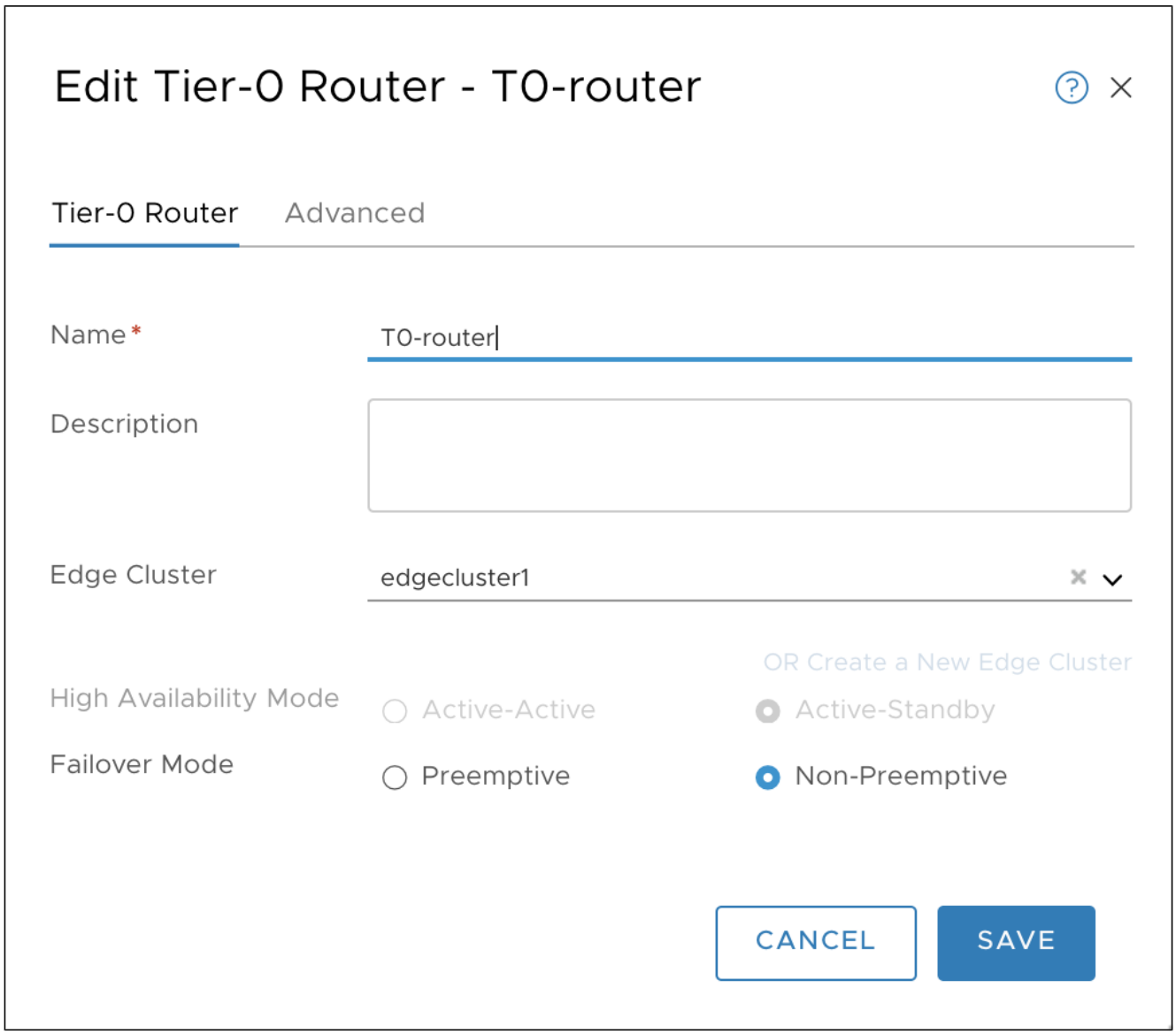
Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

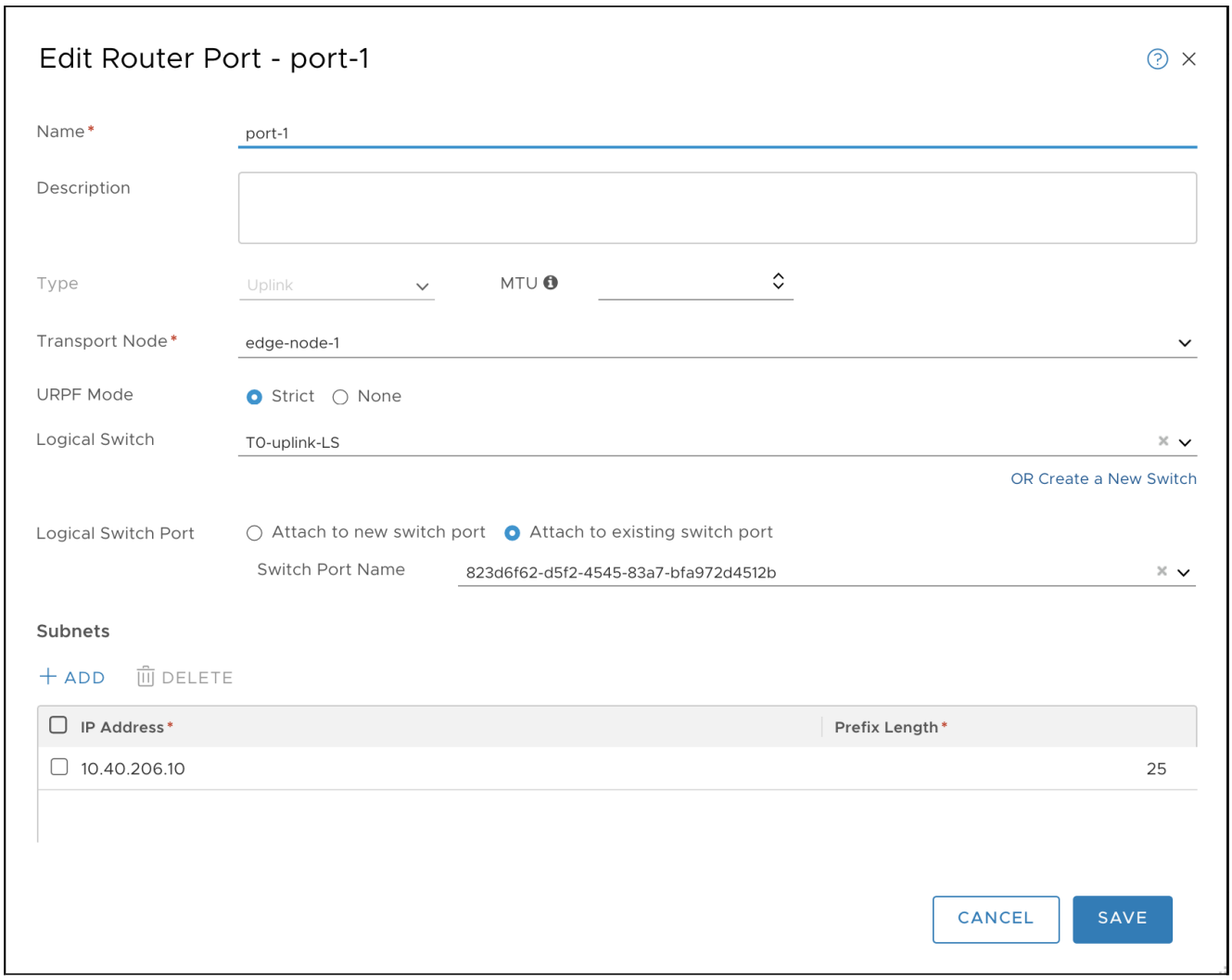
Verify the Installation of NSX-T v2.5

Complete the following operation for each ESXi host to be used by the PKS Compute Cluster.

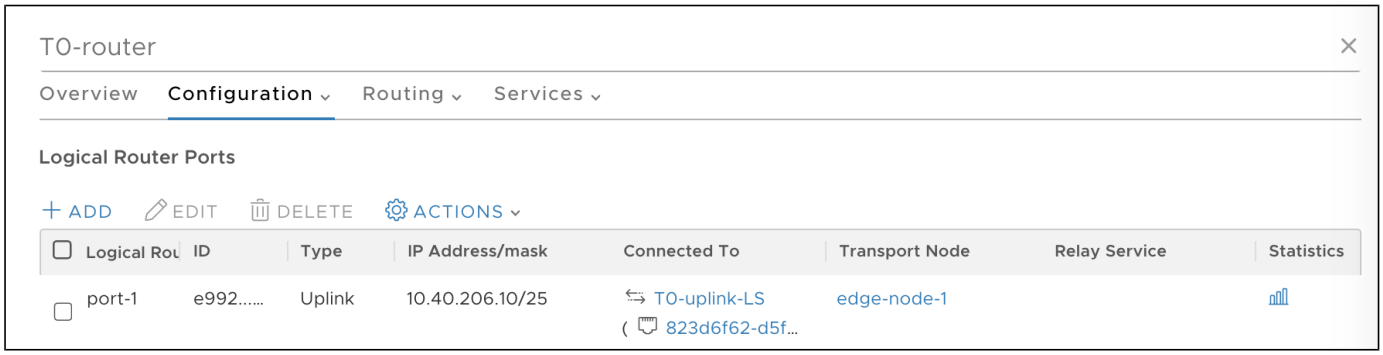
1. In NSX-T Manager, create a T0 router.
    - Name: T0-router
    - Edge Cluster: edgecluster1
    - HA mode: Active/Standby



2. Create a router port.
   - Transport Node: edge-node-1
   - LS: T0-uplink-LS
   - Subnet: For example, 10.40.206.10/25

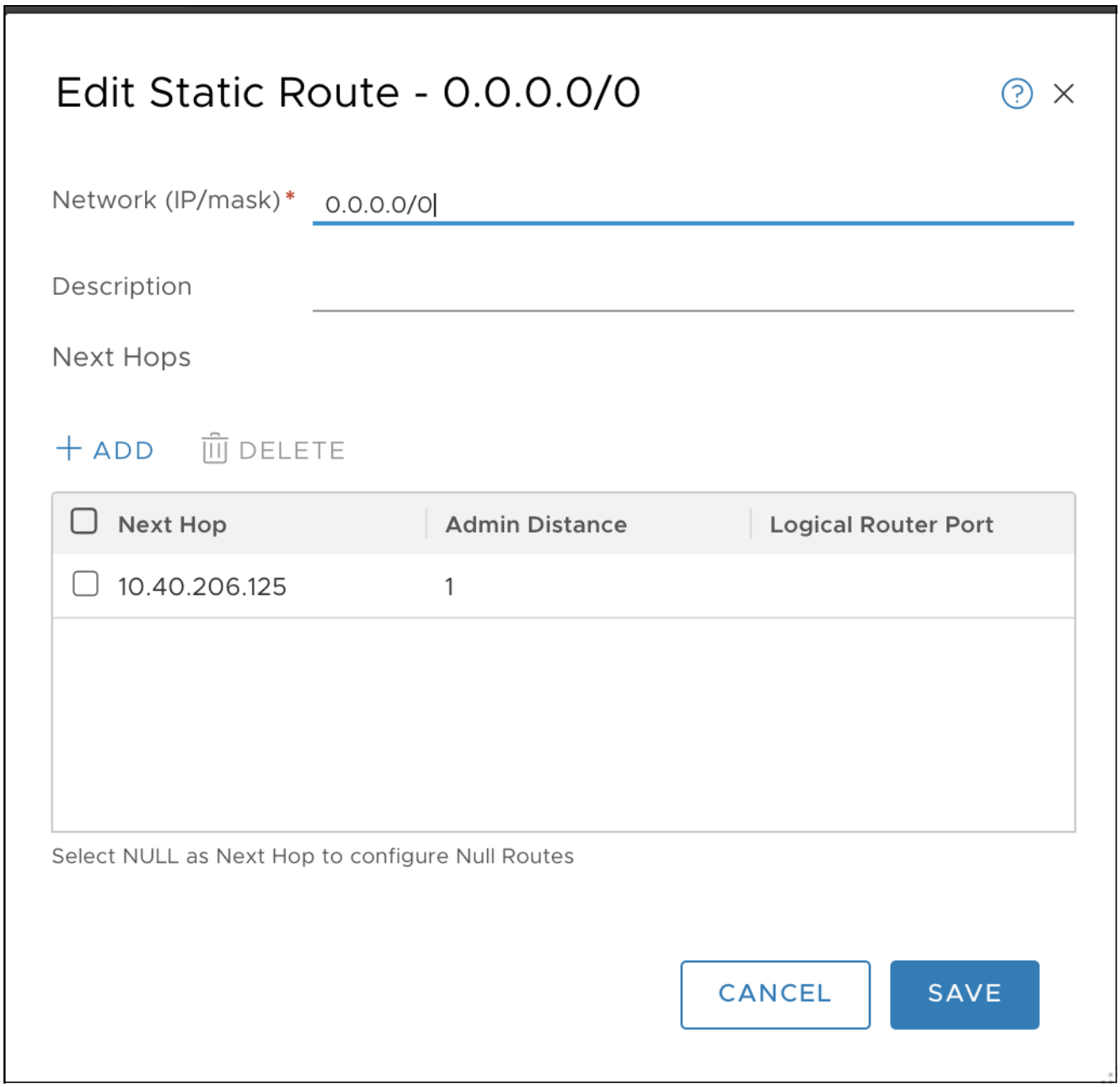


3. Click **Save** to create the T0 router.



4. Create a static route.

   - Network: 0.0.0.0/0
   - Next hop: 10.40.206.125



5. Click **Save** to save the T0 router.

6. Verify that you can ping the T0 interface IP from the network.

```
PING 10.40.206.10 (10.40.206.10): 56 data bytes
64 bytes from 10.40.206.10: icmp_seq=0 ttl=55 time=24.407 ms
64 bytes from 10.40.206.10: icmp_seq=1 ttl=55 time=23.984 ms
64 bytes from 10.40.206.10: icmp_seq=2 ttl=55 time=24.170 ms
64 bytes from 10.40.206.10: icmp_seq=3 ttl=55 time=30.211 ms
```

Next Step

Assuming you have successfully completed all preceding NSX-T installation tasks, NSX-T Data Center is now be installed and configured for Enterprise PKS.

Depending on your requirements, you may want to complete the following additional tasks:

- The NSX-T Data Center administrator password expires after 90 days. To change the password or the expiration interval, see Updating the NSX-T Admin Password.
- If you require scalability for the NSX Management Cluster, Provision a Load Balancer for the NSX-T Management Cluster.

Installation Instructions Home

See Installing and Configuring NSX-T v2.5 for Enterprise PKS.

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

Updating the NSX-T Admin Password

In this topic

Page last updated:

This topic provides instructions for updating the NSX-T administrator password for use with VMware Enterprise PKS.

Prerequisites

Make sure you have completed all preceding NSX-T installation and configuration tasks.

About the NSX-T Admin Password

VMware NSX-T Data Center v2.4 and later introduces the following password policy enhancements:

- Enforces minimum password length of 12 characters for default passwords.
- Introduces ability to set password expiration times and generates alarms when password is about to expire.

The default password expiration interval is 90 days. After this period, the NSX-T administrator password will expire on all NSX-T Manager Nodes and all NSX-T Edge Nodes.

> **Note:** For existing Enterprise PKS deployments, anytime the NSX-T password is changed you must update the BOSH and PKS tiles with the new passwords. See Adding Infrastructure Password Changes to the Enterprise PKS Tilefor more information.

Update the Password for NSX Manager Nodes

To update the NSX Manager password, perform the following actions on **one** of the NSX Manager nodes. The changes will be propagated to all NSX Manager nodes.

SSH to the NSX Manager Node

To manage user password expiration, you use the CLI on one of the NSX Manager nodes.

To access a NSX Manager node, from Unix hosts use the command `ssh USERNAME@IP_ADDRESS_OF_NSX_MANAGER`.

For example:

```
ssh admin@10.196.188.22
```

On Windows, use Putty and provide the IP address for NSX Manager. Enter the user name and password that you defined during the installation of NSX-T.

Get the password expiration interval

To get the password expiration interval, use the following command:

get user USERNAME password-expiration

For example:

```
NSX CLI (Manager, Policy, Controller 2.4.1.0.0.13716579). Press ? for command list or enter: help
nsx-manager> get user admin password-expiration
Password expires 90 days after last change
```

Update the admin password

To update the user password, use the following command:

set user USERNAME password NEW-PASSWORD old-password OLD-PASSWORD.

For example:

```
set user admin password my-new-pwd old-password my-old-pwd
```

Set the admin password expiration interval

To set the password expiration interval, use the following command:

set user USERNAME password-expiration PASSWORD-EXPIRATION.

For example, the following command sets the password expiration interval to 120 days:

```
set user admin password-expiration 120
```

Remove the admin password expiration interval

To remove password expiration, use the following command:

clear user USERNAME password-expiration.

For example:

```
clear user admin password-expiration
```

To verify:

```
nsx-manager-1> clear user admin password-expiration
nsx-manager-1> get user admin password-expiration
Password expiration not configured for this user
```

Update the Password for NSX Edge Nodes

To update the NSX Edge Node password, perform the following actions on **each** NSX Edge Node.

Enable SSH

SSH on the Edge Node is disabled by default. You have to enable SSH on the Edge Node using the Console from vSphere.

```
start service ssh
set service ssh start-on-boot
```

SSH to the NSX Edge Node

For example:

```
ssh admin@10.196.188.25
```

Get the password expiration interval for the Edge Node

For example:

```
nsx-edge> get user admin password-expiration
Password expires 90 days after last change
```

Update the user password for the Edge Node

For example:

```
nsx-edge> set user admin password my-new-pwd old-password my-old-pwd
```

Set the password expiration interval

For example, the following command sets the password expiration interval to 120 days:

```
nsx-edge> set user admin password-expiration 120
```

Remove the password expiration interval

For example:

```
nsx-edge> clear user admin password-expiration
nsx-edge> get user admin password-expiration
Password expiration not configured for this user
```

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

Provisioning a Load Balancer for the NSX-T Management Cluster

In this topic

Page last updated:

This topic describes how to deploy a load balancer for the NSX-T Management Cluster for Enterprise PKS.

About the NSX-T Management Cluster

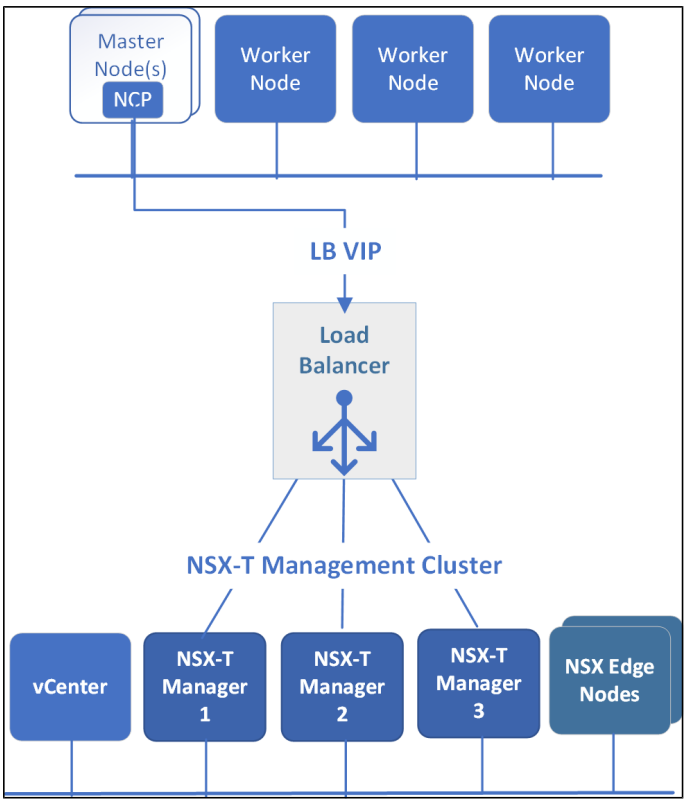
This section describes the NSX-T Management Cluster and the external load balancer for use with Enterprise PKS.

Overview

Starting in NSX-T v2.4, NSX-T supports a converged management and control plane that is referred to as the **NSX-T Management Cluster**. The new deployment model delivers high availability of the NSX-T Manager node, reduces the likelihood of operation failures of NSX-T, and provides API and UI clients with multiple endpoints or a single VIP for high availability.

While using a VIP to access the NSX-T Management layer provides high-availability, it does not balance the workload. To avoid overloading a single NSX-T Manager, as may be the case when HA VIP addressing is used, an NSX-T load balancer can be provisioned to allow NCP and other components orchestrated by Enterprise PKS to distribute load efficiently among NSX Manager nodes.

The diagram below shows an external load balancer fronting the NSX Manager nodes. The load balancer is deployed within the NSX-T environment and intercepts requests to the NSX-T Management Cluster. The load balancer selects one of the NSX-T Manager nodes to handle the request and rewrites the destination IP address to reflect the selection.



Note: The load balancer VIP load balances traffic to all NSX-T Manager instances in round robin fashion. A Cluster HA VIP, on the other hand, only sends traffic one of the NSX-T Manager instances that is mapped to the Cluster IP VIP; the other NSX-T manager instances do not receive any traffic.

Note: If you are using VMware Identity Manager (VIDM) to authenticate with the NSX Management environment, you need two separate load balancer VIPs: one for VIDM and one for PKS. Refer to Configure VMware Identity Manager Integration in the VMware NSX-T Data Center documentation.

Component Interaction with the NSX-T Management Cluster

Various components in an Enterprise PKS deployment interact with the NSX Management Cluster.

PKS Control Plane components:

- Ops Manager
- BOSH CPI (Cloud Provider Interface)
- NSX-T OSB Proxy

Kubernetes Cluster components:

- BOSH jobs running on Kubernetes master nodes to prepare and update Kubernetes clusters
- NSX-T Container Plugin (NCP)

The interaction of the PKS Control Plane components and the BOSH jobs with the NSX-T Management Cluster is sporadic. However, the NCP component may demand a high level of scalability for the NSX-T API processing capability of the NSX Management Cluster, and NCP is vital to the networking needs of each Kubernetes cluster. When a high number of Kubernetes clusters are subjected to concurrent activities, such as Kubernetes Pod and Service lifecycle operations, multiple NCP instances may tax the system and push NSX-T API processing to its limits.

Load Balancer Provisioning for the NSX-T Management Cluster

For scalability, consider deploying a load balancer in front of the NSX-T Manager nodes. As a general rule of thumb, if you are using Enterprise PKS with NSX-T to deploy more than 25 Kubernetes clusters, you should use a load balancer in front of the NSX-T Management Cluster.

> **Note:** If you do not require scalability, you can configure a Cluster VIP to achieve HA for the NSX-T Management Cluster. See HA VIP addressing.

For general purposes, a small NSX-T load balancer is sufficient. However, refer to the Scaling Load Balancer Resources to ensure that the load balancer you choose is sufficient to meet your needs.

When provisioning the load balancer, you configure a virtual server on the load balancer, and associate a virtual IP address with the virtual server. This load balancer VIP can be used as the entry-point for PKS- and NCP-related API requests on the NSX-T Control Plane. The virtual server includes a member pool where all NSX-T Management Cluster nodes belong. Additionally, health monitoring is enabled for the member pool to quickly and efficiently address potential node failures detected among the NSX-T Management Cluster.

Prerequisites for Provisioning a Load Balancer for the NSX-T Management Cluster

Before you provision a load balancer for the NSX-T Management Cluster, ensure that your environment is configured as follows:

- NSX-T is installed and configured for servicing Enterprise PKS. See Installing and Configuring NSX-T for Enterprise PKS.
- Transport zone, transport node, Edge Cluster, Edge Node connectivity, and Tier-0 Router are deployed and operational with proper static routes or BGP. See Installing and Configuring NSX-T for Enterprise PKS.
- A NSX-T Management Cluster with 3 NSX Manager nodes is provisioned. See Installing and Configuring NSX-T for Enterprise PKS.
- Your NSX-T environment has enough Edge Cluster resources to deploy a new small-size load balancer VM.
- A dedicated IP Address is available to be used as the VIP and SNAT IP address for the new load balancer. The load balancer VIP address must be globally routable from networks external to NSX-T. This IP address can be carved out from the standard IP Pool required by Enterprise PKS PKS.

Provision the NSX-T Load Balancer for the Management Cluster

To provision the load balancer for the NSX-T Management Cluster, complete the following steps.

Step 1: Log in to an NSX-T Manager Node

1. Log in to an NSX-T Manager Node.

   > **Note**: You can connect to any NSX-T Manager Node in the management cluster to provision the load balancer.

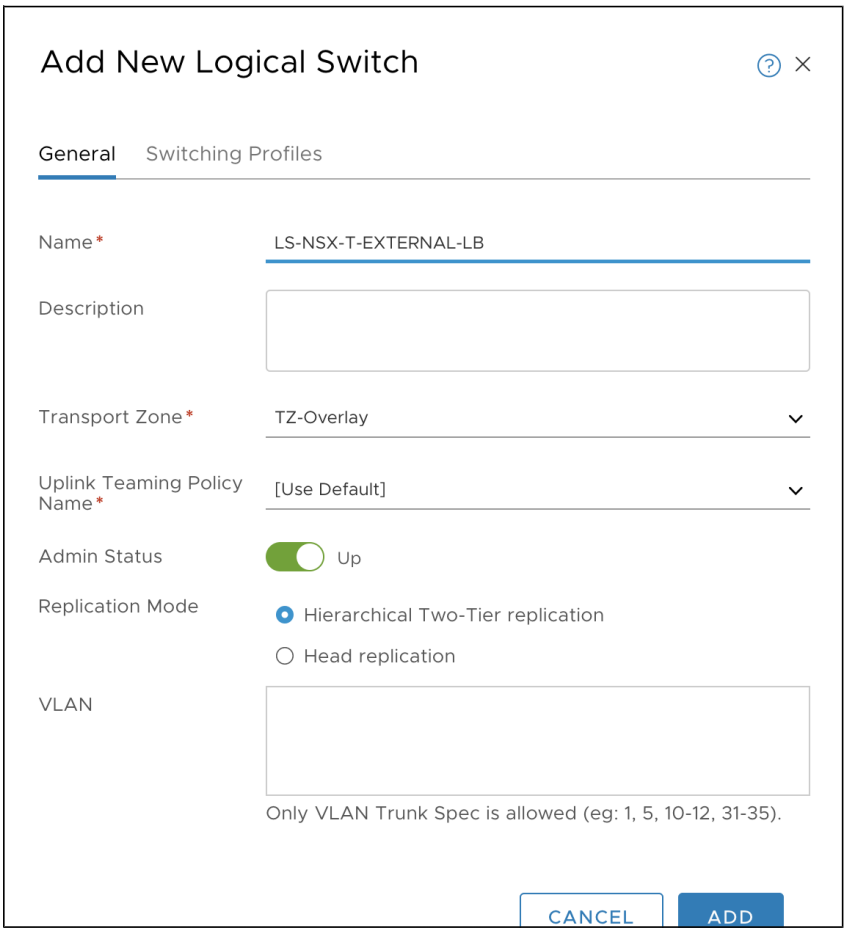
2. Select the **Advanced Networking & Security** tab.

   > **Note**: You must use the **Advanced Networking and Security** tab in NSX-T Manager to create, read, update, and delete all NSX-T networking objects used for Enterprise PKS.

Step 2: Configure a Logical Switch

Add and configure a new logical switch for the load balancer.

- Select **Switching**.
- Click **Add**.
- Configure the logical switch:
  - **Name**: Enter a name for the logical switch, such as **LS-NSX-T-EXTERNAL-LB**.
  - **Transport Zone**: Select the overlay transport zone, such as **TZ-Overlay**.
- Click **Add**.

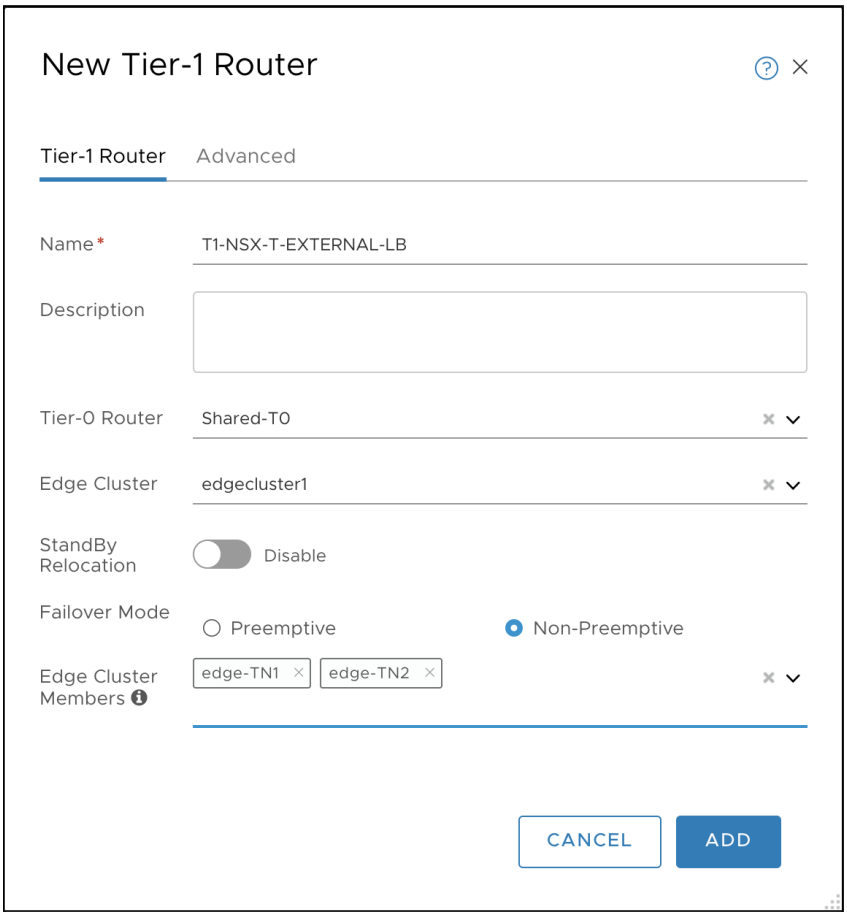


Step 3: Configure a Tier-1 Logical Router

Configure a new Tier-1 Router in Active/StandBy mode. Create the Tier-1 Router on the same Edge Cluster where the Tier-0 Router that provides external connectivity to vCenter and NSX Manager is located.

- Select **Routers**.
- Click **Add** > **Tier-1 Router**.
- Configure the new Tier-1 Router and click **Add**.
  - **Name**: **T1-NSX-T-EXTERNAL-LB**, for example.
  - **Tier-0 Router**: Connect the Tier-1 Router to the Tier-0 Router, for example **Shared-T0**.

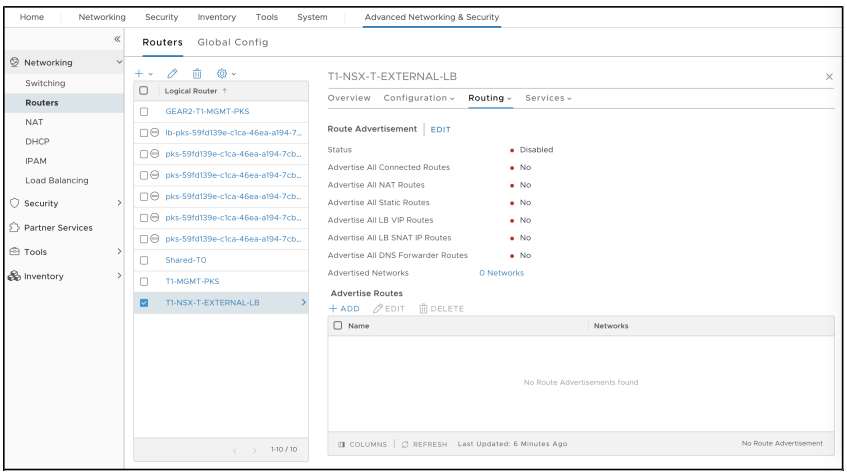
- **Edge Cluster**: Select the same Edge Cluster where the Tier-0 Router is located, such as **edgecluster1**.
- **Edge Cluster Members**: Select **nsx-edge-1-tn** and **nsx-edge-2-tn**, for example.

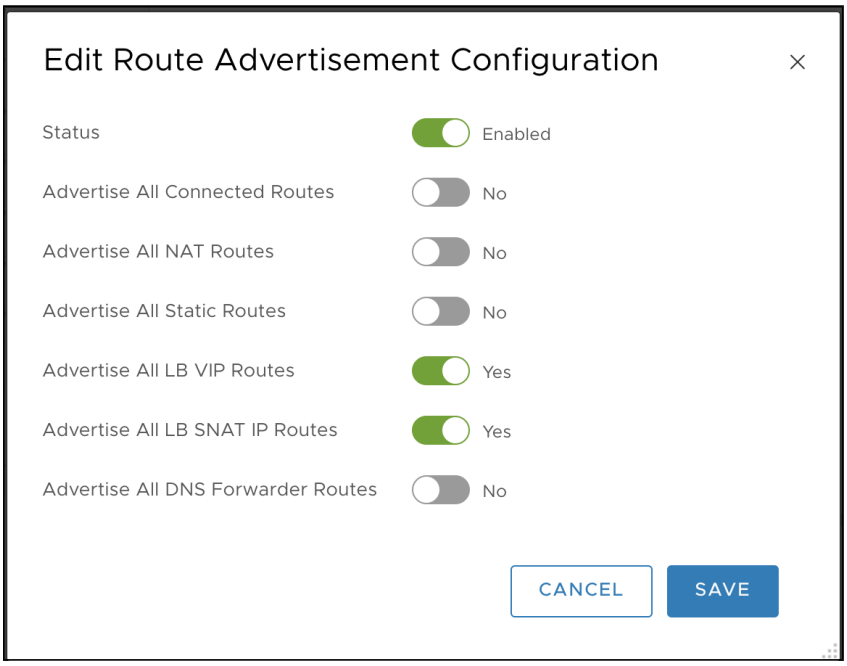


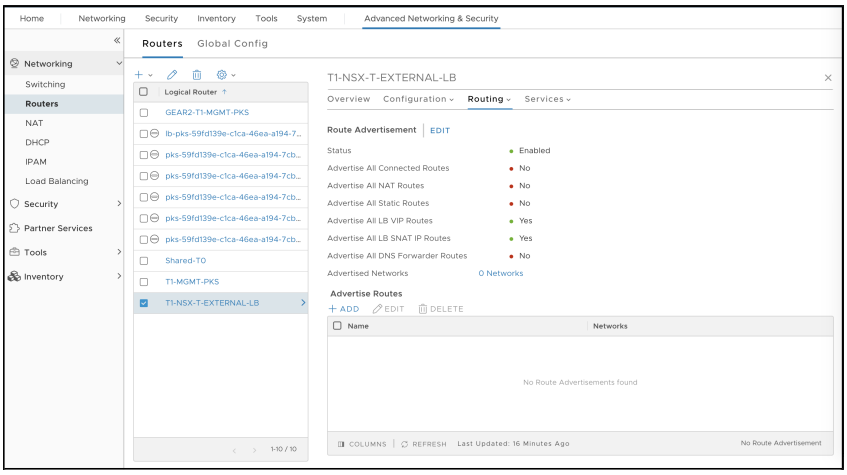
Step 4: Advertise the Routes

Configure Route Advertisement for the Tier-1 Router.

- Select the Tier-1 Router.
- Select the **Routing** tab.
- Select **Route Advertisement** > **Edit**.
- Enable Route Advertisement for all load balancer VIP routes for the Tier-1 Router:
  - **Status**: enabled
  - **Advertise all LB VIP routes**: yes
  - **Advertise all LB SNAT IP routes**: yes
  - Click **Save**



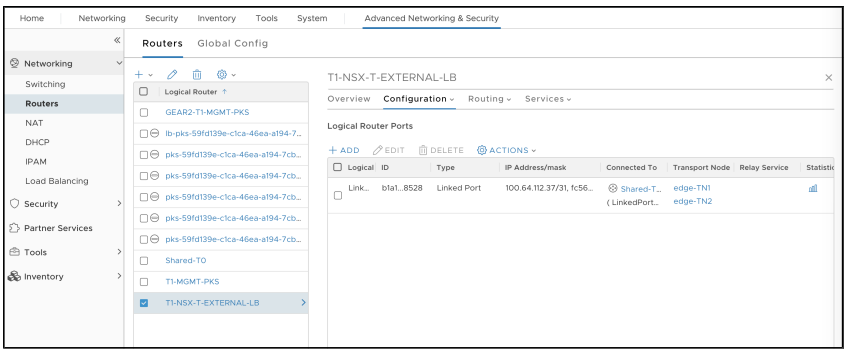




Step 5: Verify Router and Switch Configuration

Verify successful creation and configuration of the logical switch and router.

- Select the Tier-1 Router.
- Select the **Configuration** tab and the **Ports** option.
- Verify that the router has a single linked port connecting the Tier-1 Router to the Tier-0 Router.

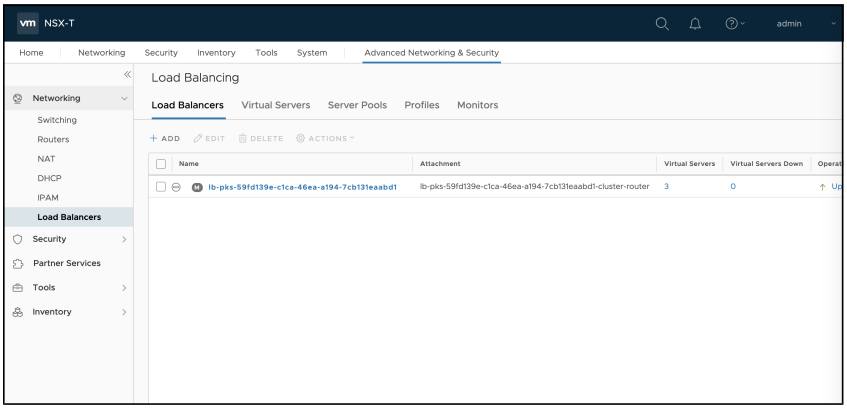


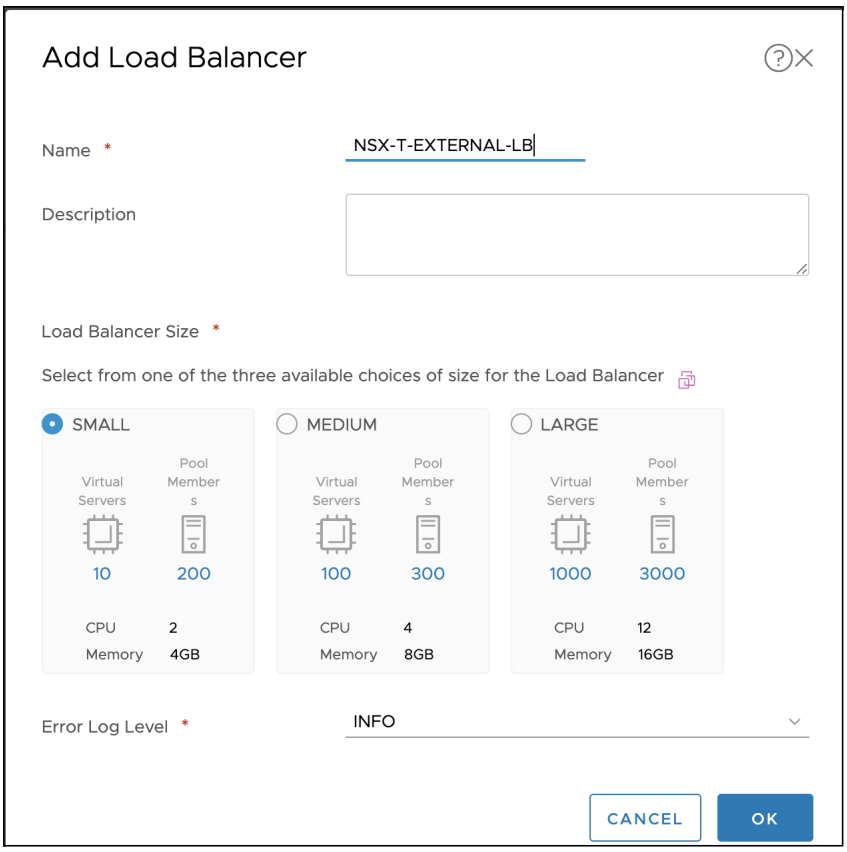
Step 6: Configure a Small Load Balancer

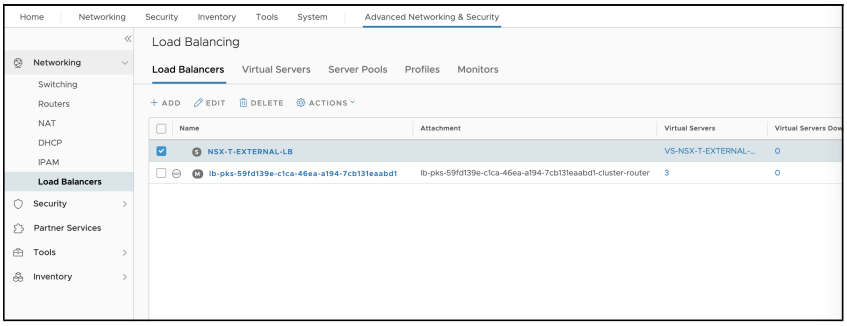
Create a new small-size Load Balancer and attach it to the Tier1 router previously created.

> **Note**: The small-size VM is suitable for the NSX Management Cluster load balancer. Make sure you have enough Edge Cluster resources to provision a small load balancer.

- Select **Load Balancers**.
- Click **Add**.
- Enter a **Name** for the load balancer.
- Select the **SMALL** size load balancer.
- Click OK.



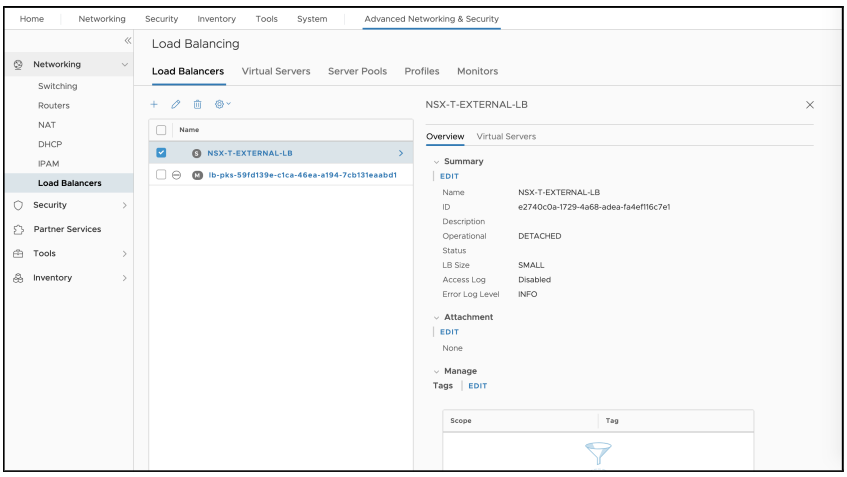


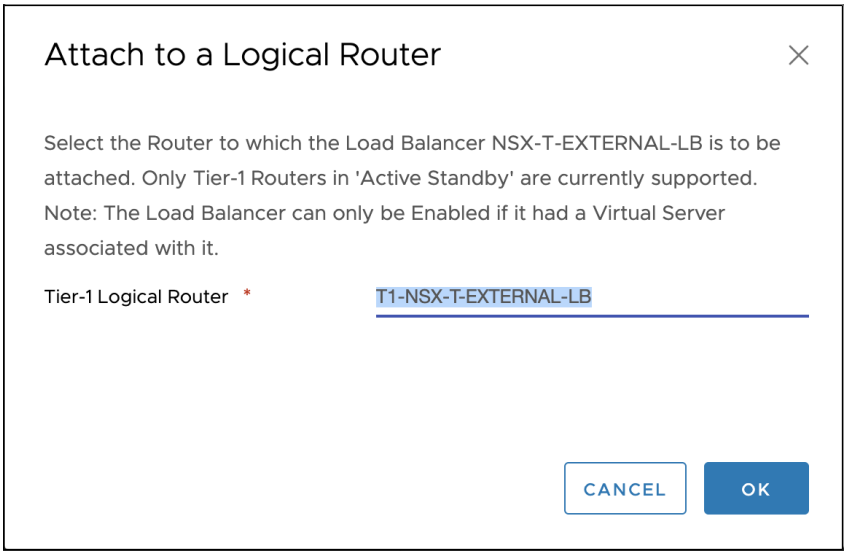


Step 7: Attach the Load Balancer to the Router

Attach the load balancer to the Tier-1 Router previously created.

- Select the load balancer you just provisioned.
- Select the **Overview** tab.
- Select **Attachment** > **Edit**.
- **Tier-1 Logical Router**: Enter the name of the Tier-1 Router you configured, for example `T1-NSX-T-EXTERNAL-LB`.
- Click **OK**.

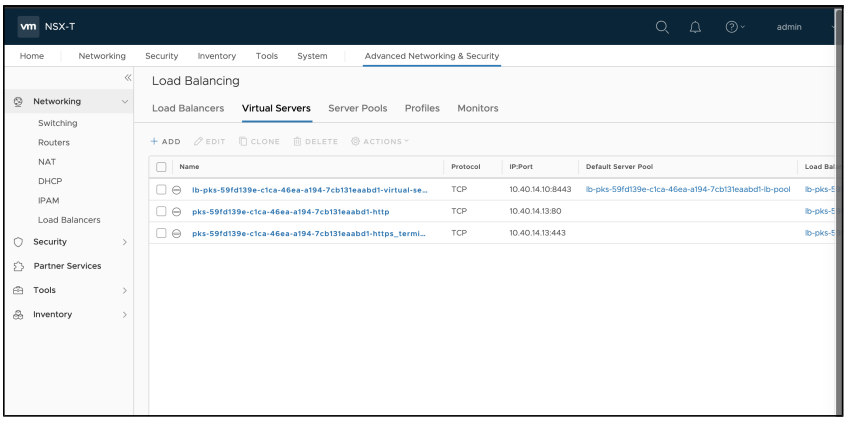




Step 8: Configure a Virtual Server

Add and configure a virtual server for the load balancer.

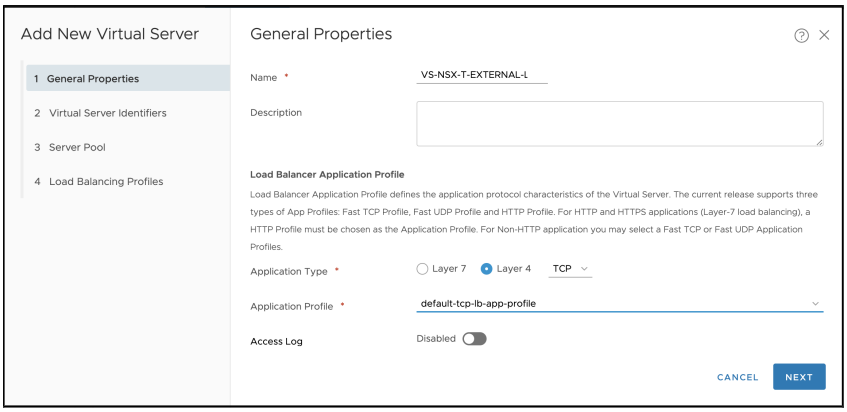
- Select **Load Balancers** > **Virtual Servers**.
- Click **Add**.



Configure **General Properties** for the virtual server:

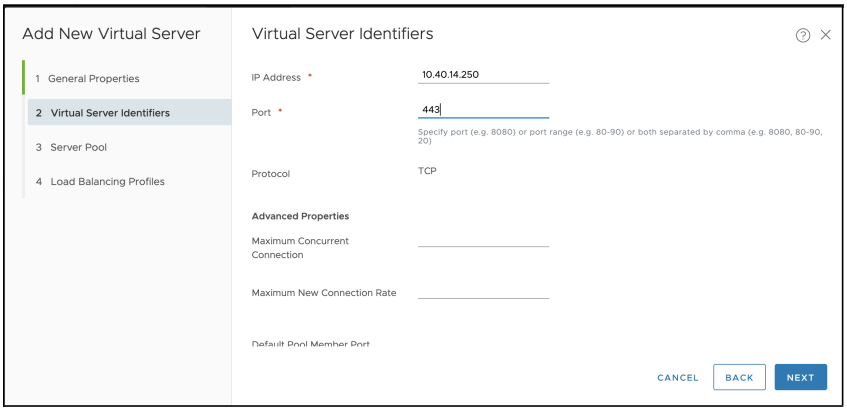
- **Name**: VS-NSX-T-EXTERNAL-LB

- **Application Types**: Layer 4 TCP
- **Application Profile**: default-tcp-lb-app-profile
- **Access Log**: Disabled
- Click **Next**



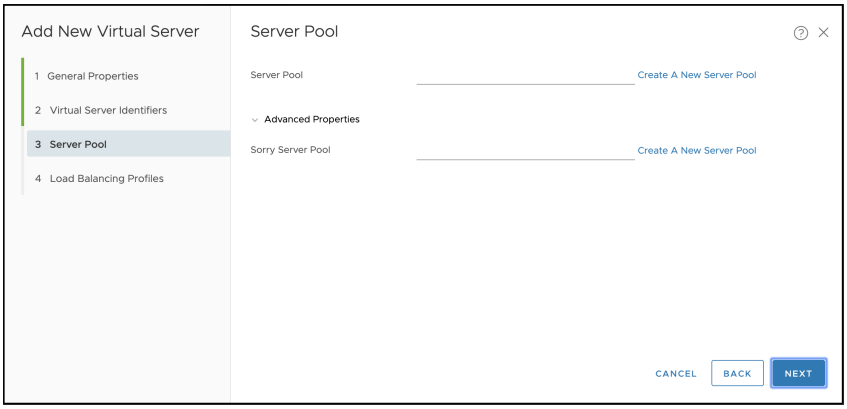
Configure **Virtual Server Identifiers** for the virtual server:

- **IP Address**: Enter an IP address from the floating pool, such as `10.40.14.250`.
- **Port**: 443
- Click **Next**.



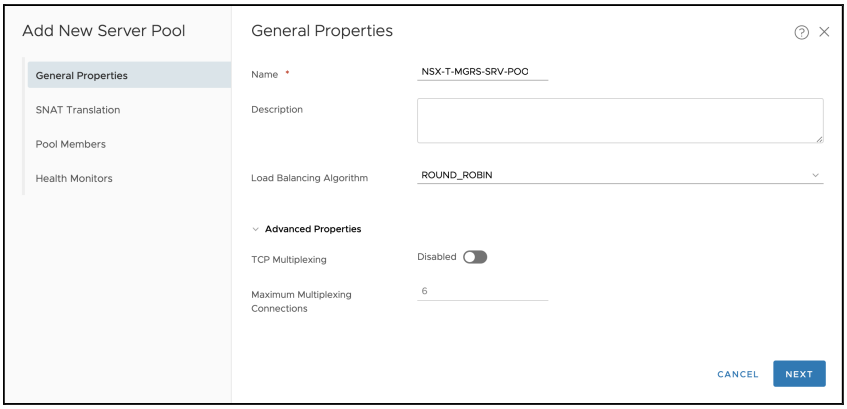
Configure **Virtual Server Pool** for the virtual server:

- Click **Create a New Server Pool**.



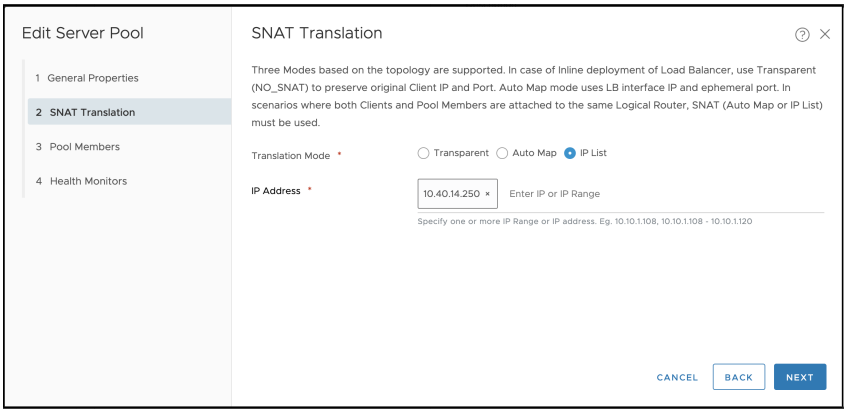
Configure **General Properties** for the server pool:

- **Name**: For example `NSX-T-MGRS-SRV-POOL`
- **Load Balancing Algorithm**: ROUND_ROBIN
- Click **Next**



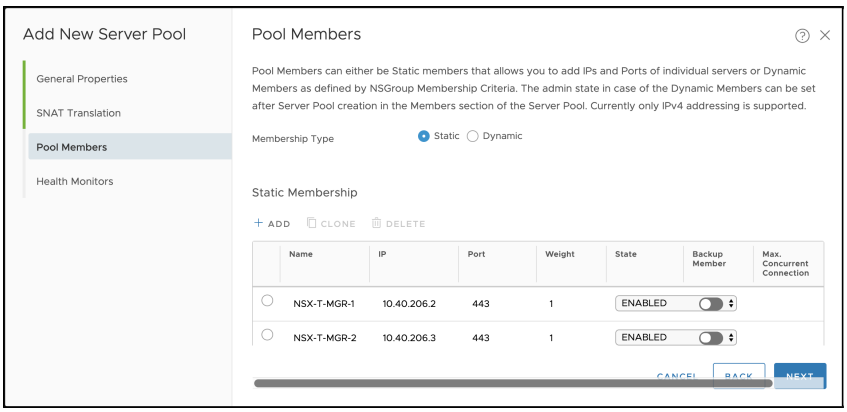
Configure **SNAT Translation** for the server pool:

- **Translation Mode**: IP List
- **IP address**: Enter the Virtual Switch IP (VIP) address here, for example `10.40.14.250`.
- Click **Next**.



Configure **Pool Members** for the server pool:

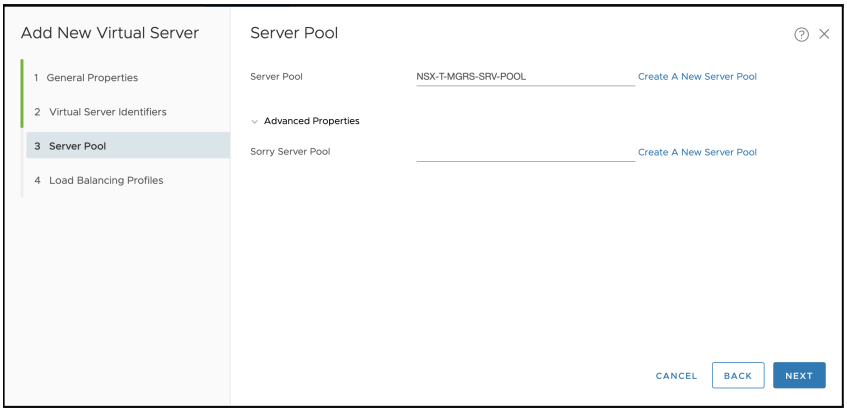
- **Membership Type**: Static.
- **Static Membership**: Add all 3 NSX Managers as members by entering the node name, IP address, and port (443) for each node.
- Click **Next**.



Configure **Health Monitors**:

- We will create the Health Monitors separately.
- Click **Finish**.

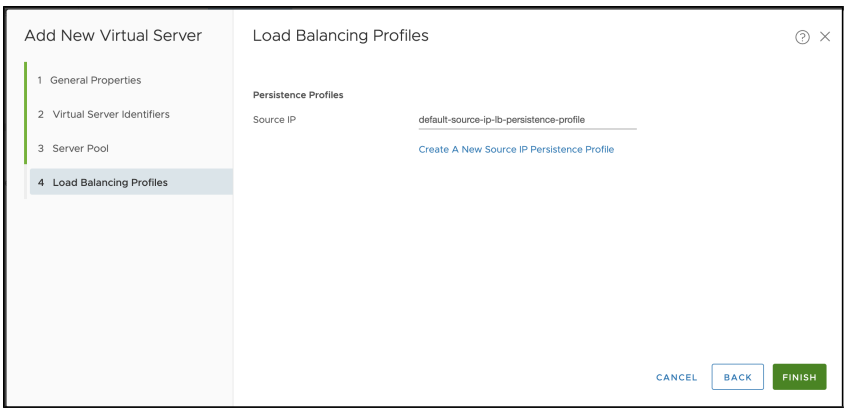
Back at the Server Pool screen, click **Next**.

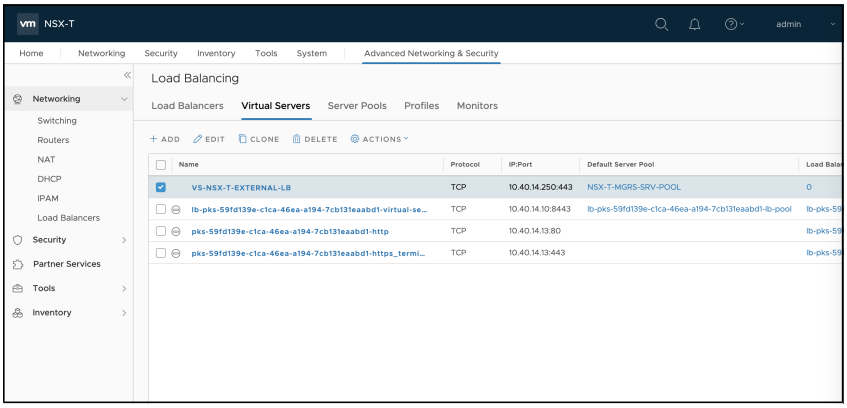


Configure **Load Balancing Profiles** for the load balancer:

- **Persistence Profile** > **Source IP**: Select **default-source-ip-lb-persistence-profile**
- Click **Finish**.

> **Note:** If a proxy is used between the NSX Management Cluster and the PKS Control Plane, do not configure a persistence profile.

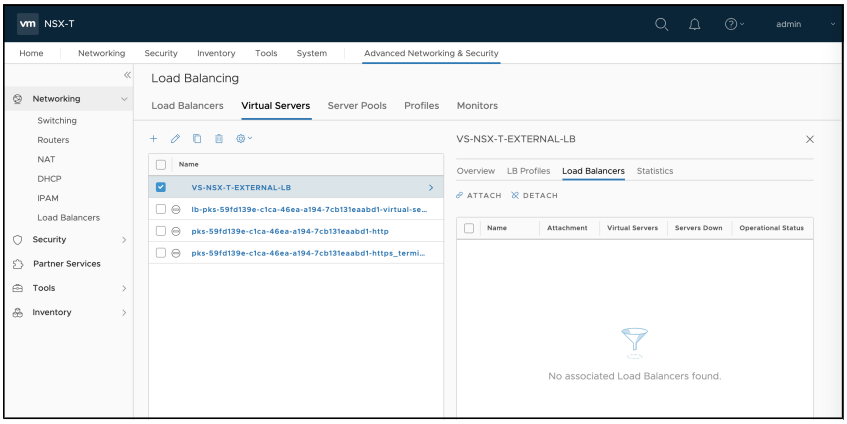


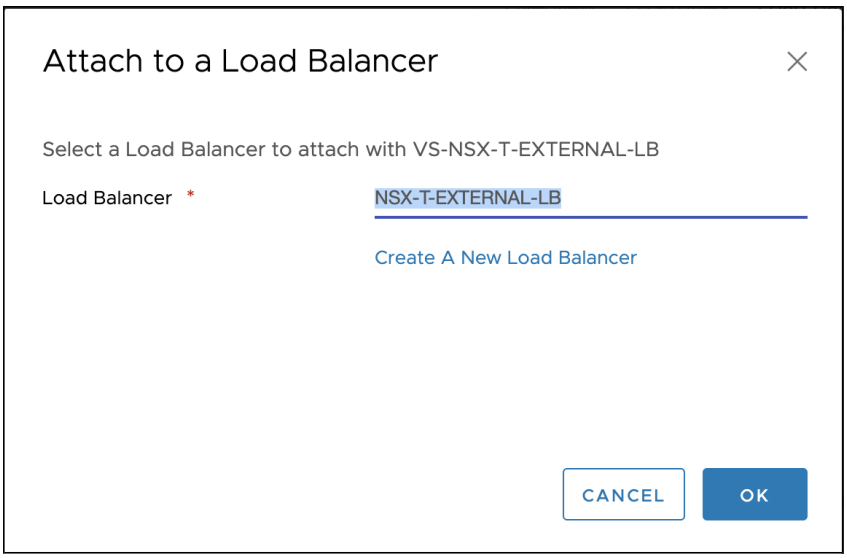


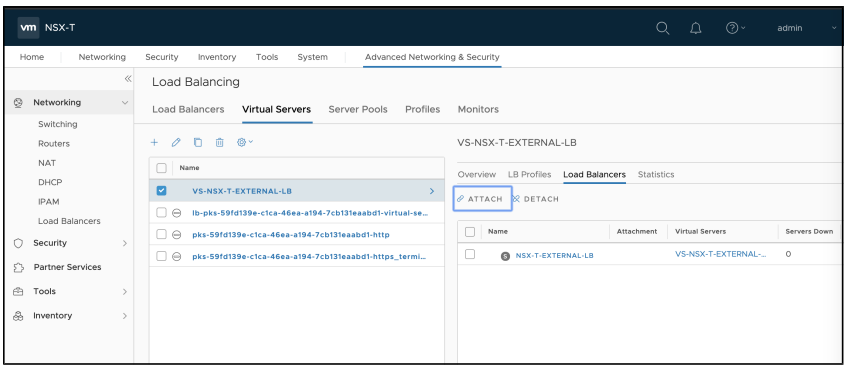
Step 9: Attach the Virtual Server to the Load Balancer

Attach the virtual switch to the NSX-T load balancer.

- In the **Load Balancing** panel, select the Virtual Server you created.
- Select the **Load Balancers** tab.
- Click **Attach**.
- **Load Balancer**: Select the load balancer to attach, such as `NSX-T-EXTERNAL-LB`.
- Click **OK**.







Step 10: Verify the Load Balancer.

Once the load balancer is configured, verify it by doing the following:

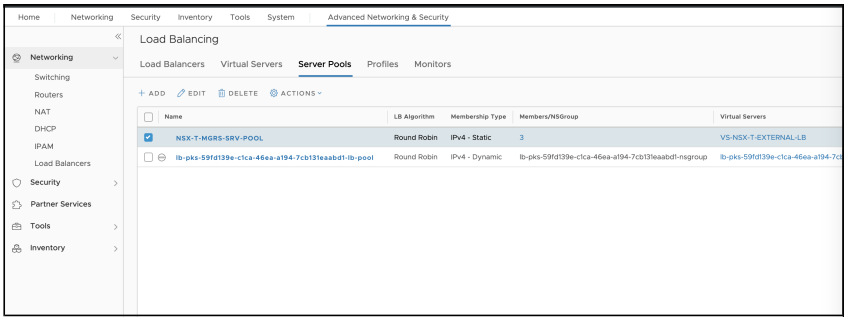
- Ping the NSX-T load balancer VIP address from your local machine.
- Access the NSX-T Manager interface using the load balancer VIP address, for example `https://10.40.14.250`.

> **Note:** The URL redirects to the same NSX-T Manager. Persistence is done on the source IP based on the persistence profile you selected.

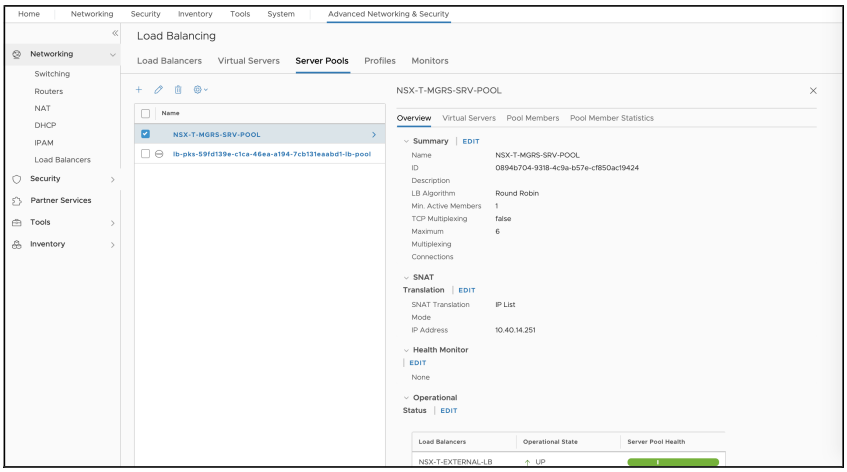
Step 11: Create an Active Health Monitor (HM)

Create a new Active Health Monitor (HM) for NSX Management Cluster members using the NSX-T Health Check protocol.

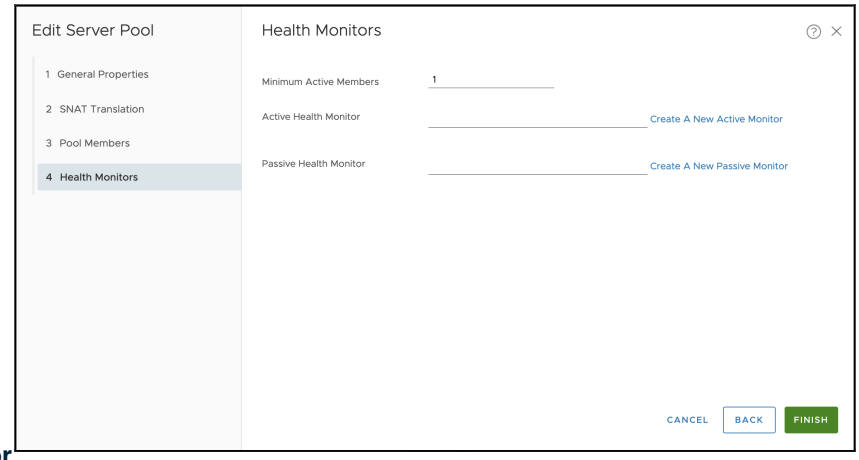
- Select **Load Balancers** > **Server Pools**.
- Select the server pool you created (for example, **NSX-T-MGRS-SRV-POOL**).



- Select the **Overview** tab



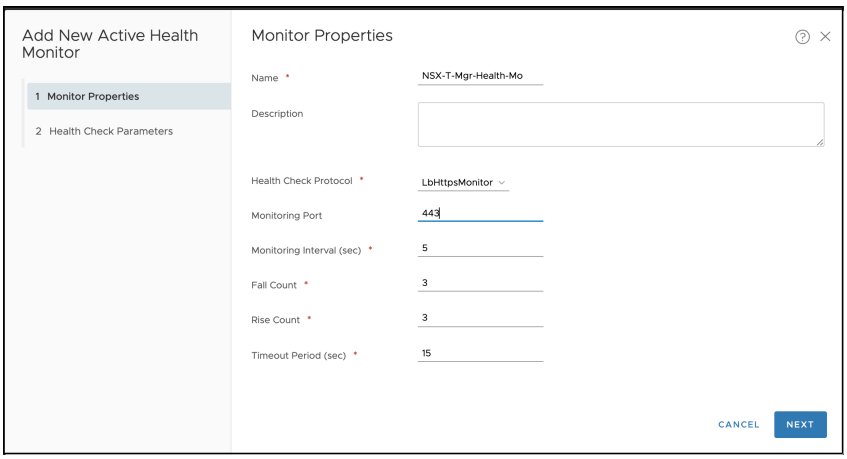
- Click **Health Monitor** > **Edit**



- Click **Create a new active monitor**

Configure **Monitor Properties**:

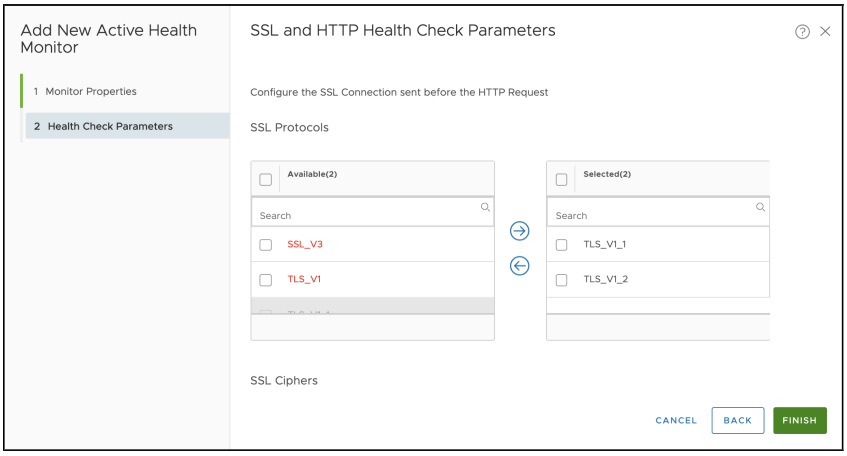
- **Name**: `NSX-T-Mgr-Health-Monitor`
- **Health Check Protocol**: `LbHttpsMonitor`
- **Monitoring Port**: `443`



Configure **Health Check Parameters**.

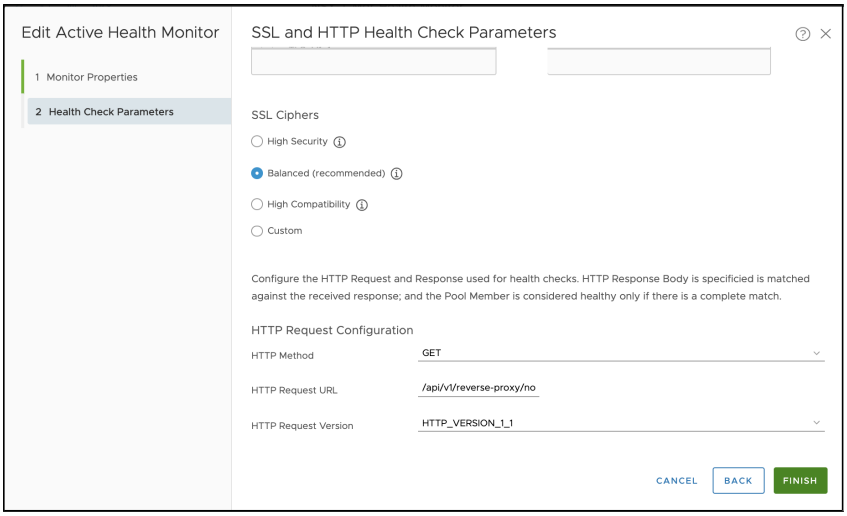
Configure the new Active HM with specific HTTP request fields as follows:

- **SSL Protocols**: Select the **TLS_v1** and **TLS_v2** protocols.
- **SSL Ciphers**: Select **Balanced** (recommended)



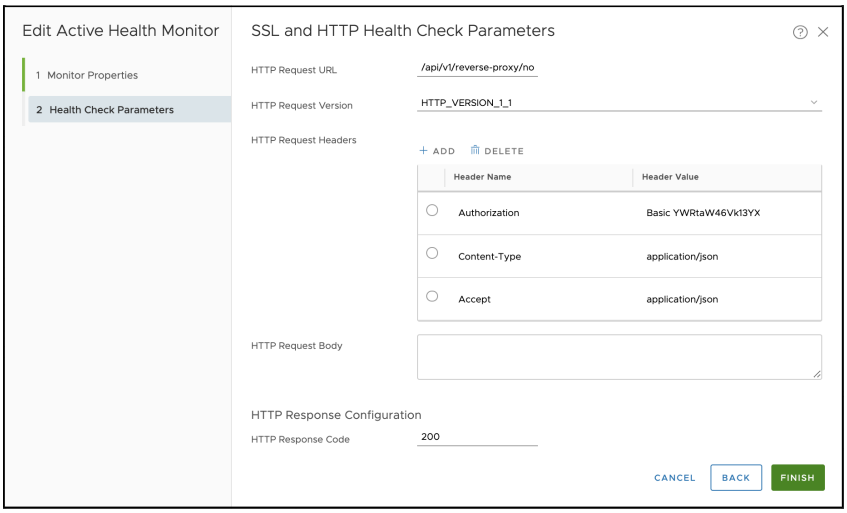
Configure the **HTTP Request Configuration** settings for the health monitor:

- **HTTP Method**: `GET`
- **HTTP Request URL**: `/api/v1/reverse-proxy/node/health`
- **HTTP Request Version**: `HTTP_VERSION_1_1`



Configure the **HTTP Request Headers** for the health monitor:

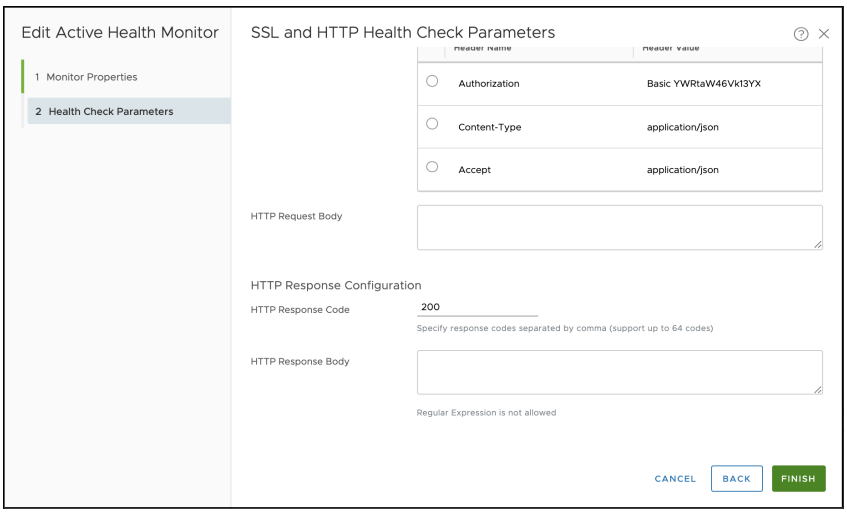
- **Authorization**: `Basic YWRtaW46Vk13YXJlMSE=`, which is the base64-encoded value of the NSX-T administrator credentials
- **Content-Type**: `application/json`
- **Accept**: `application/json`



> **Note:** In the example, *YWRtaW46Vk13YXJlMSE=* is the base64-encoded value of the NSX-T administrator credentials, expressed in the form *admin-user:password*. You can use the free online service www.base64encode.org to base64 encode your NSX-T administrator credentials.

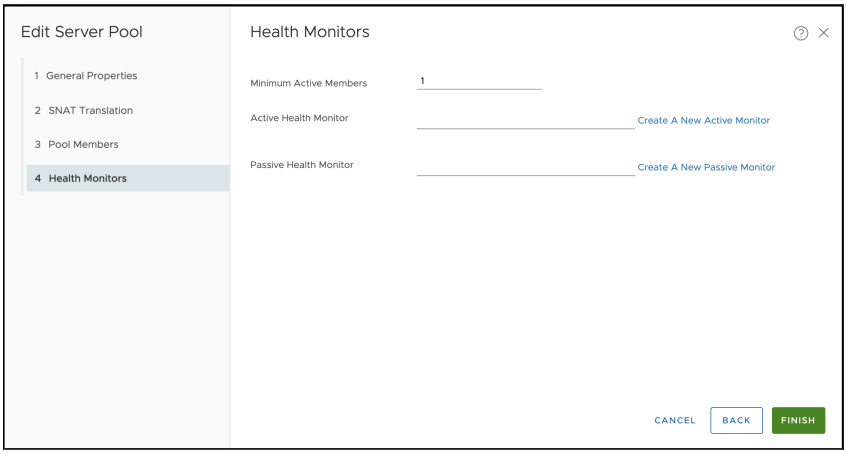
Configure the **HTTP Response Configuration** for the health monitor:

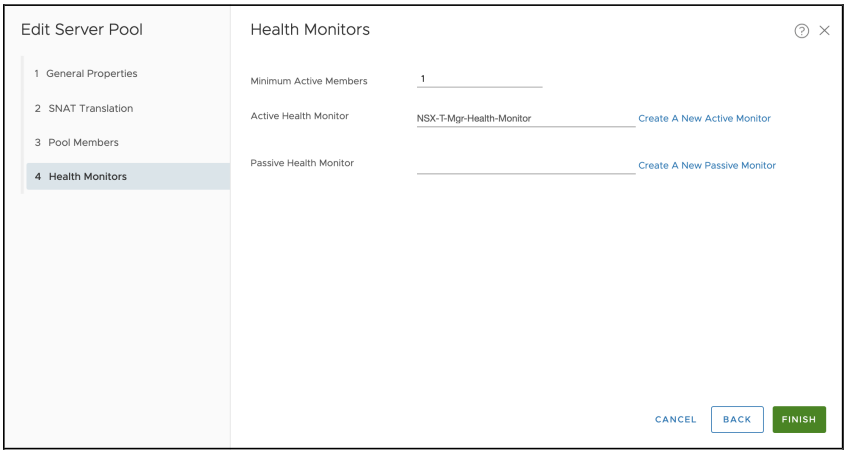
- **HTTP Response Code**: `200`
- Click **Finish**.



At the Health Monitors screen, specify the Active Health Monitor you just created:

- **Active Health Monitor**: Enter a name for the health monitor, such as **NSX-T-Mgr-Health-Monitor**.
- Click **Finish**.

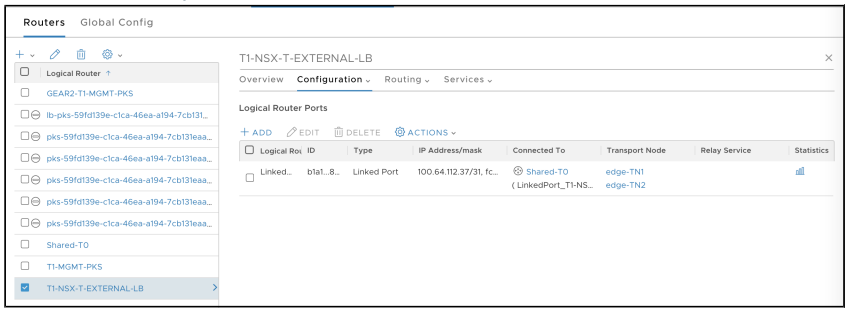




Step 12: Create SNAT Rule

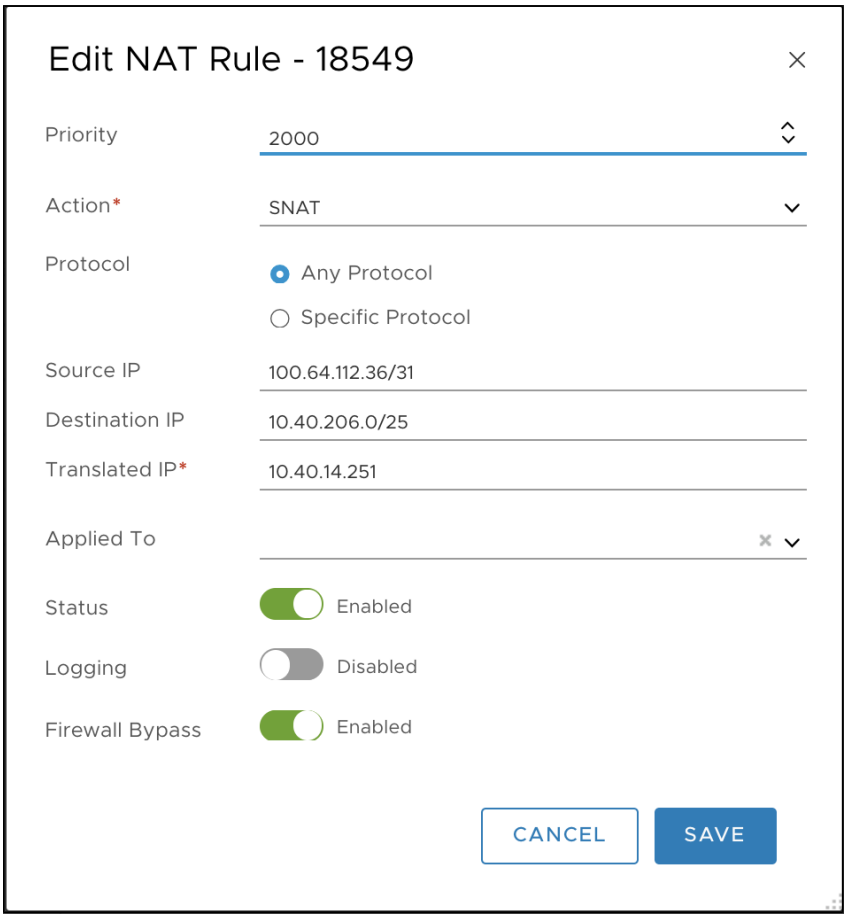
If your Enterprise PKS deployment uses NAT mode, make sure Health Monitoring traffic is correctly SNAT-translated when leaving the NSX-T topology. Add a specific SNAT rule that intercepts HM traffic generated by the load balancer and translates this to a globally-routable IP Address allocated using the same principle of the load balancer VIP. The following screenshot illustrates an example of SNAT rule added to the Tier0 Router to enable HM SNAT translation. In the example, `100.64.128.0/31` is the subnet for the Load Balancer Tier-1 uplink interface.

To do this you need to retrieve the IP of the T1 uplink (Tier-1 Router that connected the NSX-T LB instance). In the example below, the T1 uplink IP is `100.64.112.37/31`.

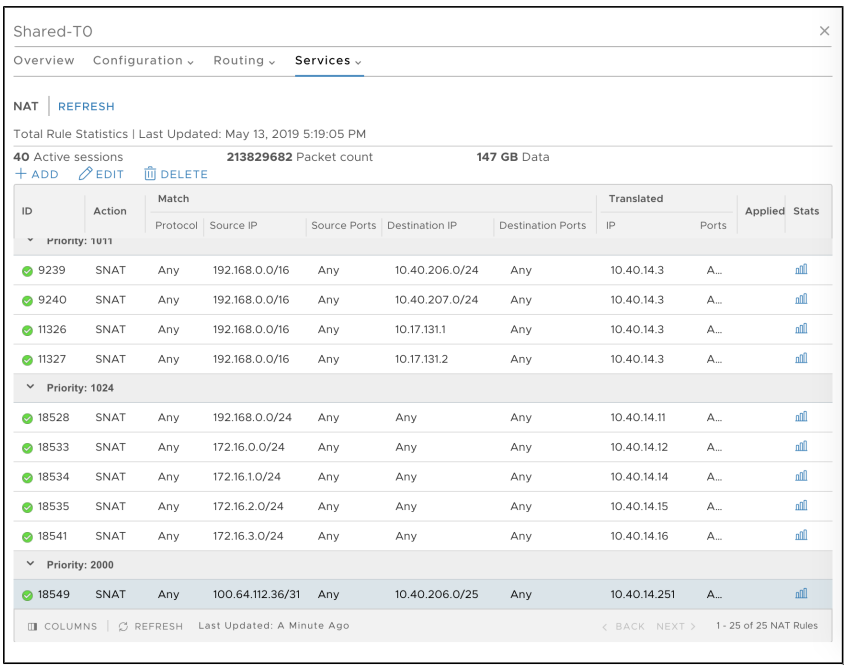


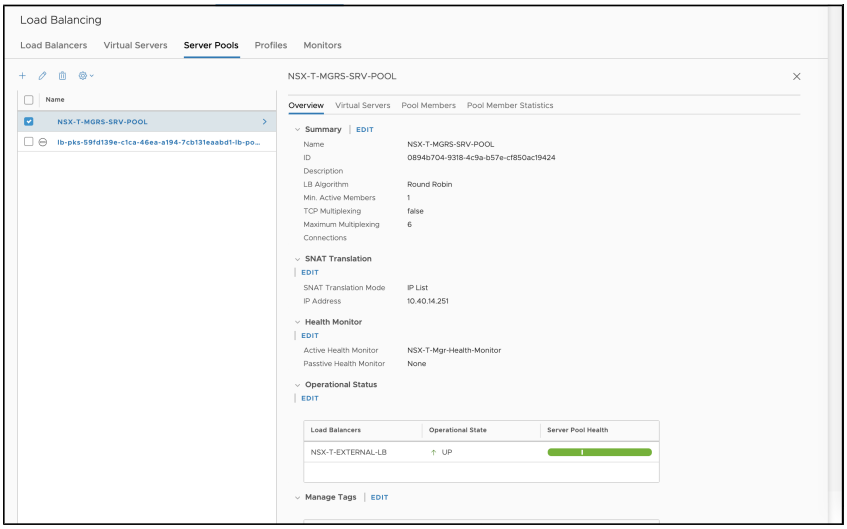
Create the following SNAT rule on the Tier-0 Router:

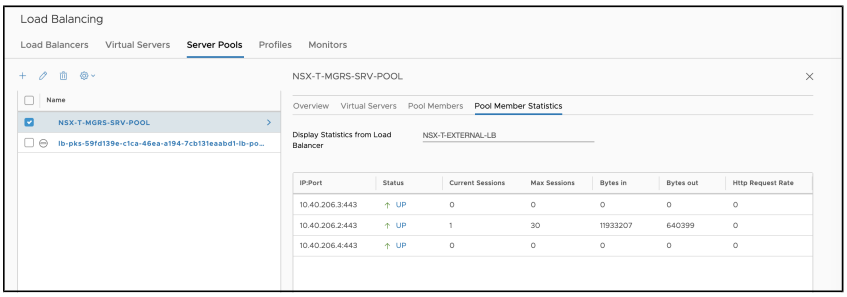
- **Priority**: `2000`
- **Action**: `SNAT`
- **Source IP**: `100.64.112.36/31`, for example
- **Destination IP**: `10.40.206.0/25`, for example
- **Translated IP**: `10.40.14.251`, for example
- Click **Save**



- Verify configuration of the SNAT rule and server pool health:



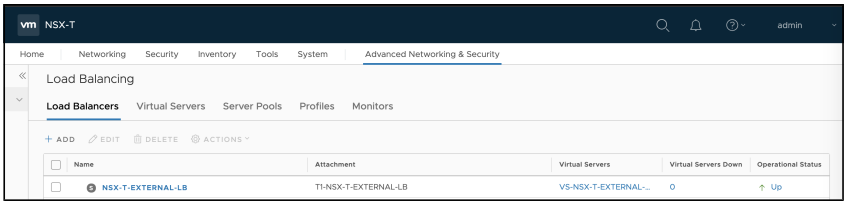




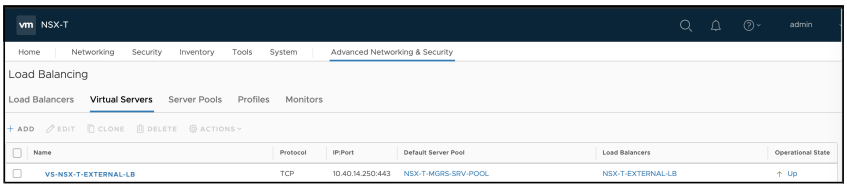
Step 13: Verify that NSX Manager Traffic Is Load Balanced

Verify the load balancer and that traffic is load balanced.

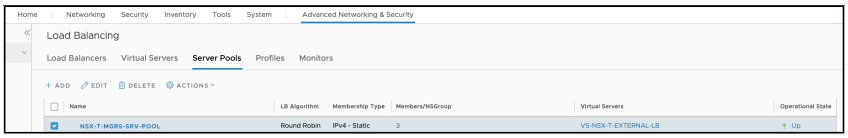
- Confirm that the status of the Logical Switch for the load balancer is Up.



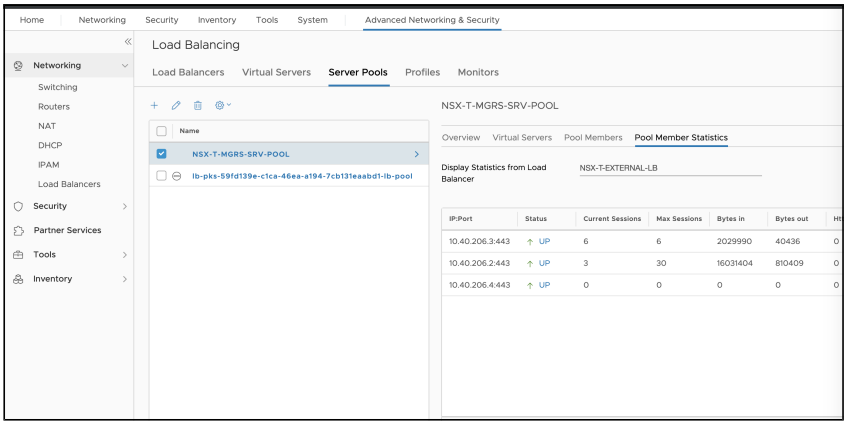
- Confirm that the status of the Virtual Server for the load balancer is Up.



- Confirm that the status of the Server Pool is Up.



- Open an HTTPS session using multiple browser clients and confirm that traffic is load-balanced across different NSX-T Managers:



You can use the NSX API to validate that secure HTTP requests against the new VIP address are associated with the load balancer's Virtual Server. Relying on the SuperUser Principal Identity created as part of PKS provisioning steps, you can cURL the NSX Management Cluster using the standard HA-VIP address or the newly-provisioned virtual server VIP. For example:

Before load balancer provisioning is completed:

```
curl -k -X GET "https://192.168.6.210/api/v1/trust-management/principal-identities" --cert $(pwd)/pks-nsx-t-superuser.crt --key $(pwd)/pks-nsx-t-superuser.key
```

After load balancer provisioning is completed:

```
curl -k -X GET "https://91.0.0.1/api/v1/trust-management/principal-identities" --cert $(pwd)/pks-nsx-t-superuser.crt --key $(pwd)/pks-nsx-t-superuser.key
```

Key behavioral differences among the two API calls is the fact that the call toward the Virtual Server VIP will effectively Load Balance requests among the NSX-T Server Pool members. On the other hand, the call made toward the HA VIP address would ALWAYS select the same member (the Active Member) of the NSX Management Cluster.

Residual configuration step would be to change PKS Tile configuration for NSX-Manager IP Address to use the newly-provisioned Virtual IP Address. This configuration will enable any component internal to PKS (NCP, NSX OSB Proxy, BOSH CPI, etc.) to use the new Load Balancer functionality.

NSX-T Installation Instructions Home

Installing and Configuring NSX-T for Enterprise PKS.

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

Installing and Configuring NSX-T Data Center for Enterprise PKS

In this topic

Page last updated:

This topic provides the workflow for installing and configuring NSX-T Data Center for use with VMware Enterprise PKS on vSphere.

Workflow for Installing and Configuring NSX-T Data Center

To perform a new installation of NSX-T Data Center for VMware Enterprise PKS, complete the following steps in the order presented.

1. Complete the prerequisites for installing NSX-T for Enterprise PKS.
2. Install the NSX Manager Unified Appliance for Enterprise PKSusing the OVA template file.
3. Deploy Two Additional NSX Manager Nodes and Form an NSX Management Cluster.
4. Configure a Virtual IP Address for the NSX-T Management Cluster.

   Note: Configure a VIP to provide high availability (HA) for the NSX-T Management Cluster.

5. Installing NSX Edge Nodes for Enterprise PKSusing the OVA template file.

   Note: You must install either a `medium` or `large` size Edge Node VM, or the bare metal Edge Node for Enterprise PKS.

6. Join Each Edge Node with the NSX-T Management Plane.
7. Enable the Repository Service on Each NSX-T Manager Node.
8. Create an IP Pool for Tunnel Endpoint IP Addresses.
9. Create Overlay and VLAN Transport Zones.
10. Create Edge Node Uplink Profile.
11. Create Edge Transport Nodes.
12. Create an Edge Cluster.
13. Create a Tier-0 Logical Router.

    Note: If you are using NAT-mode, you must configure the T0 router in Active-Standby mode.

14. Configure Edge Node High Availability (HA).

15. Configure ESXi Hosts as Transport Nodes.
16. (Optional) Update the NSX-T Admin Password Interval.

    Note: By default the NSX-T admin password expires in 90 days.

17. (Optional) Provision a Load Balancer for the NSX-T Management Cluster.

    Note: To scale, provision an NSX-T load balancer in place of the NSX-T Management Cluster VIP.

18. (Reference) Verify NSX VM Deployment for Enterprise PKS.

    Note: Refer to this topic to verify the installation of the NSX-T Manager and Edge Node VMs.

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

Prerequisites for Installing NSX-T for Enterprise PKS

In this topic

Page last updated:

This topic provides the prerequisites for installing NSX-T Data Center for use with VMware Enterprise PKS on vSphere.

Prerequisites for Installing NSX-T for Enterprise PKS

Before you install NSX-T for Enterprise PKS, complete the following prerequisites:

1. Read the Enterprise PKS Release Notes for the supported versions of NSX-T Data Center.
2. Read the NSX-T Data Center Release Notes for the NSX-T version you are installing.
3. Review the NSX-T installation documentation for the version of NSX-T you are installing.
4. Read the topics in Preparing to Install Enterprise PKS on vSphere with NSX-T Data Centerin the Enterprise PKS documentation.

Next Step

Install the NSX Manager Unified Appliance for Enterprise PKSusing the OVA template file.

NSX-T Installation Instructions Home

Installing and Configuring NSX-T for Enterprise PKS.

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

Install the NSX Manager Unified Appliance for Enterprise PKS

In this topic

Page last updated:

This topic provides instructions for installing the NSX Manager VM on vSphere for use with Enterprise PKS.

Prerequisites

Make sure you have completed the Prerequisites for Installing NSX-T for Enterprise PKS.

About the NSX Manager Appliance

The NSX-T Manager VM is provided as an OVA file named the **NSX Unified Appliance** that you import into your vSphere environment and configure.

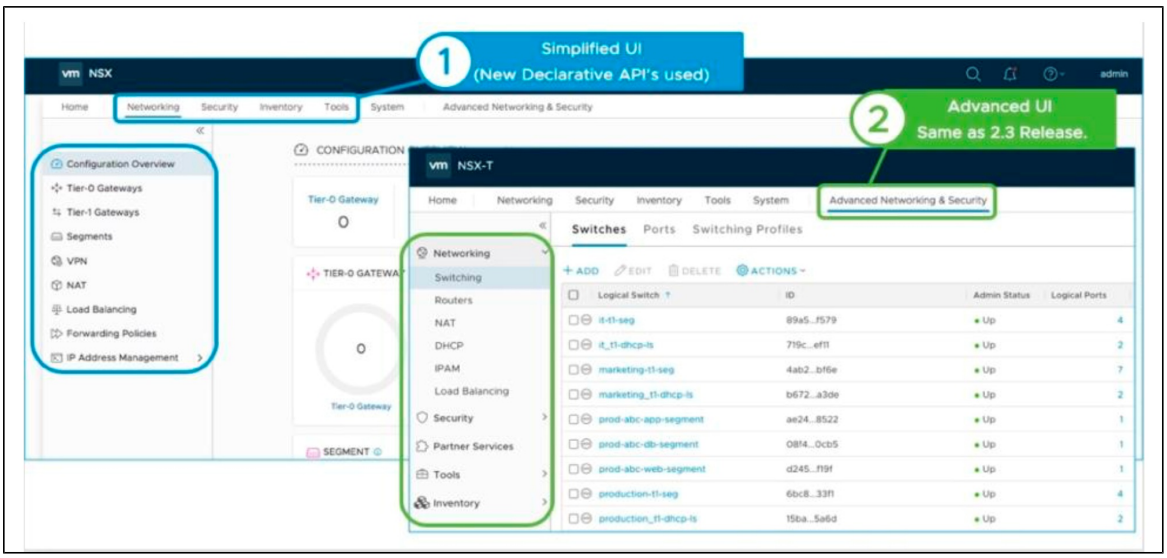
For more information, see Install the NSX Manager Unified Appliance in the NSX-T Data Center documentation. See also NSX Manager VM System Requirements.

About the NSX Manager UI

The NSX-T Manager provides the user interface and API for NSX-T Data Center. There are two options to interact with NSX-T Manager:

1. Simplified UI/API
    - Declarative interface that uses the 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.
    - 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 user interface to address Enterprise PKS installation and upgrade use cases. Currently NCP only supports the Management API.
    - Will be deprecated over time; all features and use cases will eventually be transfered to the Simplified UI/API.

Enterprise PKS does not support the **Simplified UI/API**. For Enterprise PKS, you must use the **Advanced Networking and Security** tab to create, read, update, and delete required network objects. For NSX-T host preparation and configuration, such as deploying NSX-T Managers and Edge Nodes, use the **System** tab.



Install and Configure the NSX Manager Appliance

Complete the following procedure to deploy the NSX Manager appliance.

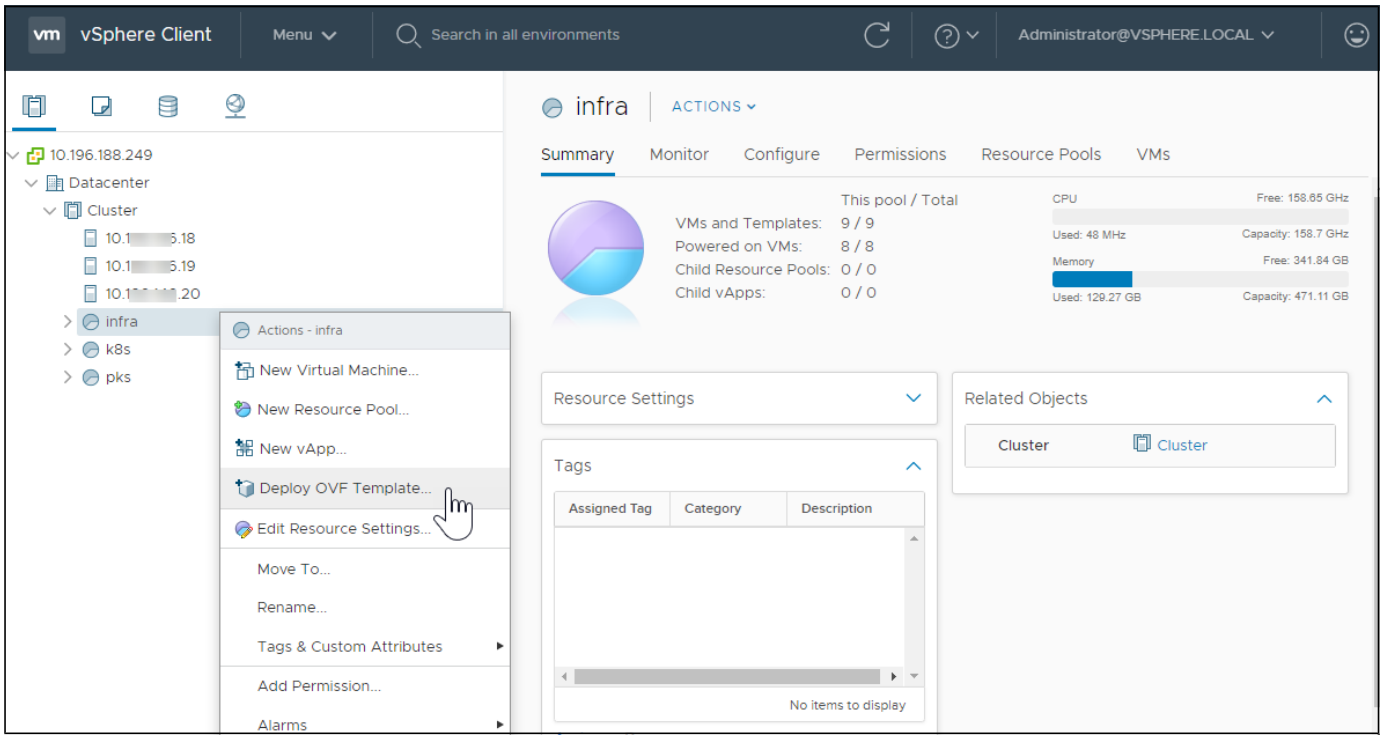
1. Locate the NSX-T Data Center OVA file (`nsx-unified-appliance-VERSION.ova`) on the VMware download portal and download it to your local machine.

2. Log in to vCenter using the vSphere Client.

3. Create a **Resource Pool** for NSX-T infrastructure, such as `infra` .

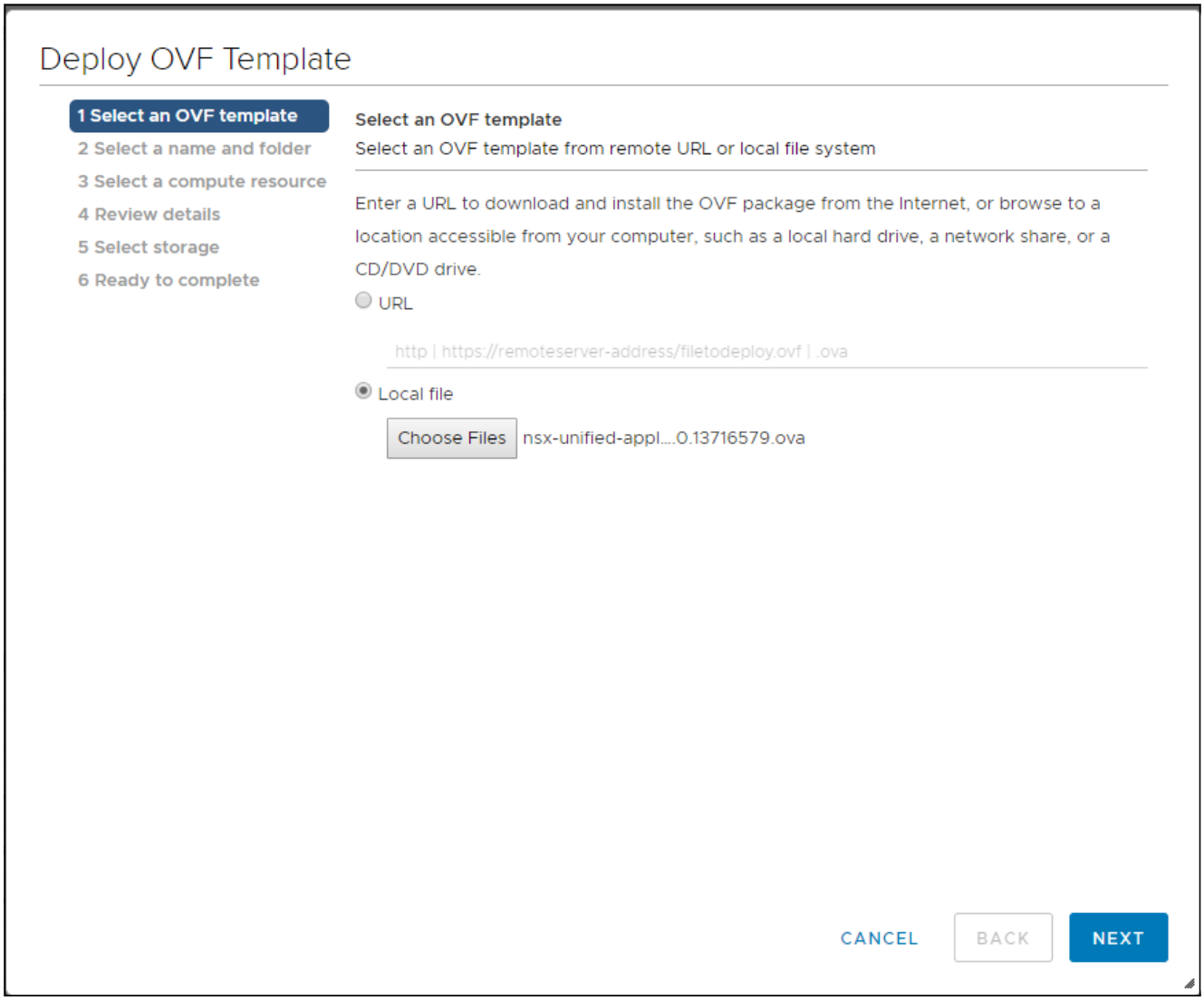
4. In the vSphere Client, select the **Resource Pool** where you want to install NSX-T Data Center.

5. Right-click and select **Deploy OVF Template** to start the installation wizard.



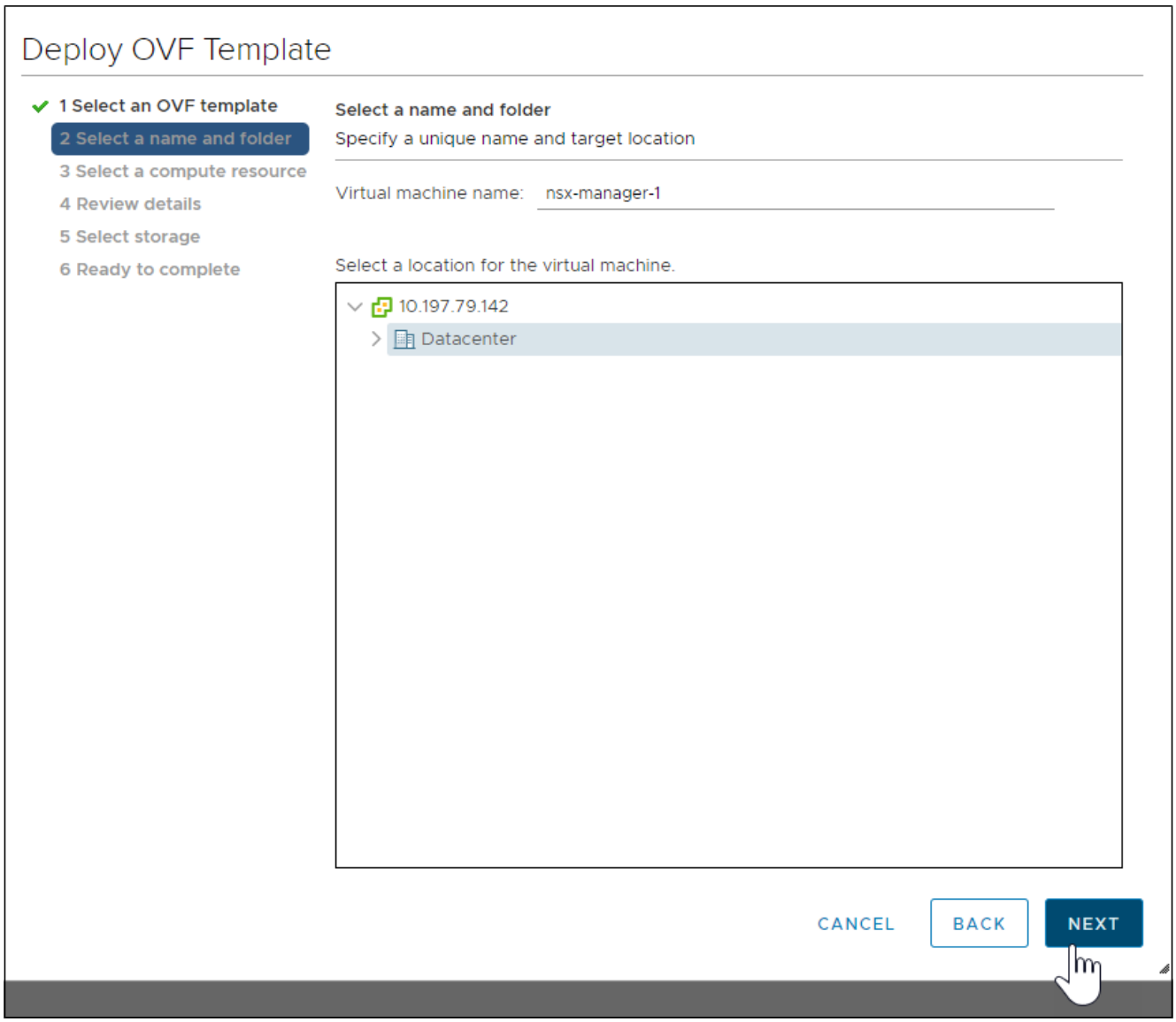
6. At the **Select an OVF template** screen:

- Select the **Local file** option
- Click **Choose Files**
- Navigate to where you downloaded the OVA file and select it
- Click **Next**

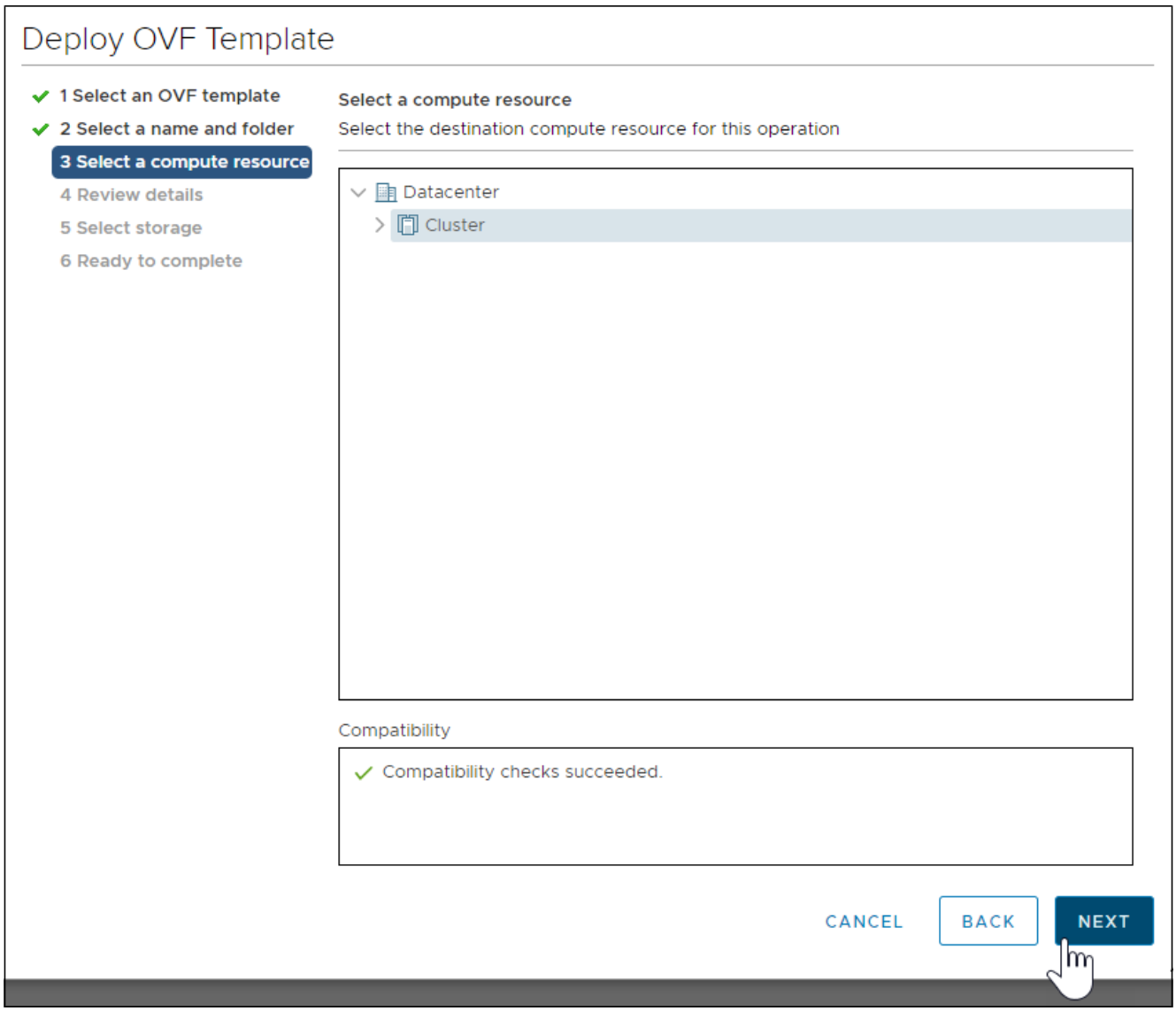


7. At the **Select a name and folder** screen:

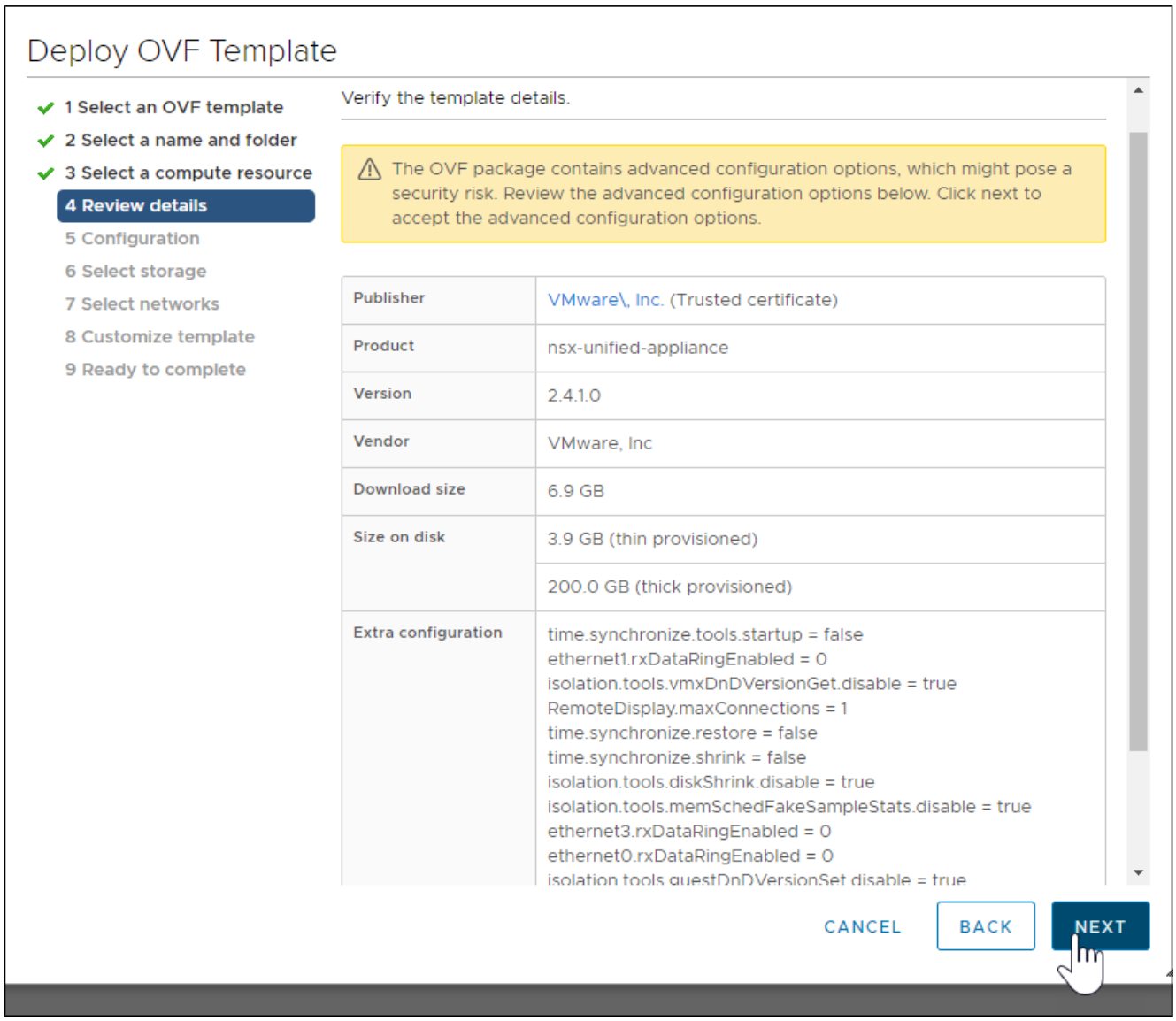
- Enter a name for the NSX Manager VM, such as `nsx-manager-1`
- Select the **Datacenter** for the VM deployment
- Click **Next**



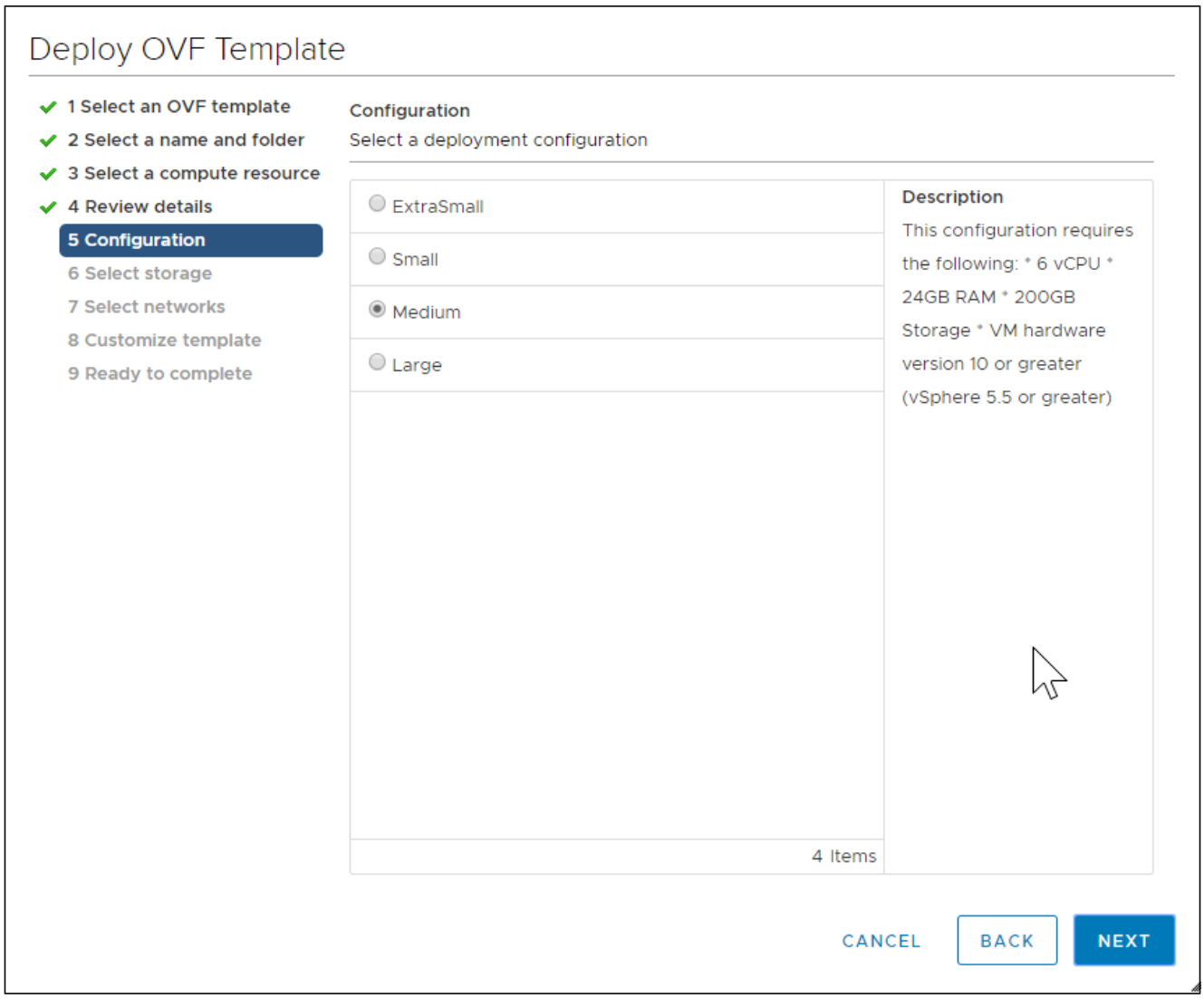
8. At the **Select a compute resource** screen:
   - Select the **Resource Pool** where the NSX Manager VM will be deployed
   - Click **Next**



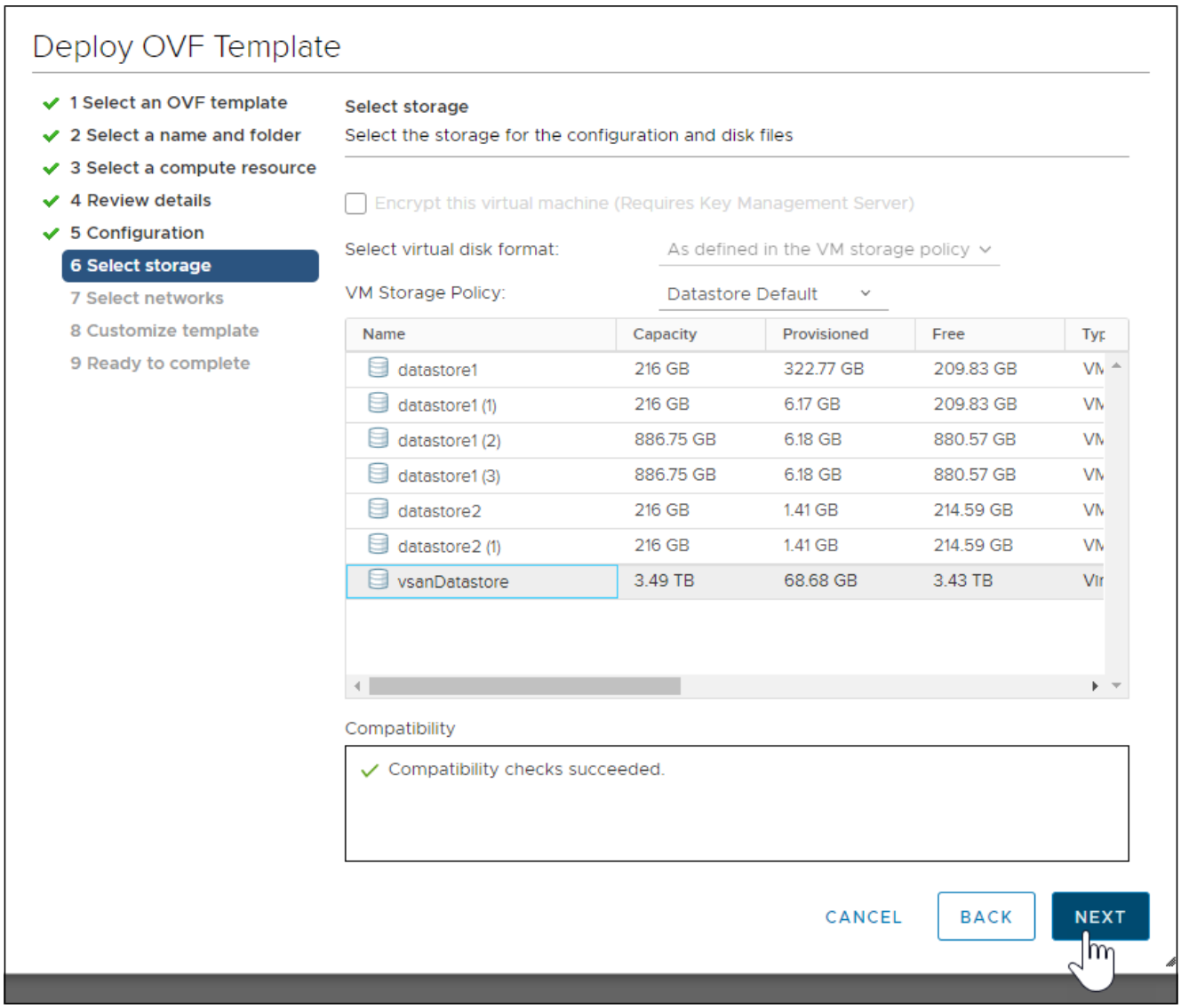
9. At the **Review details** screen:
   - Verify the OVF template details
   - Click **Next**



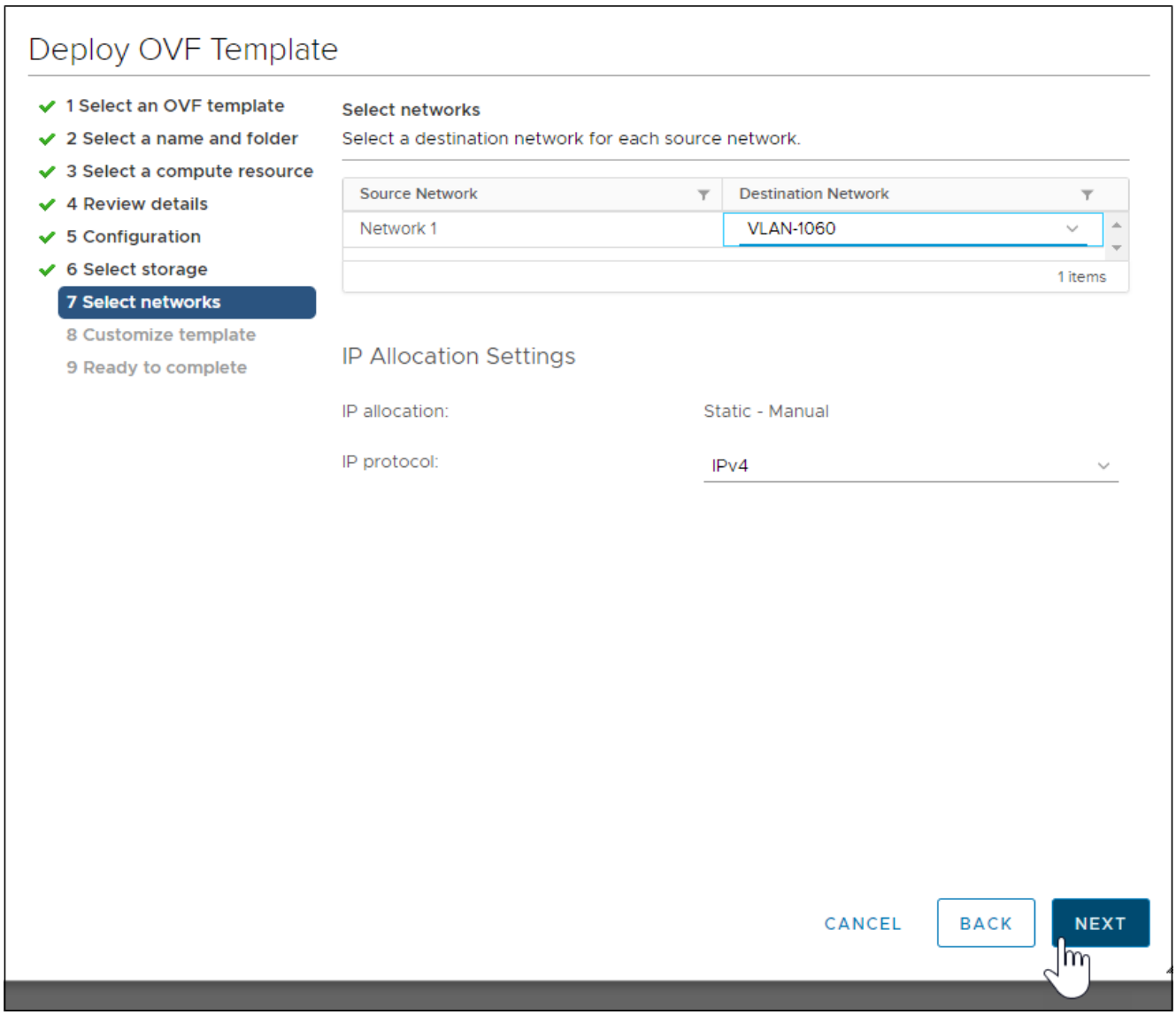
10. At the **Configuration** screen:
    - Select either the **Medium** size VM or the **Large** size VM
    - Click **Next**



11. At the **Select storage** screen:
    - Select the **vsanDatastore** if you are using vSAN, or a dedicated datastore if you are not
    - Click **Next**

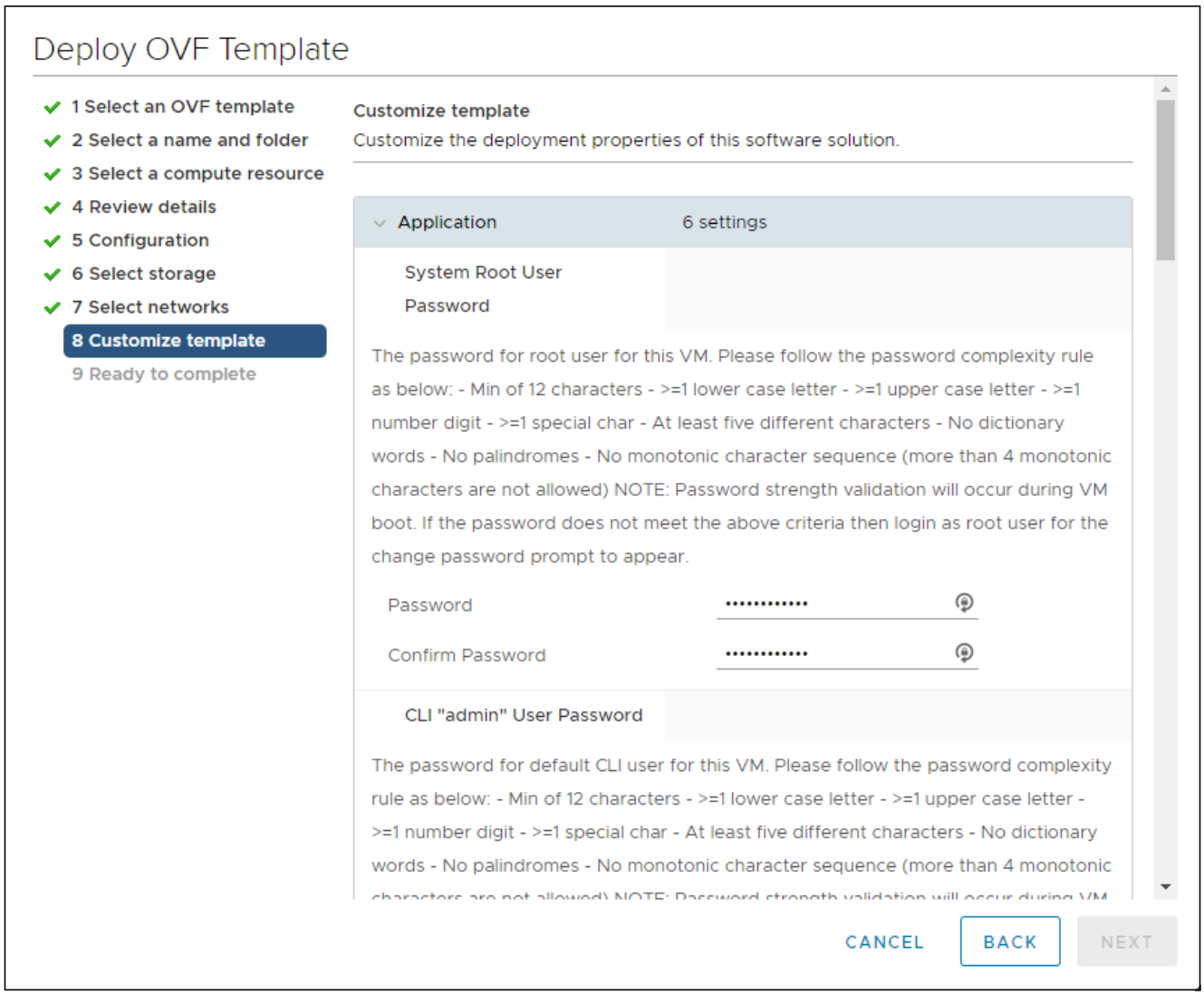


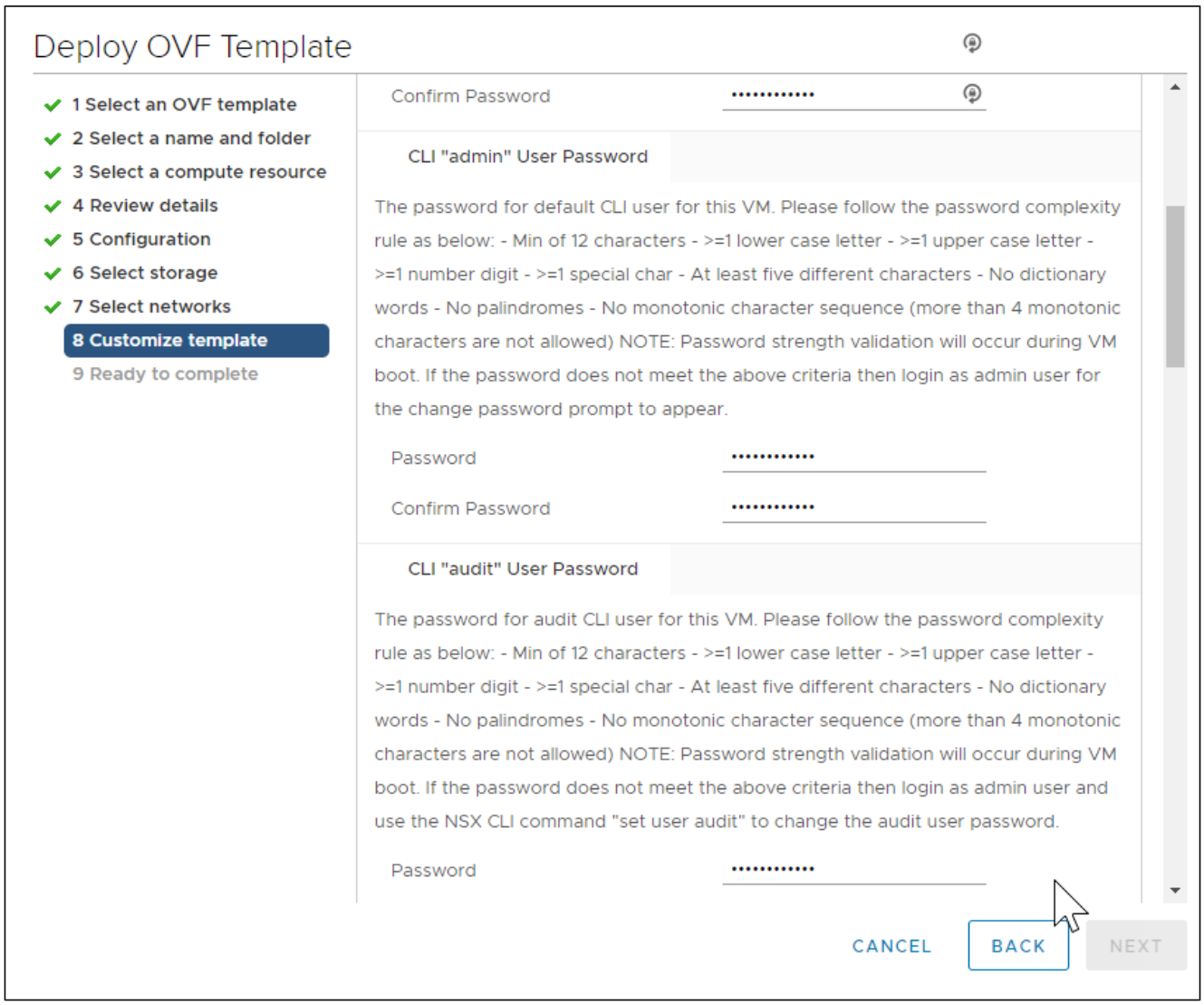
12. At the **Select networks** screen:
    - Select **Destination Network** for the NSX Manager VM
    - Click **Next**

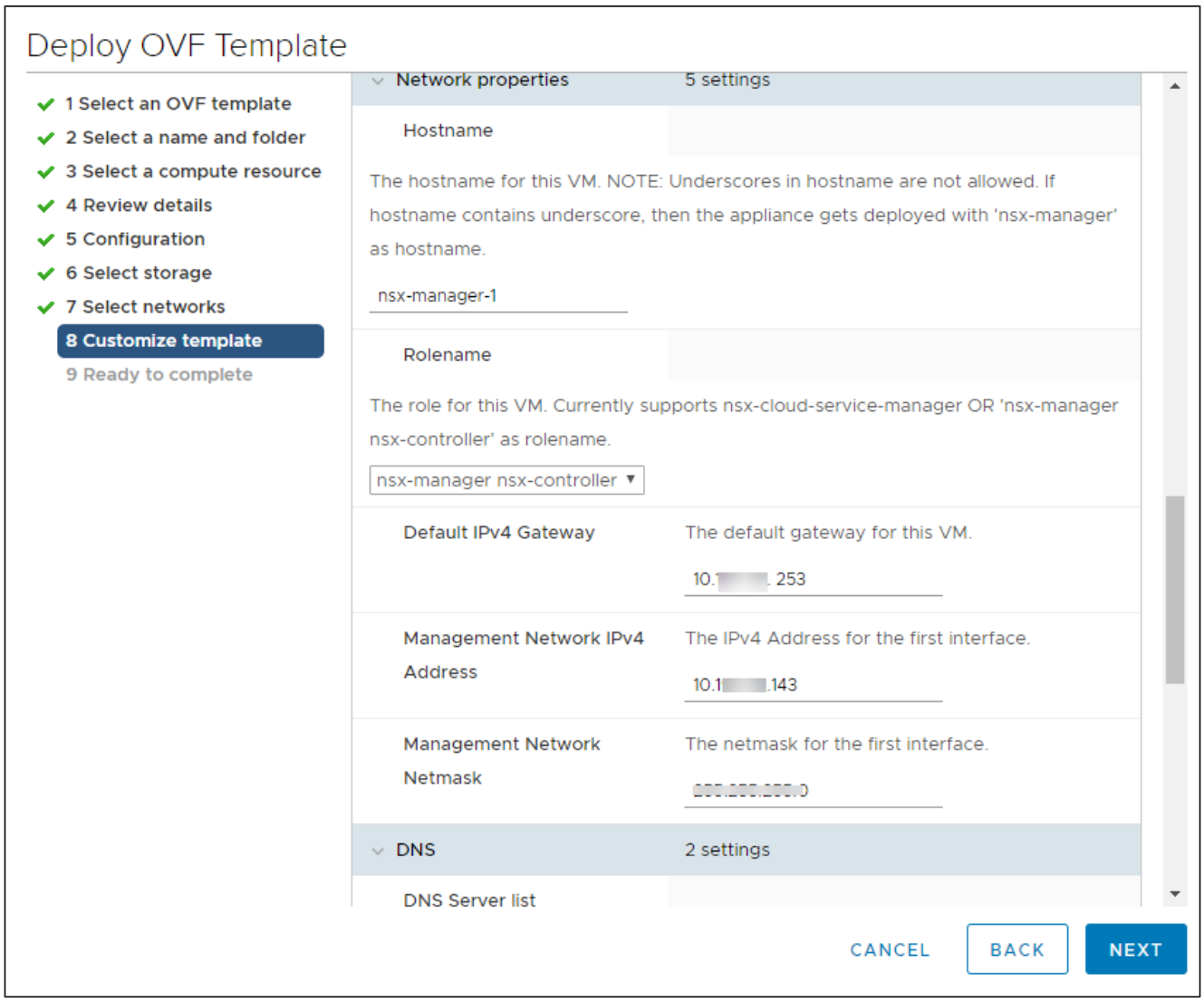


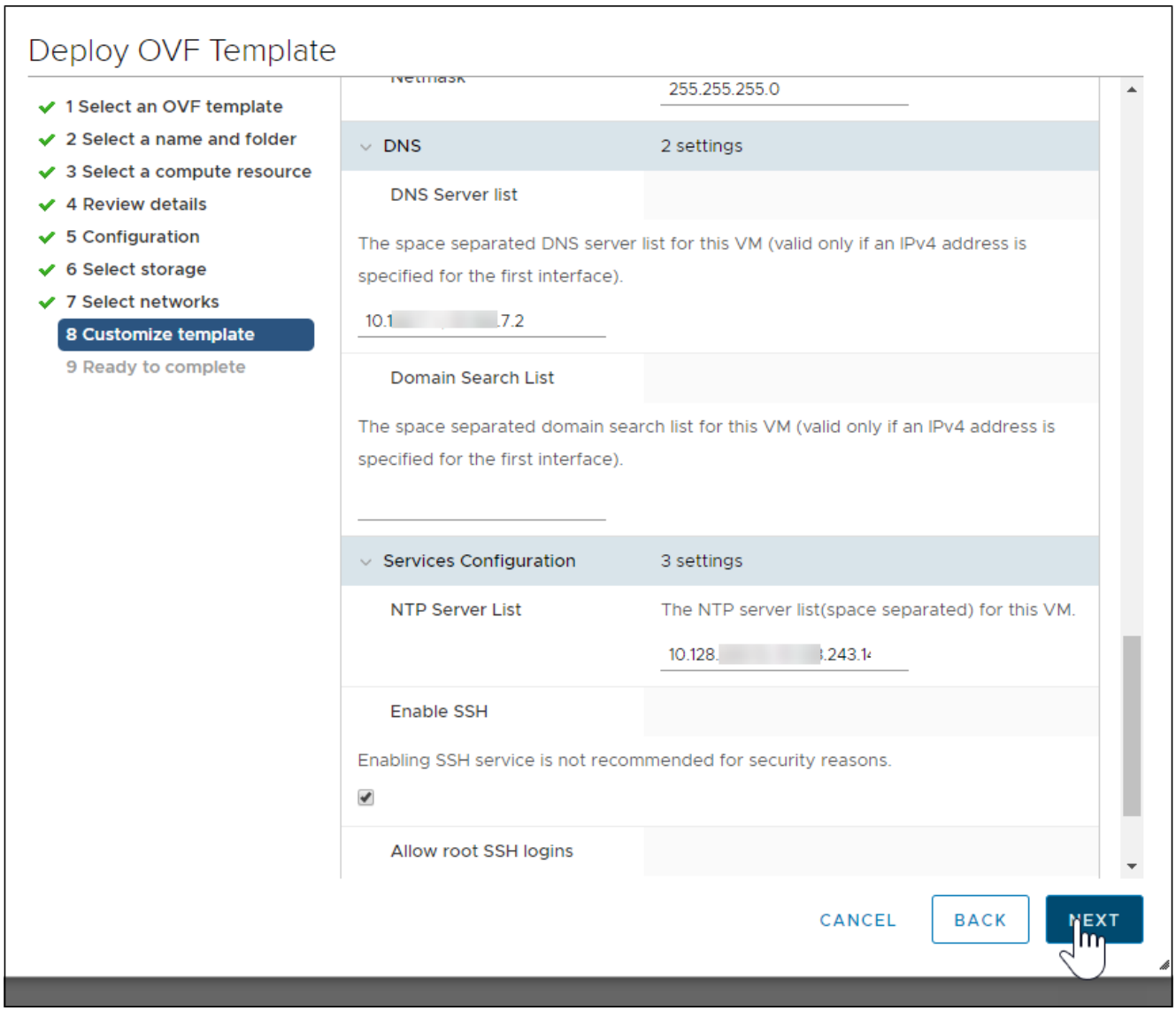
13. At the **Customize Template** screen, configure the following settings:

    - **System Root User Password** (must comply with password strength restrictions)
    - **CLI "admin" User Password** (must comply with password strength restrictions)
    - **CLI "audit" User Password** (must comply with password strength restrictions)
    - **Hostname**: Enter the hostname for the NSX Manager VM, such as `nsx-manager-1`
    - **Default IPv4 Gateway**: Enter the default gateway IPv4 address for the NSX Manager VM
    - **Management Network IPv4 Address**: Enter the IPv4 address for the first network interface
    - **Management Network Netmask**: Enter the netmask for the first network interface
    - **DNS Server List**: Enter one or more DNS servers (space-separated if multiple)
    - **NTP Server List**: One or more NTP servers (space-separated if multiple)
    - **Enable SSH**: Select Enable SSH (by default this option is disabled for security reasons)
    - **Allow root SSH logins**: Enable this option (by default this option is disabled for security reasons)
    - Click **Next**

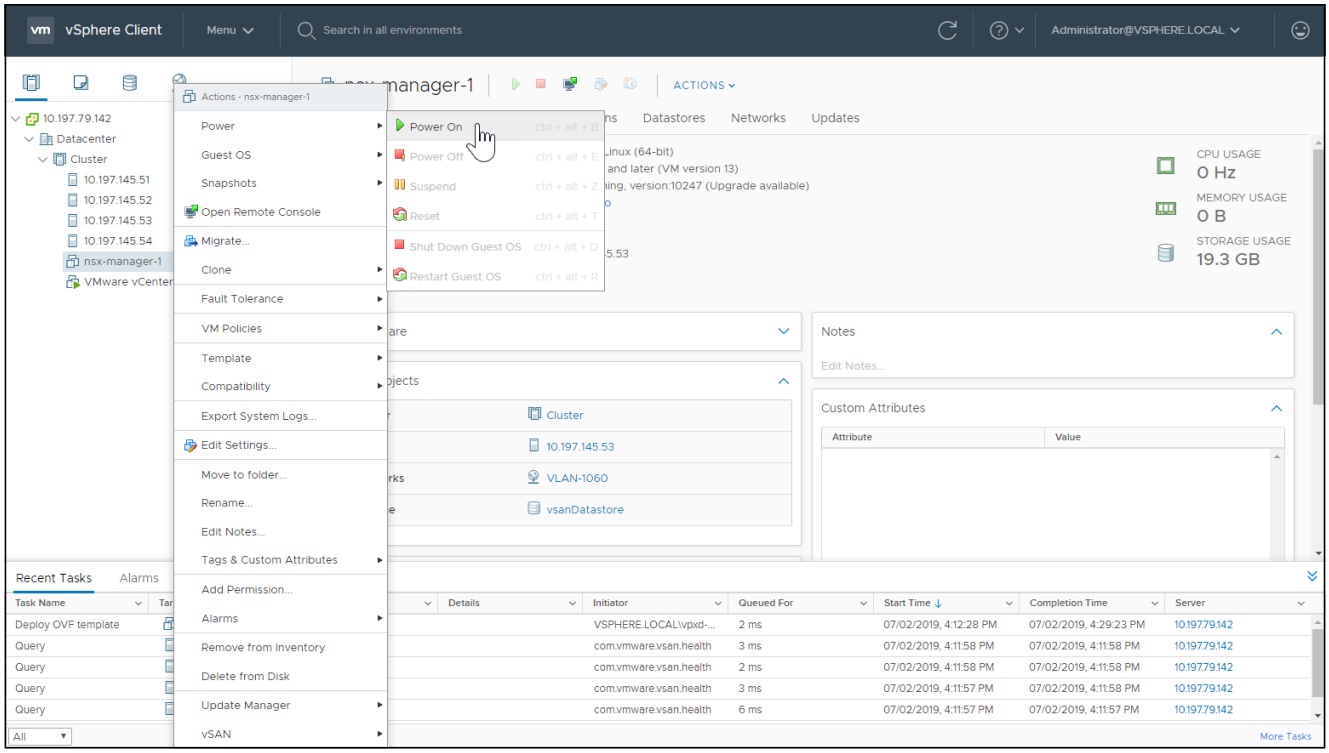








14. At the **Ready to complete** screen:
    - Verify that the OVF template specification is accurate
    - Click **Finish** to begin the installation. It will taker approximately 10 minutes to complete.



15. Expand the **Recent Tasks** panel at the bottom of the vCenter screen to view the progress of the OVA deployment

Verify NSX-T Manager Installation

See Verify NSX-T VM Deployment for Enterprise PKS.

Next Step

See Deploy Two Additional NSX Manager Nodes and Form an NSX-T Management Cluster

Installation Instructions

See Installing and Configuring NSX-T for Enterprise PKS.

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

Install Additional NSX Manager Nodes and Configure the NSX Management Cluster

In this topic

Page last updated:

This topic provides instructions for installing additional NSX Manager nodes and forming the NSX Management Cluster for use with Enterprise PKS.

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

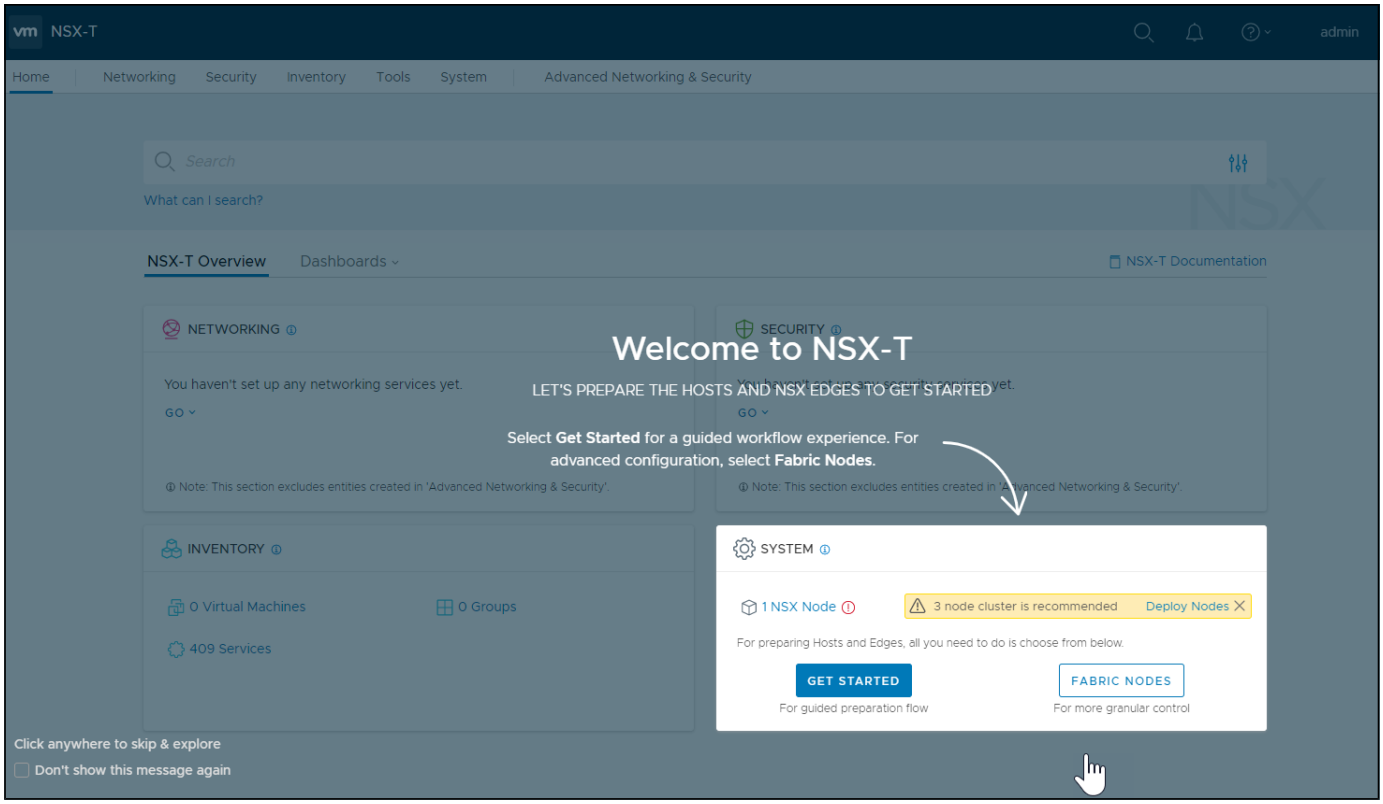
About the NSX-T Management Cluster

The NSX-T Management Cluster is a collection of three NSX Manager nodes for high-availability. For more information, see Deploy NSX Manager Nodes to Form a Cluster in the NSX-T Data Center documentation.

All repository details and user credentials are synchronized NSX-T Manager Nodes in the management cluster. After the NSX-T Management Cluster is formed, you define a virtual IP address (VIP) to access the management cluster.

Log In to the NSX Manager Web UI

1. From your browser, log in with admin privileges to NSX Manager at `https://<nsx-manager-ip-address>`.
2. For first time log ins, read and accept the EULA terms.
3. Select whether to join the VMware's Customer Experience Improvement Program (CEIP) and click **Save**.
4. At the **Welcome to NSX-T** screen, the system indicates that you have 1 NSX-T Manager node installed, and that a "3 node cluster is recommended."

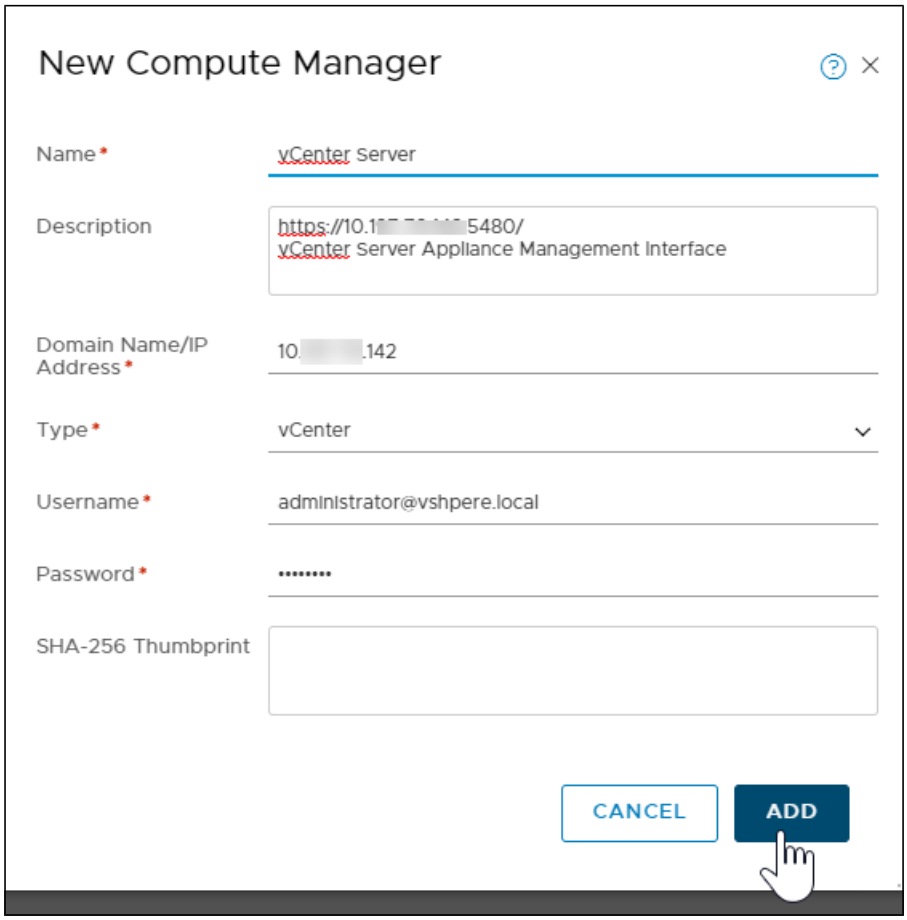


Add vCenter as the Compute Manager

Before you can add additional NSX-T Manager Nodes, you must specify a Compute Manager. A compute manager is an application that manages resources such as hosts and VMs. For Enterprise PKS we use vCenter as the compute manager.

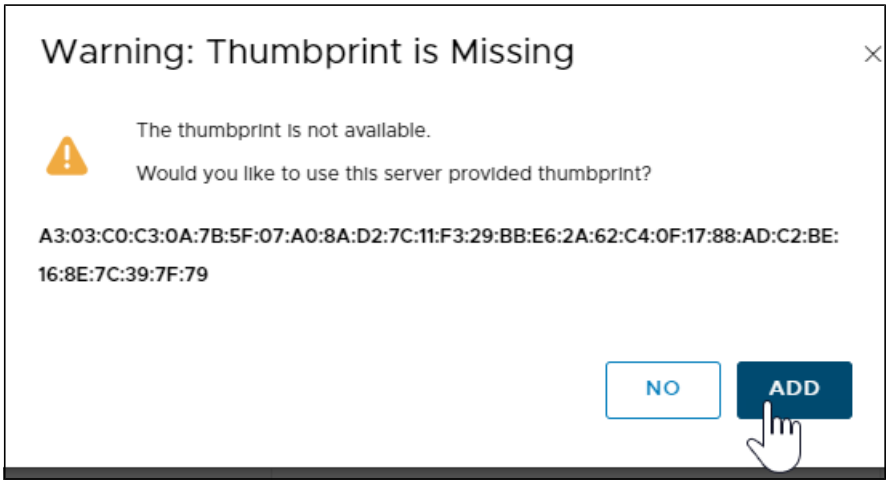
Complete the following steps to add vCenter as the Compute Manager.

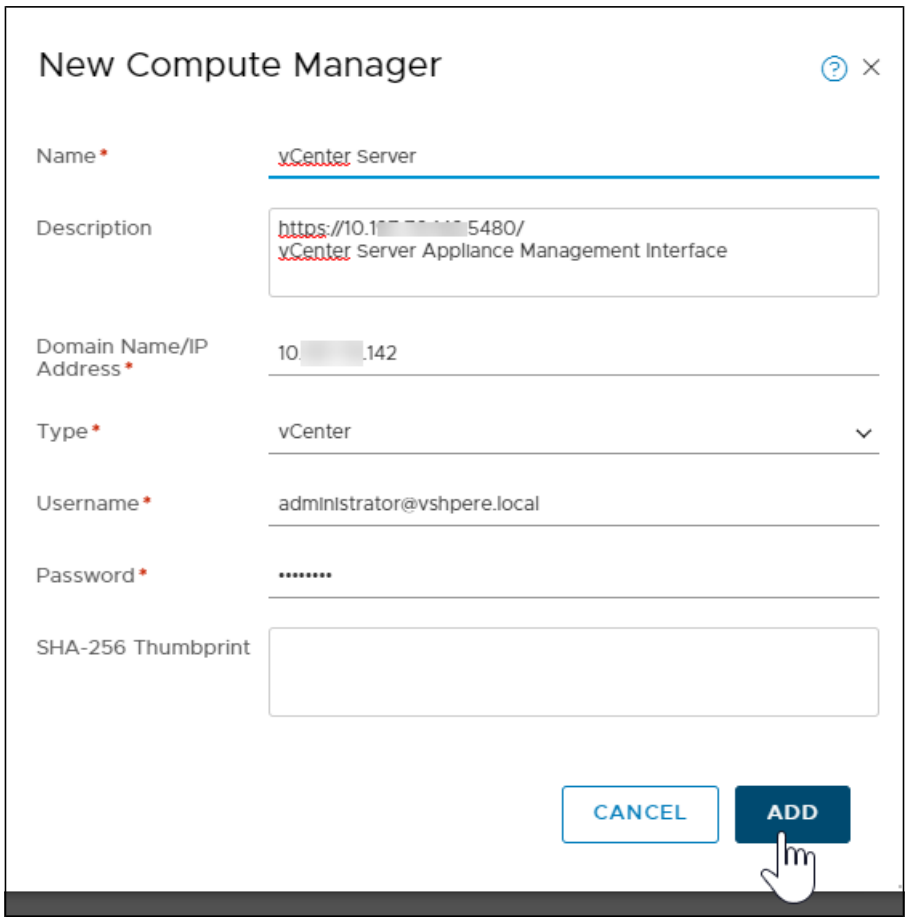
1. In NSX-T Manager, select **System** > **Fabric** > **Compute Managers** > **Add**.
2. Configure the Compute Manager as follows:
   - **Name**: Enter the name to identify the vCenter Server.
   - **Description**: Optionally add details such as the number of clusters in the vCenter Server.
   - **Domain Name/IP Address**: Enter the IP address of the vCenter Server.
   - **Type**: vCenter (keep the default option).
   - **Username** and **Password**: Enter the vCenter Server login credentials.
   - **SHA-256 Thumbprint**: Leave the thumbprint value blank.
3. Click **Add**.



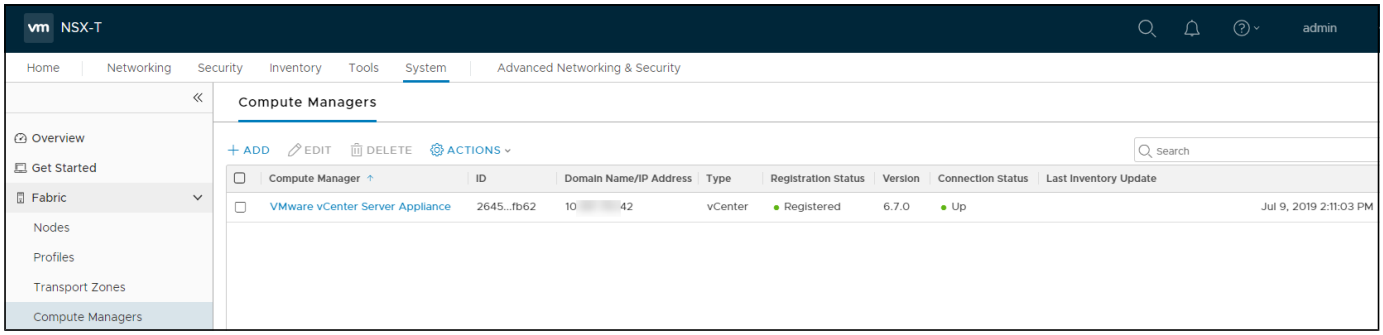
4. Click **Add** again.

   - Since you left the thumbprint value blank, you are prompted to accept the server-provided thumbprint.
   - After you accept the thumbprint, it takes a few seconds for NSX-T Data Center to discover and register the vCenter Server resources.





5. Verify that the vCenter Compute Manager is successfully **Registered** and that its connection status is **Up**.



6. If the Compute Manager is not registered, troubleshoot as necessary. See the Add a Compute Manager in the NSX-T Data Center documentation for guidance.

Deploy Additional NSX Manager Nodes

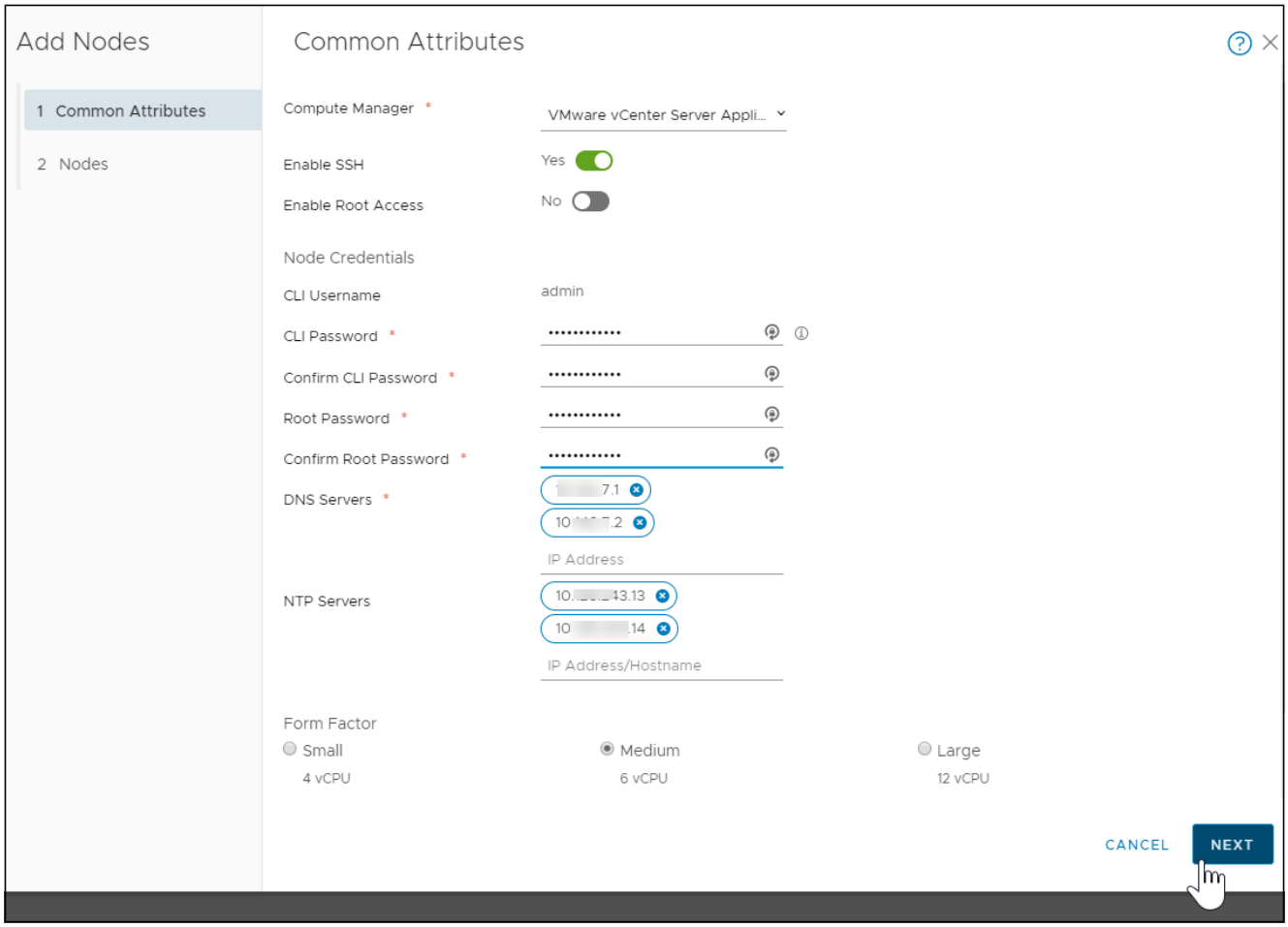
You can deploy two additional NSX-T Manager nodes using the NSX-T user interface. For more information, see Deploy NSX Manager Nodes to Form a Cluster from UI in the NSX-T Data Center documentation.

Complete the following procedure to deploy additional manager nodes and form the NSX-T Management Cluster.

1. In NSX-T Manager, select **System** > **Overview** > **Add Nodes**.

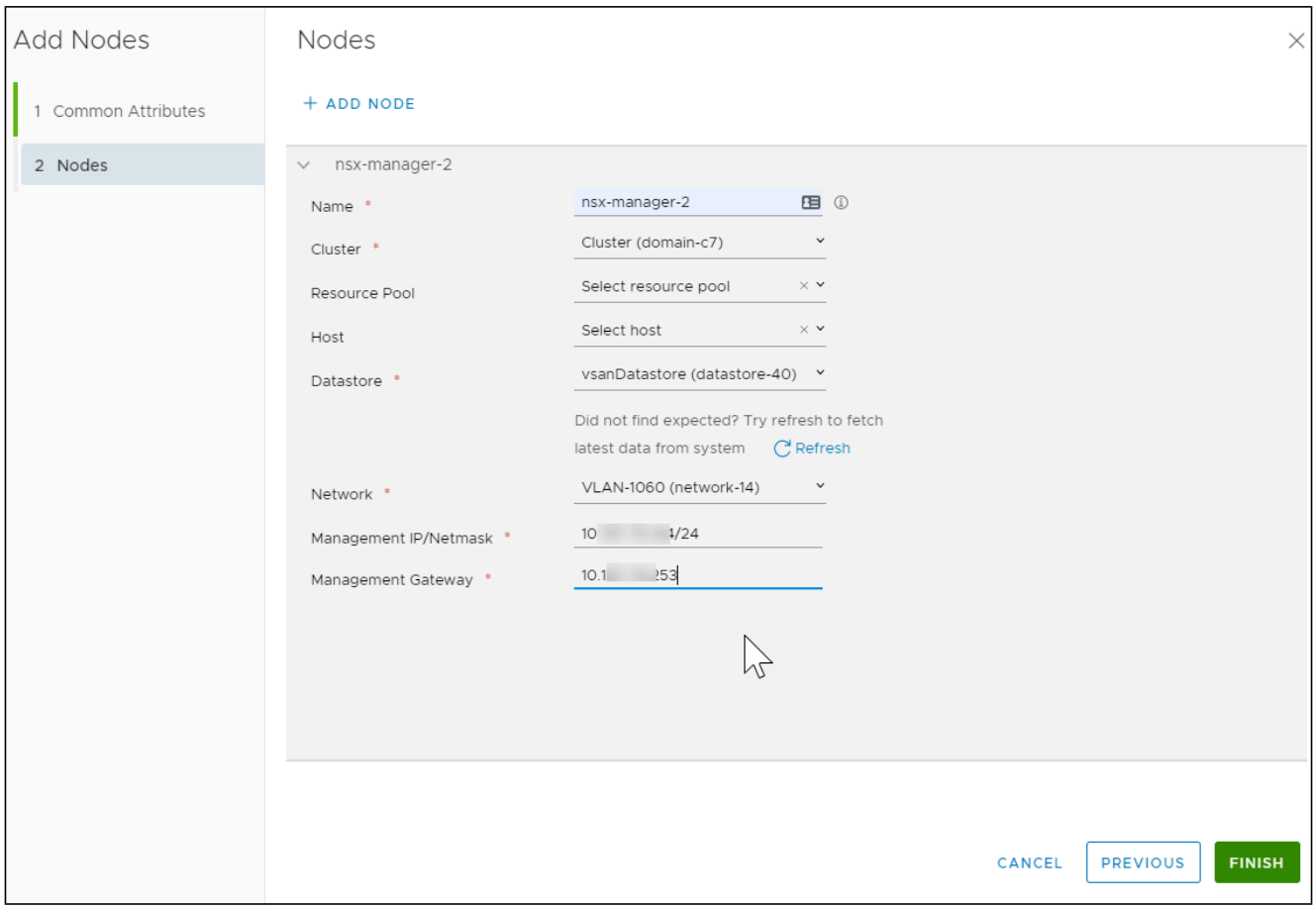
2. Enter the NSX-T Manager common attribute details as follows:

- **Compute Manager**: Registered resource compute manager is populated.
- **Enable SSH**: Toggle the button to allow an SSH login to the new NSX Manager node.
- **Enable Root Access**: Toggle the button to allow the root access to the new NSX Manager node.
- **CLI Username and Password Confirmation**: The CLI username is set to `admin`. The password must comply with the password strength restrictions.
- **Root Password and Password Confirmation**: Set the root password for the new node. The password must comply with the password strength restrictions.
- **DNS Servers**: Enter the DNS server IP address available in the vCenter Server.
- **NTP Servers**: Enter the NTP server IP address.
- **Form Factor**: Select **Medium** (default) or **Large**. Do not select **Small**.



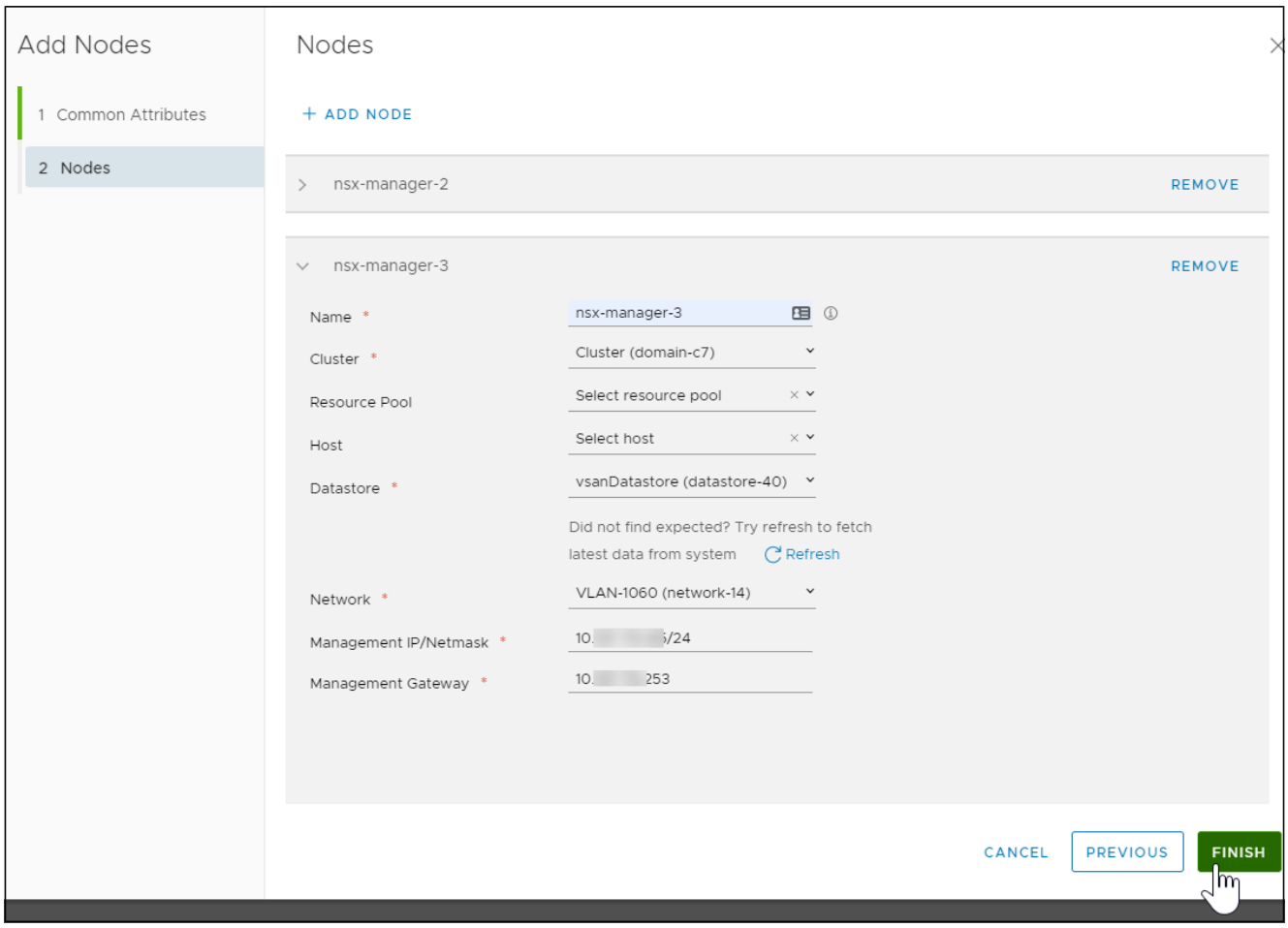
3. Enter the second NSX-T Manager node details.

- **Name**: Enter a name for the NSX-T Manager node, such as `nsx-manager-2`.
- **Cluster**: Designate the cluster the node is going to join from the drop-down menu.
- **Resource Pool or Host**: Select the **infra** resource pool from the drop-down menu.
- **Datastore**: Select a datastore for the node files from the drop-down menu.
- **Network**: Assign the network from the drop-down menu.
- **Management IP/Netmask**: Enter the IP address and netmask.
- **Management Gateway**: Enter the gateway IP address.

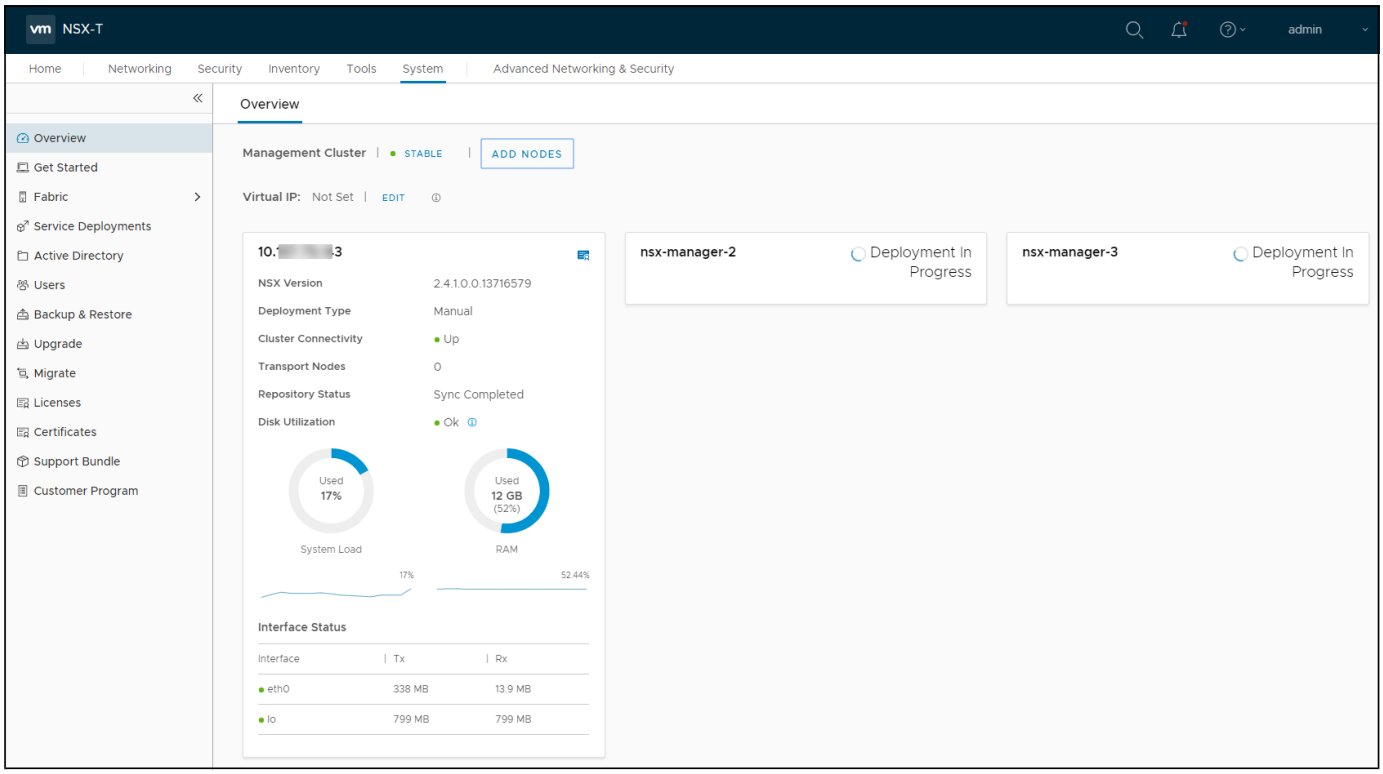


4. Click **Add Node** and configure a third NSX Manager node.

   - **Name**: Enter a name for the NSX-T Manager node, such as `nsx-manager-3`.
   - **Cluster**: Designate the cluster the node is going to join from the drop-down menu.
   - **Resource Pool or Host**: Select the **infra** resource pool from the drop-down menu.
   - **Datastore**: Select a datastore for the node files from the drop-down menu.
   - **Network**: Assign the network from the drop-down menu.
   - **Management IP/Netmask**: Enter the IP address and netmask.
   - **Management Gateway**: Enter the gateway IP address.



5. Click **Finish** and the new NSX-T Manager nodes are deployed.



Verify the NSX Management Cluster

1. Verify the deployment of the new NSX Manager nodes.
   - At the **System** > **Overview** > **Management Cluster**, you should see the deployment of the additional nodes.
   - Wait for 10-15 minutes for the deployment, cluster formation, and repository synchronization to complete.
   - Track the deployment process at the **System** > **Overview** screen in vCenter Server.
2. See Verify NSX VM Deployment for Enterprise PKS to verify the installation of each additional NSX-T Manager node.

Next Step

See Configure VIP Address for the NSX-T Management Cluster.

NSX-T Installation Instructions Home

Installing and Configuring NSX-T for Enterprise PKS.

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

Configure VIP Address for the NSX Management Cluster

In this topic

Page last updated:

This topic explains how to configure a Virtual IP address (VIP) for the NSX-T Management Cluster for VMware Enterprise PKS.

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

About the NSX Management Cluster VIP

The NSX-T Management Cluster comprises three NSX-T Manager nodes. To support a single access point for the NSX-T Manager user interface and API, assign a virtual IP Address (VIP) to the NSX-T Management Cluster. Once the VIP is assigned, any UI and API requests to NSX-T are redirected to the virtual IP address of the cluster, which is owned by the leader node. The leader node then routes the request forward to the other components of the appliance.

Using a VIP makes the NSX Management Cluster highly available. If you need to scale, an alternative to the VIP is to provision a load balancer for the NSX-T Management Cluster. Provisioning a load balancer requires that NSX-T be fully installed and configured. It is recommended that you configure the VIP now, then install a load balancer ater NSX-T is installed and configured, and only if needed.

Configure a VIP Address for the NSX-T Management Cluster

Complete the following instructions to create a VIP for the NSX Management Cluster. The IP address you use for the VIP must be part of the same subnet as the NSX-T Management nodes.

1. From your browser, log in with admin privileges to NSX Manager at `https://nsx-manager-ip-address`.
2. Go to **System > Overview**.
3. Click **Edit** next to the Virtual IP field.
4. Enter a VIP for the cluster, such as `172.16.11.81`. Ensure that the VIP is part of the same subnet as the other NSX Management nodes.
5. Click **Save**.
6. When prompted click **Refresh**.

Verify the VIP Address

To verify the VIP and troubleshoot issues, complete the following steps.

1. From your browser, log in with admin privileges to the NSX-T Management Cluster using the VIP address `https://nsx-manager-VIP-address`.

2. To check the API leader of the management cluster, enter the following command in the NSX Manager CLI `get cluster status verbose`. See Verify NSX VM Deployment for Enterprise PKS for instructions on access the NSX-T Manager CLI.

3. To troubleshoot VIP issues, verify the following logs:

   - Reverse Proxy logs at `/var/log/proxy/reverse-proxy.log`
   - Cluster manager logs at `/var/log/cbm/cbm.log`

Determine Which NSX-T Manager the VIP Is Assigned To

After the VIP is configured, you can still 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, the VIP is associated with a single NSX-T Manager. If that node goes down, the VIP attaches to another node. To determine which NSX-T Manager node the VIP is associated with, select the **Virtual IP** field in the NSX-T Manager UI.

Next Step

See Installing NSX Edge Nodes for Enterprise PKS.

NSX-T Installation Instructions Home

Installing and Configuring NSX-T for Enterprise PKS.

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

Installing NSX Edge Nodes for Enterprise PKS

In this topic

Page last updated:

This topic provides instructions for installing NSX Edge Node VMs on vSphere for use with Enterprise PKS.

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

About Deploying NSX-T Edge Nodes for Enterprise PKS

NSX Edge Nodes provide the bridge between the virtual network environment implemented using NSX-T and the physical network. Edge Nodes for Enterprise PKS run load balancers for PKS API traffic, Kubernetes pod load balancer services, and pod ingress controllers. See Load Balancers in Enterprise PKS for more information.

Enterprise PKS supports the NSX Edge Node `medium` (4 vCPU, 8 GB of RAM, 200 GB of disk space), and `large` (8 vCPU, 32 GB of RAM, 200 GB of disk space) VM form factors, and the bare metal Edge Node. The Edge Node VM can only be deployed on Intel-based ESXi hosts. See NSX Edge VM System Requirements in the NSX-T Data Center documentation.

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. In NAT mode, Enterprise PKS supports active/standby Edge Node failover. In standby mode, the standby load balancer is not available for use while the active load balancer is engaged. If No NAT mode is used, active/active can be used.

The default size of the load balancer deployed by NSX-T for a Kubernetes cluster is `small`. The size of the load balancer can be customized using Network Profiles.

In NSX-T a load balancer is deployed on the Edge Nodes as a virtual server. The following virtual servers are required for Enterprise PKS:

- 1 TCP layer 4 virtual server for each Kubernetes service of type: `LoadBalancer`
- 2 layer 7 global virtual servers for Kubernetes pod ingress resources (HTTP and HTTPS)
- 1 global virtual server for the PKS API

To determine the maximum number of load balancers per Edge Cluster, multiply the maximum number of load balancers for the Edge Node type by the number of Edge Nodes and divide by 2. For example, with 10 `large` VM Edge Nodes in an Edge Cluster, you can have up to 200 `small` load balancer instances (40 x 10/2), or up to 20 `medium` LB instances (4 x 10/2). See Scaling Load Balancer Resources in the NSX-T Data Center documentation for more information.

Note: Because of the load balancer requirements for Enterprise PKS, you cannot use the `small` Edge Node VM because this form factor does not support a sufficient number of virtual servers. You must install either a `medium` or `large` size Edge Node VM or the bare metal Edge Node for Enterprise PKS.

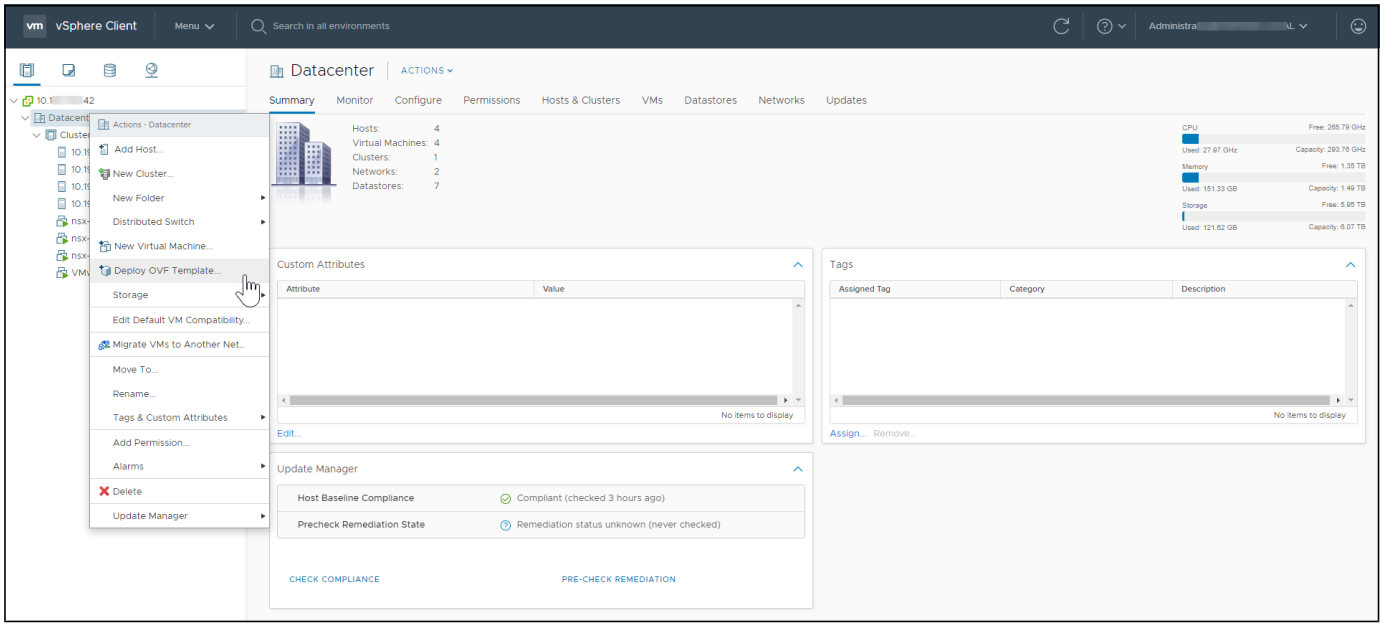
Install a Medium or Large NSX Edge Node VM Using the vSphere Client

The NSX Edge Node VM is provided as an OVA file named the **NSX Edge VM** that you import into your vSphere environment and configure.

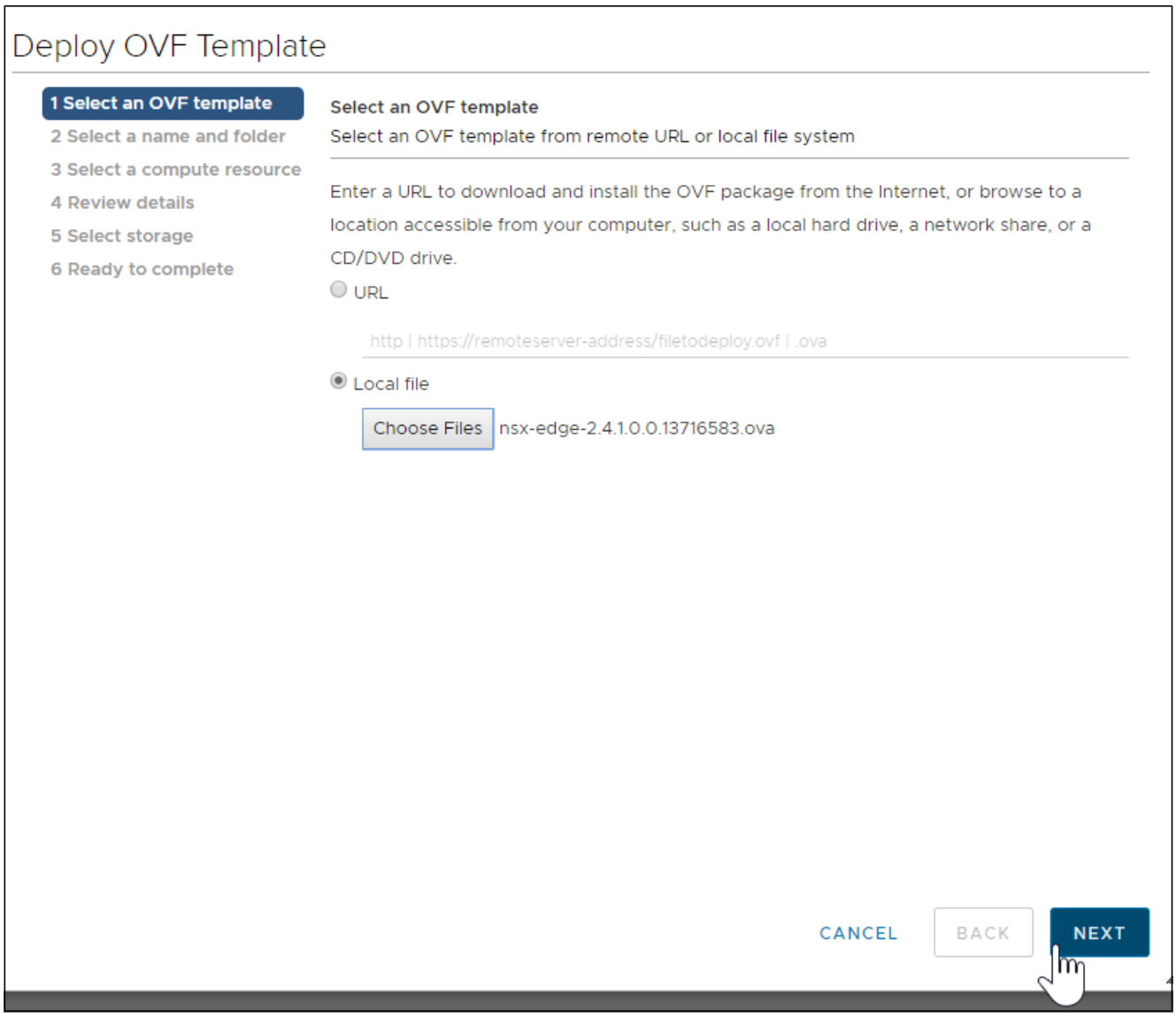
Complete the following steps to install a `medium` or `large` NSX Edge Node VM using the vSphere Client. To install an NSX Edge Node VM using the ovftool CLI, see the NSX-T Data Center documentation

Note: Repeat the deployment and verification process for each NSX Edge Node you intend to use for Enterprise PKS.

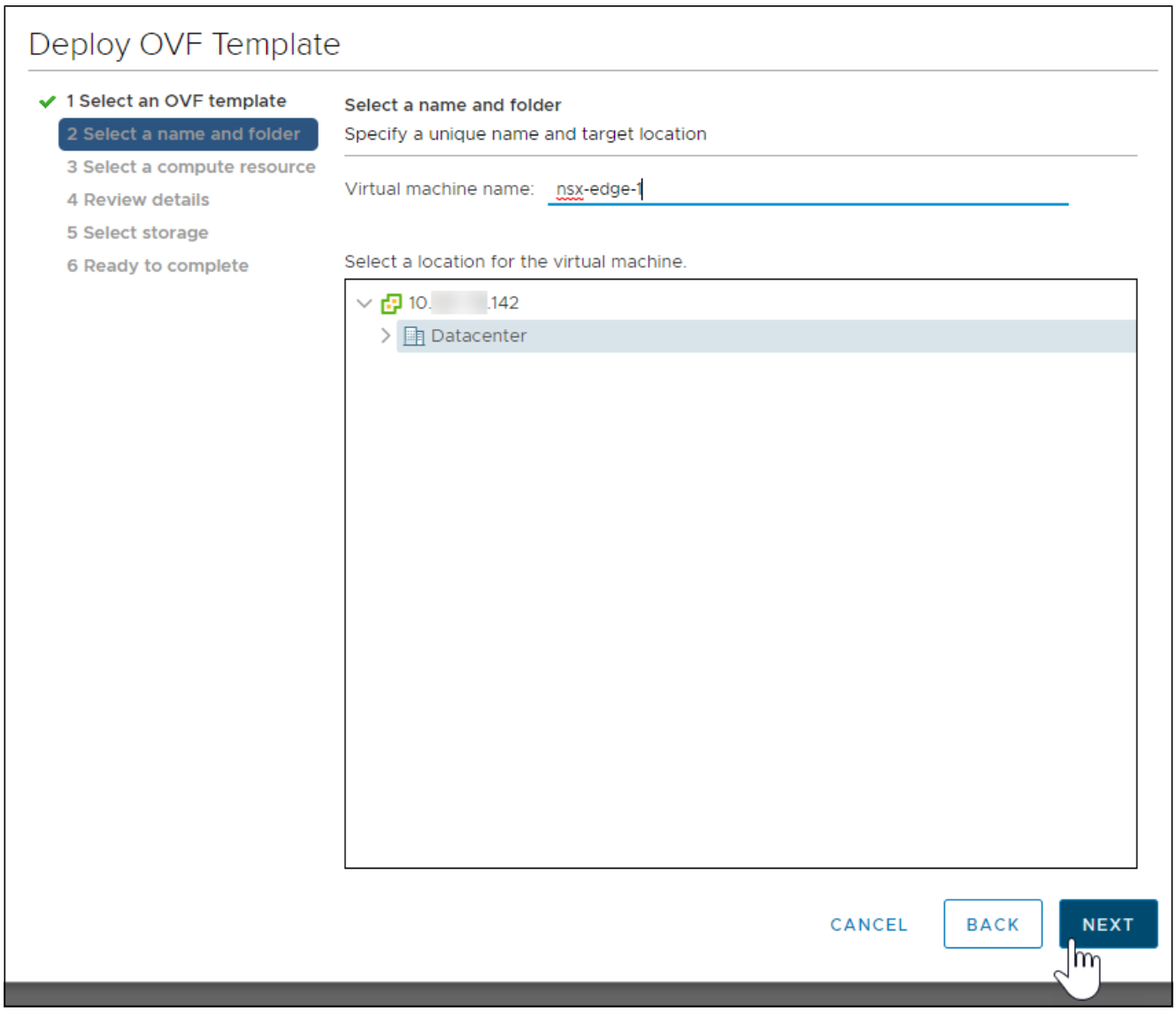
1. Locate the NSX-T Data Center OVA file and download it to your local machine, for example `nsx-edge-VERSION.ova`.
2. Log in to vCenter using the vSphere Client.
3. In the vSphere Client, select the **Resource Pool** where you want to install NSX-T Data Center.
4. Right-click and select **Deploy OVF Template** to start the installation wizard.



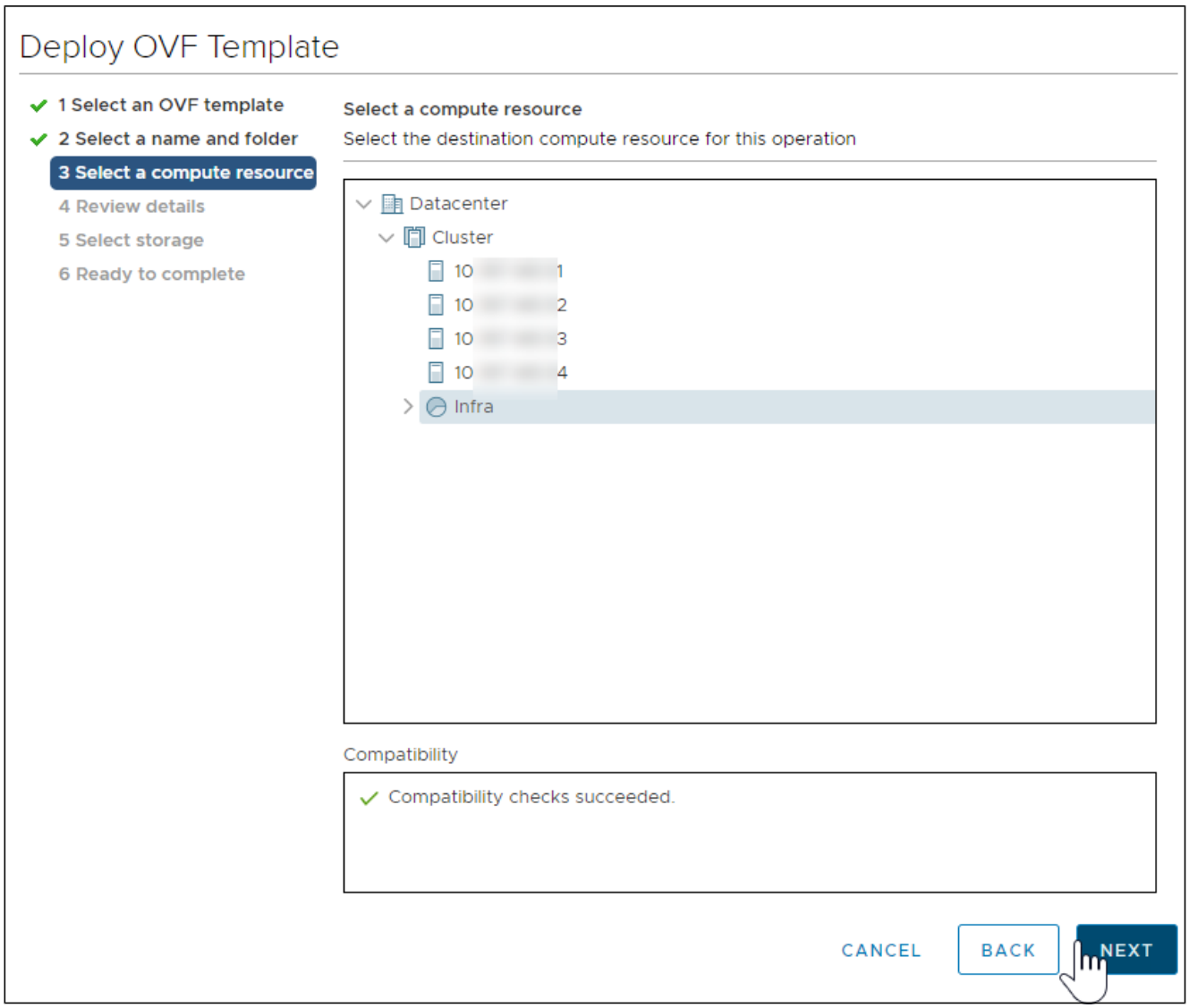
5. At the **Select an OVF template** screen:
   - Select the **Local file** option
   - Click **Choose Files**
   - Navigate to where you downloaded the OVA file and select it
   - Click **Next**



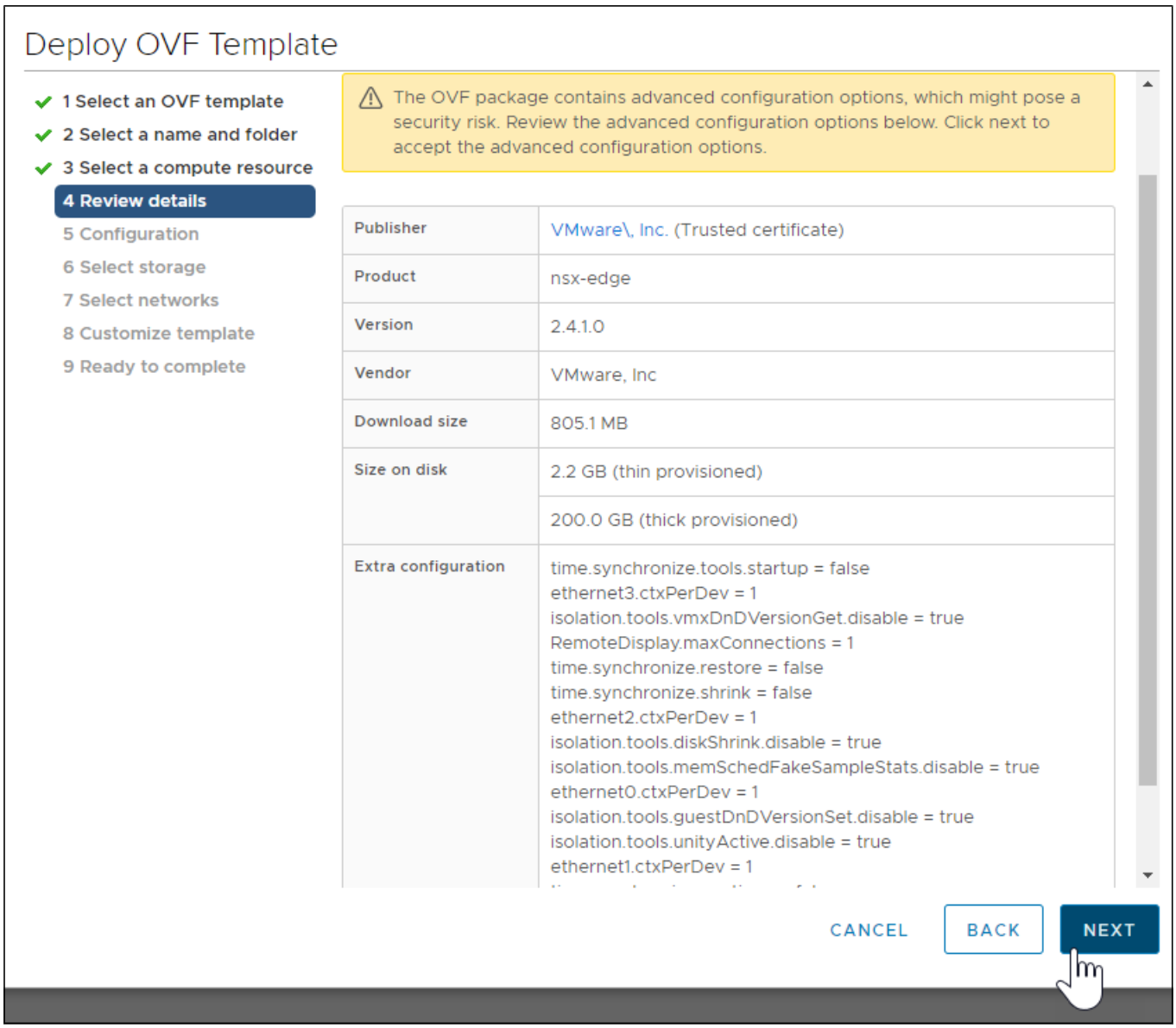
6. At the **Select a name and folder** screen:
   - Enter a name for the NSX Edge VM, such as `nsx-edge-1`
   - Select the **Datacenter** for the VM deployment
   - Click **Next**



7. At the **Select a compute resource** screen:
   - Select the **infra** resource pool where the NSX Edge VM will be deployed
   - Click **Next**

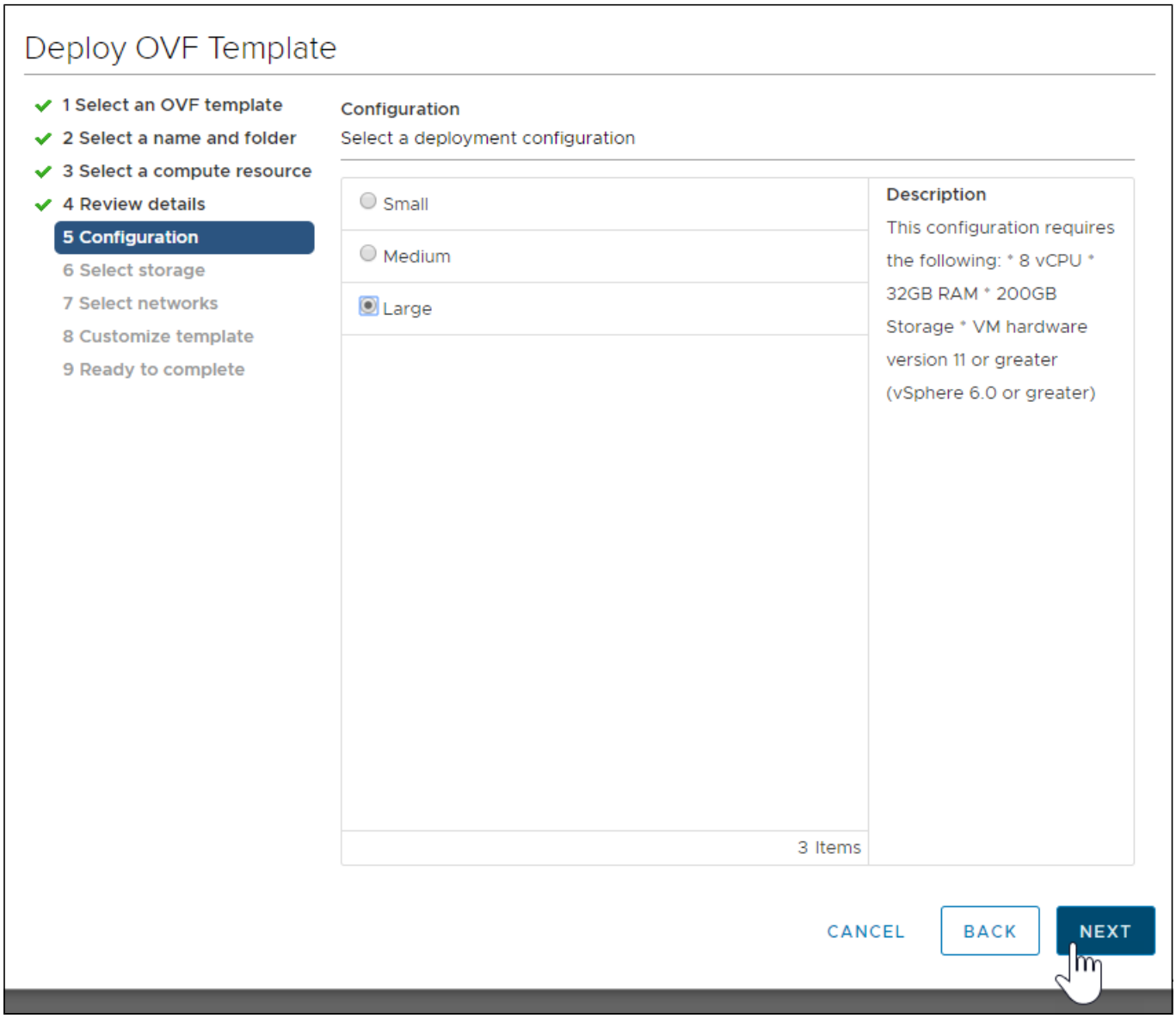


8. At the **Review details** screen, verify the OVF template details and click **Next**.

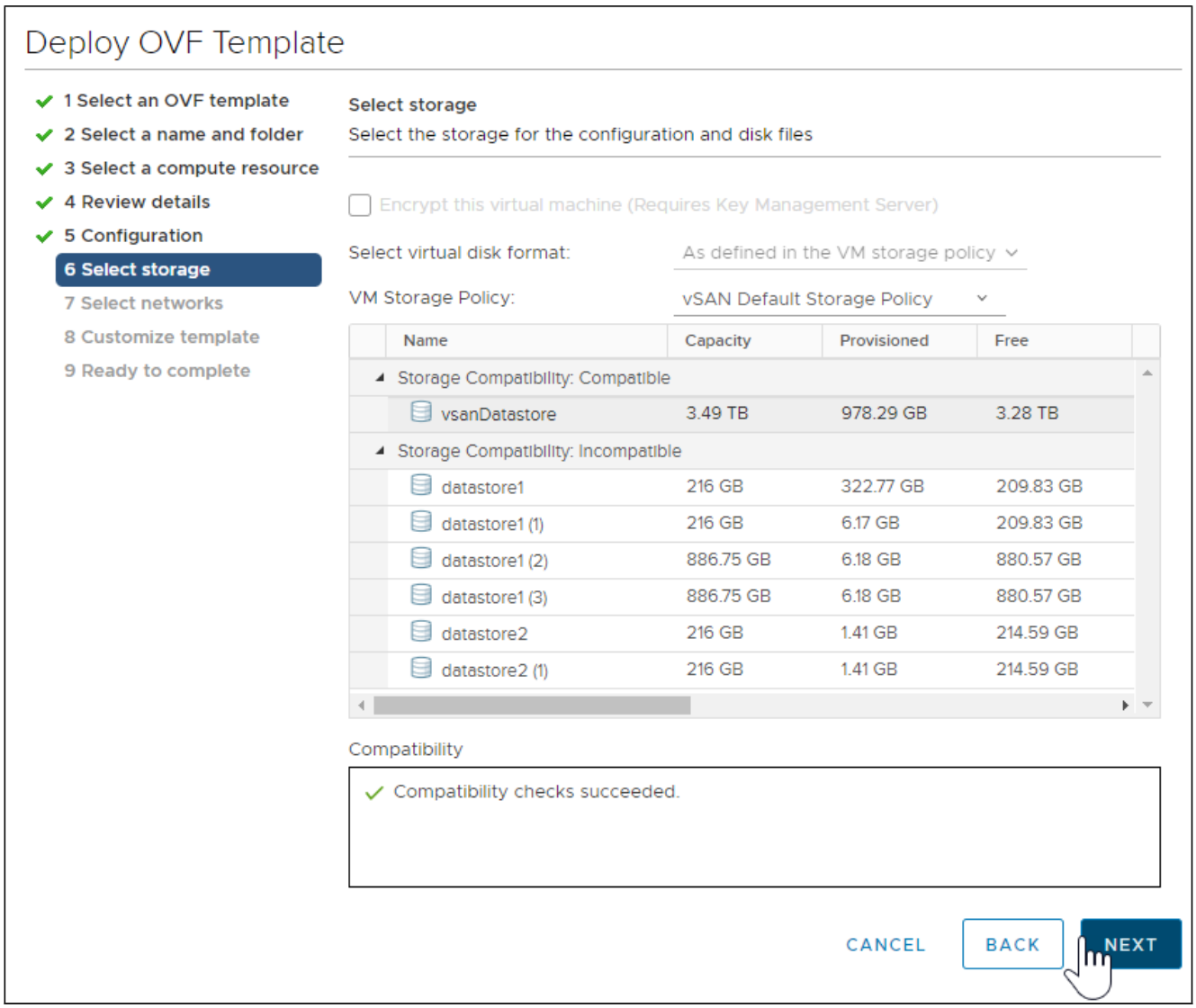


9. At the **Configuration** screen, select either **Medium** or **Large** size VM and click **Next**.

> **Warning:** You must select either **Medium** or **Large** size Edge Node VM. See About Deploying NSX-T Edge Nodes for Enterprise PKS.

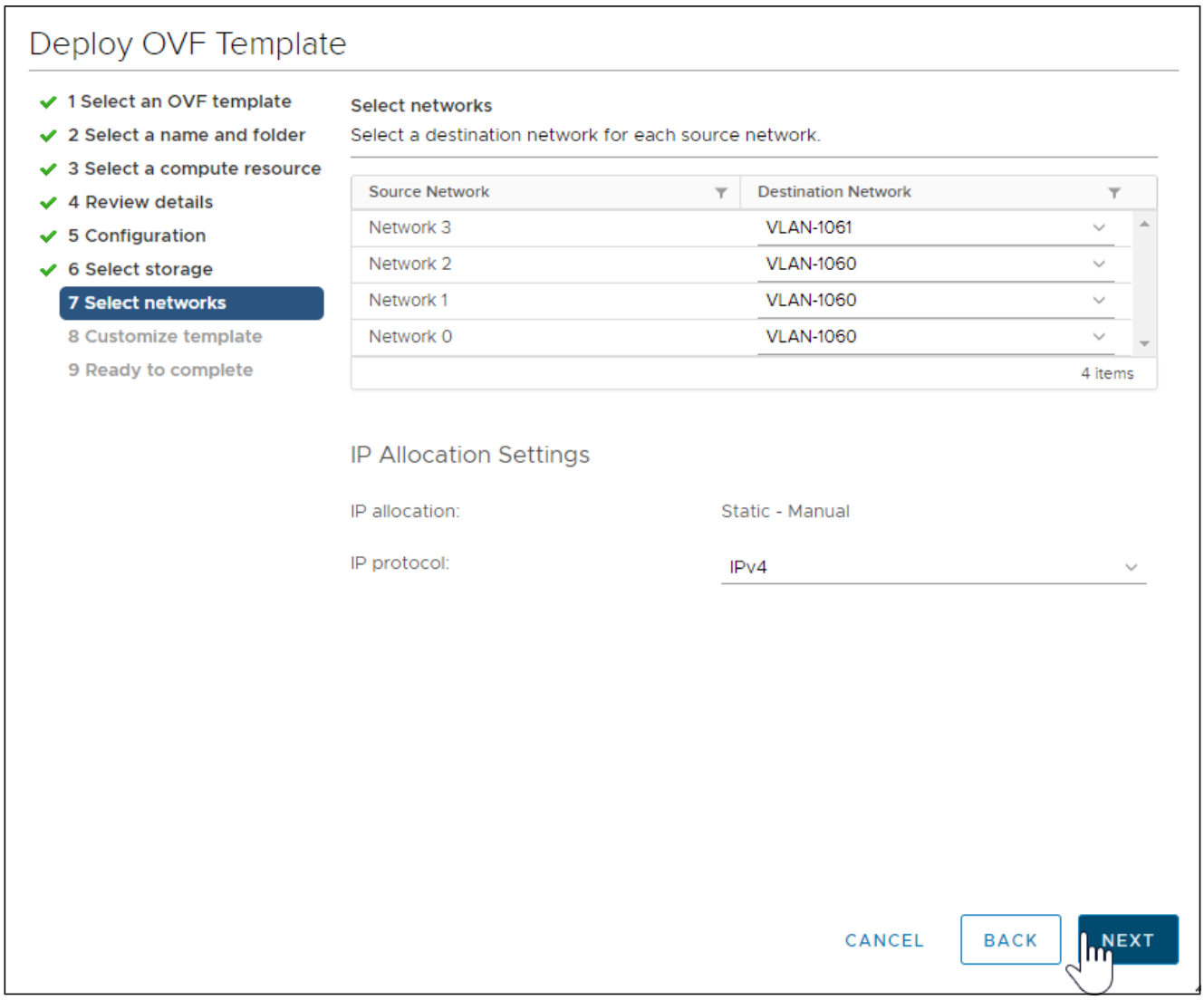


10. At the **Select storage** screen:
    - Select the **vsanDatastore** if you are using vSAN, or a dedicated datastore if you are not using vSAN
    - Click **Next**



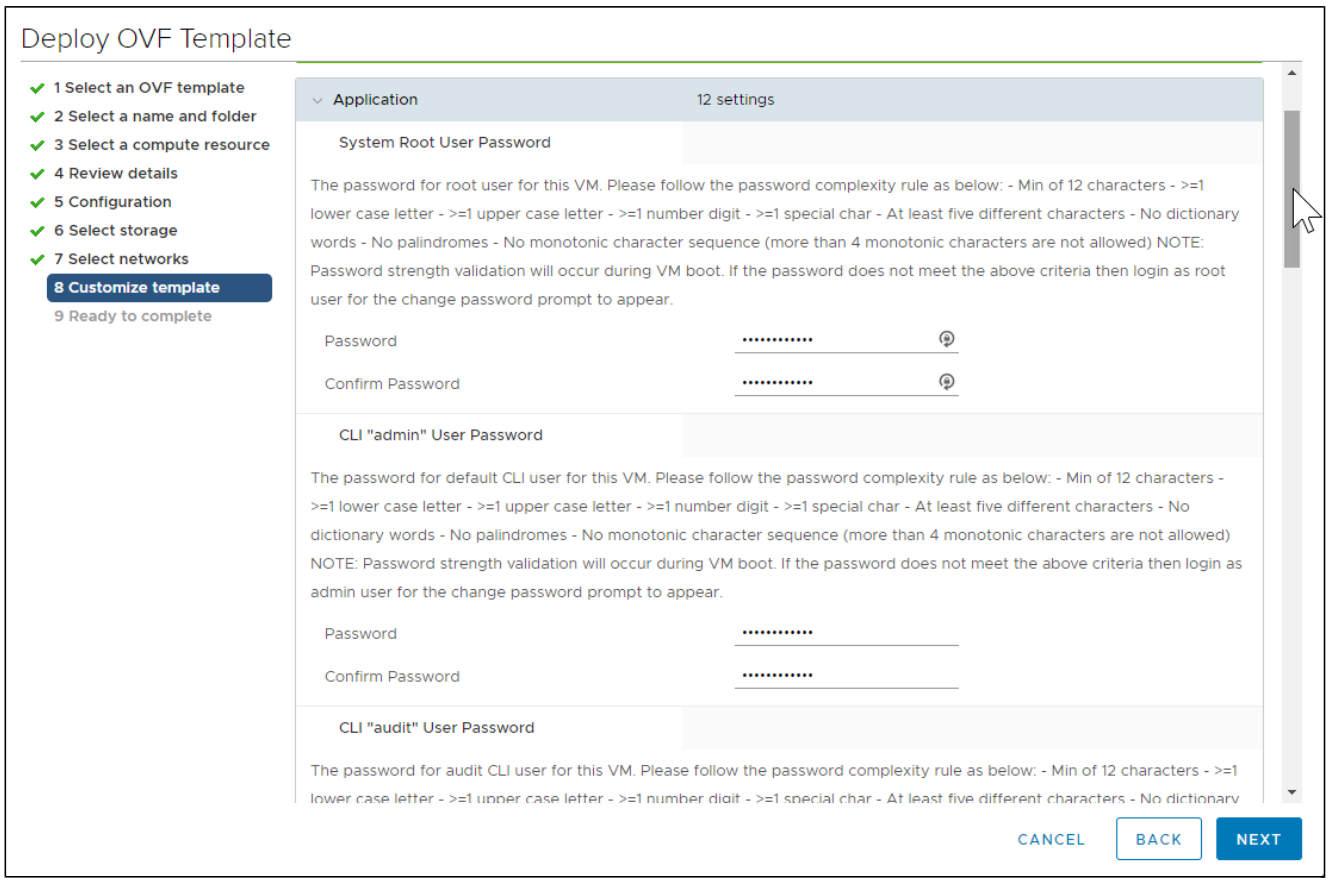
11. At the **Select networks** screen, select the **Destination Network** for each of the Source Networks. Be sure to connect the vNICs of the NSX Edge VM to an appropriate PortGroup for your environment.

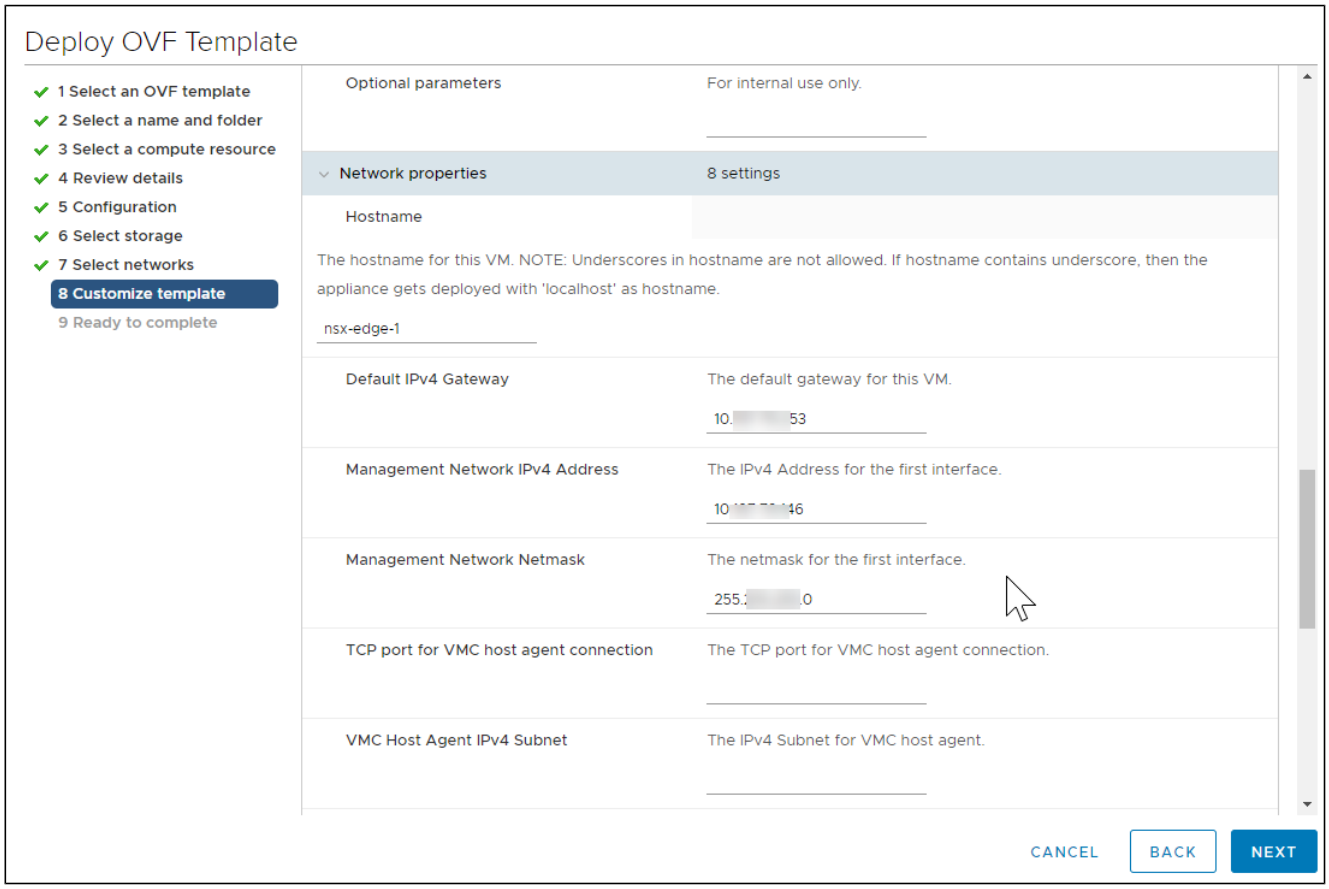
    - **Network 0**: For management purposes. Connect the first Edge interface to your environment's PortGroup/VLAN where your Edge Management IP can route and communicate with the NSX Manager.
    - **Network 1**: For TEP (Tunnel End Point). Connect the second Edge interface to your environment's PortGroup/VLAN 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)
    - Click **Next**

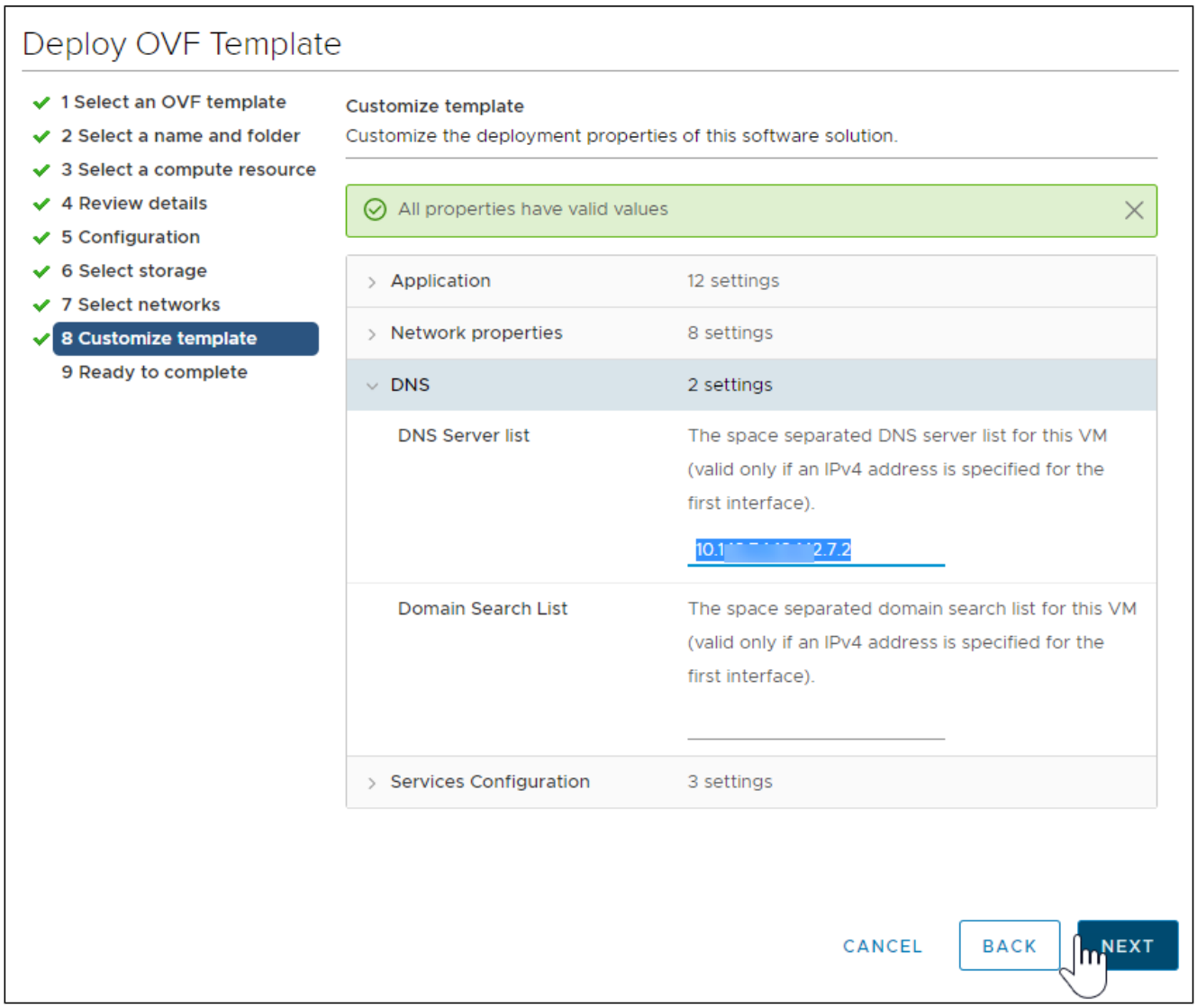


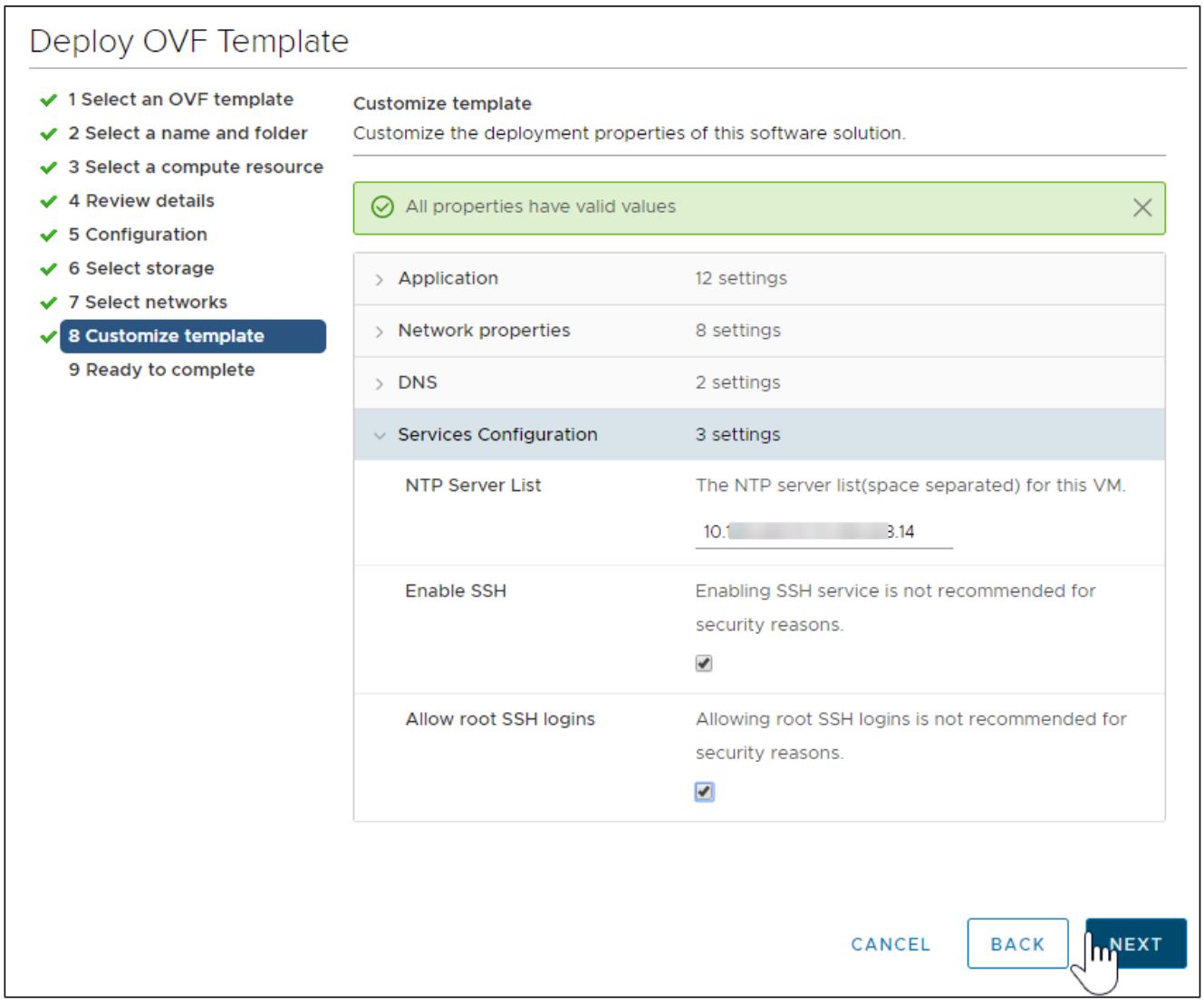
12. At the **Customize Template** screen, configure the following settings:

    - **System Root User Password** (must comply with password strength restrictions)
    - **CLI "admin" User Password** (must comply with password strength restrictions)
    - **CLI "audit" User Password** (must comply with password strength restrictions)
    - **Hostname**: for the NSX Edge VM, such as `nsx-edge-1`
    - **Default IPv4 Gateway**: The default gateway for the NSX Manager VM
    - **Management Network IPv4 Address**: The IPv4 address for the first network interface
    - **Management Network Netmask**: The netmask for the first interface
    - **DNS Server List**: One or more DNS servers (space-separated if multiple)
    - **NTP Server List**: One or more NTP servers (space-separated if multiple)
    - **Enable SSH**: Select Enable SSH (by default this option is disabled for security reasons)
    - **Allow root SSH logins**: Enable this option to (by default this option is disabled for security reasons)
    - Click **Next**

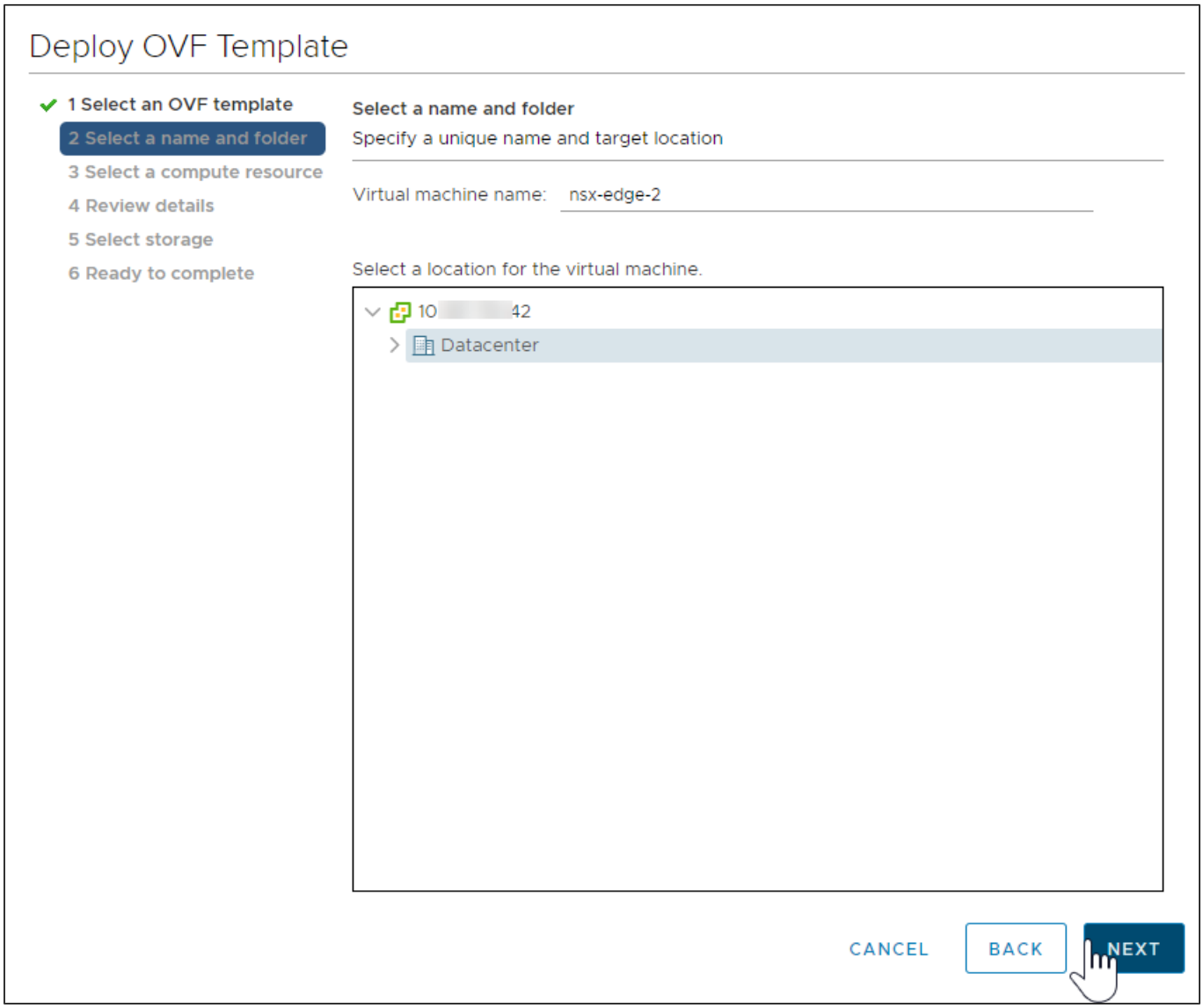








13. At the **Ready to complete** screen:
    - Verify that the OVF template specification is accurate
    - Click **Finish** to begin the installation. The installation will approximately 10 minutes to complete
14. Use the **Recent Tasks** panel at the bottom of the vCenter screen to view the progress of the OVA deployment.
15. Repeat this process for `nsx-edge-2`, and for each additional NSX Edge Node you intend to use for Enterprise PKS.



Verify NSX Edge Node VM Installation

See Verify NSX VM Deployment for Enterprise PKS.

Next Step

See Join Each NSX Edge Node with the Management Plane

NSX-T Installation Instructions Home

Installing and Configuring NSX-T for Enterprise PKS.

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

Join NSX Edge Nodes with the Management Plane for Enterprise PKS

In this topic

Page last updated:

This topic provides instructions for joining NSX-T Edge Nodes with the NSX-T Management Plane.

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

About Joining NSX Edge Nodes with the NSX Management Plane

Joining NSX Edges with the management plane ensures that the NSX Manager and NSX Edges can communicate with each other. For more information, see Join NSX Edge with the Management Plane in the NSX-T Data Center documentation.

Join NSX Edge Nodes with the NSX-T Management Plane

To register an Edge Node with NSX Manager, complete the following procedure:

1. Open an SSH session to one of the NSX-T Manager VMs, such as `nsx-manager-1`.
   - For example, if you are using Unix: `ssh admin@IP_ADDRESS_OF_NSX_MANAGER-1`
   - If you are using Windows, use Putty to SSH to the NSX-T Manager node
2. Open another SSH session to the target NSX Edge Node VM, such as `nsx-edge-1`.
3. On the NSX-T Manager appliance, run the command `get certificate api thumbprint`.
   The command output is a string of alphanumeric numbers that is unique to this NSX Manager. For example:

   ```
   nsx-manager-1> get certificate api thumbprint
   120a129d533601a6513abb...924210243d6d6bce8f8f0d84d66d
   ```

4. On the NSX-T Edge Node VM, run the following command to register the Edge Node with the NSX Management Plane.

   ```
   join management-plane <nsx-manager-ip-address> username <username> thumbprint <nsx-manager-api-thumbprint>
   ```

   For example:

   ```
   join management-plane 10.197.79.143 username admin thumbprint 120a129d533601a6...210243d6d6bce8f8f0d84d66d
   ```

5. When prompted, enter the admin password for the NSX-T Manager.

6. Verify that you see the following output, indicating that the NSX Edge Node is successfully registered.

```
Node successfully registered as Fabric Node: 2a8xYdec-aXa7-11x9-9063-0050512345f11
```

7. Repeat this join procedure for each NSX Edge Node you have installed for Enterprise PKS.

Verify NSX Edge Node Registration with the NSX Management Plane

To verify Edge Node registration with NSX Manager:

1. Verify Edge Node registration by running the command `get managers` on the NSX Edge Node.
   You should see that the Edge Node is connected to one of the NSX Manager nodes. For example:

```
nsx-edge-1> get managers
- 10.XXX.XX.144    Standby
- 10.XXX.XX.145    Connected
- 10.XXX.XX.143    Standby
```

2. In NSX Manager, select the page **System** > **Fabric** > **Nodes** > **Edge Transport Nodes**.
   You should see each registered Edge Node.

   Note: Initially the “Configuration State” is yellow because the Edge Nodes are not configured a NSX-T Transport Nodes. This is done later in the process.

3. Repeat this verification procedure for each NSX Edge Node you are deploying for Enterprise PKS.

Next Step

See Enable Repository Service on Each NSX Manager Node

NSX-T Installation Instructions Home

See Installing and Configuring NSX-T for Enterprise PKS.

Enable NSX Repository Service for VIB Installs

In this topic

Page last updated:

This topic provides instructions for enabling VIB installs from the NSX Manager repository service.

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

About VIBs

VIB stands for vSphere Installation Bundle. A VIB is a collection of files packaged into a single archive to facilitate software distribution for vSphere hosts, similar to a tarball or ZIP archive for other operating systems.

VIB packages are installed on ESXi hosts. To enable VIB installations from the NSX-T Manager repository, the repository service needs to be enabled on each NSX-T Manager node.

Enable Repository Service on Each NSX-T Manager Node

To enable VIB installation from the NSX Manager repository, the repository service needs to be be enabled in NSX Manager.

1. Open an SSH session to one of the NSX-T Manager VMs, such as `nsx-manager-1`.
   - For example, if you are using Unix: `ssh admin@IP_ADDRESS_OF_NSX_MANAGER-1`
   - If you are using Windows, use Putty to SSH to the NSX-T Manager node
2. Enable the repository service by running the following command:

   ```
   nsx-manager-1> set service install-upgrade enable
   ```

3. Repeat this procedure for each NSX Manager node you are deploying for Enterprise PKS.
   For example:

   ```
   nsx-manager-2> set service install-upgrade enable
   ```

   ```
   nsx-manager-3> set service install-upgrade enable
   ```

Next Step

See Create an IP Pool for Tunnel Endpoint IP Addresses.

NSX-T Installation Instructions Home

See Installing and Configuring NSX-T for Enterprise PKS.

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

Create TEP IP Pool

In this topic

Page last updated:

This topic provides instructions for creating an IP pool for tunnel endpoints (TEPs).

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

About TEPs

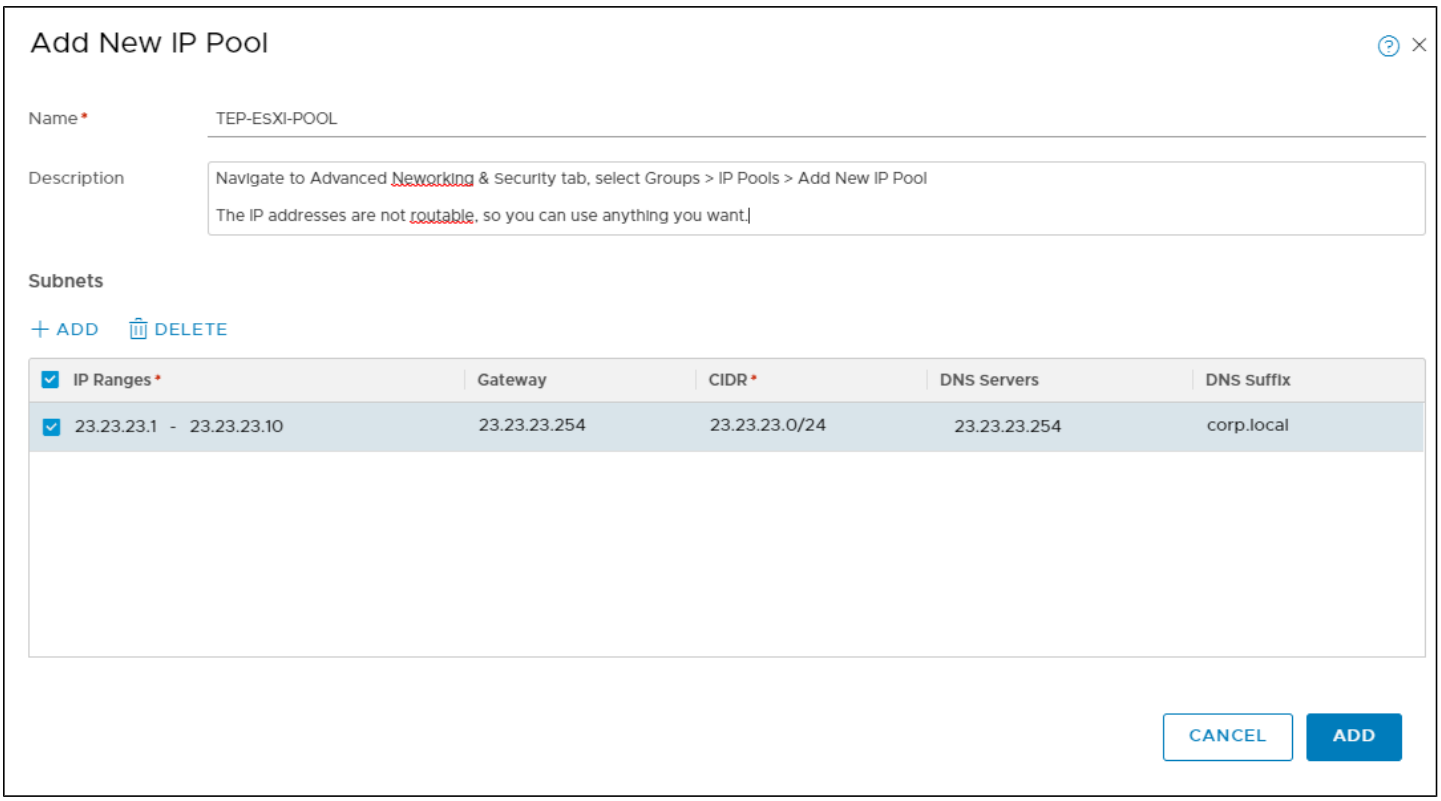
Tunnel endpoints (TEPs) are the source and destination IP addresses used in the external IP header to identify the ESXi hosts that originate and end the NSX-T encapsulation of overlay frames. You can use DHCP or manually configured IP pools for TEP addresses. The TEP addresses do not need to be routable. so you can use any random IP addressing scheme you want. For more information, see Tunnel Endpoint IP Addresses in the NSX-T Data Center documentation.

Create TEP IP Pool

To create the TEP IP Pool, complete the following procedure:

1. In NSX Manager, select **Advanced Networking & Security** > **Inventory** > **Groups** > **IP Pools** > **Add**.
2. Enter the IP pool details.

- **Name**: `TEP-ESXi-POOL`, for example
- **Description** (optional)
- **IP Ranges**: Enter the IP allocation ranges, for example `192.168.200.100 - 192.168.200.115`
- **Gateway**: `23.23.23.254`, for example
- **CIDR**: Enter the Network address in a CIDR notation, for example `192.168.200.0/24`
- **DNS Servers** (optional): Comma-separated list of DNS servers, such as `192.168.66.10`
- **DNS Suffix** (optional): Such as `corp.local`

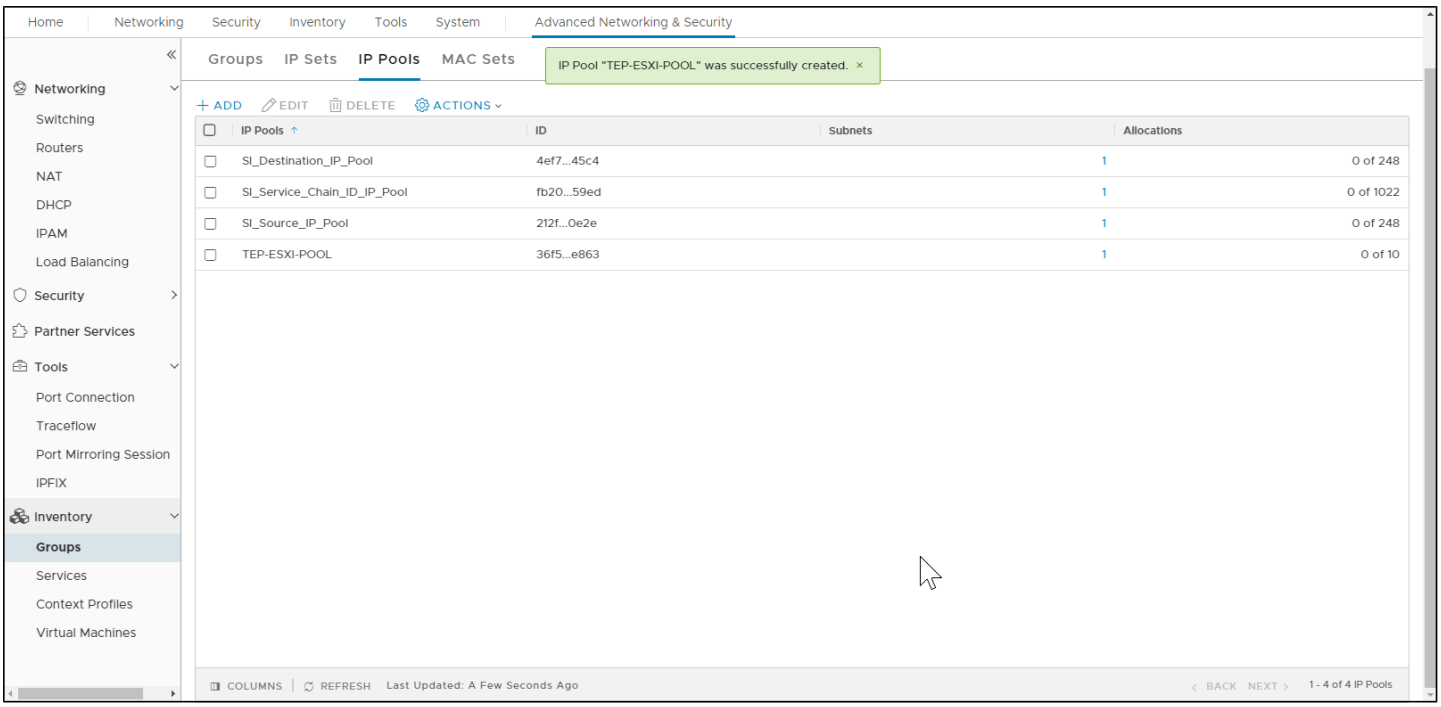


Note: TEP addresses do not need to be publicly routable.

Verify TEP IP Pool Creation

To verify TEP IP Pool configuration, complete the following steps:

1. In NSX Manager, select **Advanced Networking & Security** > **Inventory** > **Groups** > **IP Pools**.
2. Verify that the TEP IP Pool you created is present.



Next Step

See Create Overlay and VLAN Transport Zones.

NSX-T Installation Instructions Home

See Installing and Configuring NSX-T for Enterprise PKS.

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

Create Transport Zones

In this topic

Page last updated:

This topic provides instructions for creating transport zones for use with Enterprise PKS.

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

About Transport Zones

In NSX-T Data Center, a transport zone (TZ) is a logical construct that controls which hosts a logical switch can reach. A transport zone defines a collection of hosts that can communicate with each other across a physical network infrastructure. This communication happens over one or more interfaces defined as Virtual Tunnel Endpoints (VTEPs).

There are two types of transport zones: Overlay and VLAN. An overlay transport zone is used by ESXi host transport nodes and NSX-T Edge Nodes. When an ESXi host or NSX-T Edge transport node is added to an Overlay transport zone, an N-VDS is installed on the ESXi host or NSX Edge Node. The VLAN transport zone is used by NSX-T Edge Nodes and ESXi host transport nodes for its VLAN uplinks. When an NSX-T Edge Node is added to a VLAN transport zone, a VLAN N-VDS is installed on the NSX-T Edge Node.

For more information, see Transport Zones in the NSX-T Data Center documentation.

Create Overlay Transport Zone

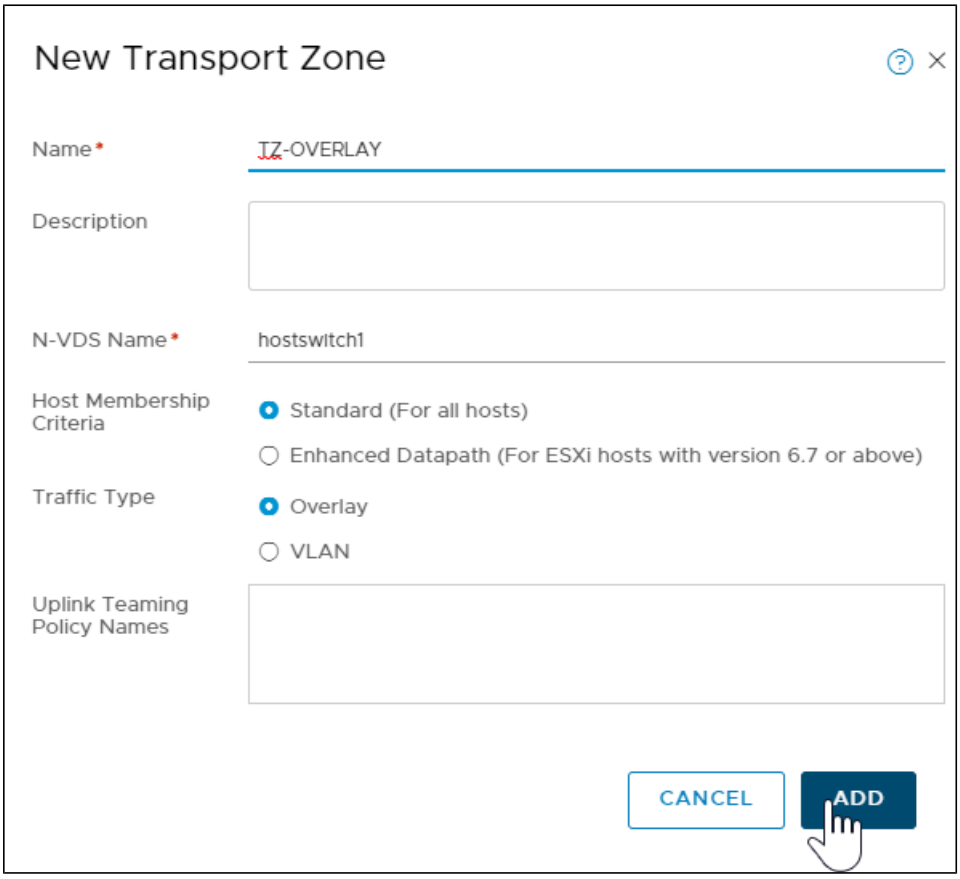
Create an Overlay Transport Zone (`TZ-OVERLAY`) for PKS control plane services and Kubernetes clusters overlay networks associated with associated with VDS `hostswitch1` .

1. In NSX Manager, select **System** > **Fabric** > **Transport Zones** > **Add**.
2. Enter a **Name** for the transport zone, such as `TZ-OVERLAY` .
3. Enter a **N-VDS Name**, such as `hostswitch1` .
4. Select a **Host Membership Criteria** (N-VDS mode): Standard or Enhanced Datapath.

> **Note:** In enhanced mode, only specific NIC configurations are supported (ESXi hosts v6.7+), and you must ensure that

you configure the supported NICs. Refer to the NSX-T Data Center documentation for more information.

5. For **Traffic Type**, select **Overlay**.
6. (Optional) Enter one or more uplink teaming policy names.
7. Click **Add**.



Create VLAN Transport Zone

Create the VLAN Transport Zone (`TZ-VLAN`) for NSX Edge Node uplinks (ingress/egress) for Kubernetes clusters associated with VDS `hostswitch2` .

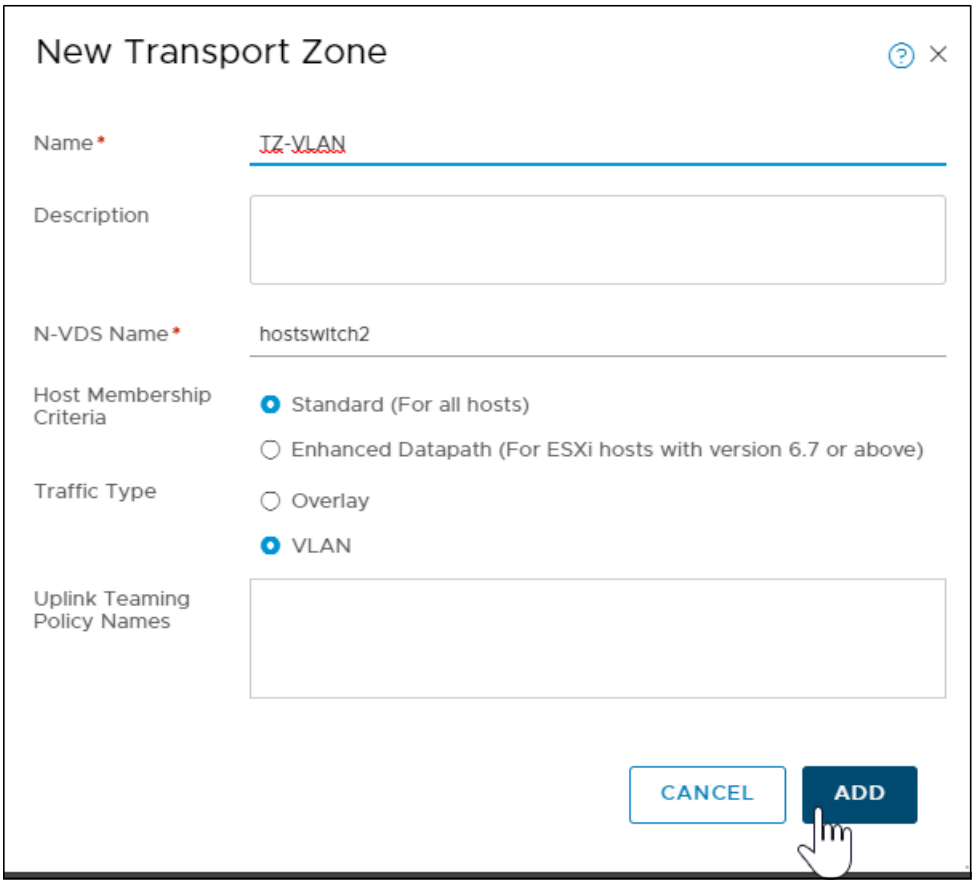
1. In NSX Manager, select **System** > **Fabric** > **Transport Zones** > **Add**.
2. Enter a **Name** for the transport zone, such as `TZ-VLAN` .
3. Enter **N-VDS Name**, such as `hostswitch2` .
4. Select a **Host Membership Criteria** (N-VDS mode): Standard or Enhanced Datapath.

   Note: In enhanced mode, only specific NIC configurations are supported (ESXi hosts v6.7+), and you must ensure that you configure the supported NICs. Refer to the NSX-T Data Center documentation for more information.

5. For **Traffic Type**, select **VLAN**.

6. (Optional) Enter one or more uplink teaming policy names.

7. Click **Add**.

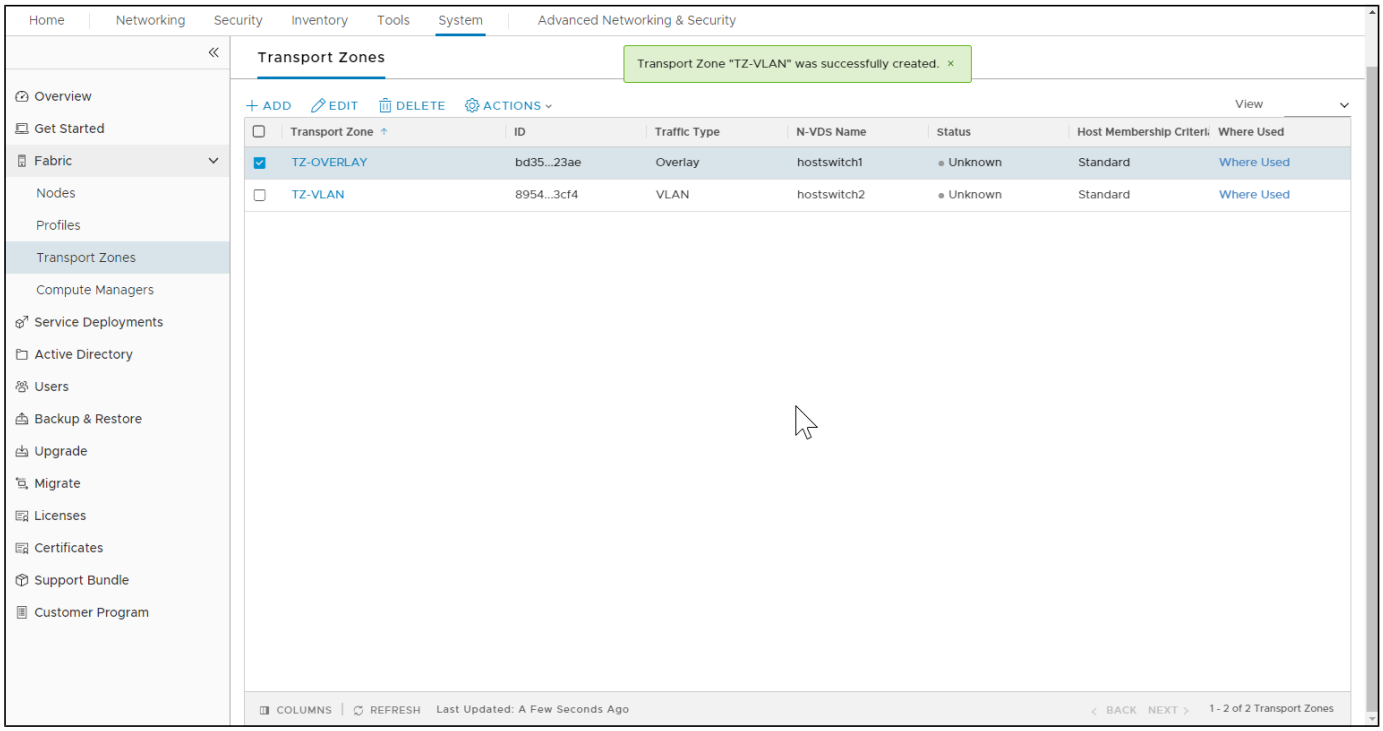


Verify Transport Zone Creation

To verify transport zone creation:

1. In NSX-T Manager select **System** > **Fabric > Transport Zones**.

2. Verify that you see the TZ-OVERLAY and TZ-VLAN transport zones you created:



View a larger version of this image.

Next Step

Create Edge Node Uplink Profile

NSX-T Installation Instructions Home

Installing and Configuring NSX-T for Enterprise PKS.

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

Create an Uplink Profile

In this topic

Page last updated:

This topic provides instructions for creating an NSX-T uplink profile for use with Enterprise PKS.

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

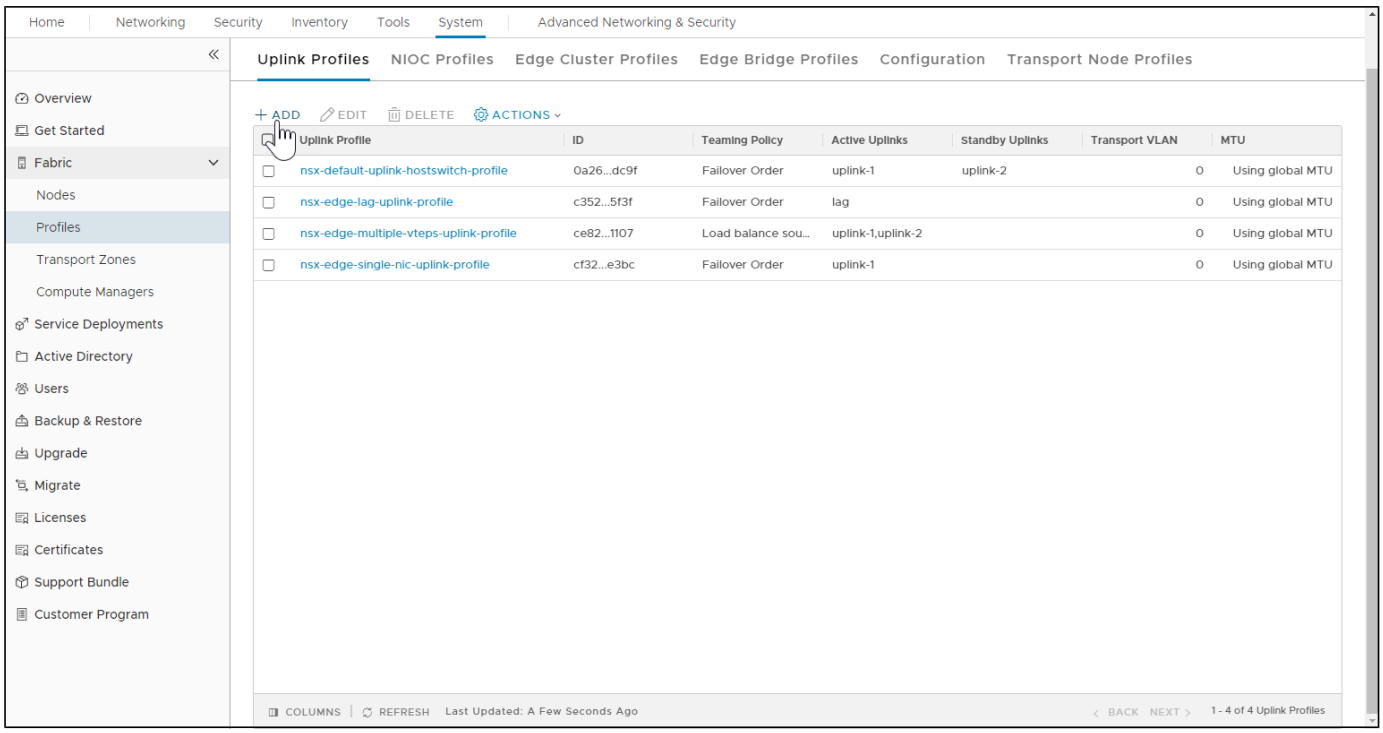
About Uplink Profiles

An uplink profile defines policies for the uplinks from ESXi hosts to NSX-T logical switches. For more information, see Uplink Profile in the NSX-T Data Center documentation.

Create Uplink Profile for Edge Nodes

Complete the following steps to create an uplink profile:

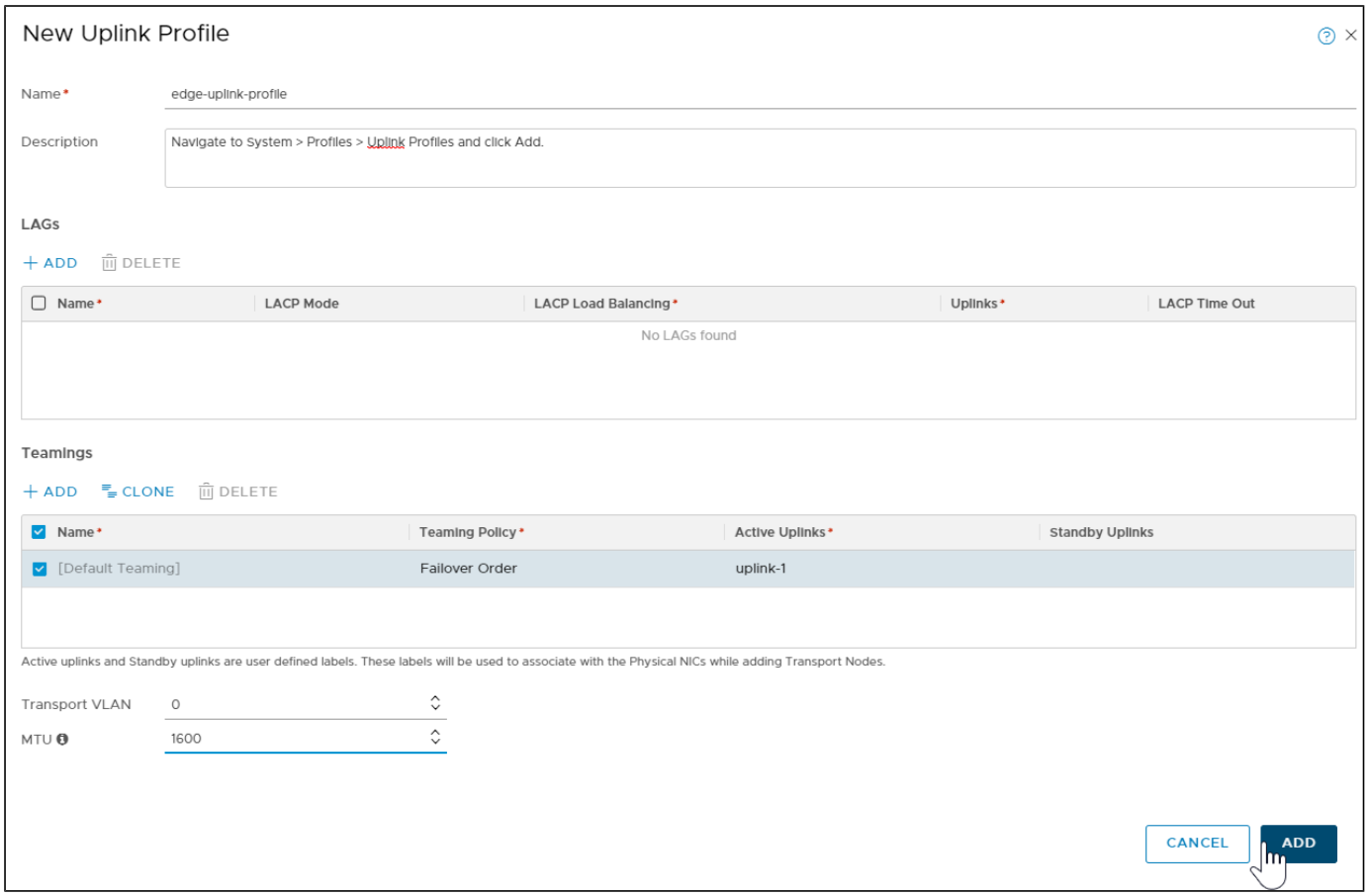
1. In NSX-T Manager, select **System** > **Fabric** > **Profiles** > **Uplink Profiles** > **Add**.



2. Enter a **Name** for the new uplink profile, such as `edge-uplink-profile`.

3. Enter the MTU value. The uplink profile MTU default value is 1600.

4. Click **Add**.

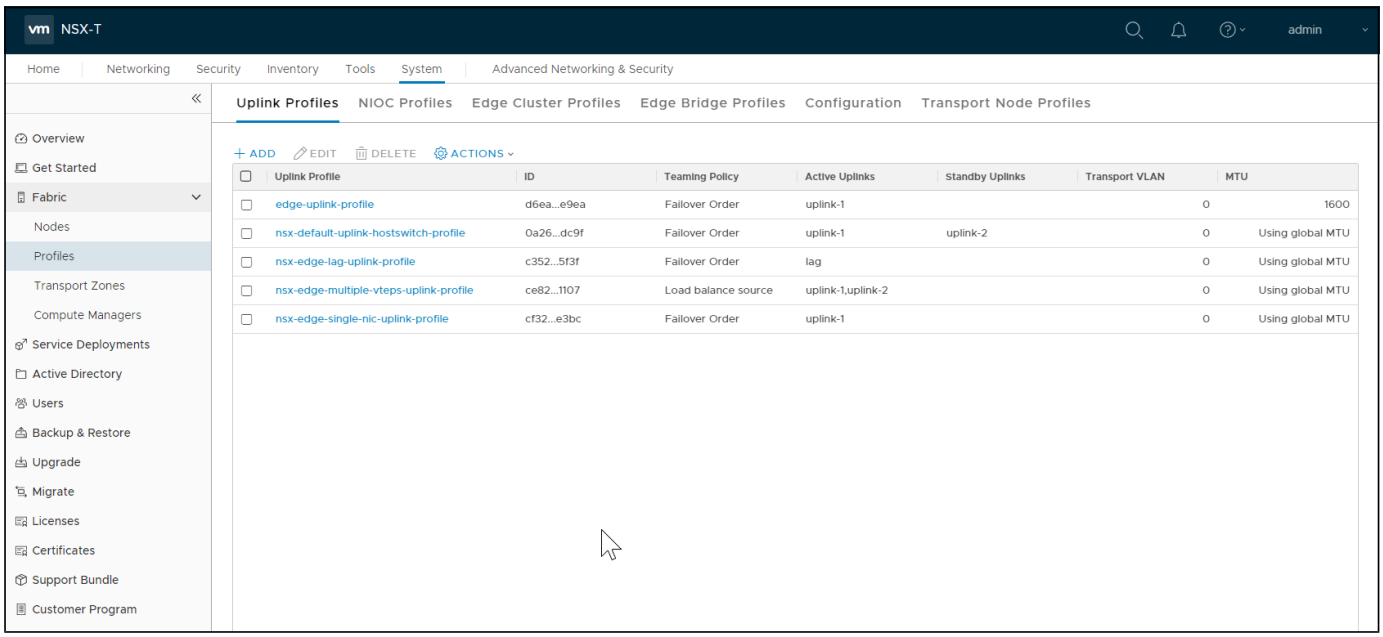


Verify Uplink Profile Creation

To verify Uplink Profile creation:

1. In NSX-T Manager, select **System** > **Fabric** > **Profiles** > **Uplink Profiles**.

2. Verify that you see the Edge Node uplink profile you created:



Next Step

Configure Edge Nodes as NSX Transport Nodes

NSX-T Installation Instructions Home

Installing and Configuring NSX-T for Enterprise PKS.

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

Create Edge Transport Nodes

In this topic

Page last updated:

This topic provides instructions for configuring Edge Transport Nodes for use with Enterprise PKS.

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

About Transport Nodes

In NSX-T transport nodes are the hosts running the local control plane daemons and forwarding engines implementing the NSX-T data plane. A transport node runs a NSX-T Virtual Distributed Switch (N-VDS) that is responsible for switching packets according to the configuration of available network services.

A host can serve as a transport node if it contains at least one NSX-T-managed virtual distributed switch (N-VDS). When you create a host transport node and then add the node to a transport zone, NSX-T Data Center installs an N-VDS on the host. For each transport zone that the host belongs to, a separate N-VDS is installed. The N-VDS is used for attaching VMs to NSX-T Data Center logical switches and for creating NSX-T Data Center logical router uplinks and downlinks.

In this portion of the NSX-T installation for Enterprise PKS, you create NSX Edge Transport Nodes that allow Edge Nodes to exchange virtual network traffic with other nodes. You add both the VLAN and OVERLAY NSX Transport Zones to the NSX Edge Transport Nodes and confirm NSX-T Manager connectivity.

We will 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.

For more information, see NSX Edge Transport Node in the NSX-T Data Center documentation.

Configure Each Edge Node as as Transport Node

To create an Edge Transport Node for Enterprise PKS, complete the following steps.

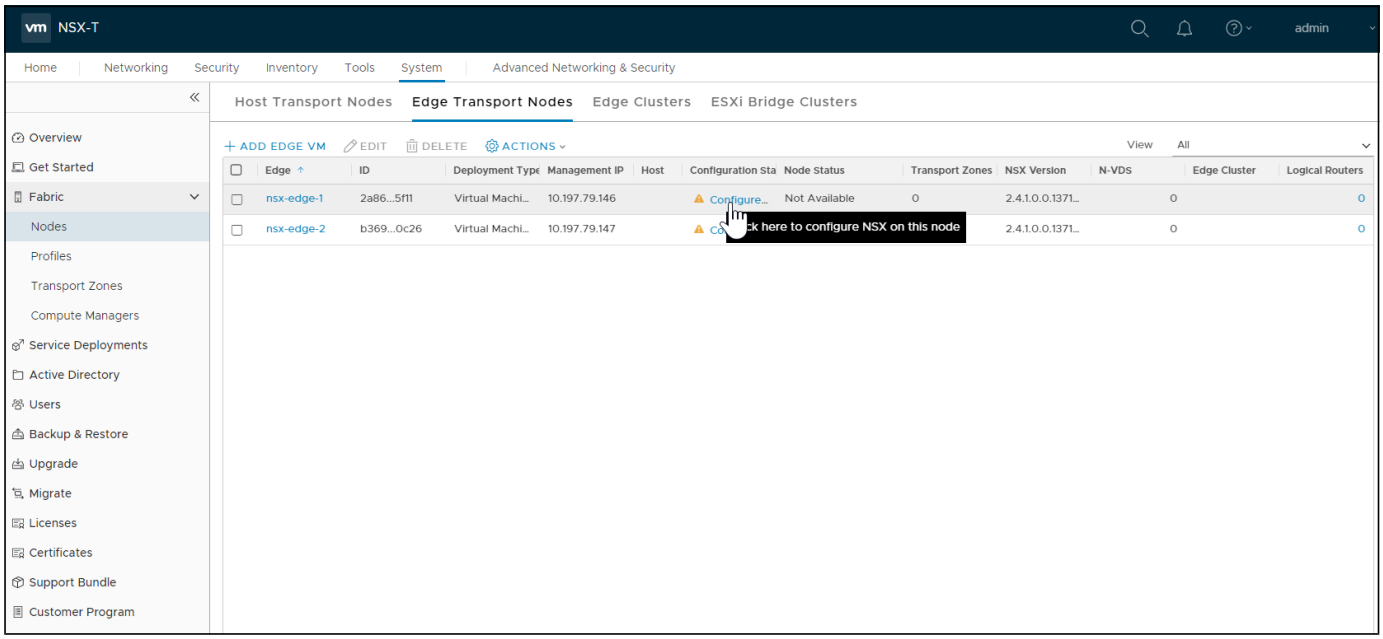
Note: Perform this procedure for each Edge Node pair you deploy for Enterprise PKS.

1. Log in to NSX Manager at `https://VIP_ADDRESS_OF_NSX_MANAGEMENT_CLUSTER`.

2. Go to **System > Fabric > Nodes > Edge Transport Nodes**

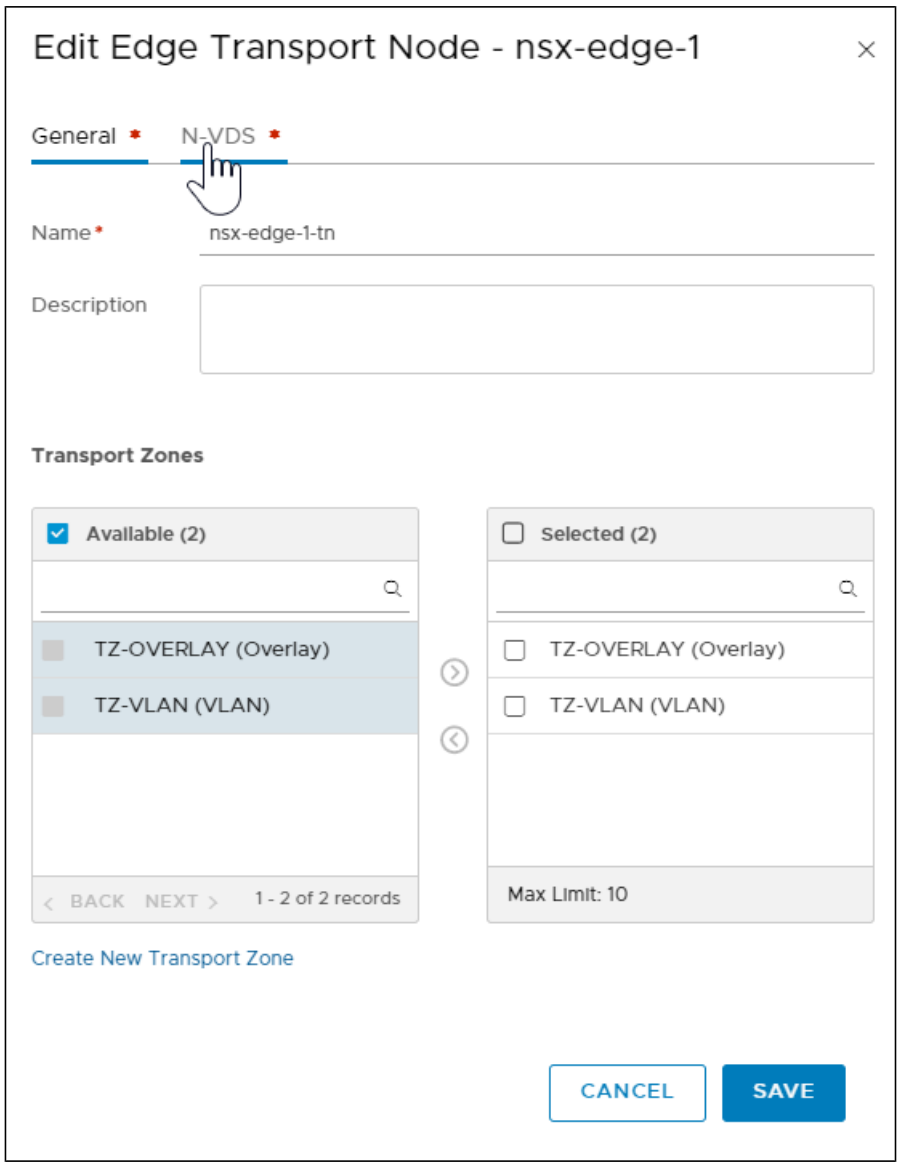
3. Verify that you see both of the NSX-T Edge Nodes you have installed.

4. Click the **Configure** link in **Configuration State** column for the first Edge Node.



5. In the **General** tab, configure the following:

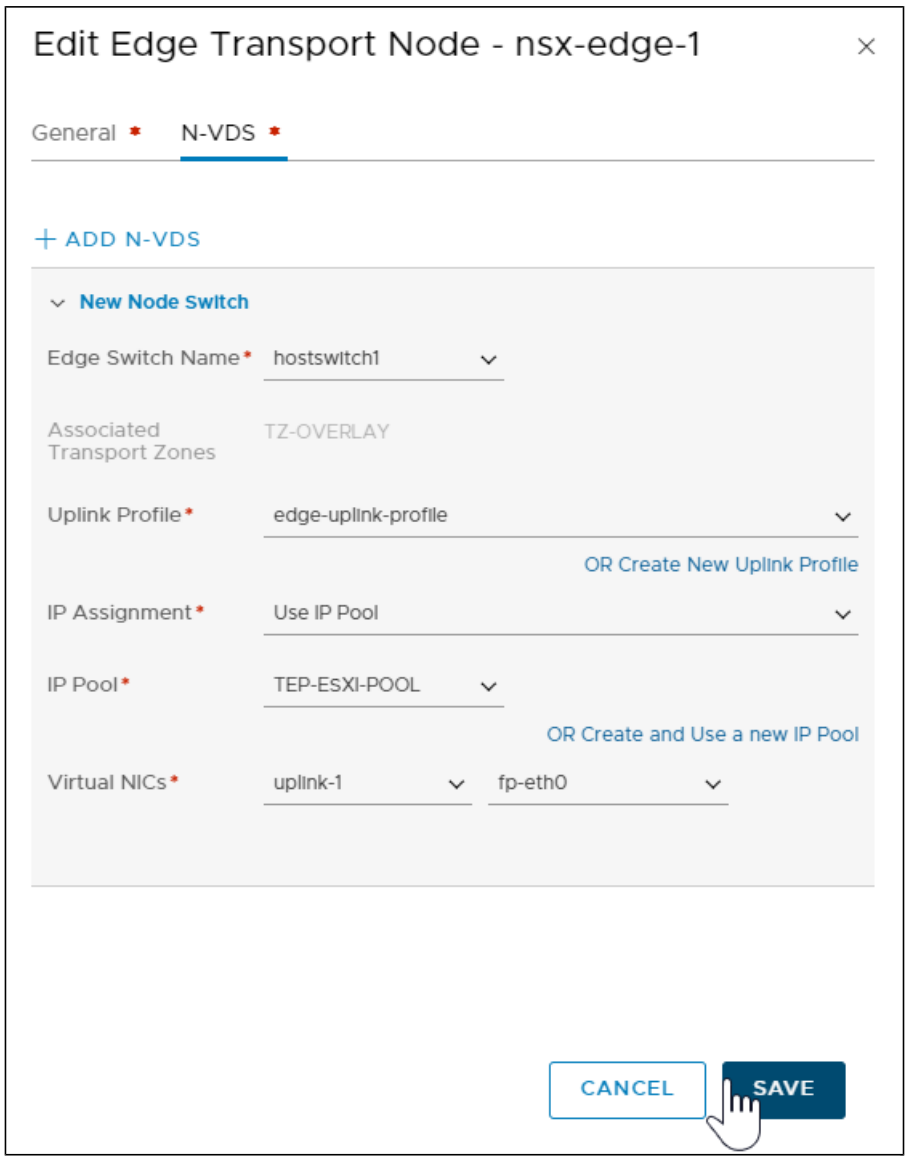
   - **Name**: Enter a name for the Edge Transport Node, such as `nsx-edge-1-tn`.
   - **Transport Zones**: Select both Transport Zones: TZ-Overlay (Overlay) and TZ-VLAN (VLAN).



6. Select the **N-VDS** tab.

7. Configure the distributed switch for the overlay network.

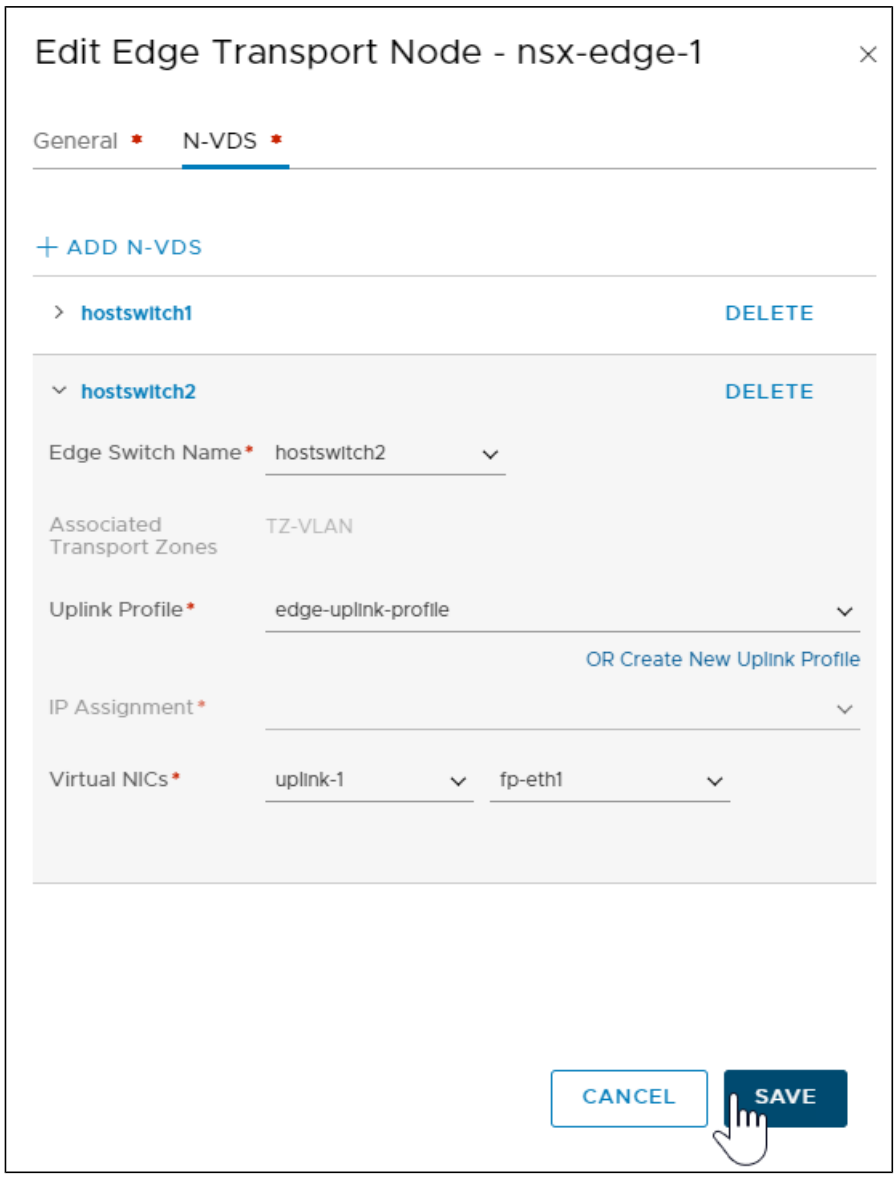
   - **Edge Switch Name**: `hostswitch1`
   - **Associated Transport Zones**: `TZ-OVERLAY`
   - **Uplink Profile**: `edge-uplink-profile`
   - **IP Assignment**: `Use IP Pool`
   - **IP Pool**: `TEP-ESXi-POOL`
   - **Virtual NICs**: `uplink-1` and `fp-eth0` (corresponds to Edge VM vnic1 (second vnic))



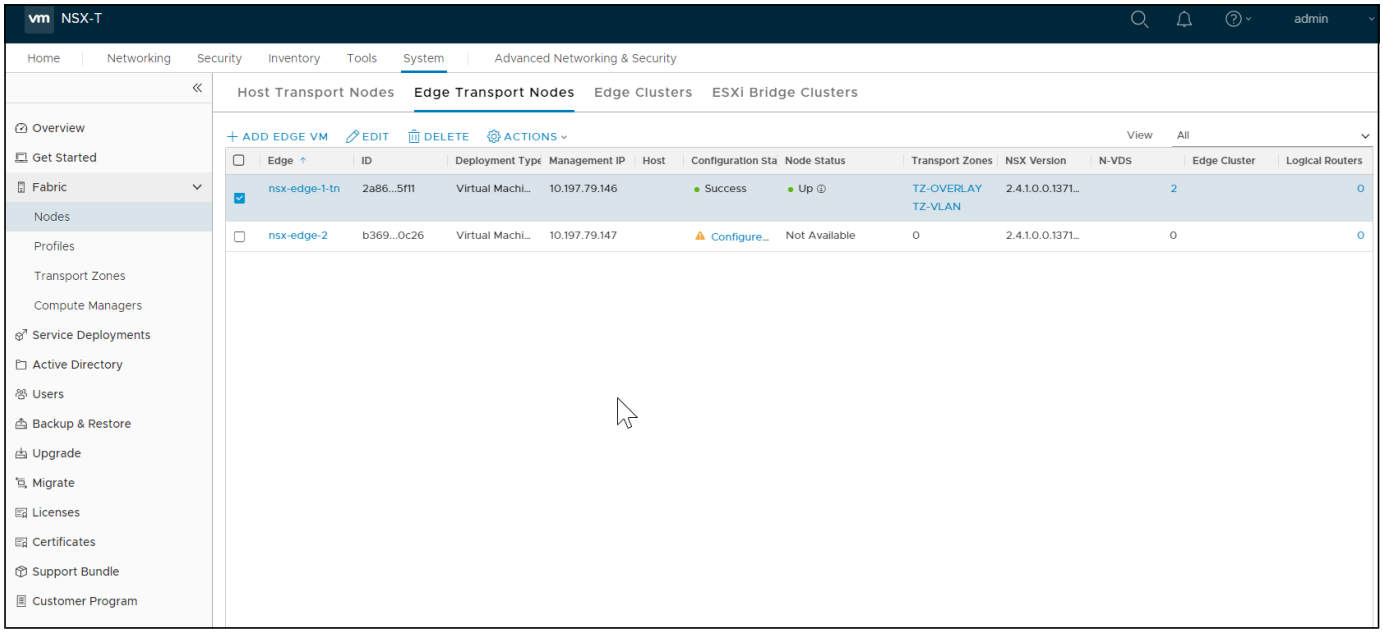
8. Click **Add N-VDS** to add the second virtual distributed switch.

9. Configure the second transport node switch.

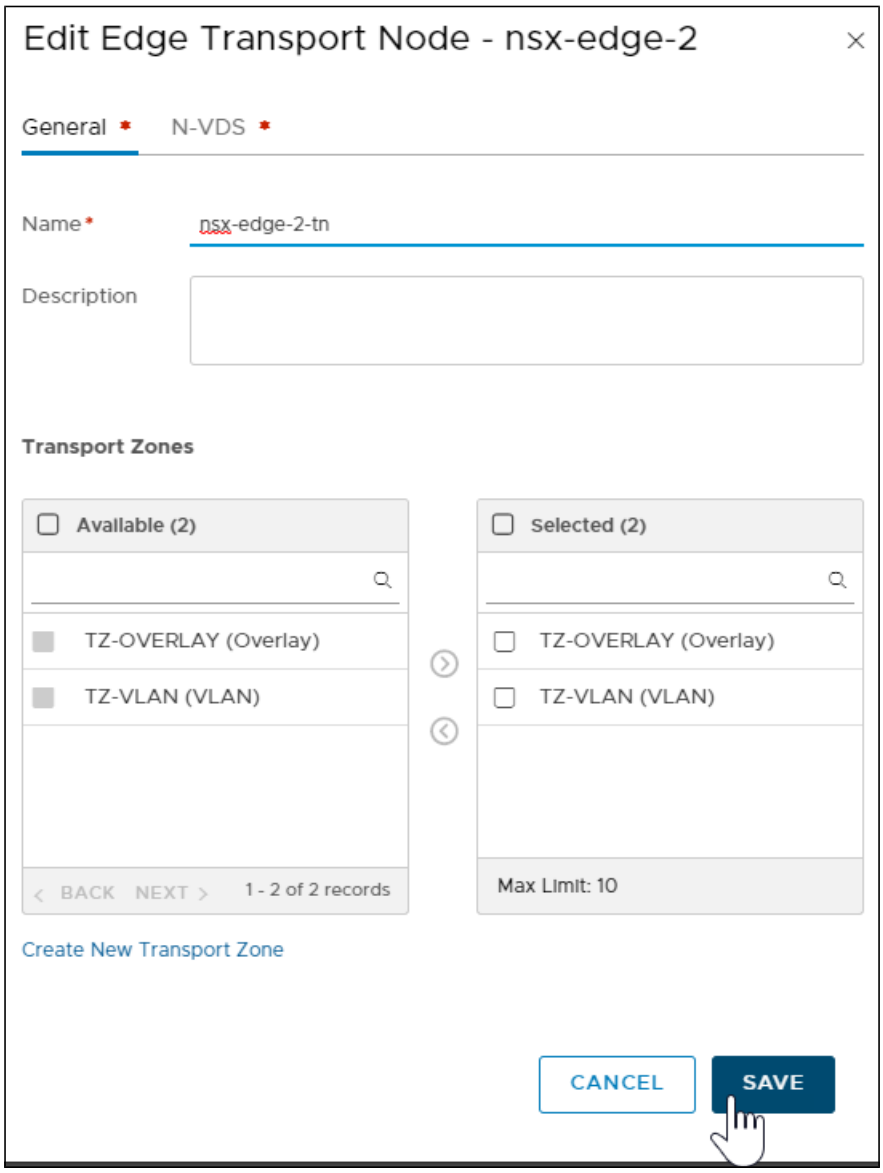
   - **Edge Switch Name**: `hostswitch2`
   - **Associated Transport Zones**: `TZ-VLAN`
   - **IP Assignment**: leave blank
   - **Uplink Profile**: `edge-uplink-profile`
   - **Virtual NICs**: `uplink-1` and `fp-eth1` (corresponds to Edge VM vnic2 (third vnic))

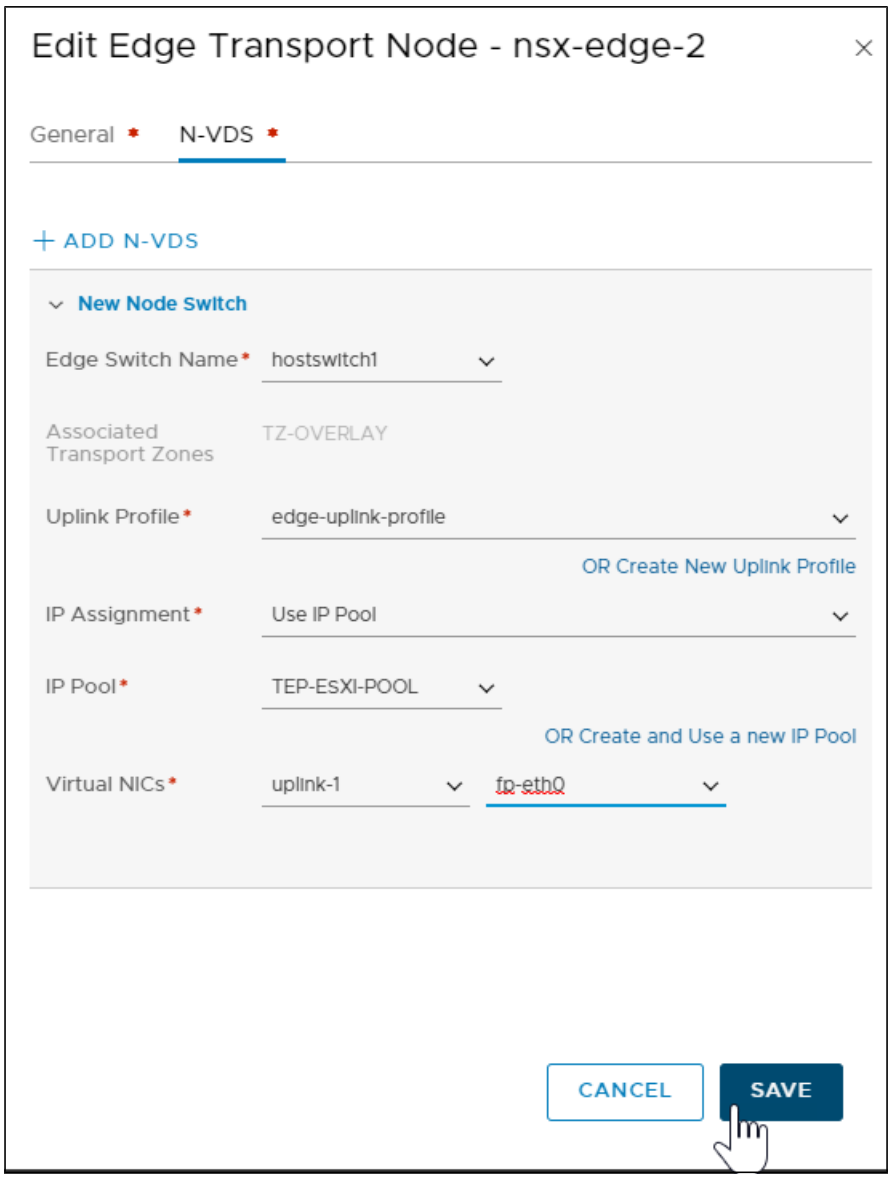


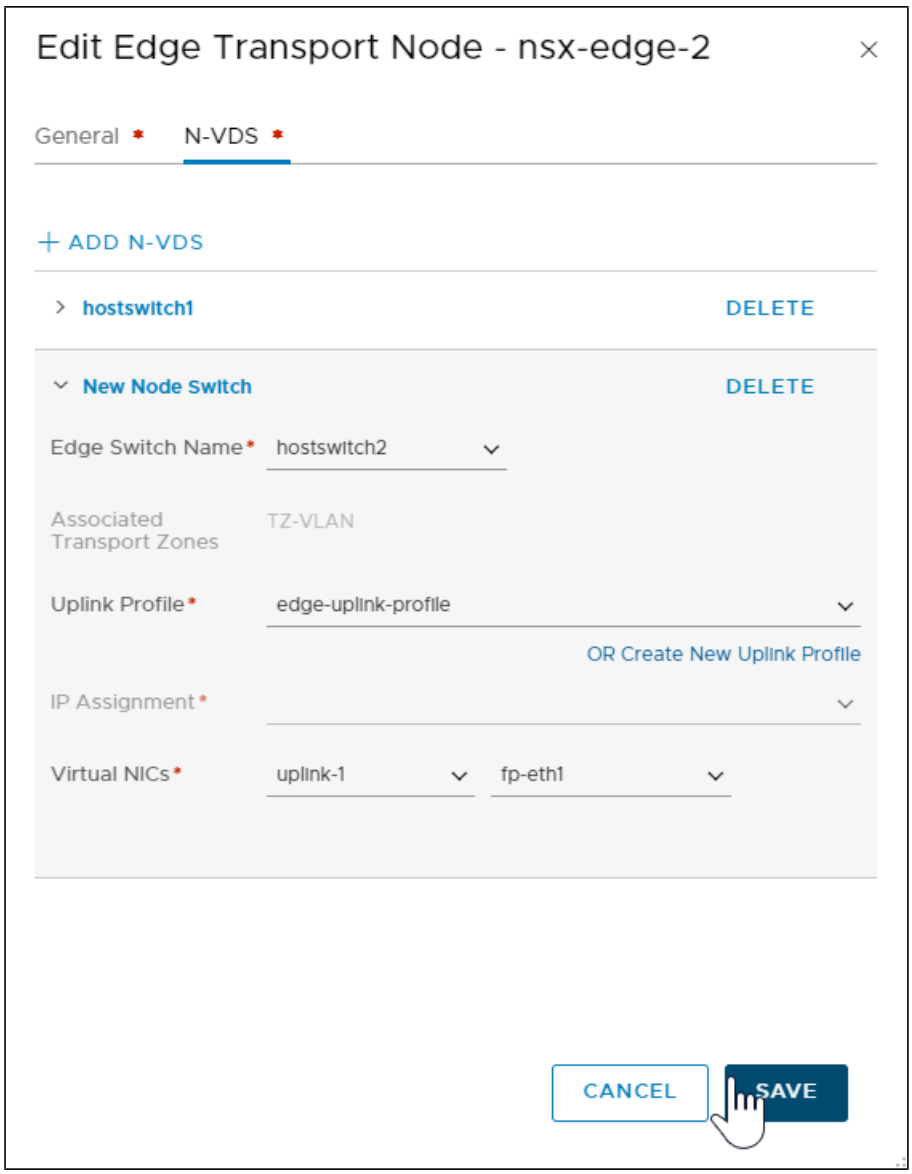
10. Click **Save**. You should see that the **Configuration State** is Success and the **Node Status** is initially Up.

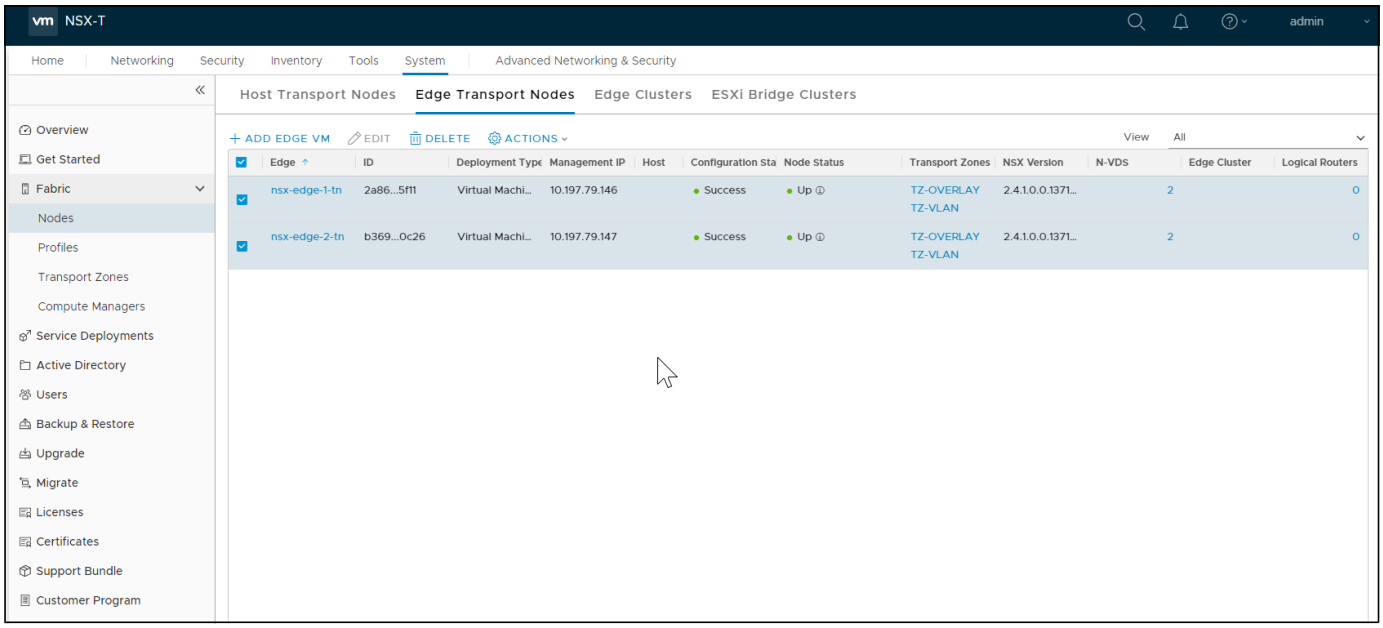


11. Repeat this procedure for the second Edge Transport Node.









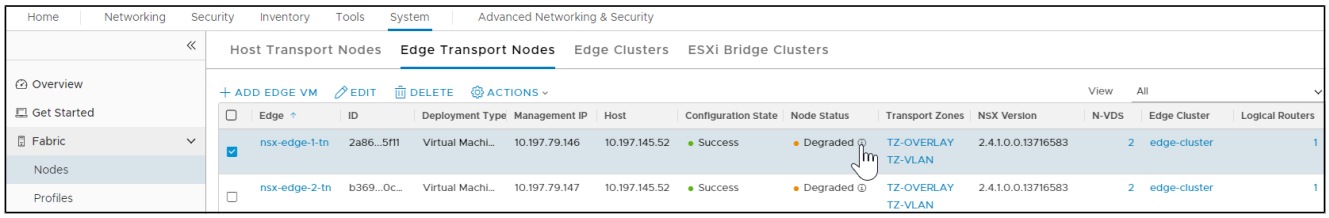
Verify Edge Transport Node Configuration

To verify the configuration of each Edge Transport Node, complete the following steps:

1. In NSX-T Manager, select **System > Fabric > Nodes > Edge Transport Nodes**

2. For the first Edge Transport Node:

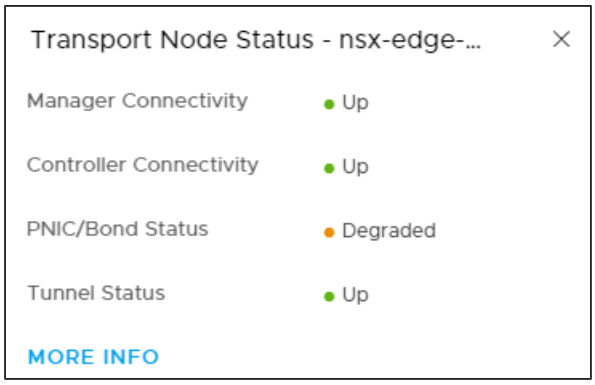
   - Verify that the **Configuration State** is `Success`.
   - Verify that the **Node Status** is **Degraded**, which is expected because there is no traffic yet.



3. Click the **Information** icon in the **Node Status** column for the first Edge Node.

4. Verify that Manager Connectivity, Controller Connectivity, and Tunnel Status are `UP` for the selected Edge Transport Node.

> **Note:** The PNIC/Bond Status is expected to be **Degraded**. It will change when you will have traffic over the tunnel.



5. Repeat this verification process for the second Edge Transport Node.

Next Step

See Create Edge Cluster.

NSX-T Installation Instructions Home

See Installing and Configuring NSX-T for Enterprise PKS.

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

Create Edge Cluster

In this topic

Page last updated:

This topic provides instructions for creating an NSX-T Edge Cluster for use with Enterprise PKS.

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

About Edge Clusters

In NSX-T Data Center, an Edge Cluster is a logical boundary for resourcing. Having a multi-node cluster of NSX-T Edge Nodes helps ensure that at least one NSX Edge is always available. To create a Tier-0 logical router or a Tier-1 router with stateful services such as NAT, a traffic load balancer, and other virtual network objects, you must associate them with an NSX Edge cluster.

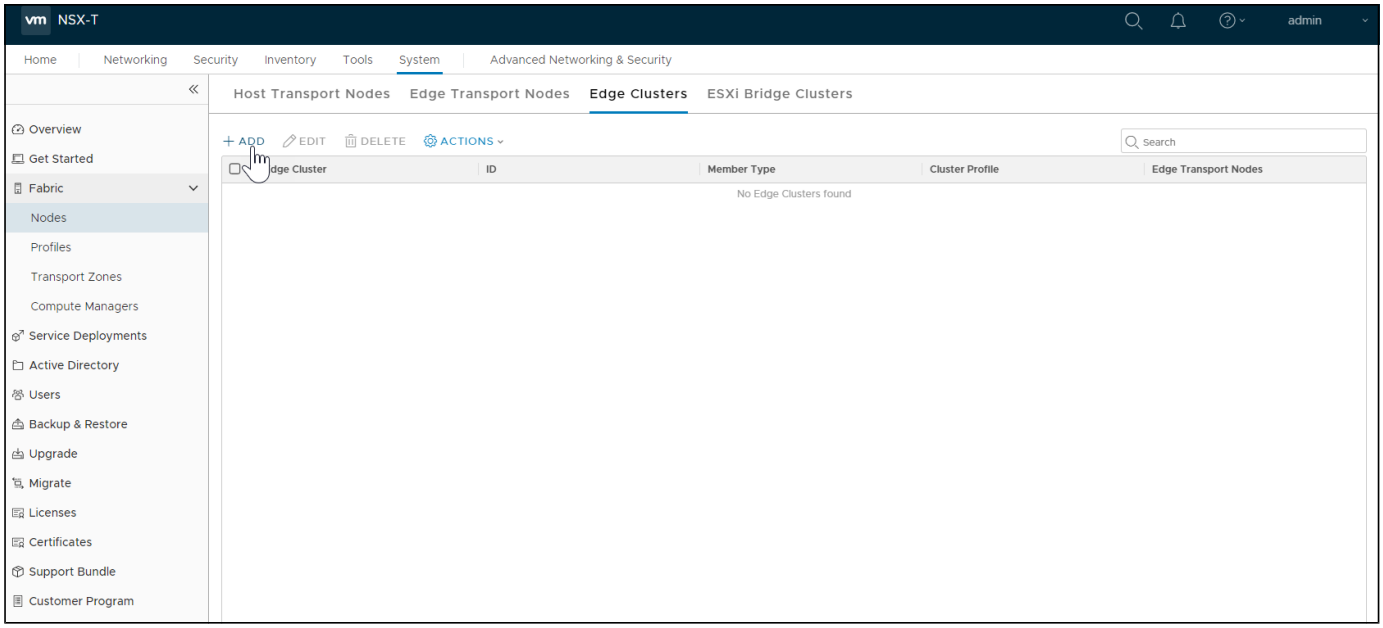
An Edge Cluster can have a maximum of 10 Edge Nodes. If the provisioning requires more Edge Nodes than what a single Edge Cluster can support, multiple Edge Clusters must be deployed. An NSX-T Edge Transport Node can be added to only one NSX-T Edge cluster. After creating the NSX-T Edge cluster, you can later edit it to add additional NSX-T Edge Nodes. An NSX-T Edge cluster can be used to back multiple logical routers.

For more information, see Edge Clusters in the NSX-T Data Center documentation.

Create Edge Cluster

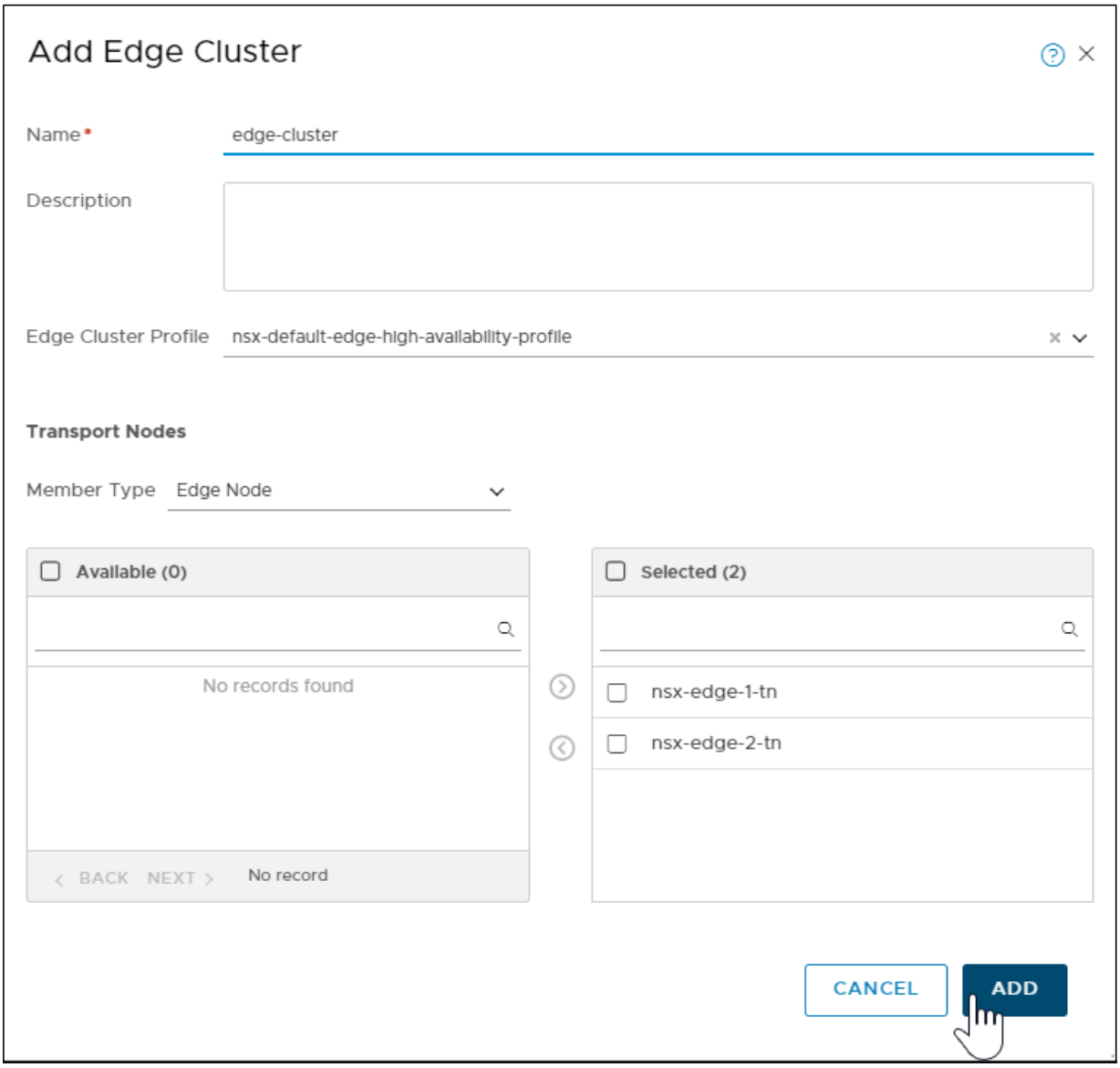
Create an NSX Edge Cluster and add each Edge Transport Node to the Edge Cluster by completing the following procedure:

1. In NSX Manager, select **System > Fabric > Nodes > Edge Clusters** and click **Add**.



2. Configure the NSX Edge Cluster as follows:

- **Name**: Enter the NSX Edge cluster a name.
- **Edge Cluster Profile**: Select `nsx-default-edge-high-availability-profile` from the menu.
- **Transport Nodes > Member Type**: Select `Edge Node` from the menu.
- From the **Available** column, select both NSX-T Edge Nodes and click the right-arrow to move them to the **Selected** column.

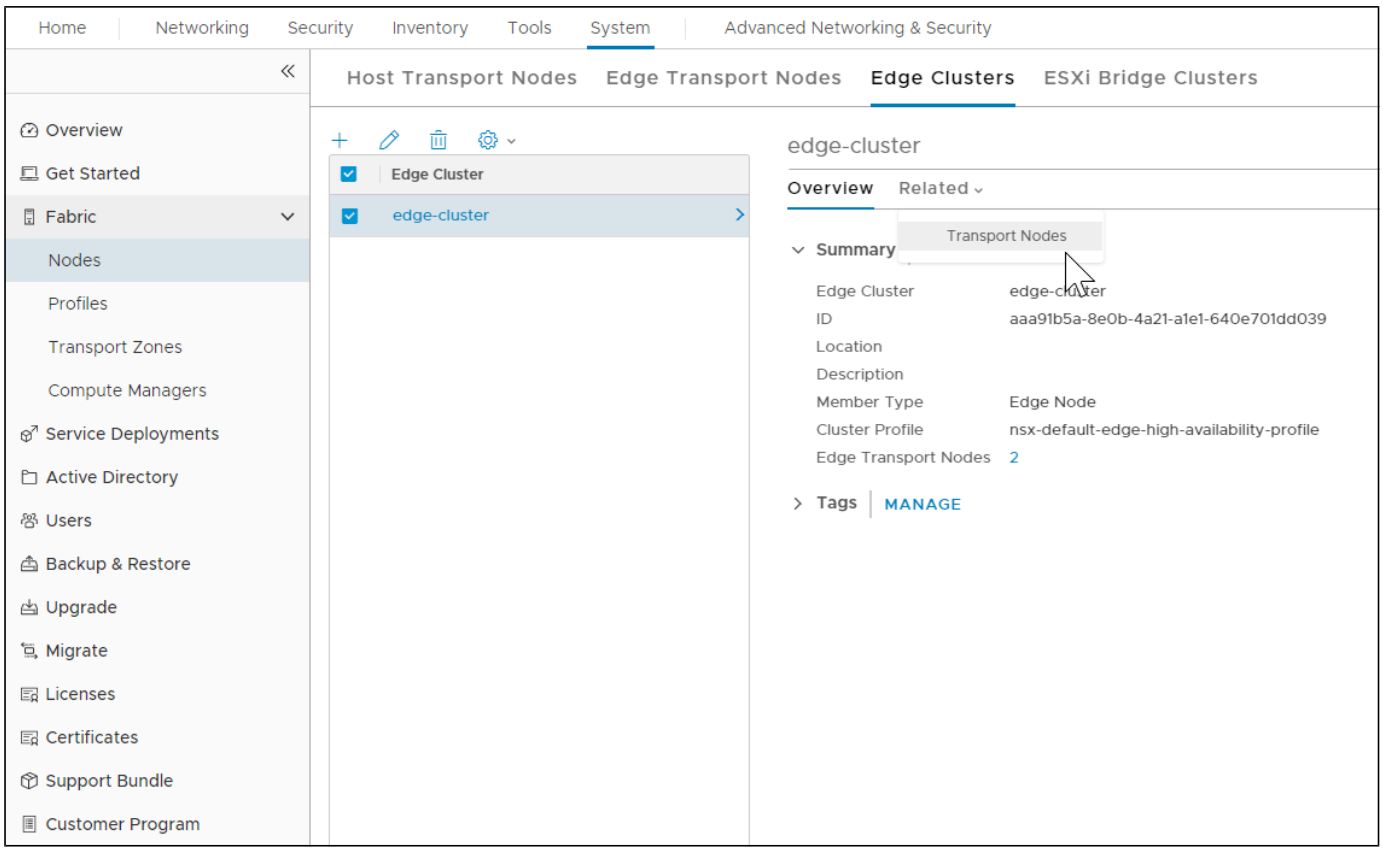


- Click **Add**.

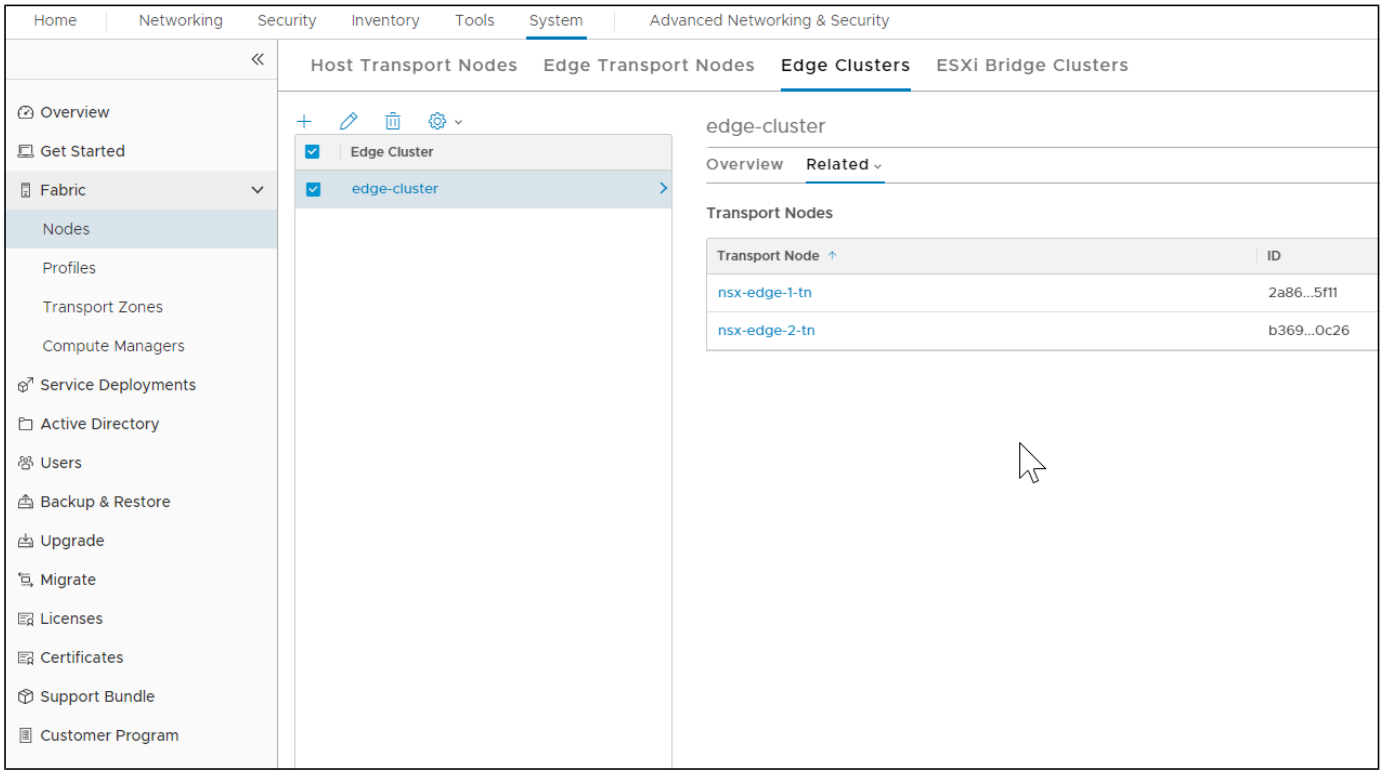
Verify Edge Cluster Creation

To verify Edge Cluster creation, complete the following steps:

1. In NSX-T Manager, select **System > Fabric > Nodes > Edge Clusters**
2. Verify that you see the new Edge Cluster.
3. Select **Edge Cluster > Related > Transport Nodes**.



4. Verify that both 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. SSH to NSX Edge Node 2 and repeat the above commands to verify proper connectivity.

7. Get the TEP IP addresses.

   - Navigate to **System > Fabric > Nodes > Edge Transport Nodes**
   - Select the Edge Transport Node, such as `nsx-edge-1-tn`.
   - Select the **Monitor tab**.

8. 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 TEP-IP-ADDRESS-EDGE-2
```

You should be able to ping Edge Transport Node 2 using the TEP address.

Next Step

See Create Tier-0 Logical Router.

NSX-T Installation Instructions Home

See Installing and Configuring NSX-T for Enterprise PKS.

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

Create Tier-0 Logical Router

In this topic

Page last updated:

This topic provides instructions for creating an NSX-T Tier-0 Logical Router for use with Enterprise PKS.

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

About Tier-0 Logical Routers

In NSX-T Data Center, a Tier-0 logical router provides a gateway service between the logical and physical network.

There are several steps involved in the process of creating a Tier-0 router, summarized as follows. Detailed step-by-step instructions are provided after the summary. For additional information, see Tier-0 Logical Router in the NSX-T documentation.

1. Define a T0 logical switch with an ingress/egress uplink port.
2. Attach the T0 LS to the VLAN Transport Zone.
3. Create a logical router port and assign to it a routable CIDR block (`10.172.1.0/28`, for example) that your environment uses to route to all Enterprise PKS-assigned IP pools and IP blocks. Work with your network administrator to get the IP address and subnet mask (prefix length) to specify for the T0 Router Port.
4. Connect the T0 router to the uplink VLAN logical switch.
5. Attach the T0 router to the Edge Cluster and set HA mode to **Active-Standby**, if you are using NAT mode. 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. If you are using No NAT mode, you can use **Active-Active** mode for the T0 router.
6. Configure T0 routing to the rest of your environment using the appropriate routing protocol for your environment or by using static routes.

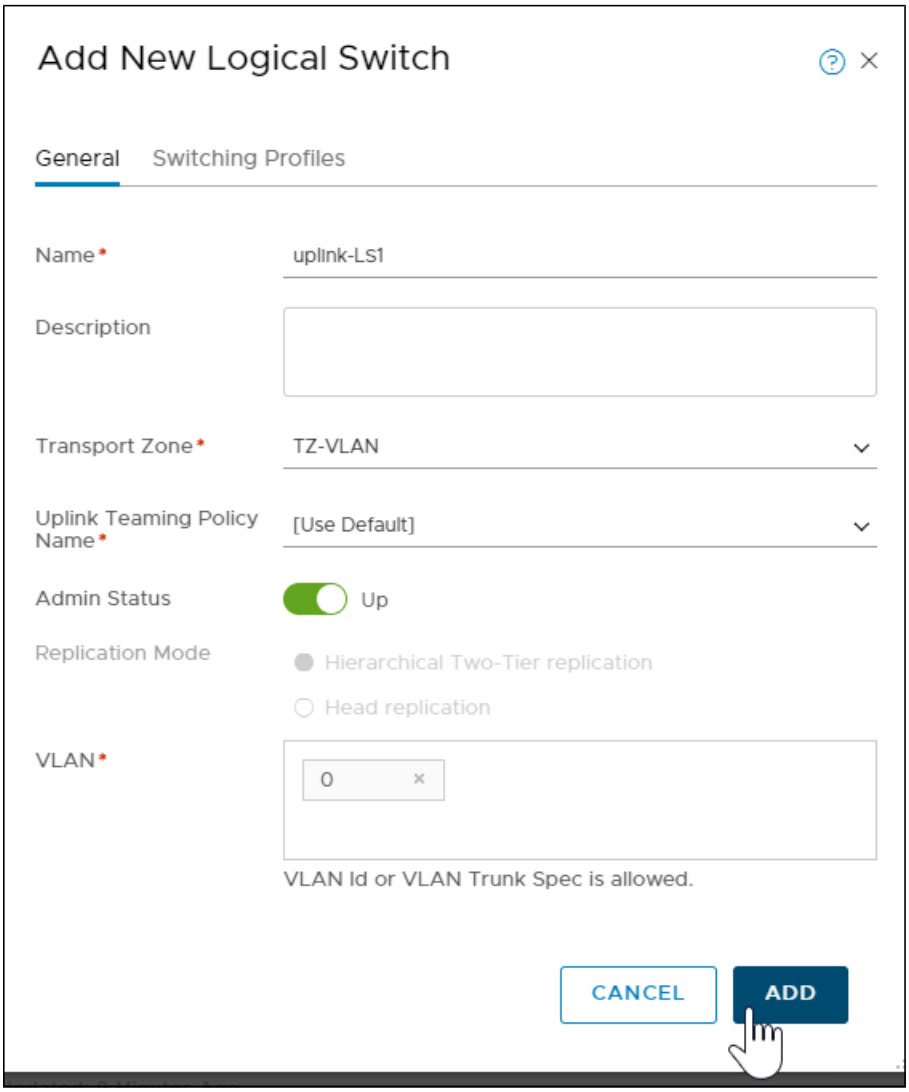
Create T0 Router

Create a Tier-0 Logical Router for Enterprise PKS by completing the following procedure.

Create VLAN Logical Switch (LS)

1. In NSX Manager, go to **Advanced Networking & Security** > **Switching** > **Switches**.
2. Click **Add** and create a VLAN logical switch (LS).
3. Configure the logical switch as follows and click **Add** to save the configuration:

- **Name**: `uplink-LS11`, for example
- **Transport Zone**: `TZ-VLAN`, for example
- **Uplink Teaming Policy Name**: [Use Default]
- **Admin Status**: Up
- **VLAN**: `0`



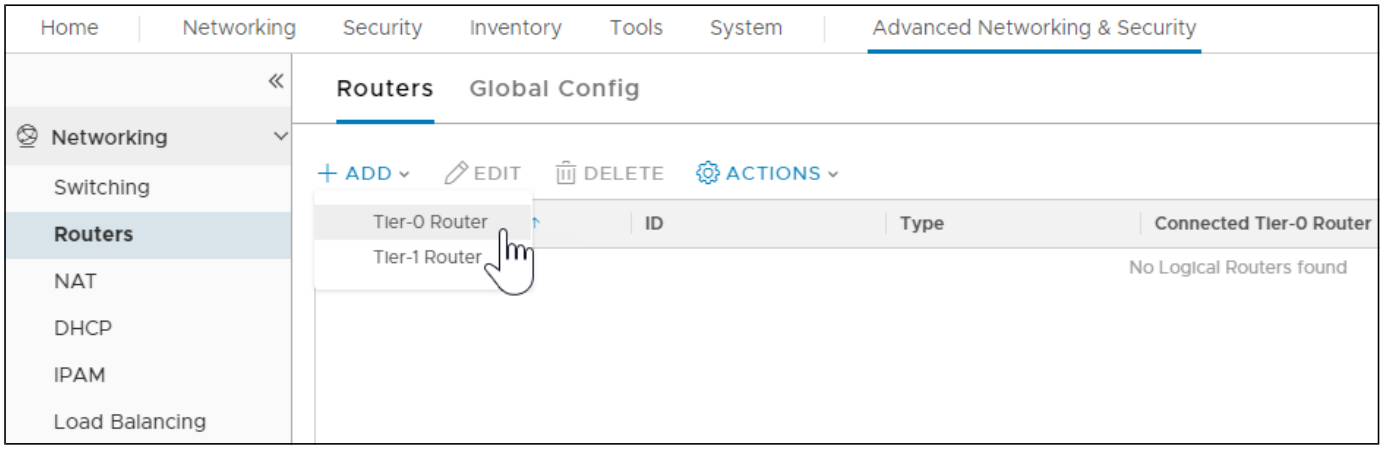
Create T0 Router Instance

When you create the Tier-0 Router, you must specify the HA mode. If you are using NAT mode, you must select **Active-Standby**. NAT

rules are applied on the T0 by the NSX-T Container Plugin (NCP) component. If the T0 Router is not set to **Active-Standby**, NCP will not be able to create NAT rules on the T0 Router. If you are using routable IP addresses for Kubernetes nodes (that is, No NAT mode), you can use **Active-Active**.

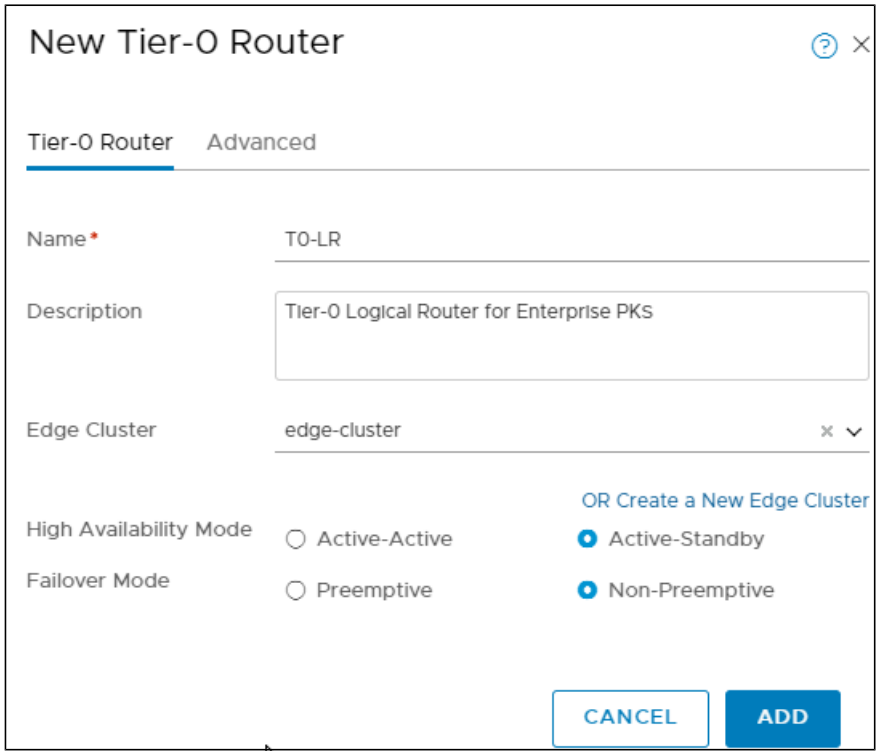
1. In NSX Manager, go to **Advanced Networking & Security** > **Routing > Routers**.

2. Click **Add** and select the **Tier-0 Router** option.

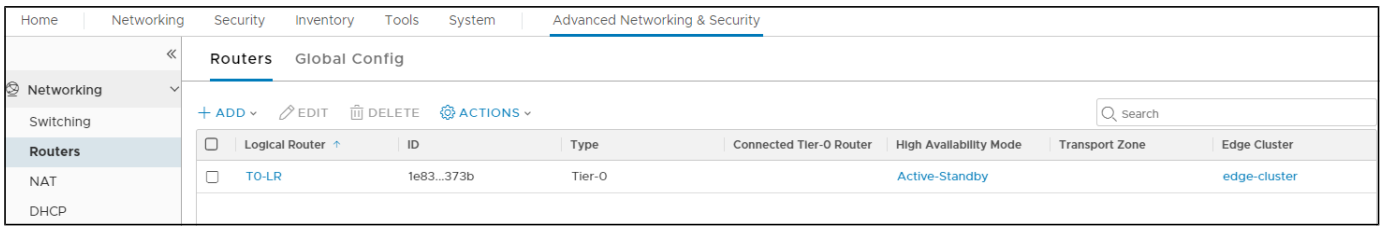


3. Configure the new T0 router as follows:

   - **Name**: Enter a name for the T0 router, `T0-LR` for example.
   - **Edge Cluster**: Select the Edge Cluster, `edge-cluster` for example.
   - **High Availability Mode**: Select `Active-Standby` (if NAT mode) or `Active-Active` (if No NAT mode).
   - **Failover Mode**: Select `Preemptive` or `Non-Preemptive` based on your requirements.



4. Click **Add** and verify you see the new T0 Router instance:



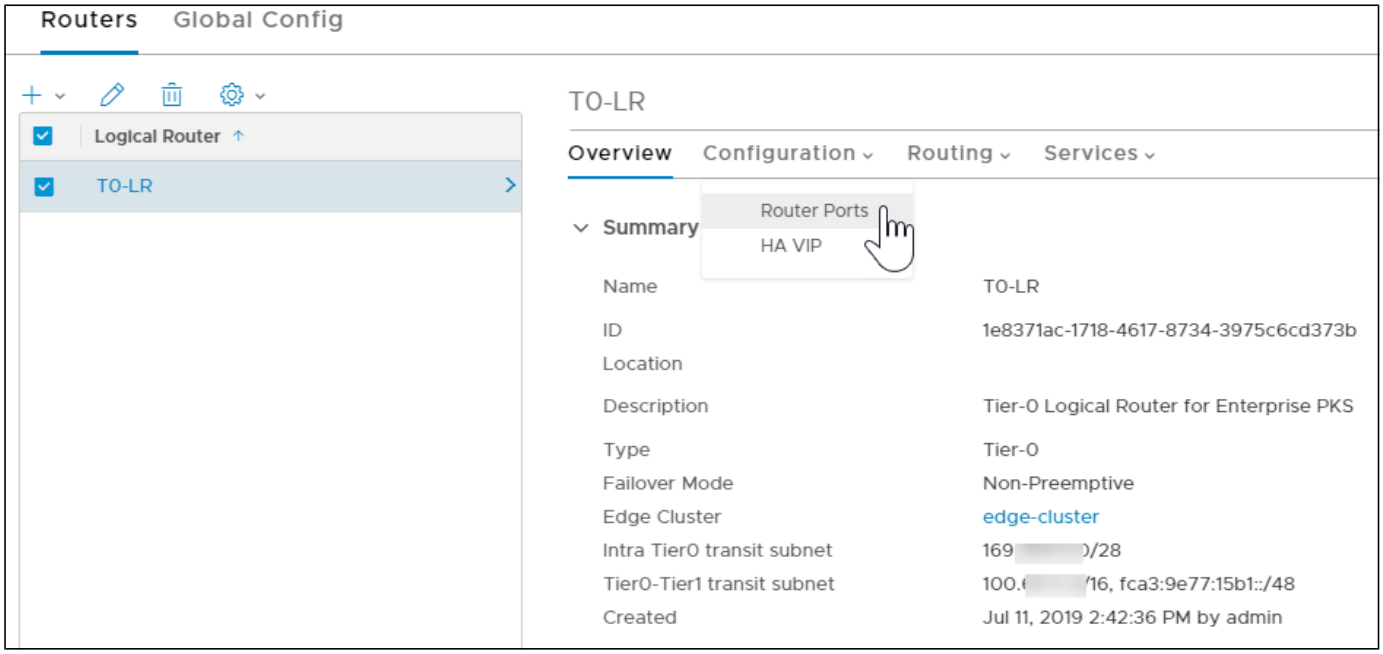
Create T0 Router Ports for both Edge Transport Nodes

In this section you configure a new T0 Router port by attaching the T0 Router port to the logical switch you created at the beginning of this procedure (`uplink-LS1` , for example). You then assign an IP address and CIDR that your environment uses to route to all PKS-assigned IP pools and IP blocks.

1. In NSX Manager, go to **Advanced Networking & Security** > **Routing > Routers**.

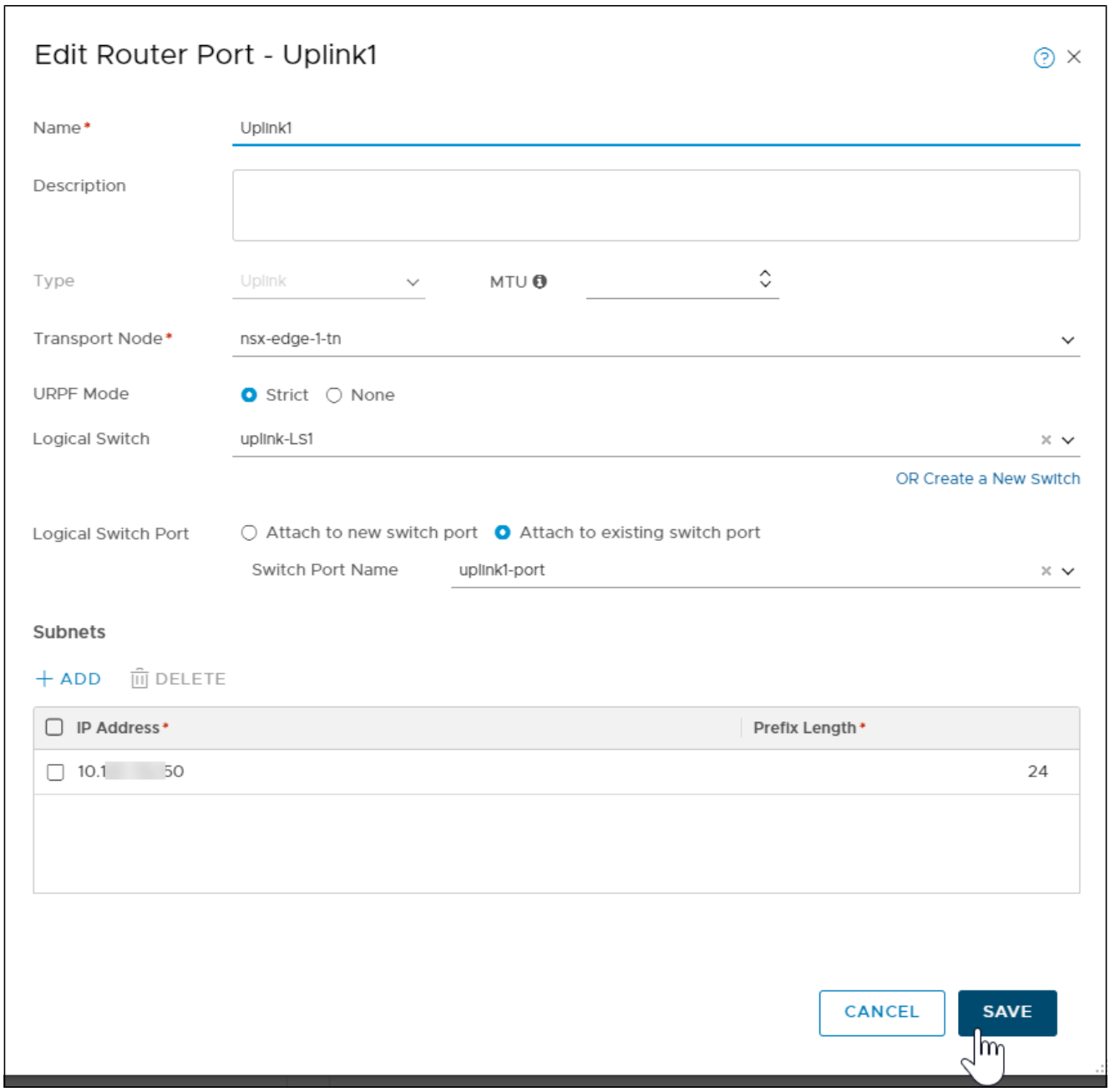
2. Select the T0 Router you just created.

3. Select **Configuration > Router Ports**.



4. Click **Add** and configure the new T0 router port as follows:

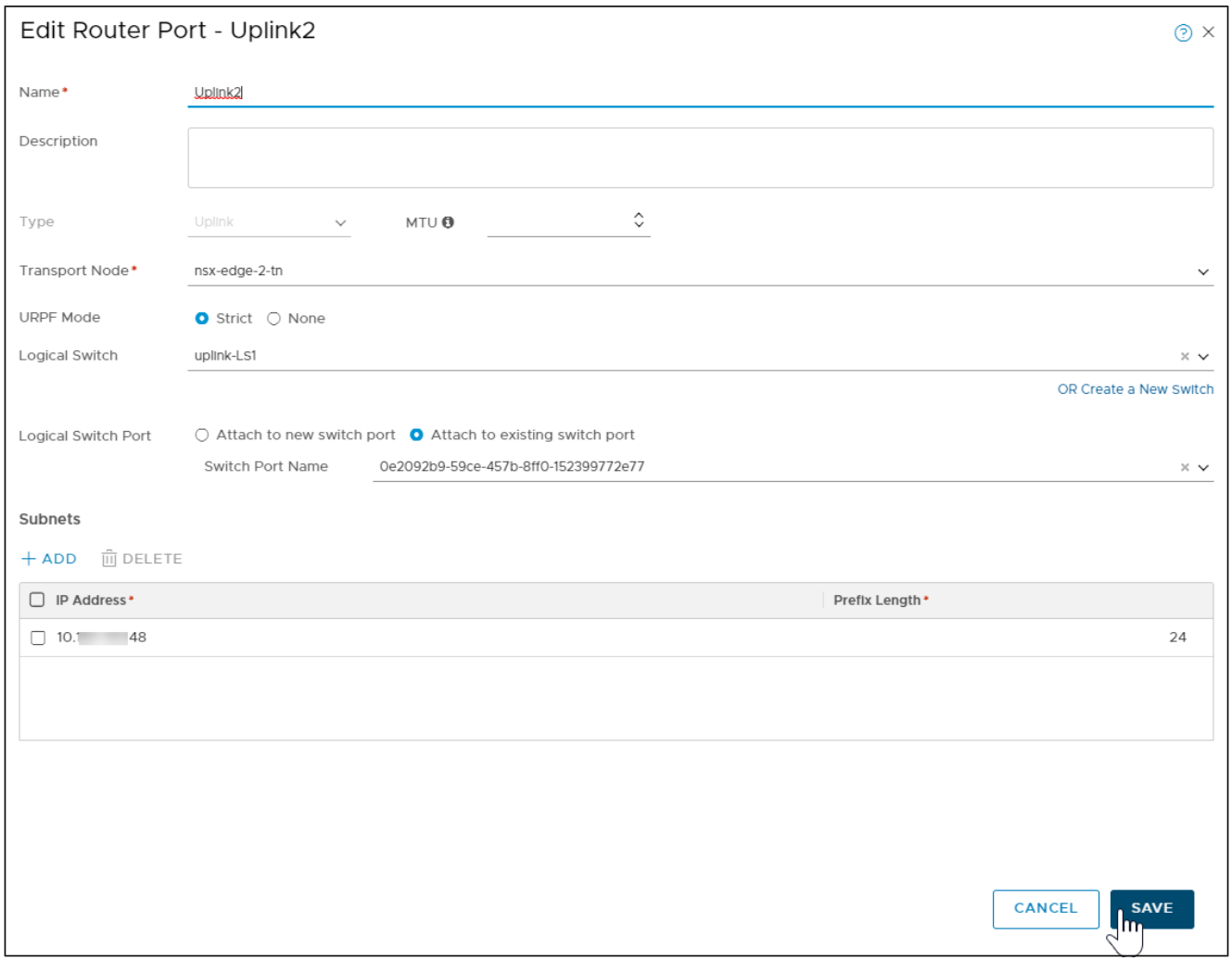
   - **Name**: `Uplink1`
   - **Type**: Uplink
   - **Transport Node**: `nsx-edge-1-tn` , for example
   - **URPF Mode**: Strict (typically)
   - **Logical Switch**: `uplink-LS1` (the logical switch you created earlier in this procedure)
   - **Logical Switch Port**: Attach to new switch port
   - **Logical Switch Port**: `uplink1-port` , for example
   - **IP Address**: `10.145.22.115` , for example
   - **Prefix Length** `24` , for example



5. Click **Add** and verify that you see the new T0 Router Port interface.

6. Repeat this procedure for the second Edge Transport Node.

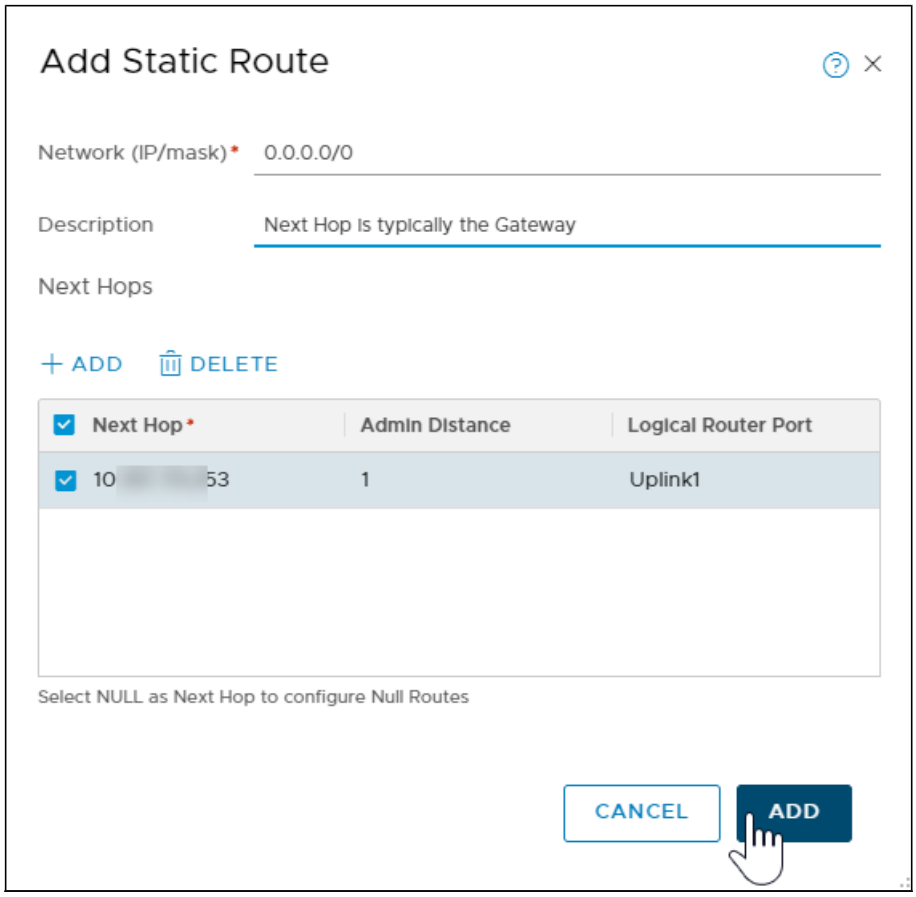
   - **Name**: `Uplink2`
   - **Type**: Uplink
   - **Transport Node**: Select the second Edge Node, `nsx-edge-2-tn`, for example
   - **URPF Mode**: Strict (typically)
   - **Logical Switch**: `uplink-LS1` (the logical switch you created earlier in this procedure)
   - **Logical Switch Port**: Attach to existing switch port
   - **Logical Switch Port**: Select the UUID of the switch port you created for Uplink1
   - **IP Address**: `10.145.22.116`, for example
   - **Prefix Length** `24`, for example



Add a Static Route

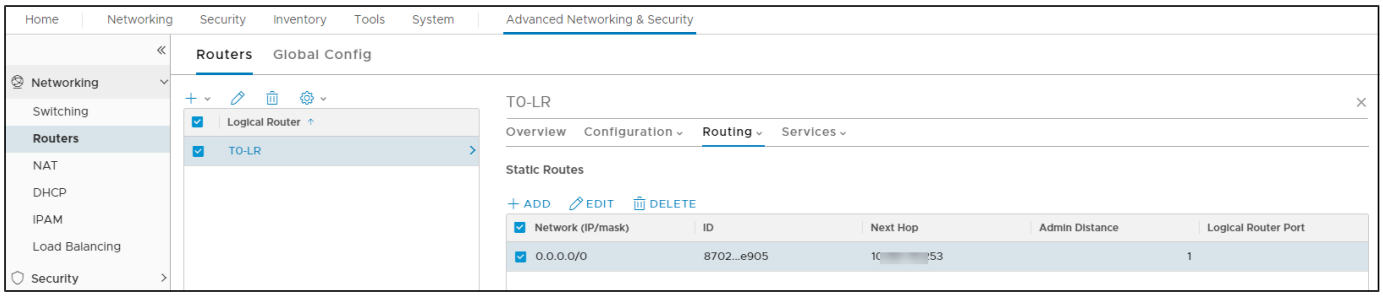
Configure T0 routing to the rest of your environment using static routes (if you are using NAT-mode) or the appropriate routing protocol (if you are using no-NAT-mode). The following example uses static routes for the T0 router. The CIDR used must route to the IP address you just assigned to your T0 uplink interface.

1. Go to **Advanced Networking & Security** > **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.
   - **Network (IP/mask)**: `0.0.0.0/0`
   - **Next Hop**: `10.145.22.117`, for example
   - **Admin Distance**: `1`



- **Logical Router Port**: `Uplink1`

4. Click **Add** and verify that see the newly created static route:



Verify T0 Router Creation

If successfully configured, the T0 Router uplink port IP address should be reachable from your corporate network.

1. Go to **Advanced Networking & Security** > **Routing > Routers** and select the T0 Router.

2. In the **Overview** tab, review the **Summary** section and **High Availability Mode**.

3. From your local laptop or workstation, ping the uplink IP address. For example:

```
PING 10.40.22.24 (10.40.22.24): 56 data bytes
64 bytes from 10.40.22.24: icmp_seq=0 ttl=53 time=33.738 ms
64 bytes from 10.40.22.24: icmp_seq=1 ttl=53 time=36.965 ms
```


Next Step

See Configure Edge Node HA.

NSX-T Installation Instructions Home

See Installing and Configuring NSX-T for Enterprise PKS.

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

Configure Edge Nodes for High Availability (HA)

In this topic

Page last updated:

This topic provides instructions for configuring Edge Node high-availability (HA) for Enterprise PKS.

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

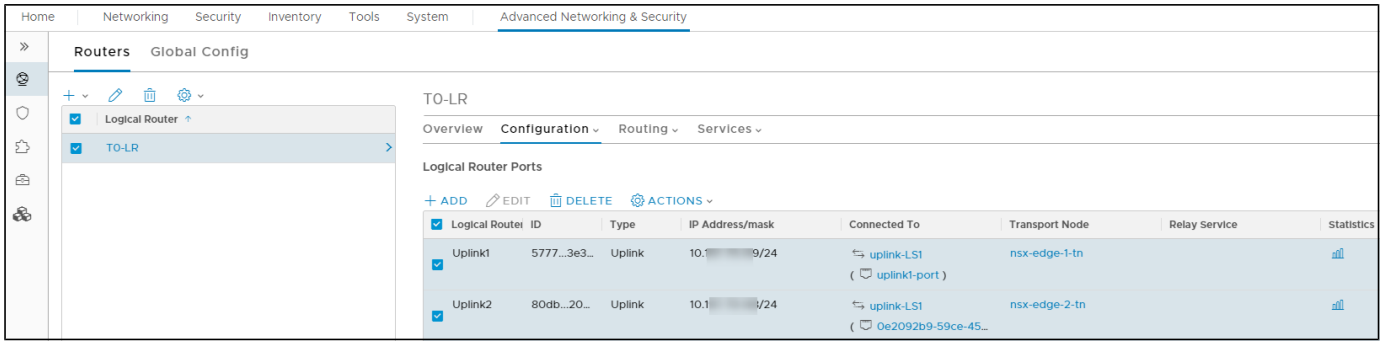
About Edge Node HA

If Enterprise PKS is deployed in NAT mode, Edge Nodes for high availability (HA) use Active/Standby mode to support failover. Properly configuring the Edge Nodes for HA requires two uplinks on the T0 router: one attached to Edge Transport Node 1, and the other attached to Edge Transport Node 2. 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.

Verify Tier-0 Router Configuration

If the T0 Router is not correctly configured for HA, failover to the standby Edge Node will not occur. Before you configure Edge Node HA, verify the following for the T0 Router:

1. In NSX Manager, go to **Advanced Networking & Security** > **Routing > Routers**.
2. Select the T0 Router you created for Enterprise PKS.
3. Select **Configuration > Router Ports**.
4. Verify that you created two uplink router ports for the Edge Nodes as described in the topic Create Tier-0 Logical Router.



5. With the T0 router selected, select **Routing** > **Static Routes**.

6. Verify that you defined a default static route so that the next hop points to the physical router.

   - **Network**: `0.0.0.0/0`
   - **Next Hop**: `10.179.67.233`
   - **Logical Router Port**: empty

7. If you did not create the required router ports and static routes for the Edge Transport Nodes, see Create Tier-0 Logical Router.

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, that peer with the T0 router must use the VIP address.

> **Note**: The IP addresses for the uplink-1 router port, uplink-2 router port, and the HA VIP must belong to same subnet.

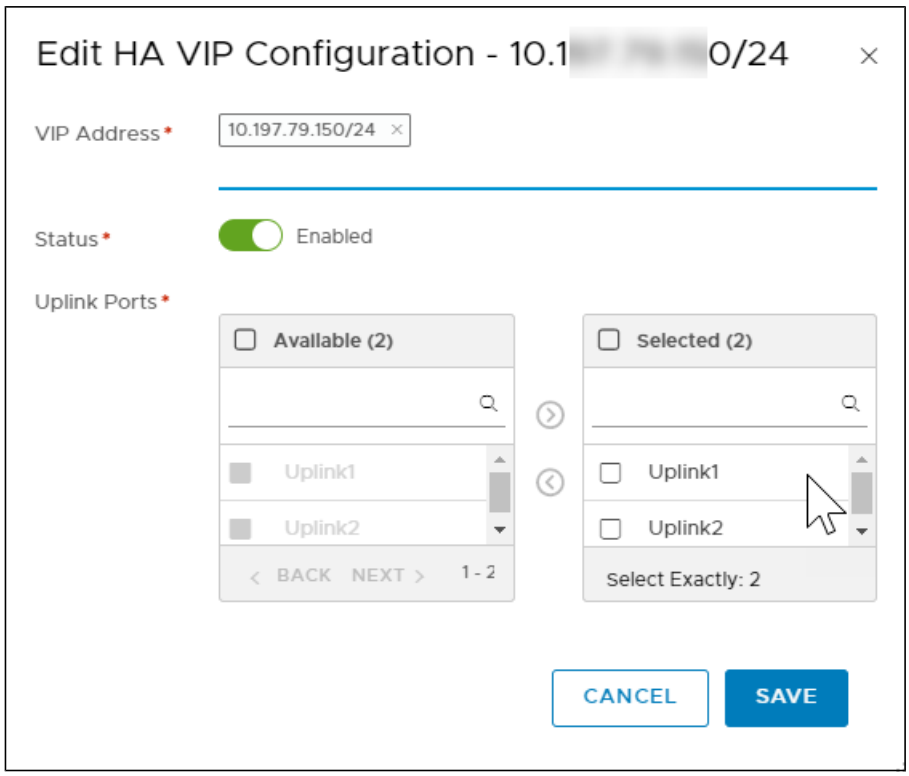
1. In NSX Manager, go to **Advanced Networking & Security** > **Routing > Routers**.

2. Select the T0 Router you created for Enterprise PKS.

3. Select **Configuration** > **HA VIP**.

4. Create the VIP as follows.

   - **VIP Address**: `10.179.67.235/24`, for example
   - **Status**: Enabled
   - **Uplinks Ports**: `Uplink1` and `Uplink2`, for example



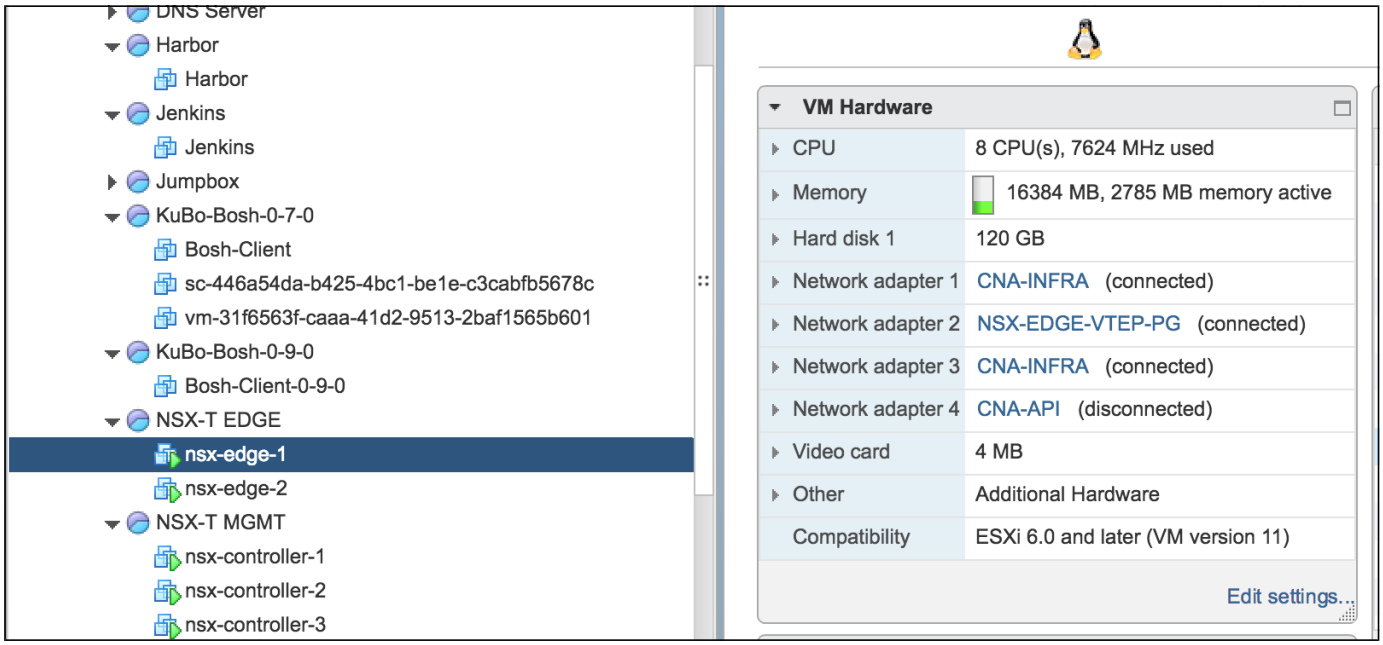
5. Click **Save** and verify the creation of the HA VIP.

Disconnect Unused NICs

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.

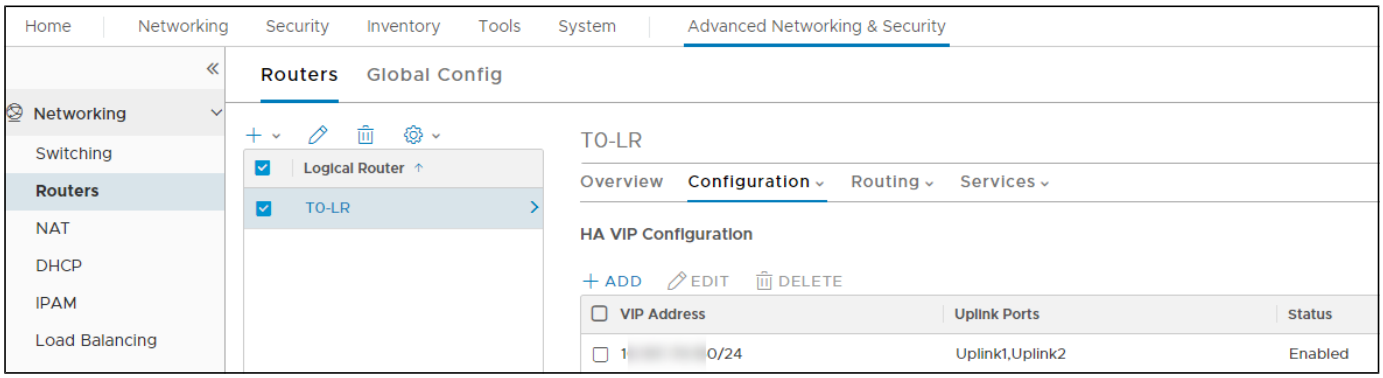
1. Using vCenter, disconnect any unused vNIC interface in each Edge Node VM (this interface can cause duplicate packets.)

2. For example, in the screenshot below, **Network adapter 4** is not being used, so it is disconnected:



Verify Edge Node HA Configuration

1. The T0 router should display both Edge Node uplink ports in active/standby pairing.



2. Run the following commands to verify HA channels:

```
nsx-edge-1> get high-availability channels
nsx-edge-1> get high-availability channels stats
nsx-edge-1> get logical-router
nsx-edge-1> get logical-router ROUTER-UUID high-availability status
```

3. Repeat for Edge Node 2.

Next Step

See Configure ESXi Hosts as NSX Transport Nodes.

NSX-T Installation Instructions Home

See Installing and Configuring NSX-T for Enterprise PKS.

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

Prepare ESXi Hosts as NSX-T Transport Nodes

In this topic

Page last updated:

This topic provides instructions for preparing ESXi hosts as NSX transport nodes.

Prerequisites

Make sure you have completed all preceding NSX-T installation tasks.

About ESXi Host Preparation

In NSX-T Data Center, a Transport Node allows nodes to exchange traffic for virtual networks. ESXi hosts dedicated to the Enterprise PKS Compute Cluster must be prepared as transport nodes.

For each ESXi host in the NSX-T Fabric to be used for Enterprise PKS-provisioned Kubernetes clusters, create an associated transport node. For example, if you have three ESXi hosts in the vCenter Cluster reserved for Enterprise PKS use, create three transport nodes. Add the Overlay Transport Zone to each ESXi Host 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-T Host Transport Nodes that is not in use by other vSwitches on the ESXi host. Make sure the `vmnic1` (second physical interface) of the ESXi host is not used. NSX-T will take ownership of it (opaque NSX vswitch will use it as uplink).

> **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.

Create Host Transport Nodes

Complete the following operation for each ESXi host to be used by the PKS Compute Cluster.

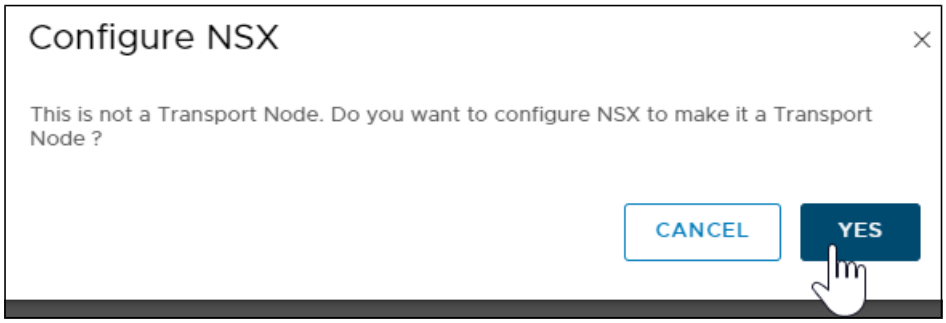
1. In NSX-T Manager, go to **System > Fabric > Nodes > Host Transport Nodes**
2. For the **Managed by** field, select **VMware vCenter Server App** from the dropdown menu.

   > **Note**: This assumes that you configured vCenter as the Compute Manager.

3. Expand the Cluster icon so the ESXi hosts are displayed.

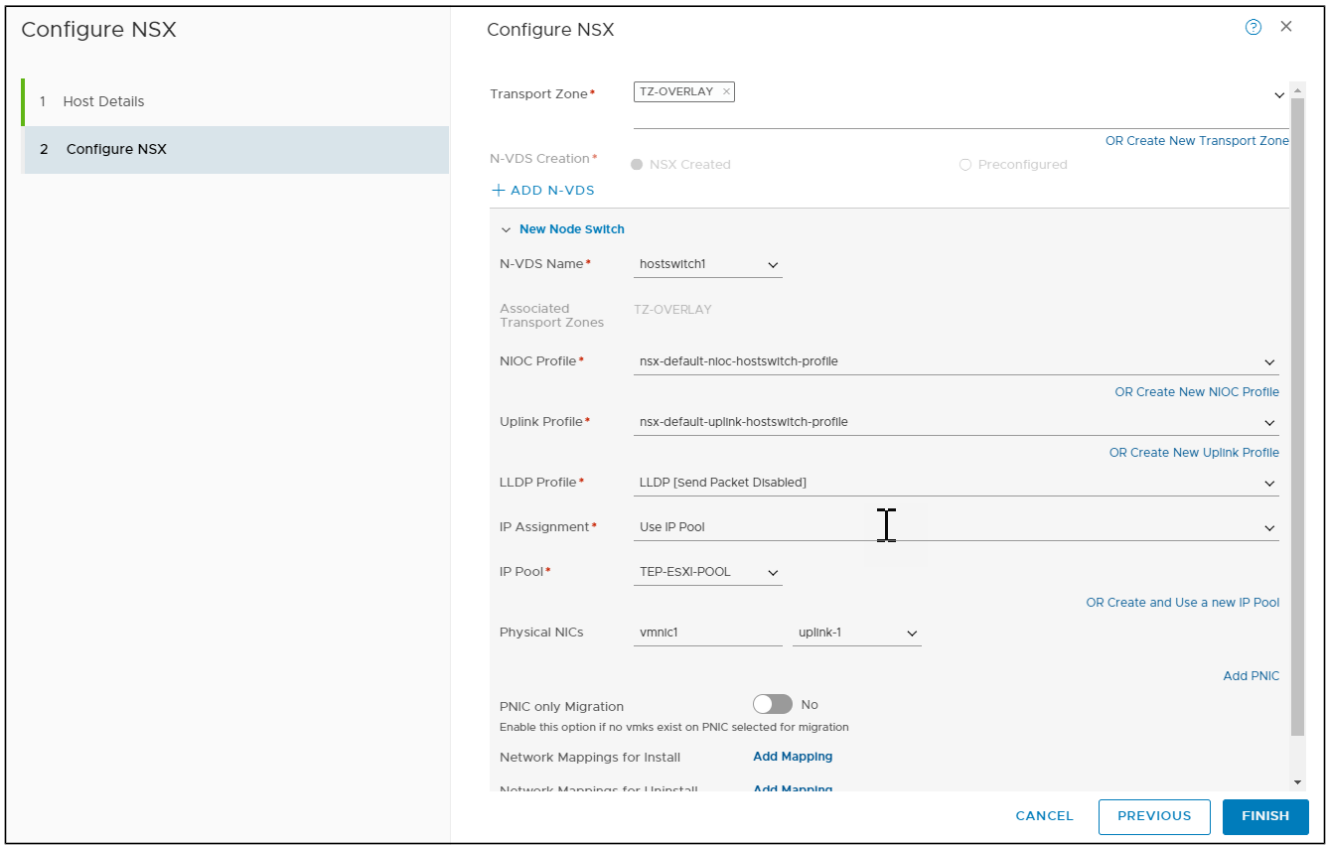
4. In the **NSX-T Configuration** column, click the **Not Configured** icon to begin the process.

5. Click **Yes** to configure the ESXi host as a transport node.



6. Configure the ESXi host as follows:

   - **Transport Zone**: TZ-Overlay
   - **N-VDS Name**: `hostswitch1`
   - **NIOC Profile**: `nsx-default-noic-hostswitch-profile`
   - **Uplink Profile**: `nsx-default-uplink-hostswitch-profile`
   - **LLDP Profile**: LLDP [Send Packet Disabled]
   - **IP Assignment**: `Use IP Pool`
   - **IP POOL**: `TEP-ESXi-POOL`
   - **Physical NICs**: `vmnic1` and `uplink-1`
   - **PNIC only Migration**: No

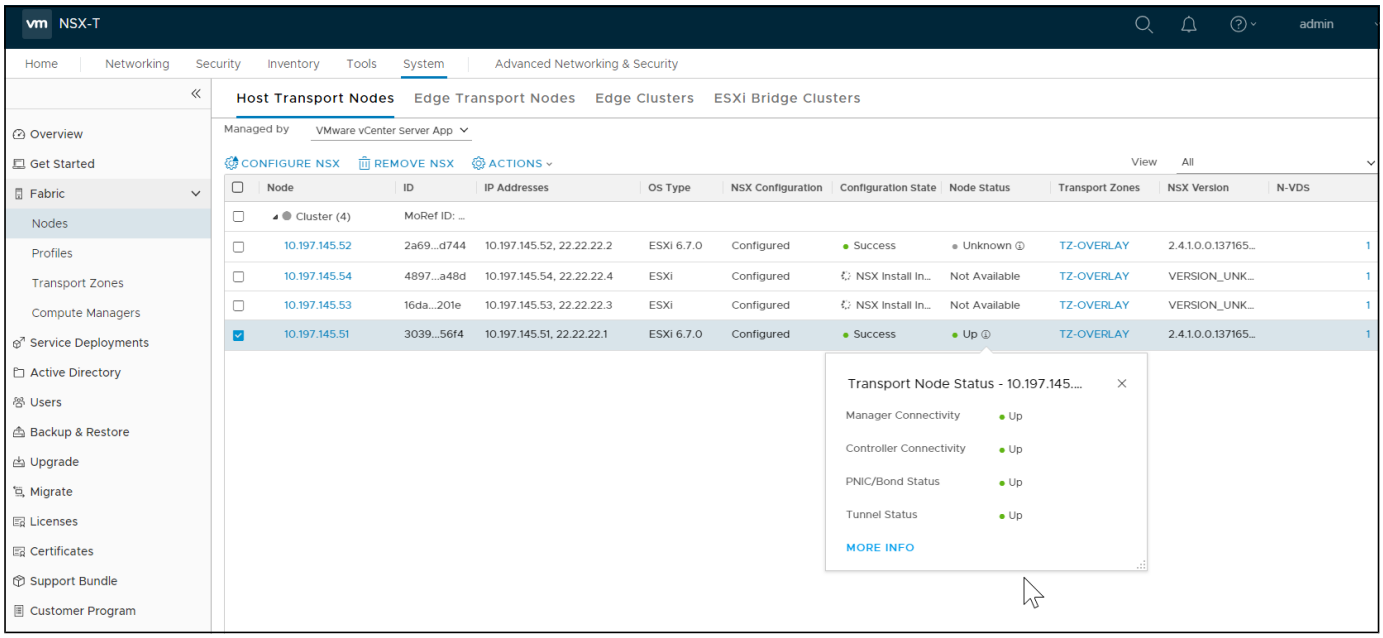


7. Click **Finish** to create the ESXi Transport Node.

8. Repeat the process for each ESXi host in the vSphere Cluster dedicated for Enterprise PKS.

Verify ESXi Host Preparation for Enterprise PKS

1. In NSX Manager, go to **System > Fabric > Nodes > Host Transport Nodes**
2. For the **Managed by** field, select **VMware vCenter Server App** from the dropdown menu.
3. Verify that the **Node Status** is **Up** for the ESXi host.



4. Select the information icon for the EXSi host in the **Node Status** column. Verify that the **Tunnel Status** status is **Up**.
5. Verify that 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

Assuming you have successfully completed all preceding NSX-T installation tasks, NSX-T Data Center should now be installed and configured for Enterprise PKS.

Depending on your requirements, you may want to complete the following additional tasks:

- The NSX-T Data Center administrator password expires after 90 days. To change the password or the expiration interval, see Updating the NSX-T Admin Password.
- If you require scalability for the NSX Management Cluster, Provision a Load Balancer for the NSX-T Management Cluster.

NSX-T Installation Instructions Home

Installing and Configuring NSX-T for Enterprise PKS.

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

Creating the Enterprise PKS Management Plane

In this topic

Page last updated:

Prepare the vSphere and NSX-T infrastructure for the Enterprise PKS Management Plane where the PKS, Ops Manager, BOSH Director, and Harbor Registry VMs are deployed.

Prerequisites

Before you begin this procedure, make sure you have completed the following prerequisites for installing Enterprise PKS on vSphere with NSX-T:

- Preparing to Install Enterprise PKS on vSphere with NSX-T Data Center
- Installing and Configuring NSX-T for Enterprise PKS

About the Enterprise PKS Management Plane

The Enterprise PKS Management Plane is the network for PKS Management Plane components, including the PKS API Server, Ops Manager, BOSH Director, and Harbor Registry. The Enterprise 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, and NSX-T NAT rules on the Tier-0 Router.

For all types of NSX-T deployment topologies, create a Tier-1 (T1) Logical Switch and a Tier-1 Logical Router and Port. Enable route advertisement for the T1 Logical Router and advertise All NSX connected routes for the Management Plane VMs. Lastly, link the Tier-1 Router to the Tier-0 Router.

If you are using the NAT Topology, you will also need to create the following NAT rules on the Tier-0 Router:

- 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**.
- 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 Management Plane 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`.

- Destination NAT (DNAT) rule that maps an external IP address from the **PKS MANAGEMENT CIDR** to the IP where you deploy Harbor on the Management Plane 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`.

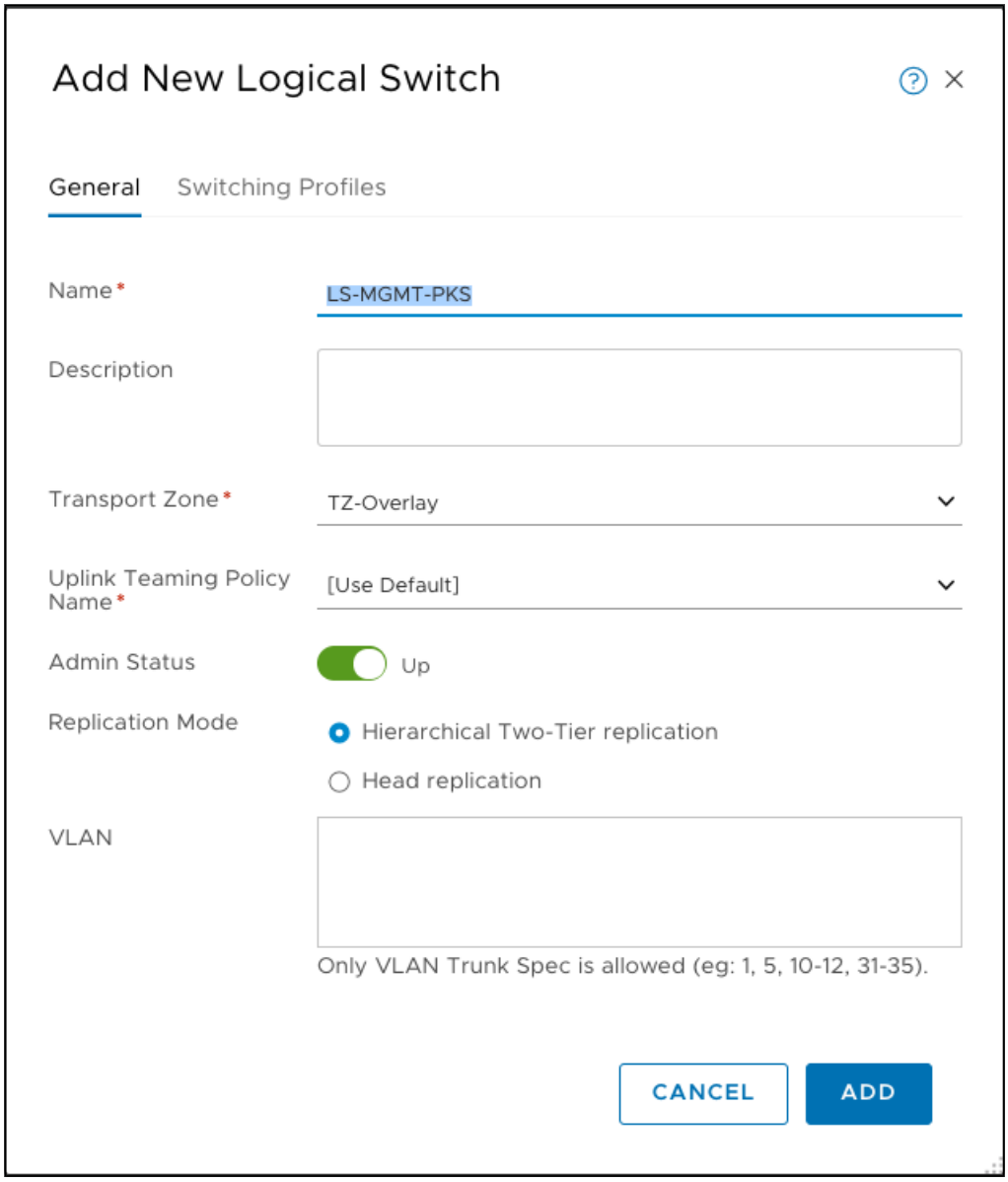
Lastly, if you want to provide users with remote access to the PKS CLI, you will need to define a DNAT rule that maps an external IP address from the **PKS MANAGEMENT CIDR** to the IP where you deploy the PKS API Server on the Management Plane logical switch. This rule lets developers use the PKS CLI remotely from their workstations or laptops. Such a rule is needed for both NAT and No-NAT topologies.

Step 1. Create vSphere Resource Pool for the Management Plane

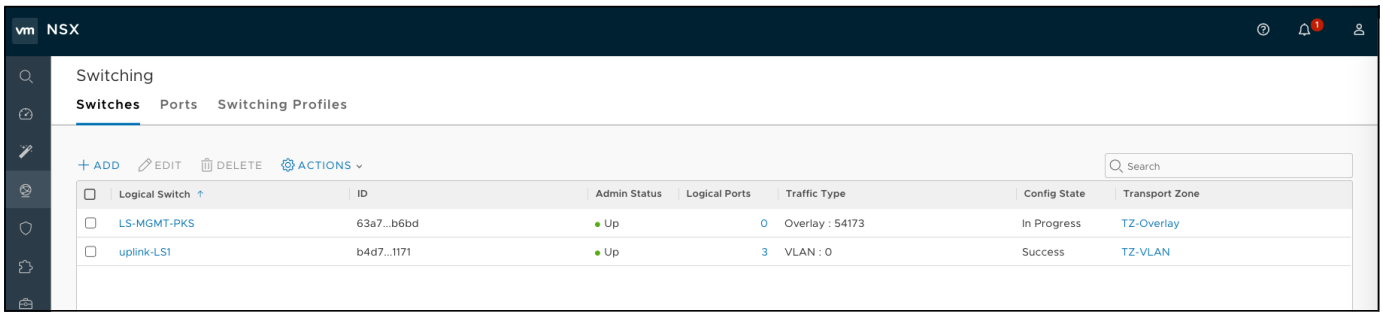
1. Log in to vCenter for your vSphere environment.
2. Select **Compute Cluster > New Resource Pool**.
3. Name the resource pool, such as `RP-MGMT-PKS`.
4. Click **OK**.
5. Verify resource pool creation.

Step 2. Create NSX-T Logical Switch for the Management Plane

1. In NSX Manager, select **Switching > Add**.



2. Create a new logical switch.
3. Click **Add**.
4. Verify logical switch creation.

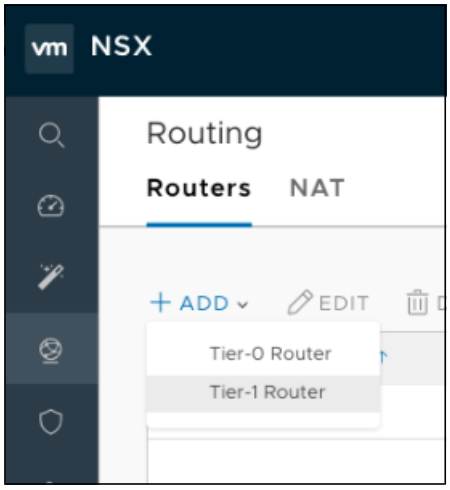


Step 3. Create NSX-T Tier-1 Router for the Management Plane

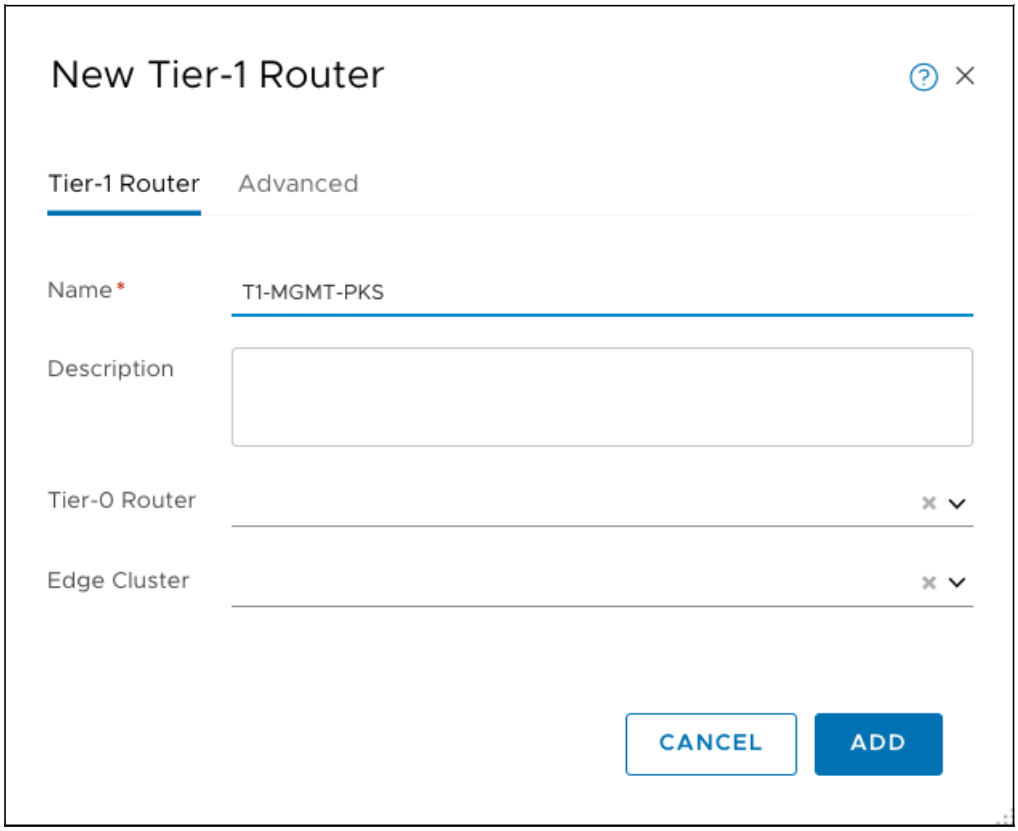
Defining a Tier-1 Router involves creating the router and attaching it to the logical switch, creating a router port, and advertising the routes.

Create Tier-1 Router

1. In NSX Manager, select **Routing > Add > Tier-1 Router**.

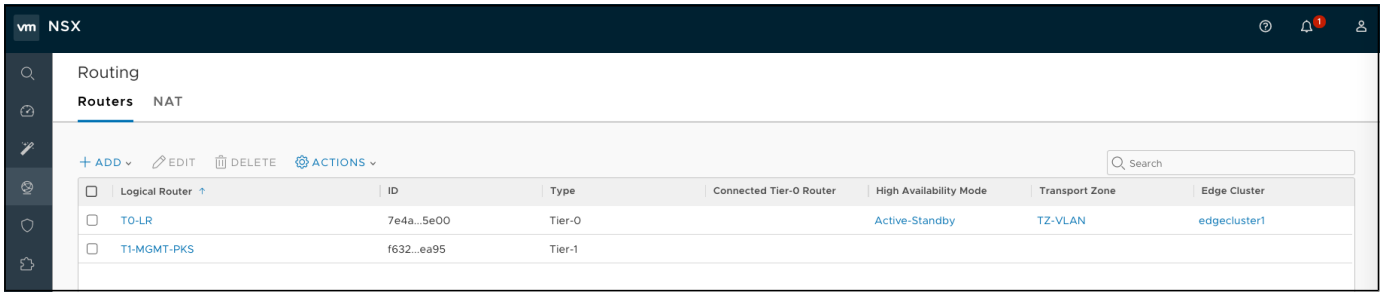


2. Configure the Tier-1 Router.



3. Click **Add**.

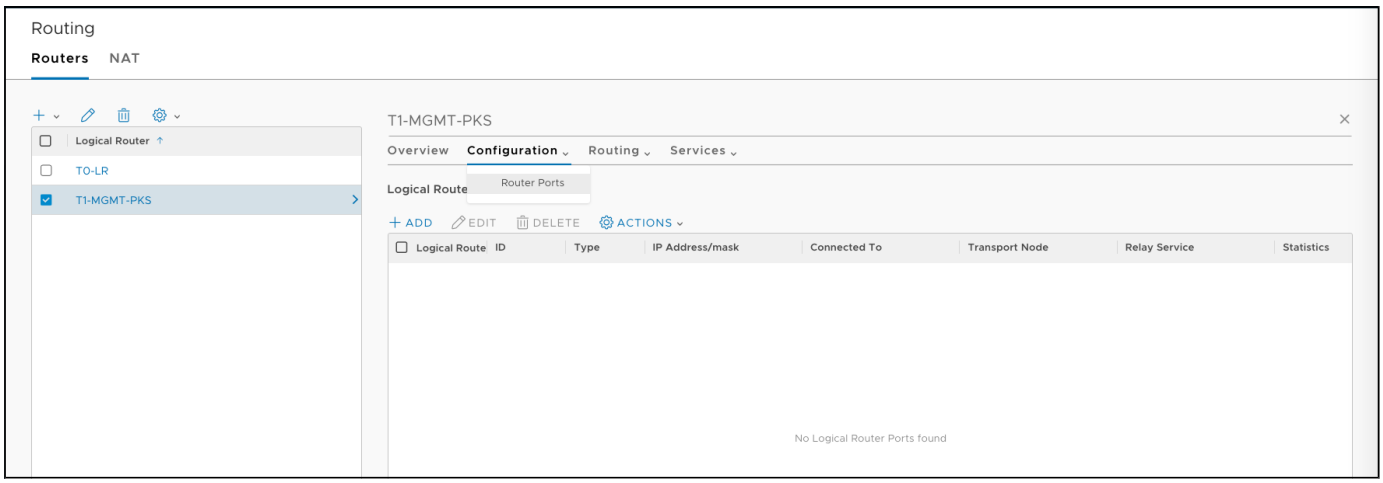
4. Verify Tier-1 Router creation.



Create Tier-1 Router Port

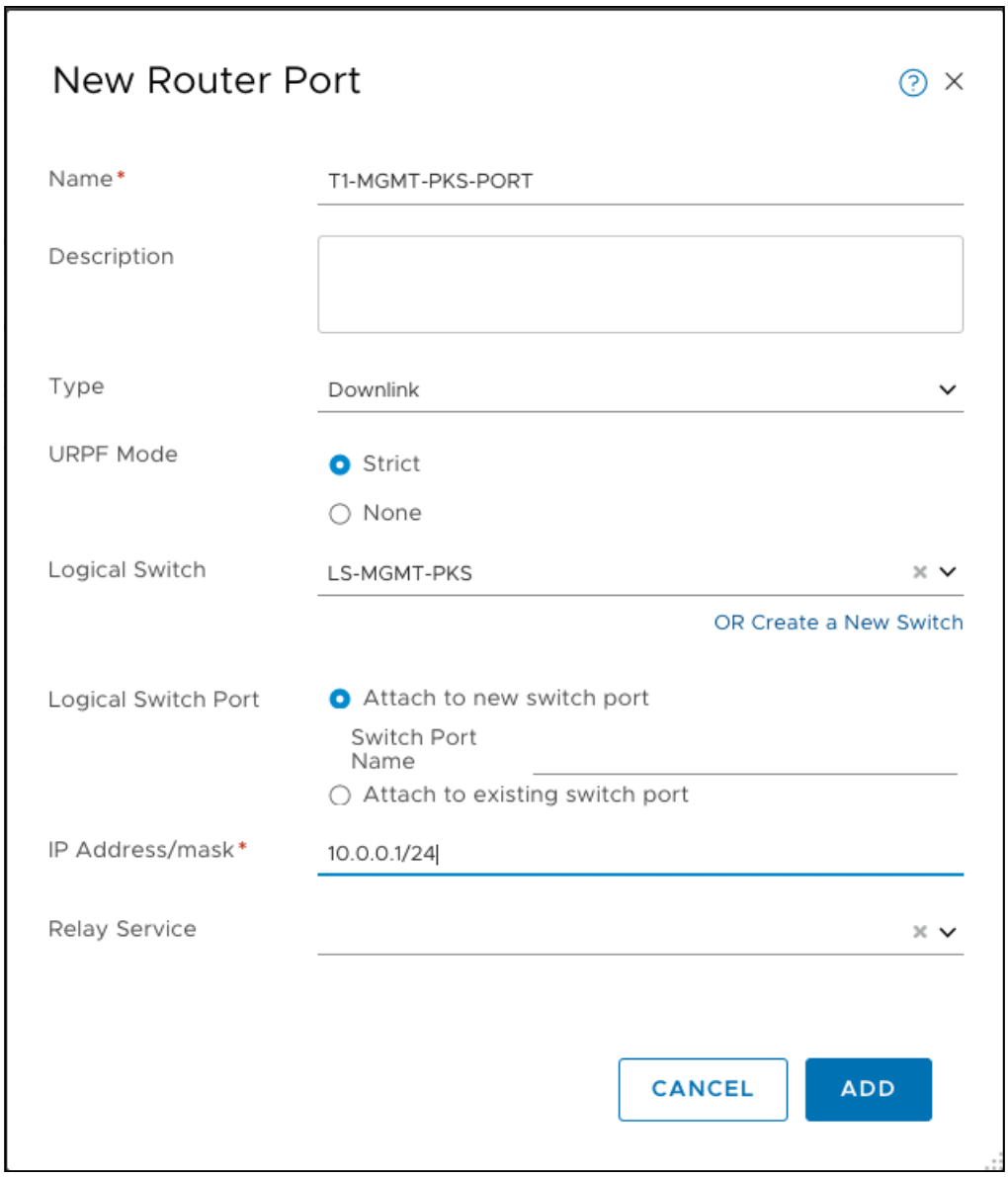
1. Select the Tier-1 Router you created.

2. Select **Configuration > Router Ports**.



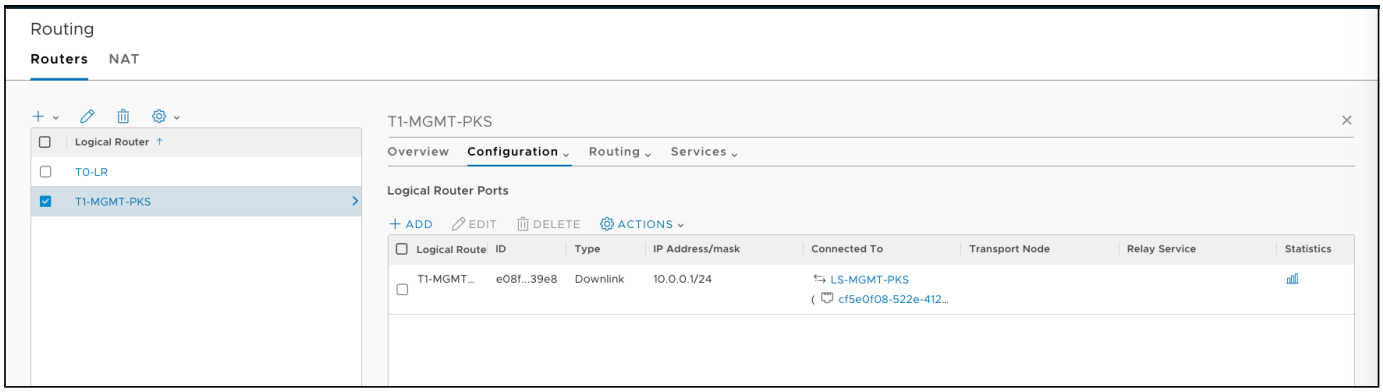
3. Click **Add** and configure the Tier-1 Router Port.

   - **Name**: `T1-MGMT-PKS-PORT`
   - **Logical Switch**: Select `LS-MGMT-PKS` from the menu list
   - **IP Address/mask**: `10.0.0.1/24` (for example)



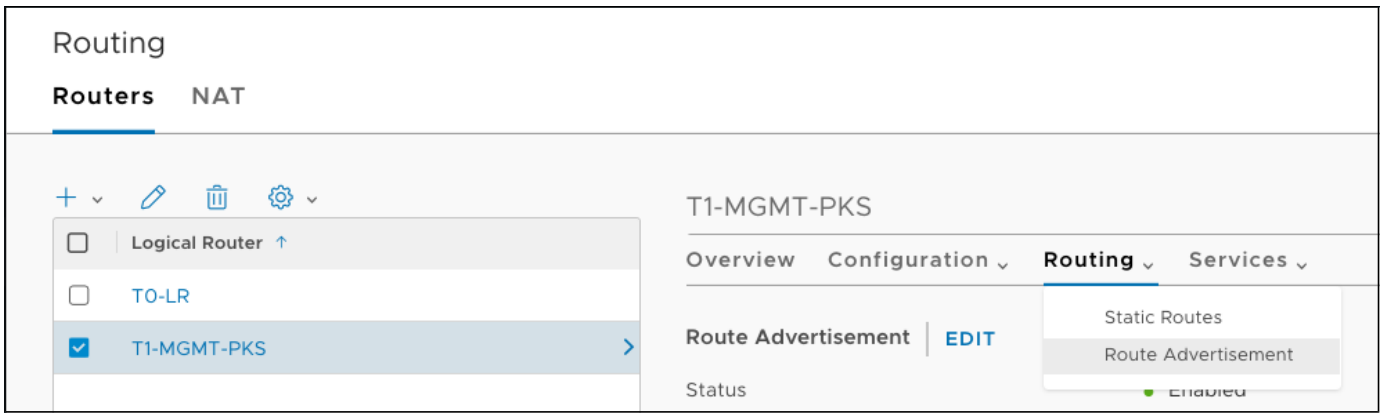
4. Click **Add**.

5. Verify Tier-1 Router Port creation.

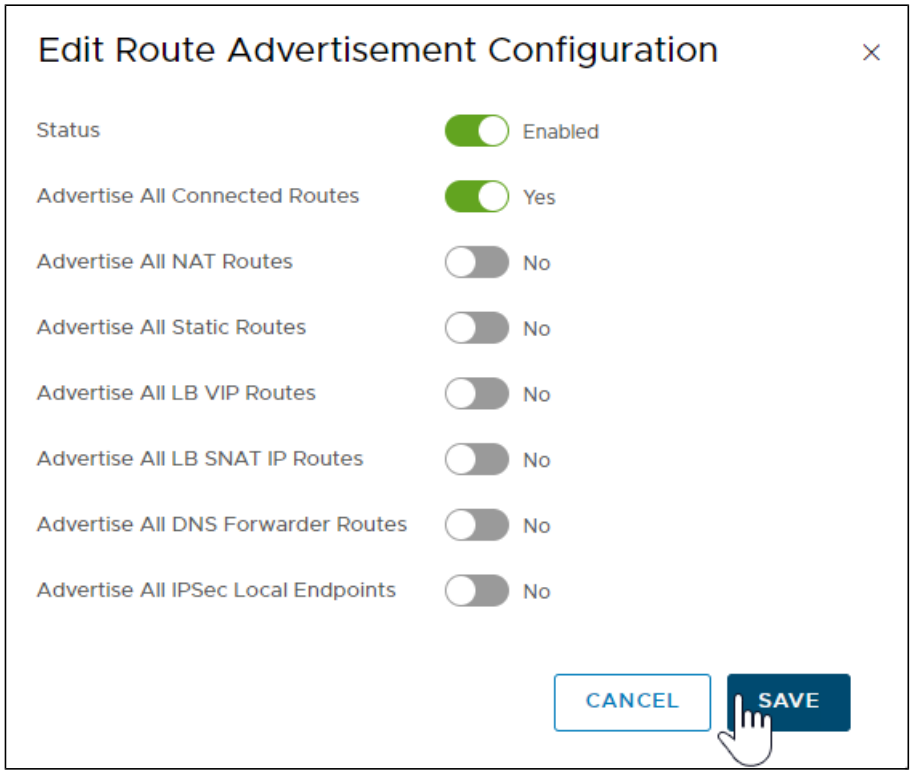


Advertise the Tier-1 Routes

1. Select the Tier-1 Router > Routing > Route Advertisement.

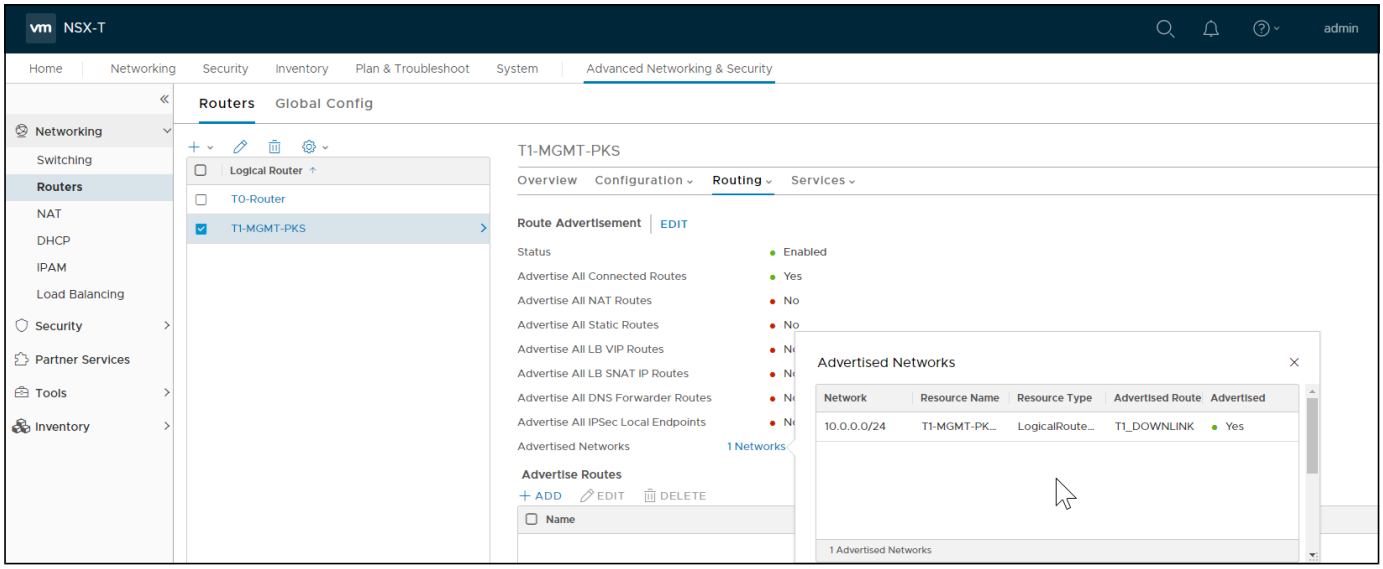


2. Advertise the Tier-1 routes as follows:
   - **Status**: enabled
   - **Advertise All Connected Routes**: yes



3. Click **Save**.

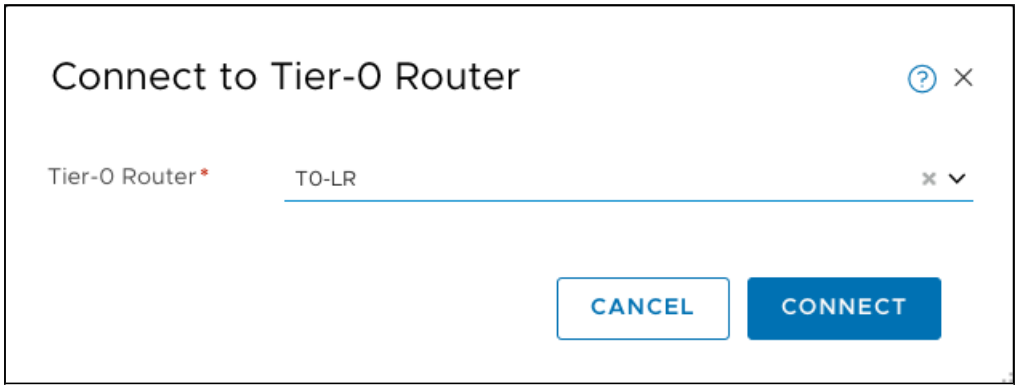
4. Verify route advertisement.



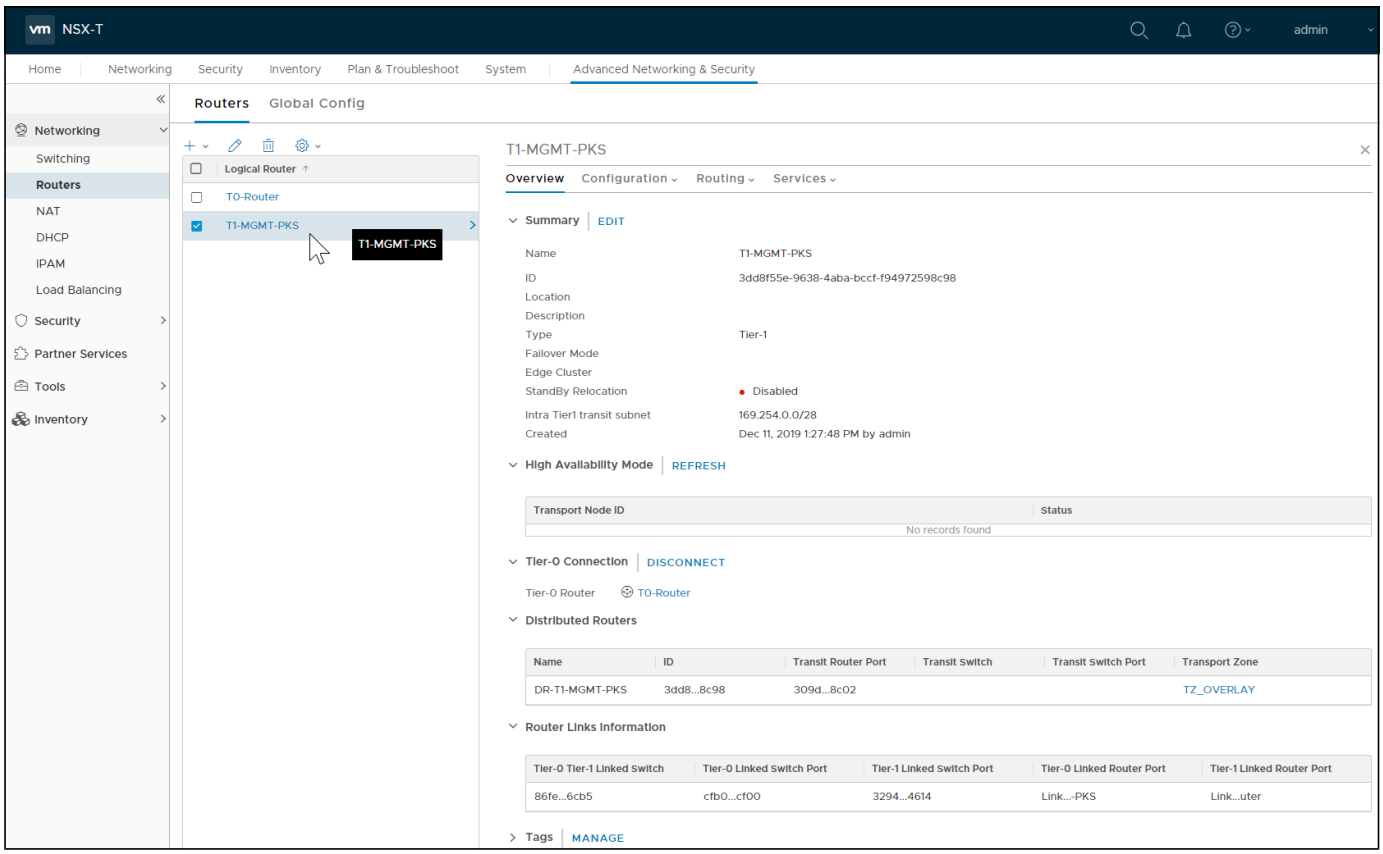
Step 4. Connect the Tier-1 Router to the Tier-0 Router

Connect the Tier-1 router to the Tier-0 router and verify router-to-router connectivity.

1. At the **Tier-1 Router > Overview** screen, select the option **Tier-0 Connection > Connect**.
2. At the **Connect to Tier-0 Router**, select the Tier-0 Router and click **Connect**.

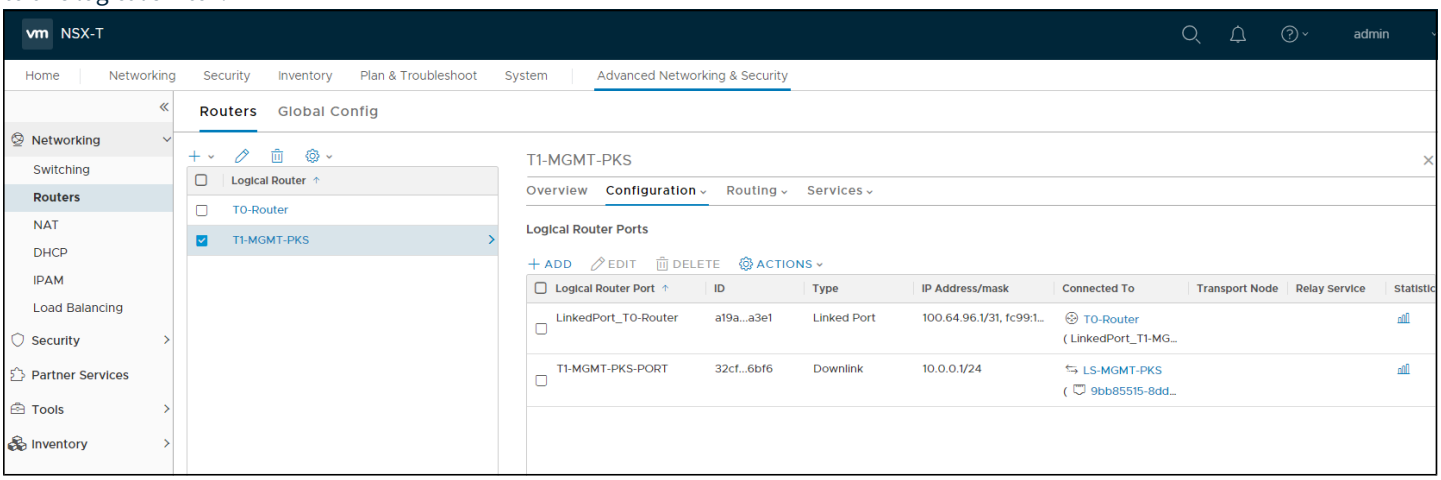


3. Verify connectivity between the Tier-1 and Tier-0 Routers.



4. Select the **Tier-1 Router > Configuration > Router ports**.

The Tier-1 Router created for the Management Plane should have 2 port connections: one connected to the Tier-0 router, and a second port connected to the logical switch defined for the Management Plane. This second port is the default gateway for all VMs connected to this logical switch.



Step 5. Create SNAT Rule on the Tier-0 Router for vCenter and NSX Manager

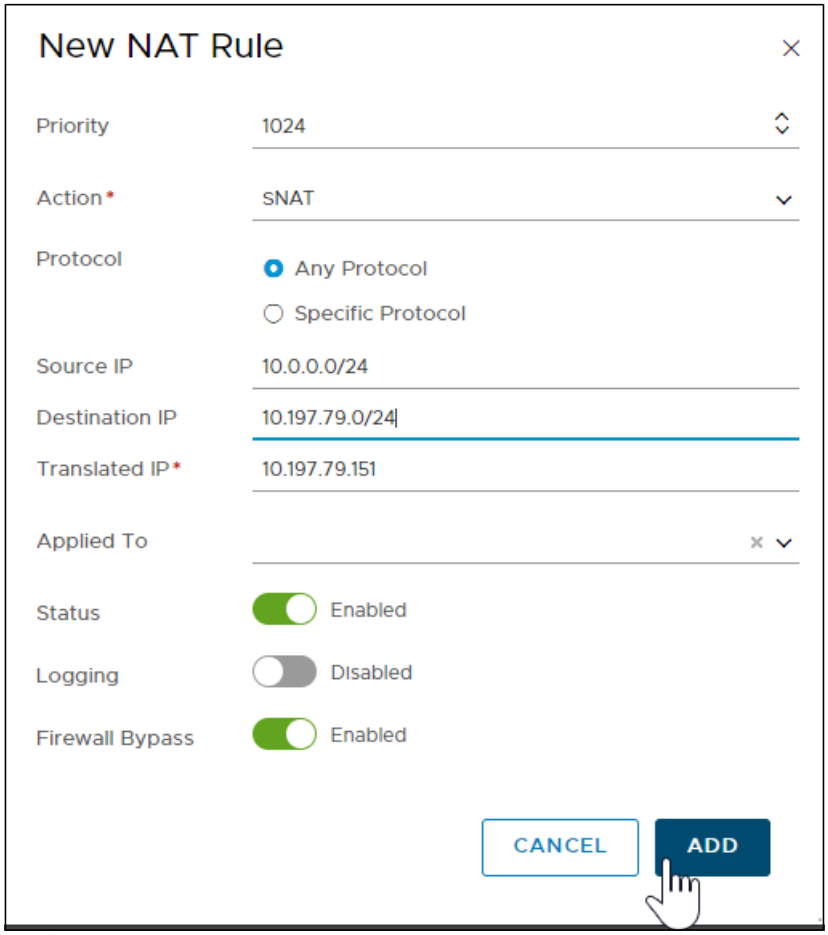
Create a Source NAT (SNAT) rule on the Tier-0 Router for Enterprise PKS management components to access vCenter and NSX manager. The SNAT rule on the Tier-0 Router allows the Management Plane VMs to communicate with the vCenter and NSX-T Management environments. For example, create a SNAT rule that maps `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 **Tier-0 Router > Services > NAT**.

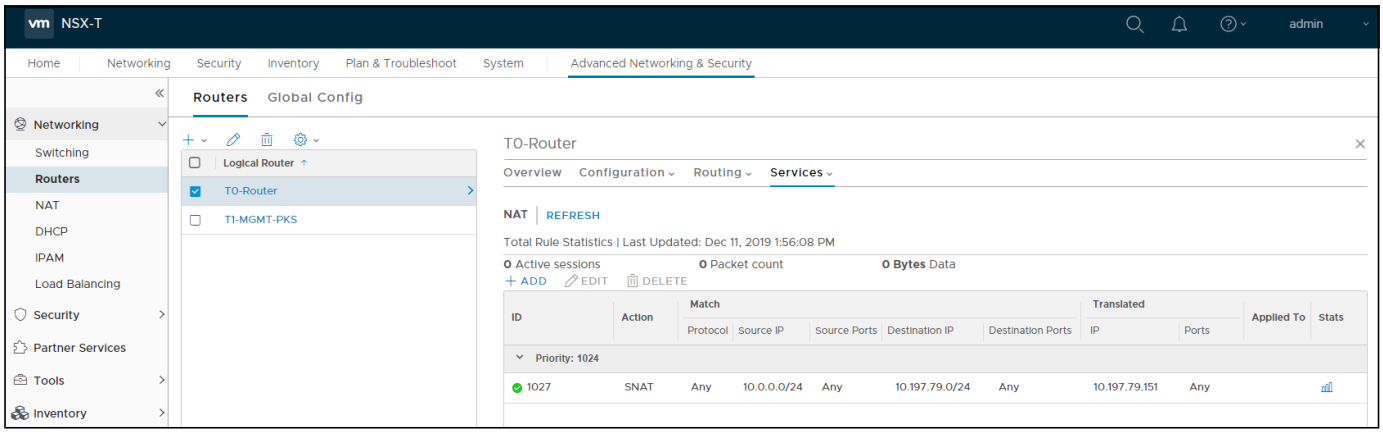
2. Click ADD and configure the SNAT rule:

   - **Priority**: `1024` (for example)
   - **Action**: `SNAT`
   - **Source**: `10.0.0.0/24` (for example)
   - **Destination IP**: `10.197.79.0/24` (for example)
   - **Translated IP**: `10.197.79.151` (for example)



3. Click **Add**.

4. Verify SNAT rule creation.



Step 6. Create DNAT Rule on the Tier-0 Router for Ops Manager

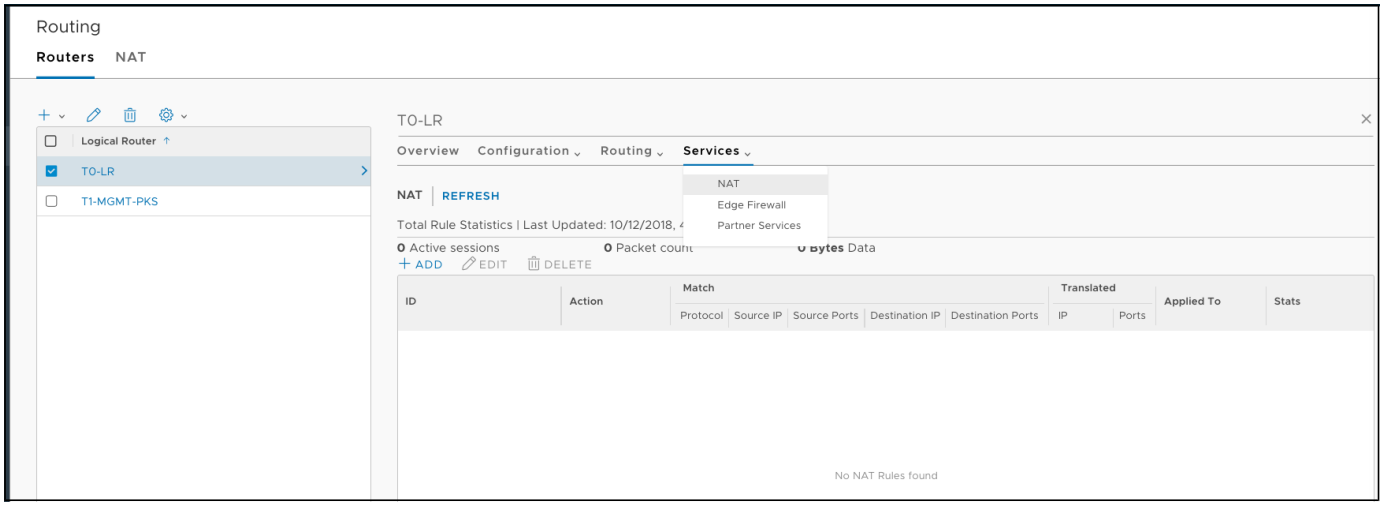
Create a DNAT rule on the T0 Router to access the Ops Manager Web UI, which is required to deploy Enterprise PKS.

The Destination NAT (DNAT) rule on the Tier-0 Router maps an external IP address from the **PKS MANAGEMENT CIDR** to the IP where you deploy Ops Manager on the Management Plane 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`.

To create a DNAT rule for Ops Manager:

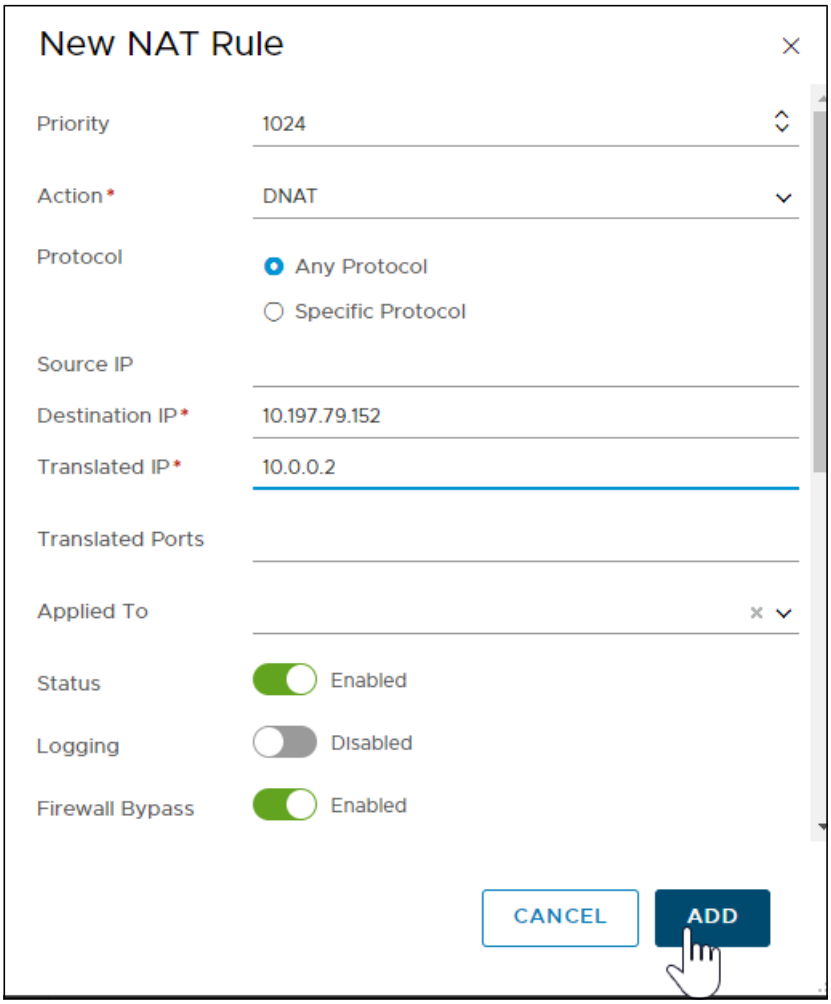
1. In the NSX-T Manager interface, select **Routing > Routers**.

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:

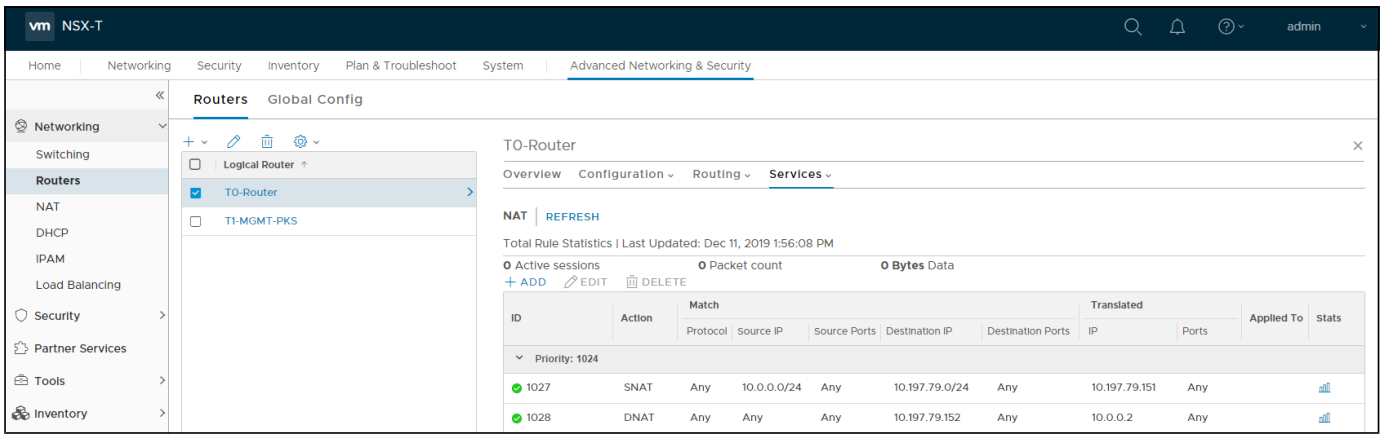
   - **Priority**: `1024` (for example)
   - **Action**: DNAT (for example)
   - **Destination IP**: `10.197.79.152` (for example)



- **Translated IP**: `10.0.0.2` (for example)

4. Click **Add**.

5. Verify the DNAT rule you defined.



Step 7. Create DNAT on the Tier-0 Router for Harbor Registry

If you are using VMware Harbor Registry with Enterprise PKS, create a DNAT rule on the Tier-0 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 Enterprise PKS Management network.

See Create DNAT Rule in the VMware Harbor Registry documentation for instructions.

Step 8. 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 Enterprise PKS is installed and the PKS API VM has an IP address.

1. When the Enterprise PKS installation is completed, retrieve the Enterprise PKS endpoint by performing the following steps:
   a. From the Ops Manager Installation Dashboard, click the **Enterprise PKS** 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 Enterprise PKS endpoint. For example, a DNAT rule that maps `10.172.1.4` to `172.31.0.4`, where `172.31.0.4` is the Enterprise 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 Enterprise PKS Management Plane from VMs in the vCenter network.

Next Step

After you complete this procedure, follow the instructions in Creating the Enterprise PKS Compute Plane.

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

Creating the PKS Compute Plane

In this topic

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 Enterprise PKS on vSphere with NSX-T, including:

- Preparing to Install Enterprise PKS on vSphere with NSX-T Data Center
- Installing and Configuring NSX-T v2.5 for Enterprise PKS Installing and Configuring NSX-T v2.4 for Enterprise PKS
- Creating the Enterprise PKS Management Plane

About the PKS Compute Plane

Installing VMware Enterprise PKS on vSphere with NSX-T requires the creation of vSphere resource pools that map to BOSH availability zones where Kubernetes VMs will run, as well as NSX IP blocks for Kubernetes node and pod networks, and 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 Enterprise 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).

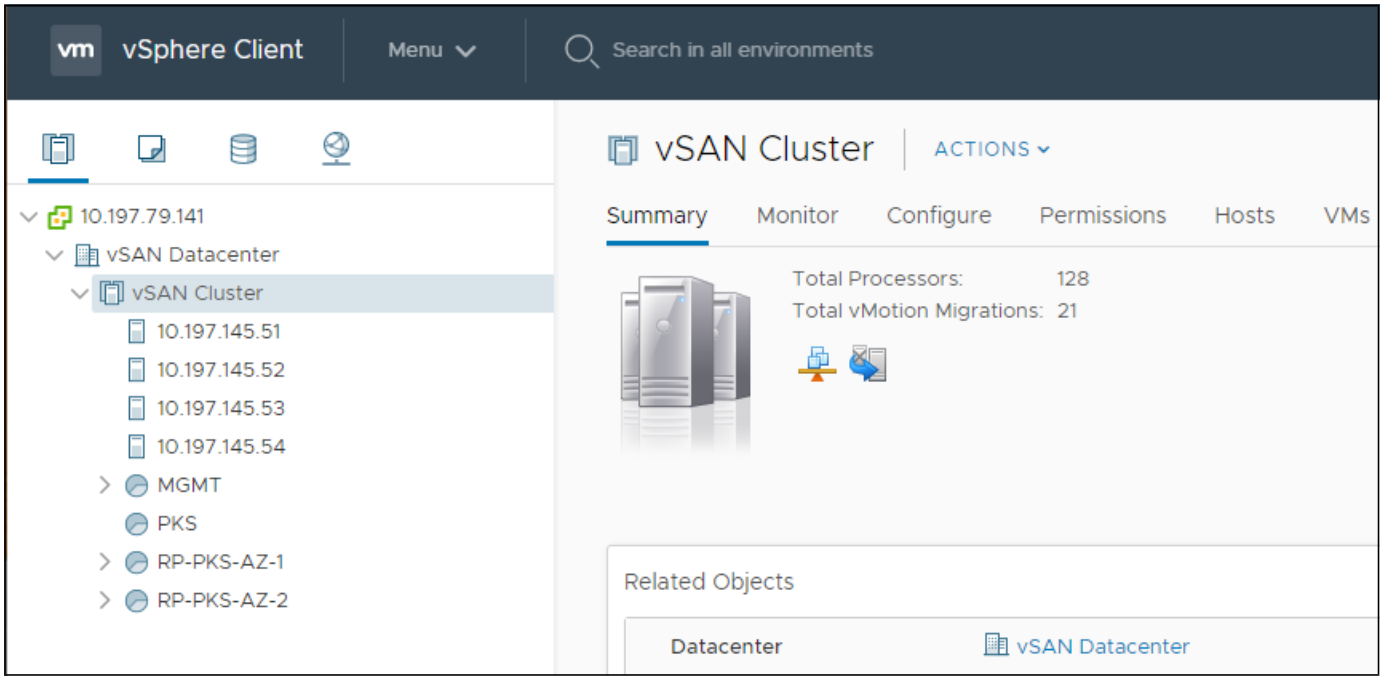
In addition, create a Floating IP Pool from which to assign routable IP addresses to components. This network provides your load balancing address space for each Kubernetes cluster created by Enterprise PKS. The network also provides IP addresses for Kubernetes API access and Kubernetes exposed services.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

Create vSphere Resource Pools for Kubernetes Cluster Nodes

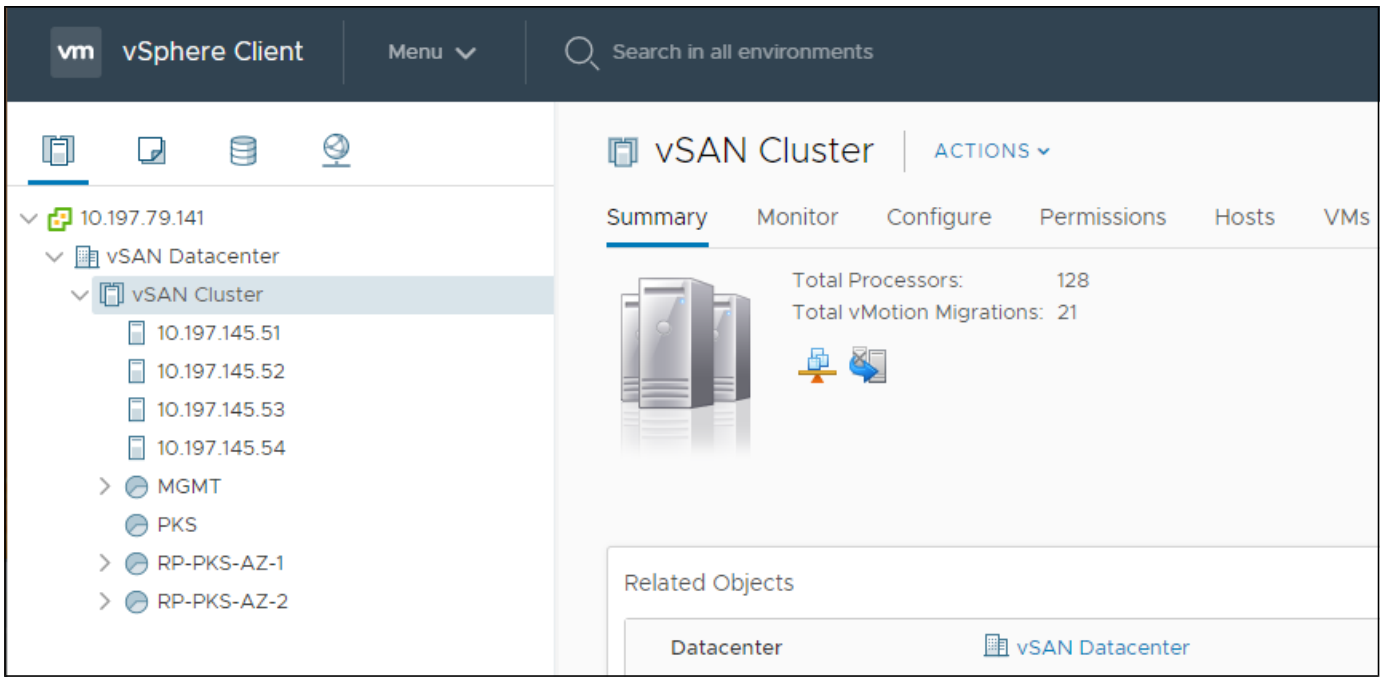
Create vSphere Resource Pool for AZ-1

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:



Step 2: Create vSphere Resource Pool for AZ-N

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-2`.
4. Click **OK** and verify resource pool creation:

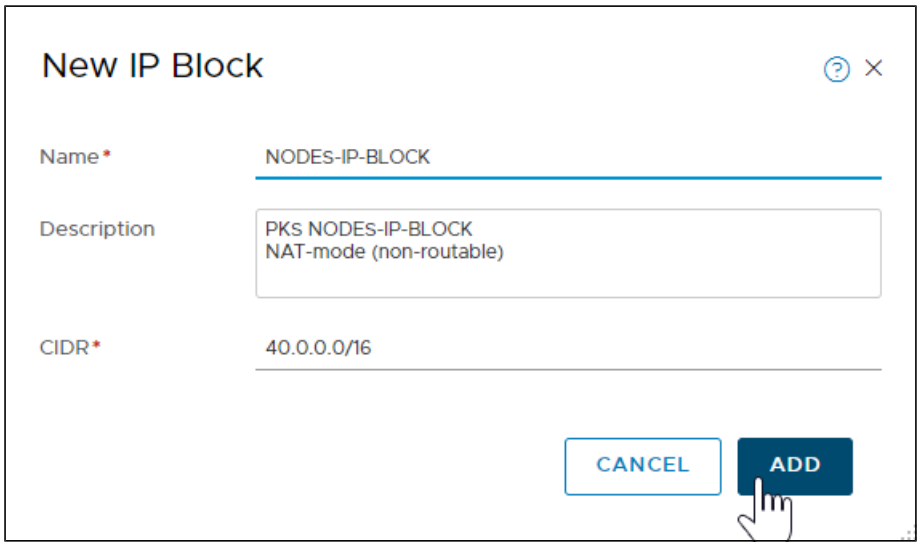


Create NSX-T Objects for Kubernetes Nodes and Pods

Complete the following instructions to create the required NSX-T network objects.

Create the Nodes IP Block

1. In NSX Manager, go to **Advanced Networking & Security** > **Networking > IPAM**.
2. Add a new IP Block for Kubernetes Nodes. If you are using NAT mode, the CIDR is non-routable. For example:
   - **Name**: NODES-IP-BLOCK

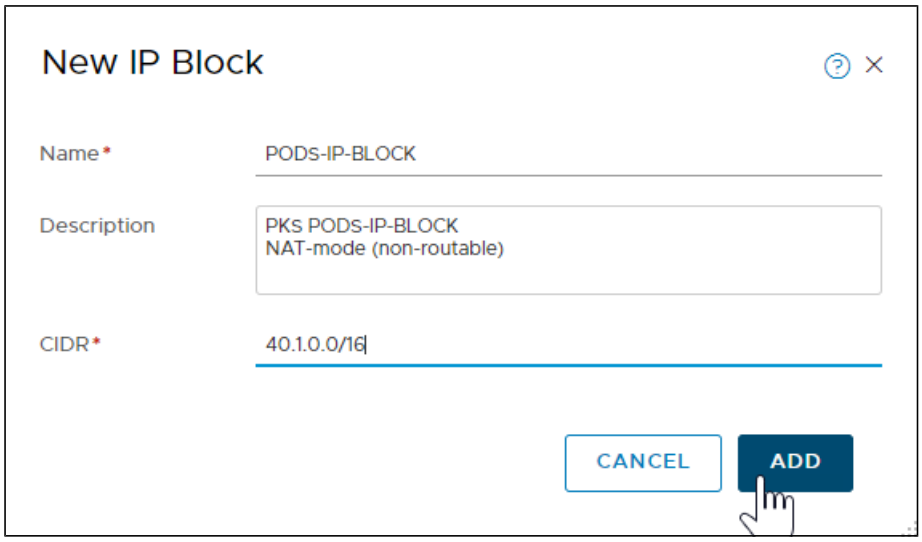


   - **CIDR**: `40.0.0.0/16` (for example)
3. Click **Add** to add the Nodes IP Block.

Create the Pods IP Block

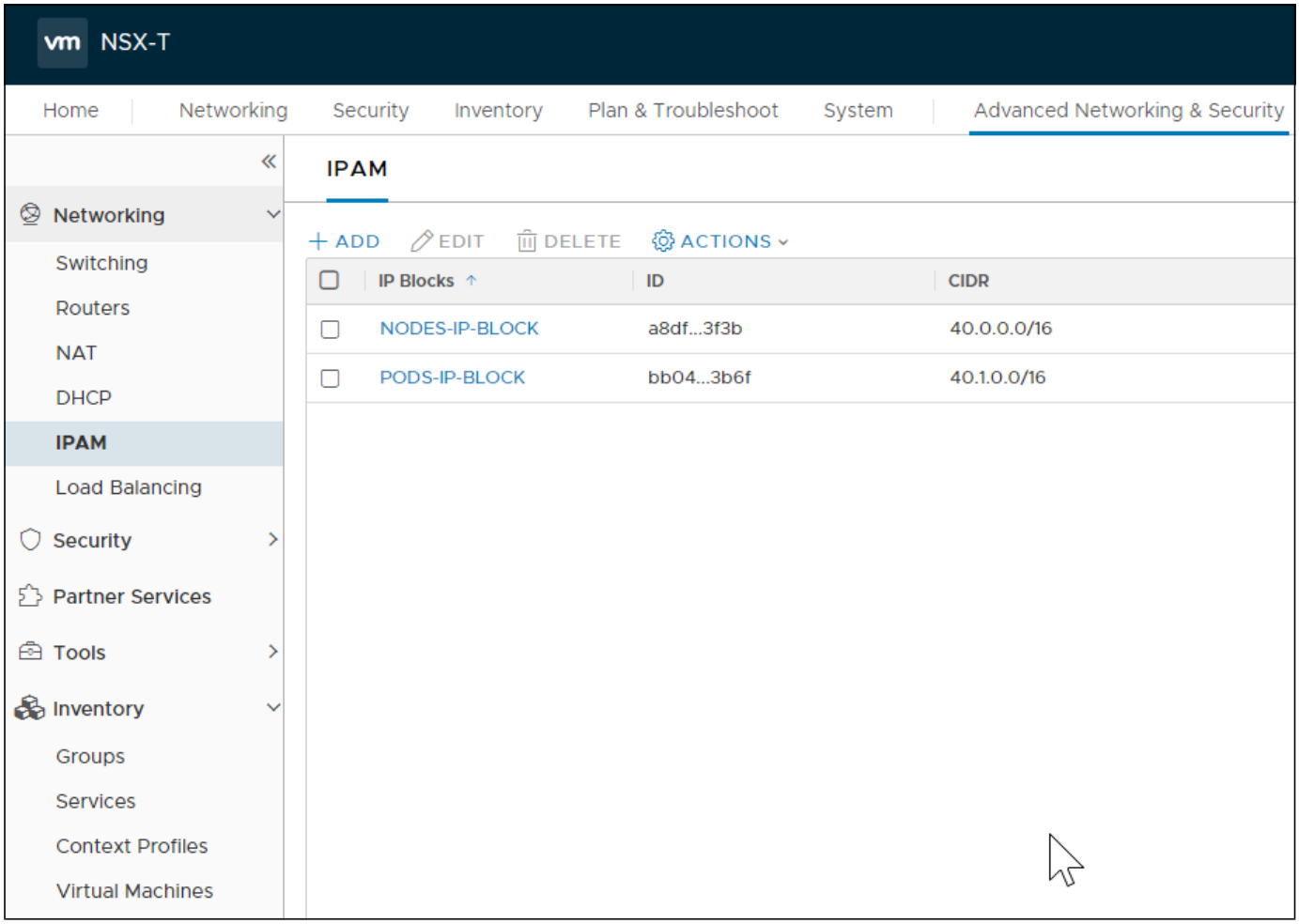
1. In NSX Manager, go to **Advanced Networking & Security** > **Networking > IPAM**.

2. Add a new IP Block for Pods. The CIDR is non-routable. For example:
   - **Name**: PODS-IP-BLOCK
   - **CIDR**: `40.1.0.0/16` (for example)

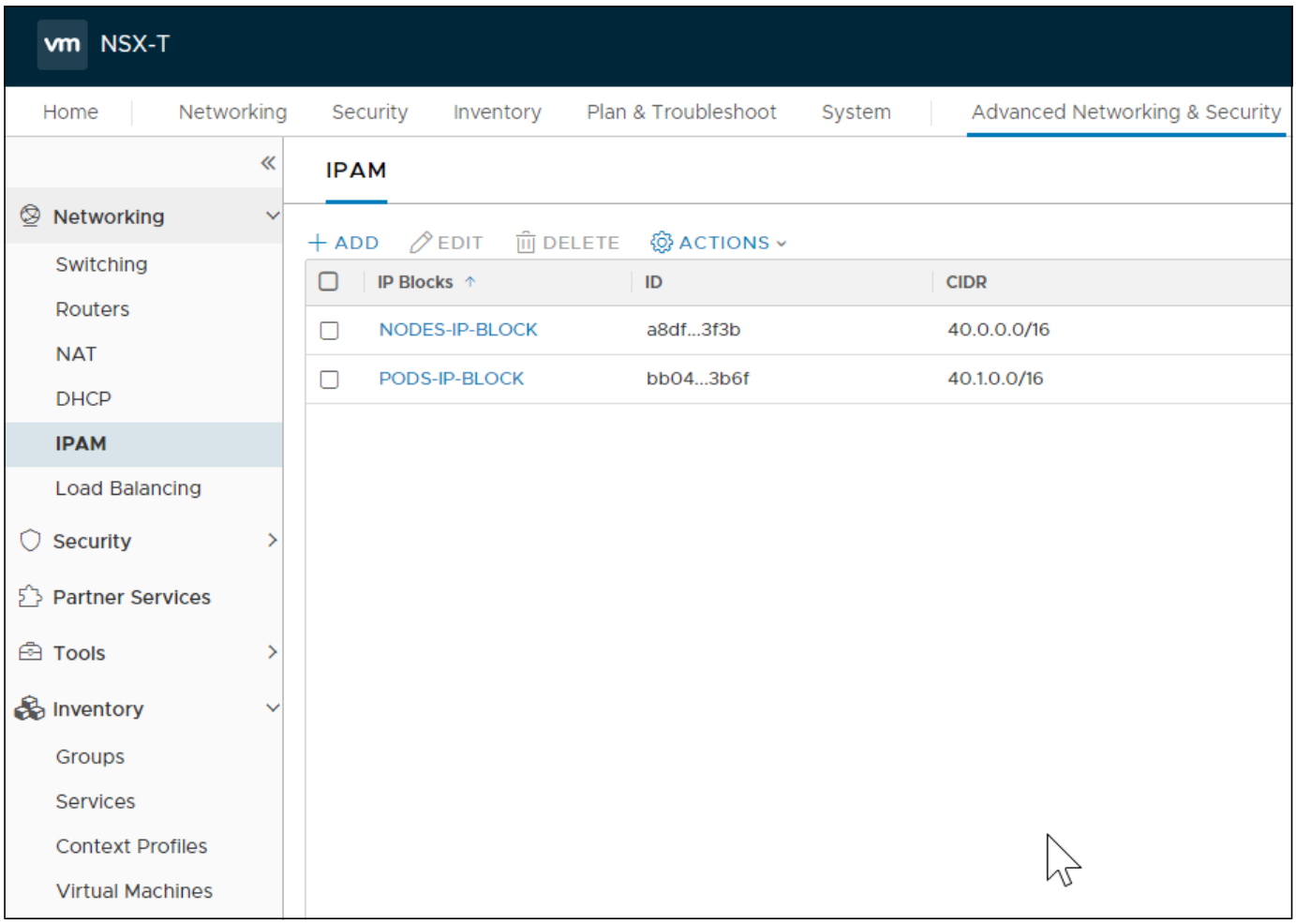


Verify IP Blocks

1. Verify creation of the IP Blocks.

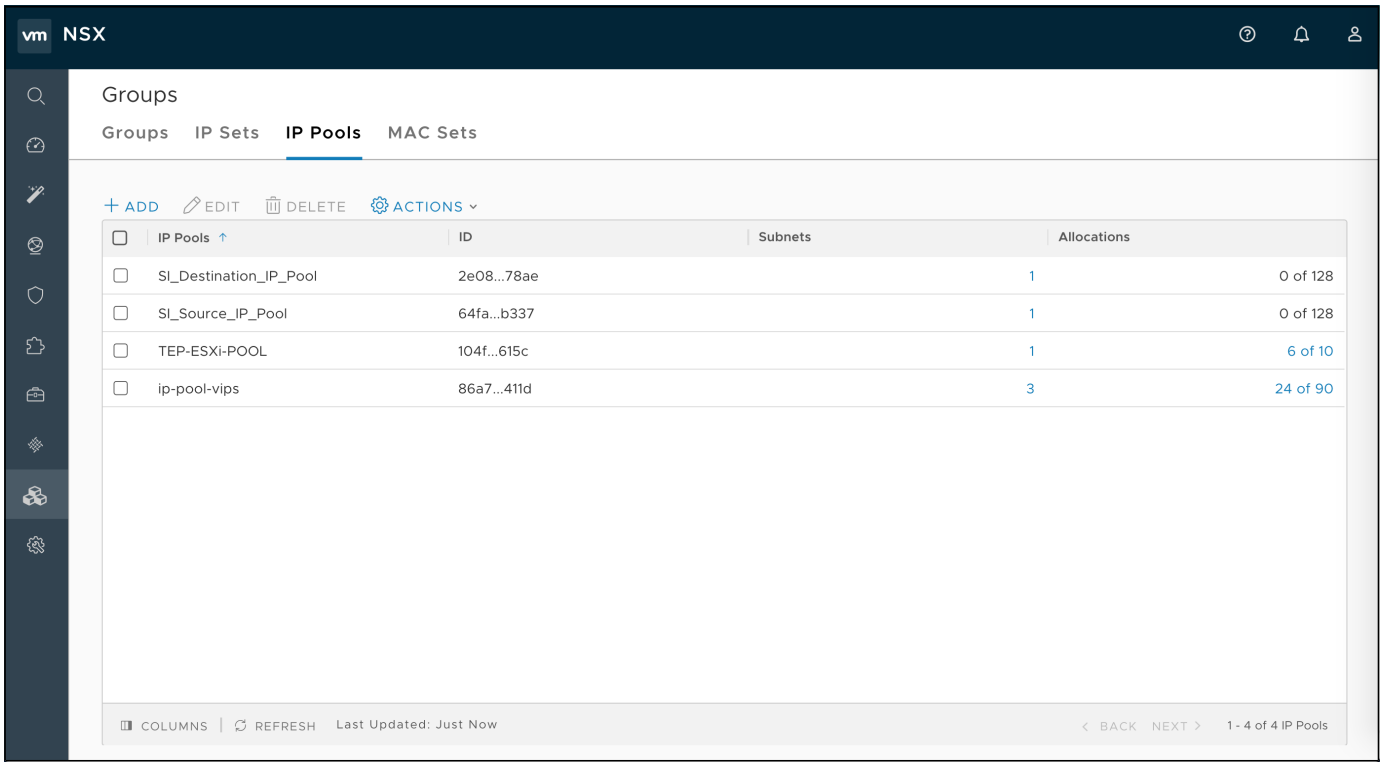


2. Record the UUID of both IP Block objects. You use the UUIDs when you install Enterprise PKS.

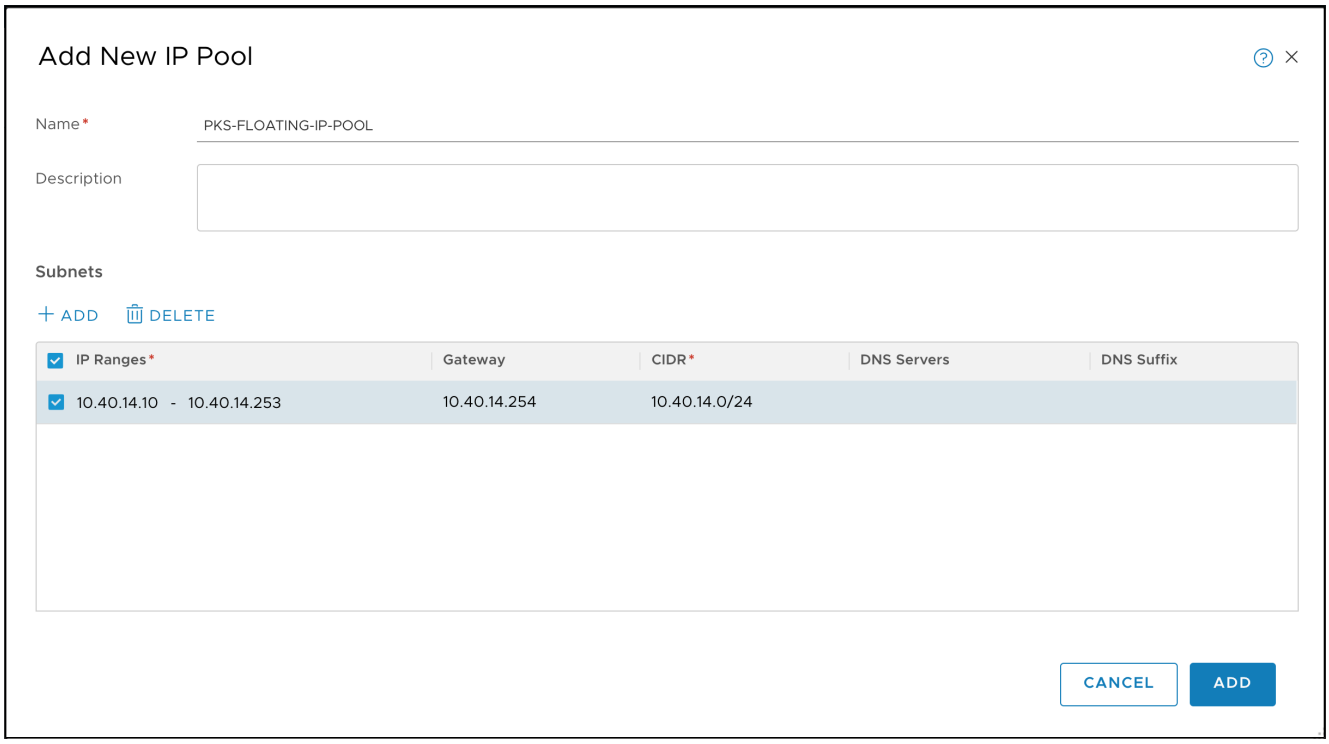


Create Floating IP Pool

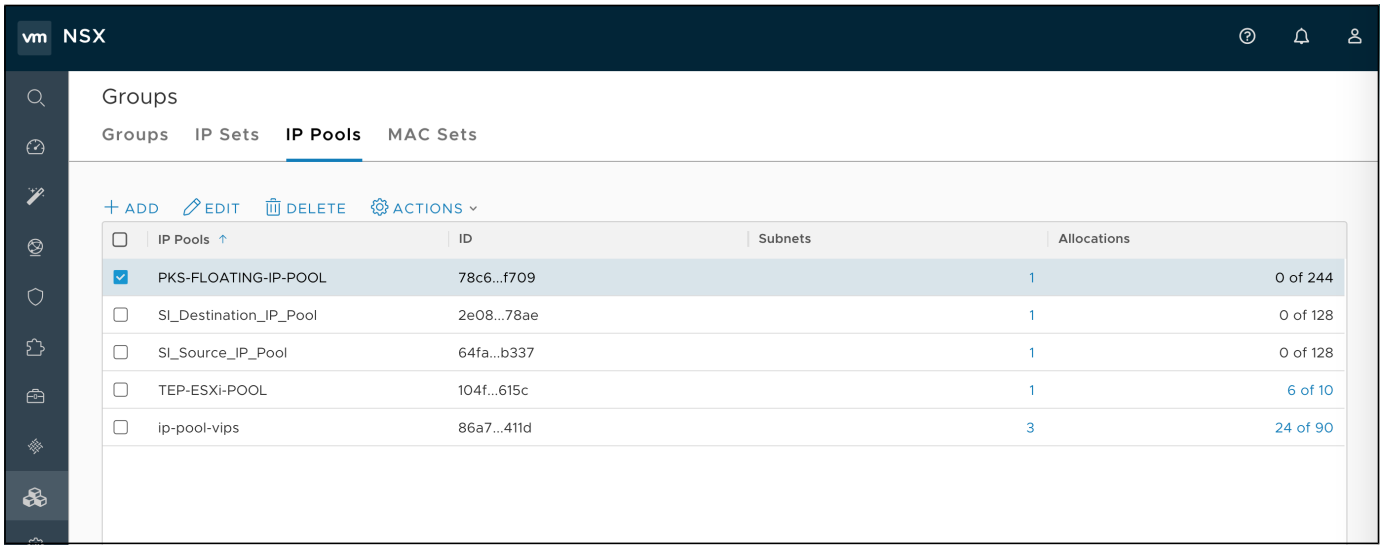
1. In NSX Manager, go to **Advanced Networking & Security** > **Inventory > Groups > IP Pool**



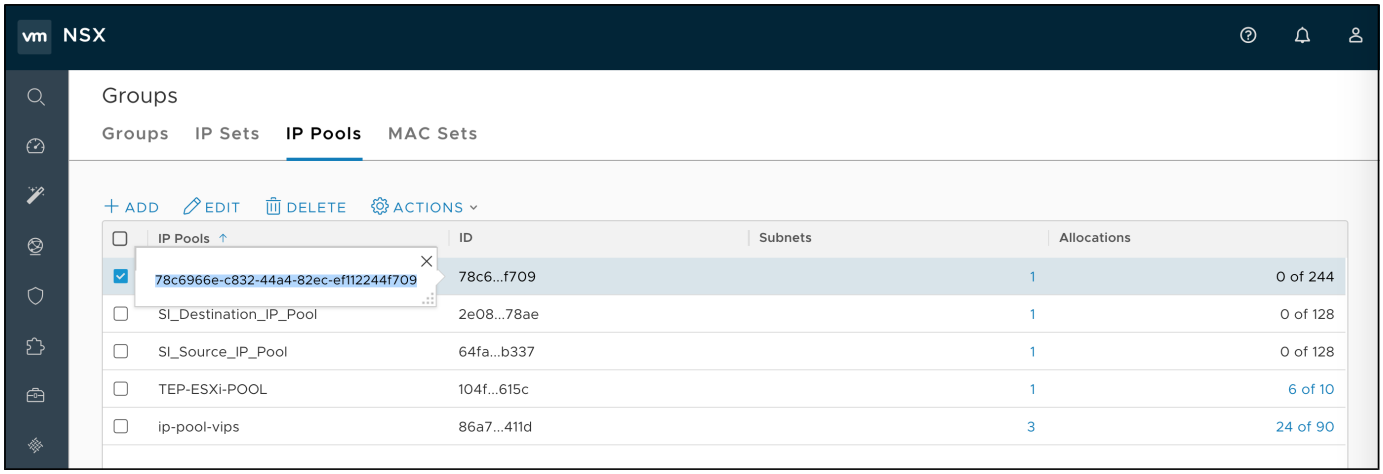
2. Add a new Floating IP Pool. For example:
   - **Name**: PKS-FLOATING-IP-POOL
   - **IP Ranges**: `10.40.14.10 - 10.40.14.253` (for example)
   - **Gateway**: `10.40.14.254` (for example)
   - **CIDR**: `10.40.14.0/24` (for example)



3. Verify creation of the Floating IP Pool.



4. Get the UUID of the Floating IP Pool object. You use this UUID when you install Enterprise PKS.



Next Step

After you complete this procedure, follow the instructions in Deploying Ops Manager with NSX-T for Enterprise PKS

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

Deploying Ops Manager with NSX-T for Enterprise PKS

In this topic

Page last updated:

This topic provides instructions for deploying Ops Manager on VMware vSphere with NSX-T integration for use with VMware Enterprise PKS.

Prerequisites

Before deploying Ops Manager with NSX-T for Enterprise PKS, you must have completed the following tasks:

- Preparing to Install Enterprise PKS on vSphere with NSX-T Data Center
- Installing and Configuring NSX-T v2.5 for Enterprise PKS Installing and Configuring NSX-T v2.4 for Enterprise PKS
- Creating the Enterprise PKS Management Plane
- Create Enterprise PKS Compute Plane

In addition, review the supported Ops Manager versions for Enterprise PKS. See Enterprise PKS Release Notes or the Product Compatibility Matrix.

Review to the known issues for your version of Ops Manager. See one of the following:

- PCF Ops Manager v2.6 Release Notes
- Pivotal Operations Manager v2.7 Release Notes

Step 1: Generate SSH Key Pair for Ops Manager v2.6 and Later

Beginning with Ops Manager v2.6 for vSphere, password authentication is replaced with SSL/TLS authentication. If you are installing Ops Manager v2.6 or later on vSphere, you cannot deploy Ops Manager successfully without adding a public SSH key in the appropriate field of the **Customize Template** screen. If you do not add a public SSH key, Ops Manager shuts down automatically because it cannot find a key and may enter a reboot loop. For more information, see Passwords Not Supported for Ops Manager VM on vSphere in the Ops Manager v2.6 release notes.

For instructions on generating the required SSH key pair for installing Ops Manager v2.6 or later for vSphere, refer to the following KB article:Generate an SSH key pair for installing Ops Manager v2.6 on vSphere.

When you add the key value to the **Public SSH Key** field, you must enter the entire public key similar to the format required for `authorized_keys`. For example, the format required is similar to the following:

```
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEAnZBapWsER/EO1hLYvV/rkZe78mUBueZGHx1kw+ByfNbLoA385Cm72L+6qq40yOIH6R42nHN/bynbeHOD4Ptes4/s2lrLJtTzEWgH9XYnId4sE5f+QTFd2kRtTzZcu8WvFudEIyCIWjO+o9yvPETs05dEl/3KDn+t9uXx
```

Step 2: Deploy Ops Manager for Enterprise PKS

1. Before starting, refer to the Enterprise PKS Release Notes for supported Ops Manager versions for Enterprise 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 v2.6 Release Notes or the Pivotal Operations Manager Release v2.7 Release Notes.

3. Download the Pivotal Ops Manager for vSphere installation file from the Pivotal Network.

   a. Open a browser to the Pivotal Operations Manager download page on the Pivotal Network.
   b. Use the dropdown menu to select the supported Pivotal Operations Manager release.
   c. Select the **Pivotal Ops Manager for vSphere** download option. This downloads the **Pivotal Ops Manager for vSphere** VM template as an OVA file.

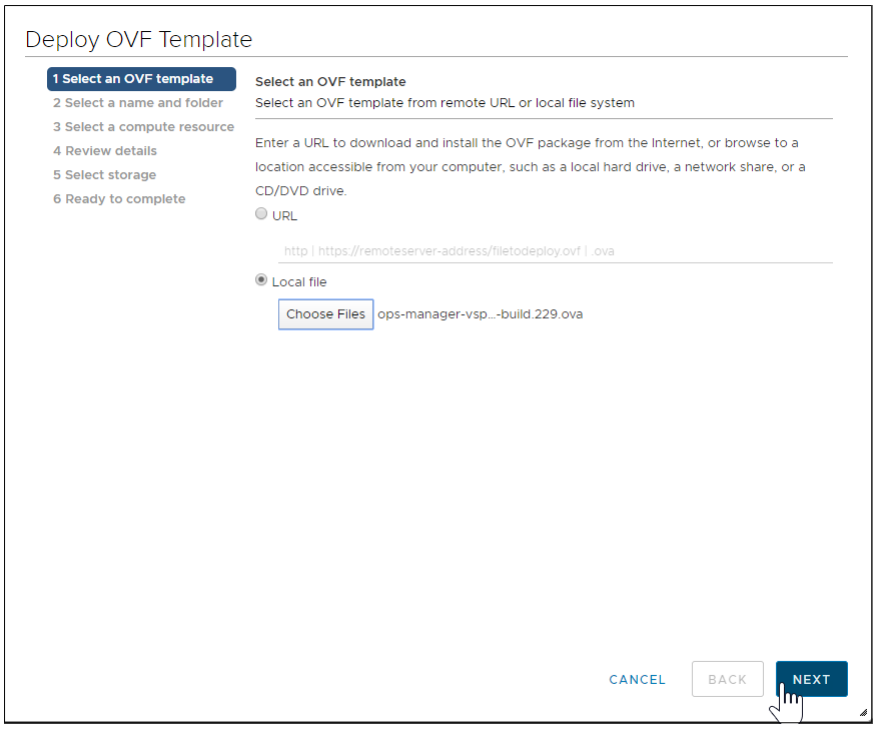
4. Log into vCenter using the vSphere Client (HTML5) to deploy the Ops Manager OVA.

5. Select the Resource Pool defined for the Enterprise PKS Management Plane. See Create Enterprise PKS Management Plane if you have not defined the Enterprise PKS Management Resource Pool.

6. Right click the Enterprise PKS Management Plane Resource Pool and select **Deploy OVF Template.**

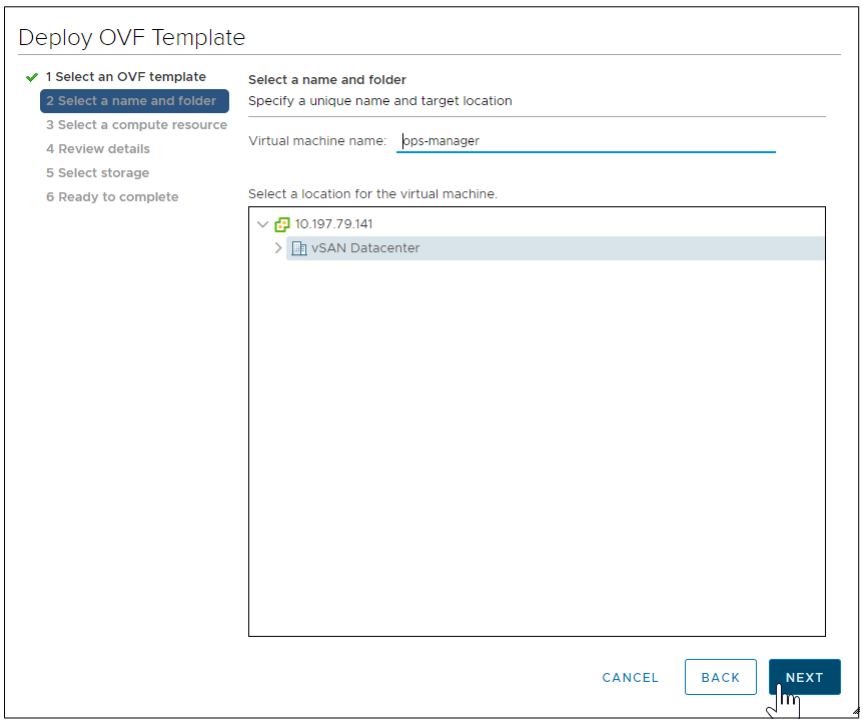
7. At the **Select an OVF template** screen:

   - Click **Browse**.
   - Select the Ops Manager OVA file you downloaded and click **Open**.

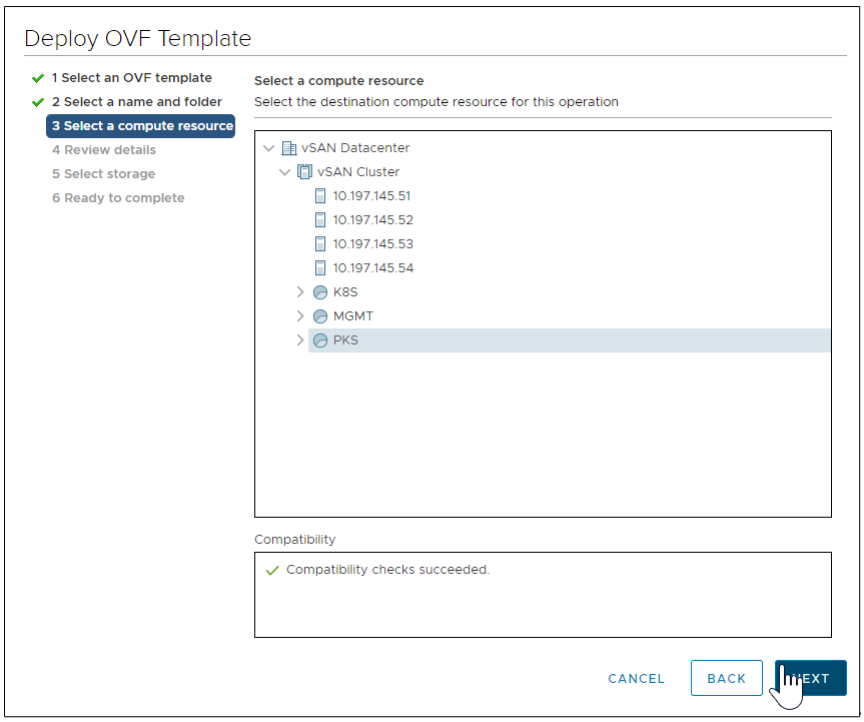


- Click **Next**.

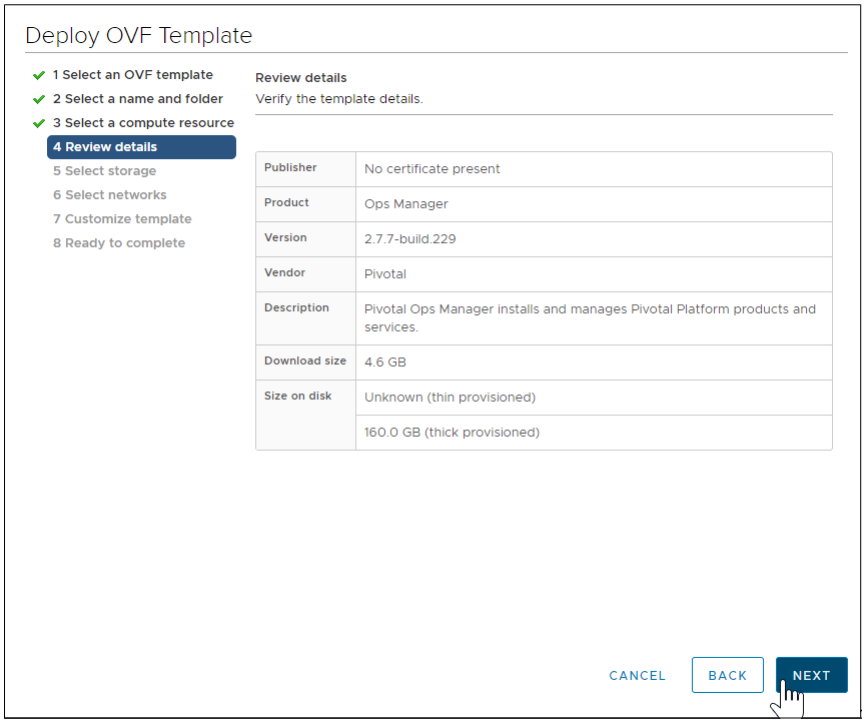
8. At the **Select Name and folder** screen, enter a name for the Ops Manager VM (or use the default name), select the **Datacenter**, and click **Next**



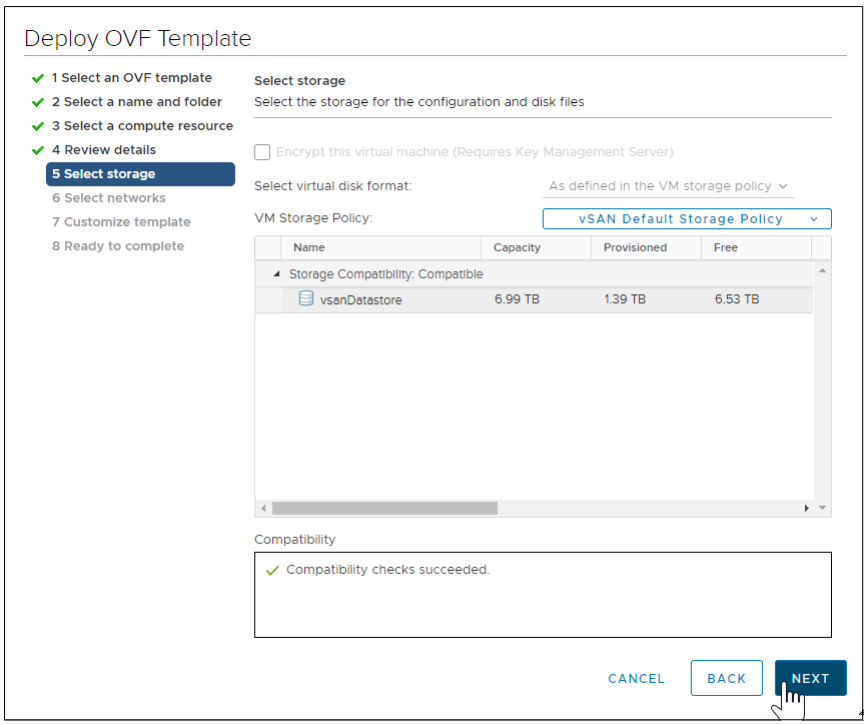
9. At the **Select a compute resource** screen, select the Enterprise PKS **Resource Pool** or **Cluster object** and click **Next**.



10. At the **Review details** screen, confirm the configuration up to this point and click **Next**.

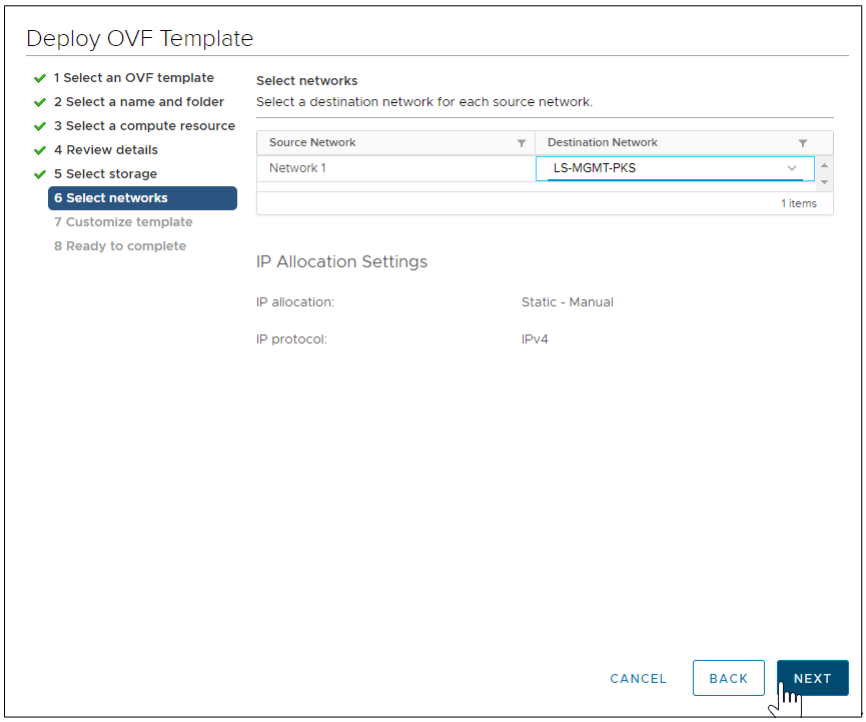


11. At the **Select Storage** screen, select the desired Datastore, and click **Next**.



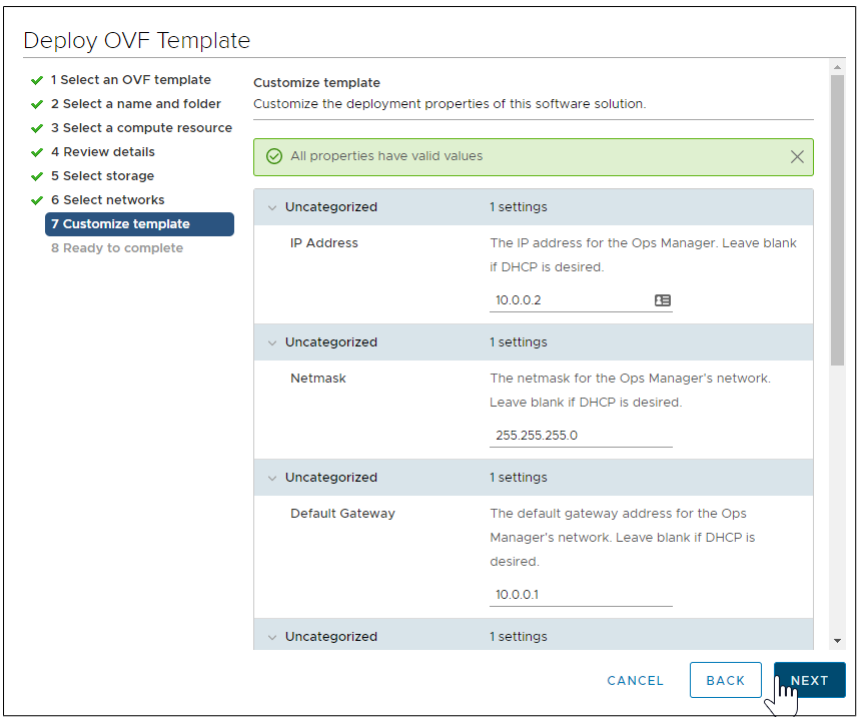
warning: Ops Manager requires a Director VM with at least 8 GB memory. For more information, see Provisioning a Virtual Disk in vSphere.

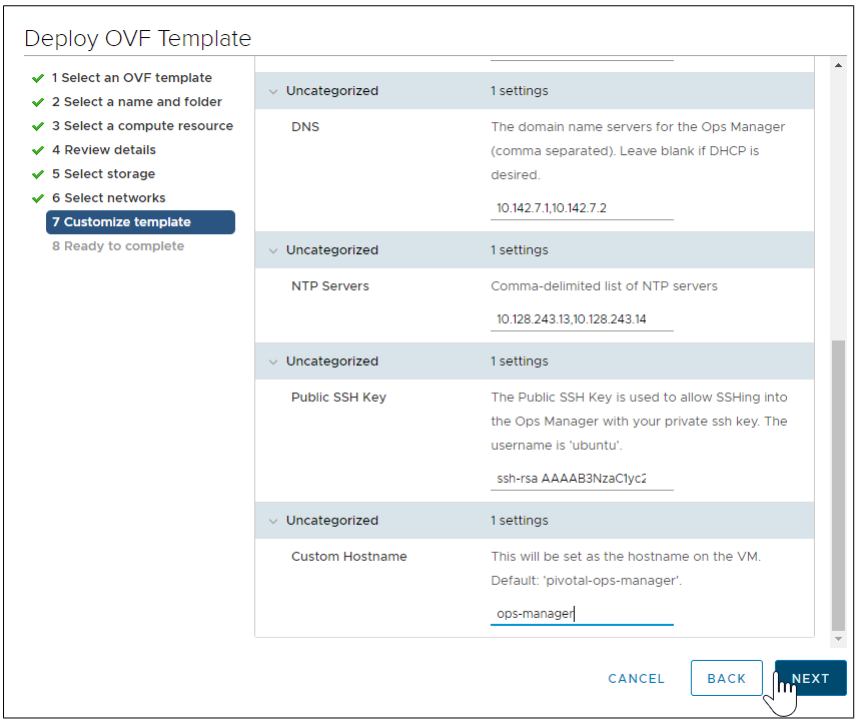
12. At the **Select Networks** screen:
    - If you are using vSphere 6.7, select the Enterprise PKS Management T1 Logical Switch that you defined when Creating the Enterprise PKS Management Plane, and click **Next**.



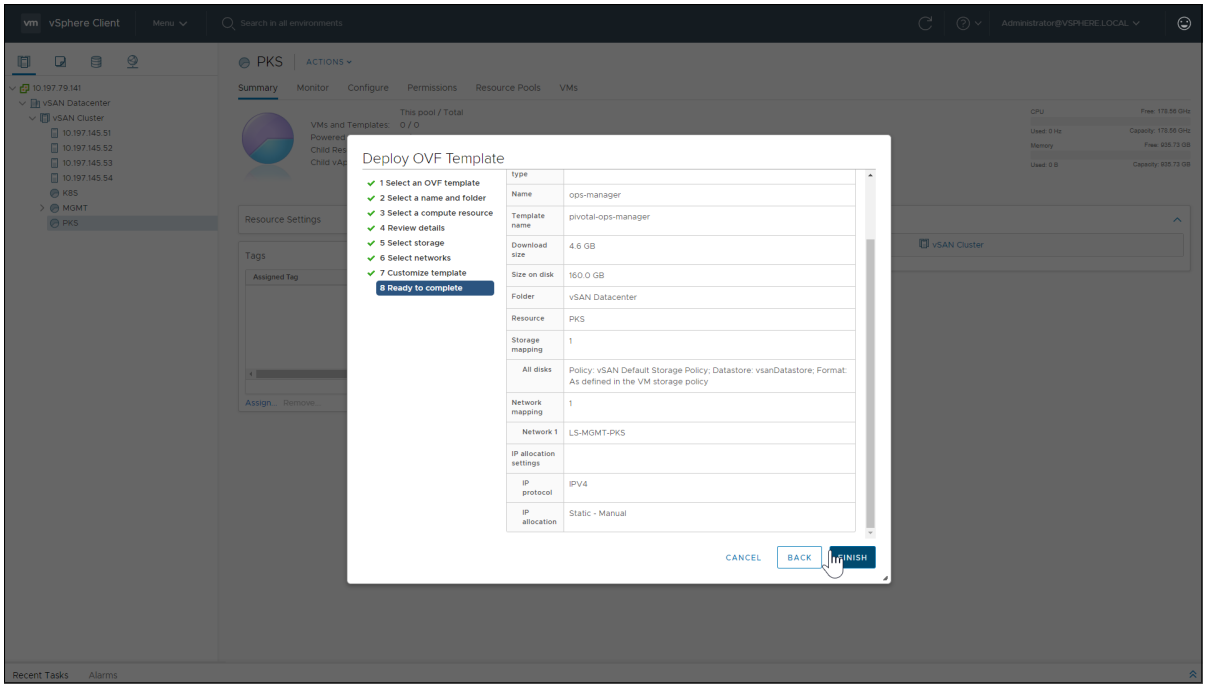
    - If you are using vSphere 6.5, see Network Selection for vSphere v6.5.

13. At the **Customize template** screen, enter the following information, and click **Next**.
    - **IP Address**: The IP address of the Ops Manager network interface, for example `10.0.0.2` (assuming non-routable NAT-mode).
    - **Netmask**: The network mask for Ops Manager, for example, `255.255.255.0`.
    - **Default Gateway**: The default gateway for Ops Manager to use, for example `10.0.0.1` (assuming non-routable NAT-mode).
    - **DNS**: One or more DNS servers for the Ops Manager VM to use, for example `10.14.7.1`.
    - **NTP Servers**: The IP address of one or more NTP servers for Ops Manager, for example `10.113.60.176`.
    - **Public SSH Key**: (Required) Enter the public SSH key to allow SSH access to the Ops Manager VM. You must enter the entire the public SSH key in the expected format. See SSH Key Requirements for Ops Manager v2.6 and Later.
    - **Custom hostname**: The hostname for the Ops Manager VM, for example `ops-manager`.

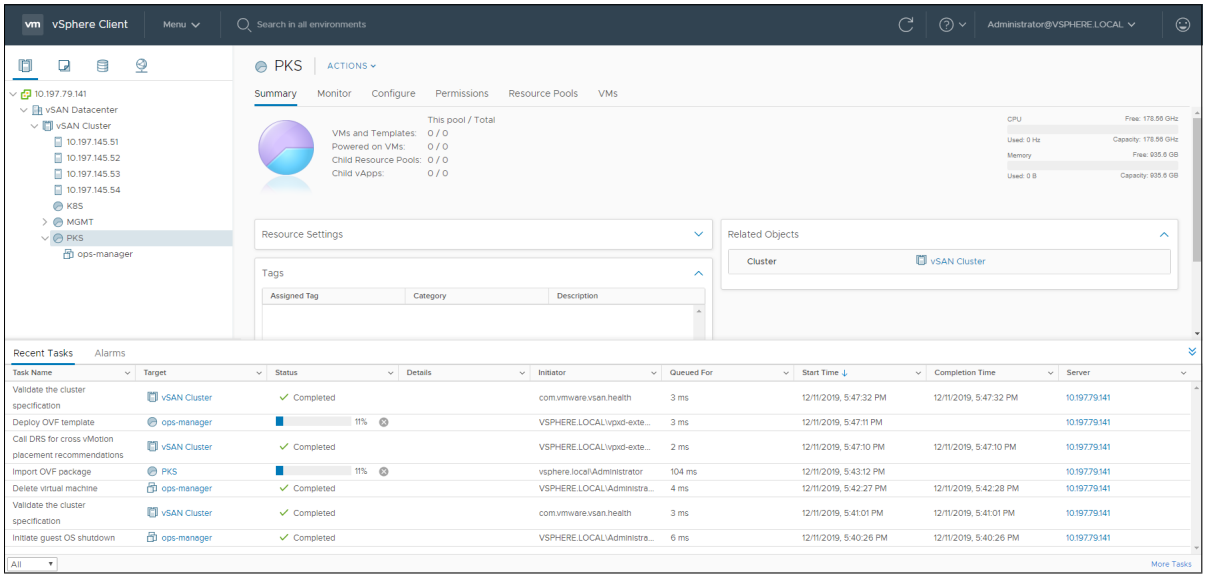




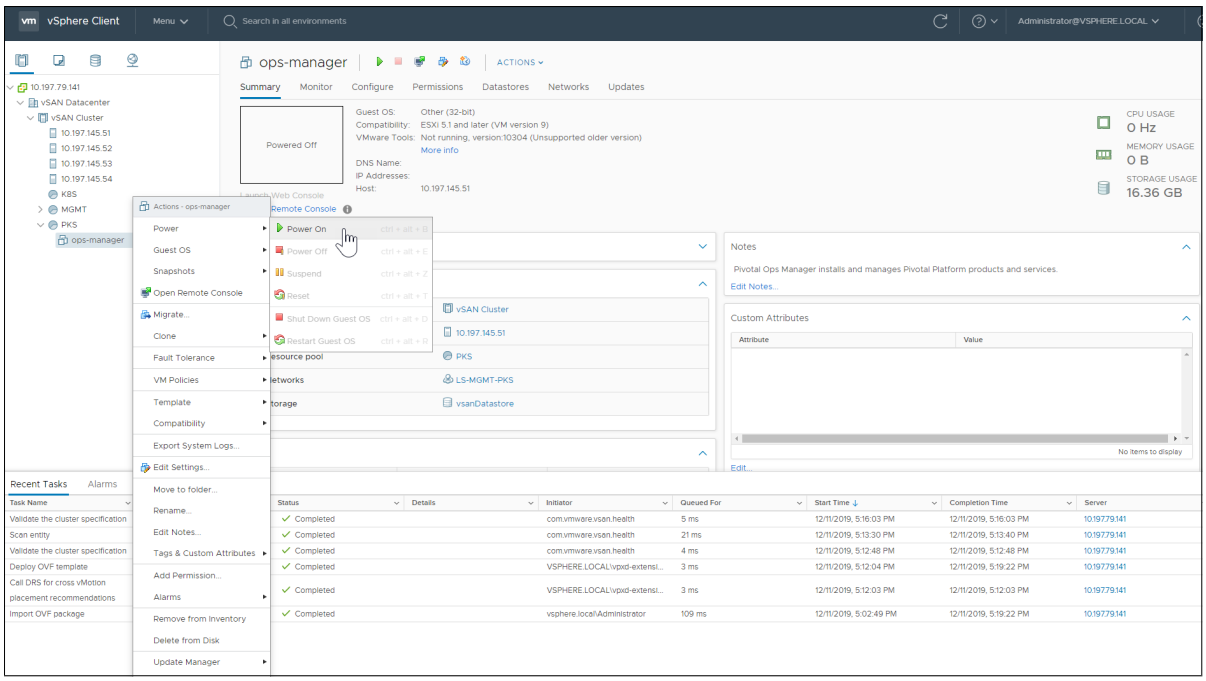
14. At the **Ready to complete** screen, review the configuration settings and click **Finish**. This action begins the OVA import and deployment process.



15. 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.



16. Right-click the Ops Manager VM and click **Power On**.



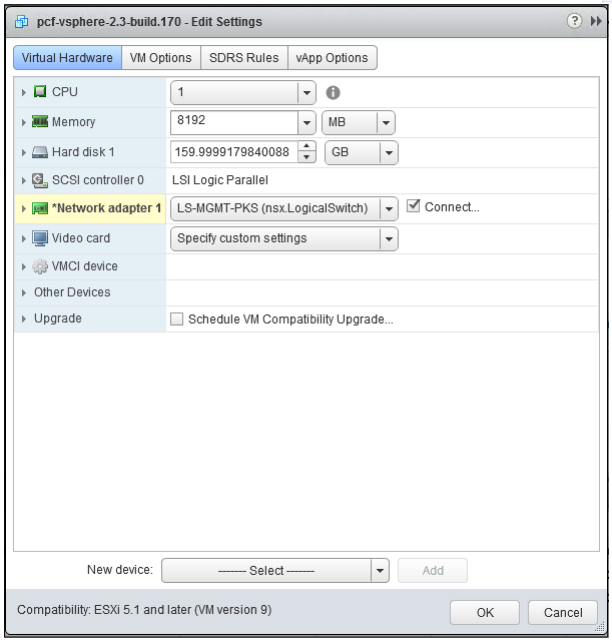
Network Selection for vSphere v6.5

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.

If you are using vSphere 6.5, at the **Select Networks** screen, select a vSS or vDS port-group such as the standard **VM Network**, and click **Next**.

Complete the remaining deployment steps as described above.

After the OVA deployment completes successfully, right-click the Ops Manager VM and select **Edit Settings**. Change the vNIC connection to use the `nsx.LogicalSwitch` that is defined for



the PKS Management Plane, for example `LS-MGMT-PKS`.

Step 3: Configure Ops Manager for Enterprise PKS

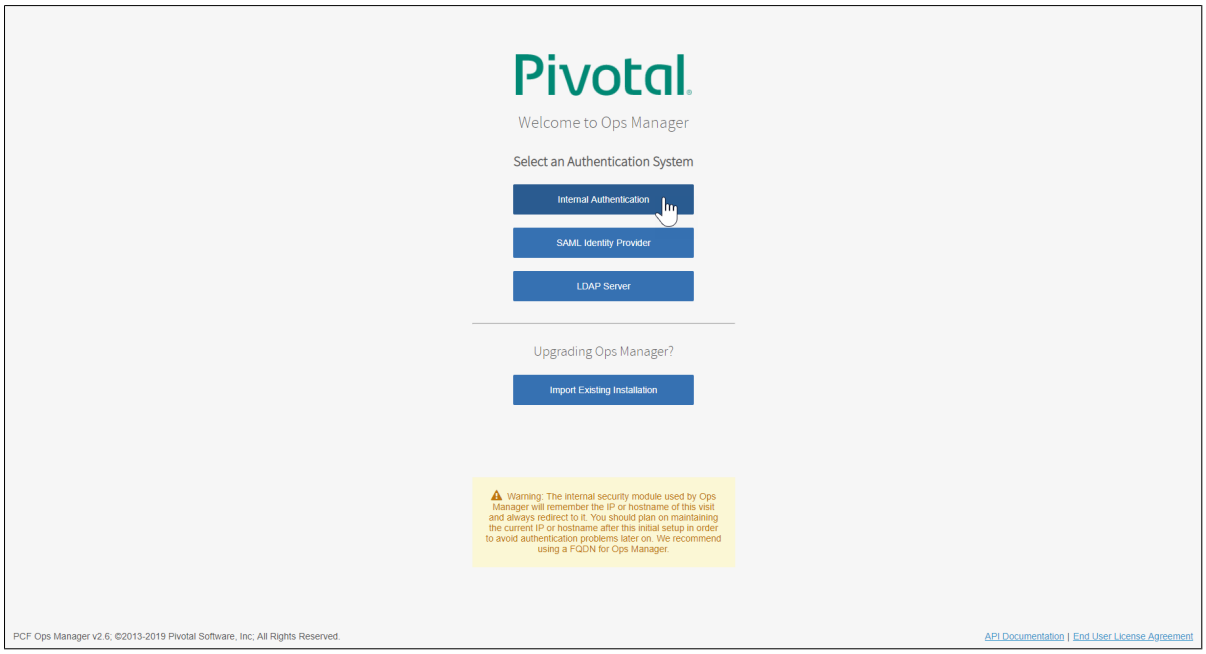
The first time you start Ops Manager, you are required select an authentication system. These instructions use **Internal Authentication**. See Set Up Ops Manager in the Pivotal Platform documentation for configuration details for the **SAML** and **LDAP** options.

1. If you are using the NAT deployment topology, create a DNAT rule that maps the Ops Manager private IP to a routable IP. See Create Enterprise PKS Management Plane for instructions.
2. If you are using the No-NAT deployment topology, create a DNS entry for the routable IP address that you set for Ops Manager. Use FQDN to log into Ops Manager.

> **Note**: Ops Manager security features require you to create a fully qualified domain name to access Ops Manager. See Installing Pivotal Cloud Foundry on vSphere.

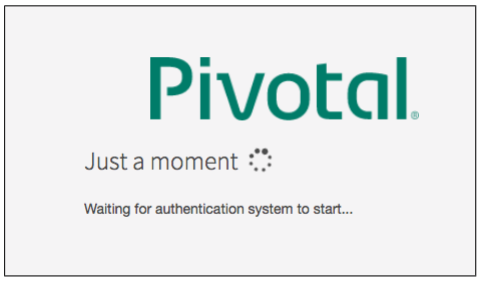
3. Navigate to the IP address (NAT mode) or FQDN (No-NAT mode) of your Ops Manager VM in a web browser. The “Welcome to Ops Manager” page should appear.

> **Note**: It is normal to experience a brief delay before the interface is accessible while the web server and VM start up.



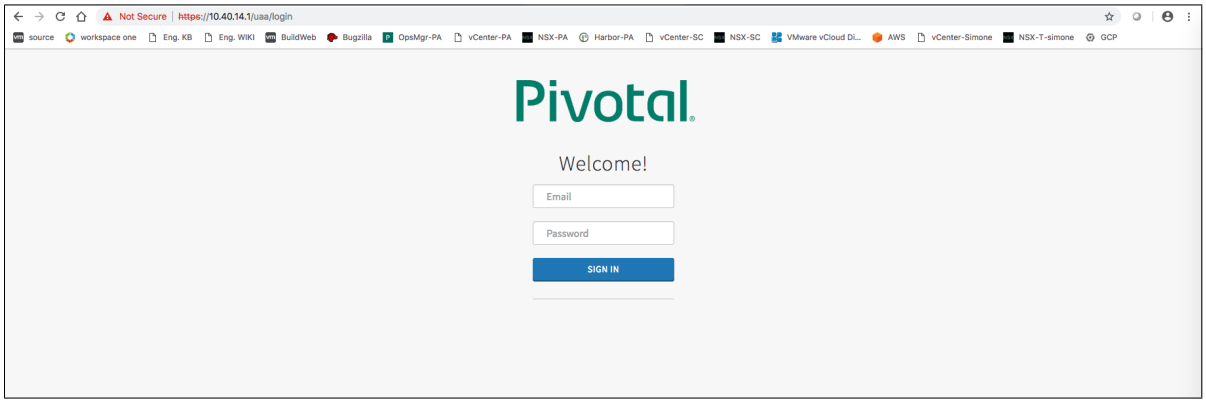
4. Select **Internal Authentication** and provide the following information:
   - **Username**, **Password**, and **Password confirmation** to create a user with administrative privileges.
   - **Decryption passphrase** and the **Decryption passphrase confirmation**. This passphrase encrypts the Ops Manager datastore, and is not recoverable.
   - **HTTP proxy** or **HTTPS proxy**, follow the instructions in Configuring Proxy Settings for the BOSH CPI.

5. Read the **End User License Agreement**, and select the checkbox to accept the terms.

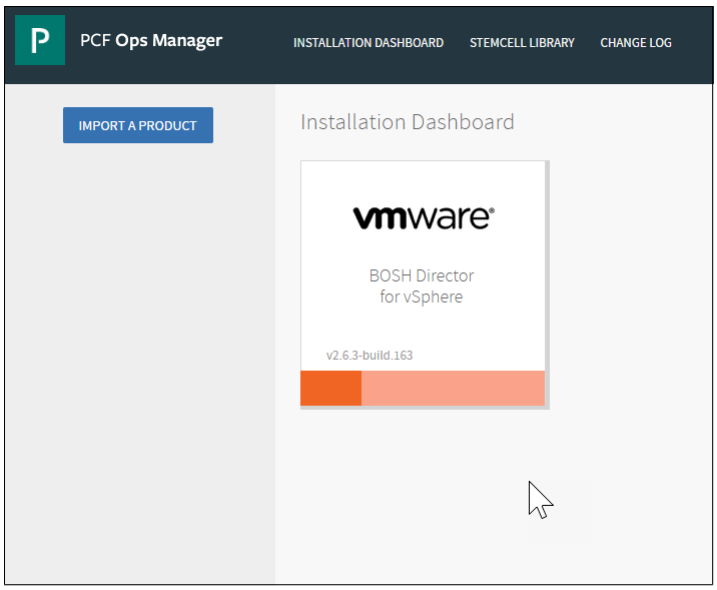


6. Click **Setup Authentication**. It takes a few minutes to initialize the database.

7. Log in to Ops Manager with the username and password that you created.



8. 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 Enterprise PKS

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

Generating and Registering the NSX Manager Certificate for Enterprise PKS

In this topic

Page last updated:

This topic describes how to generate and register the NSX Manager certificate authority (CA) certificate in preparation for installing VMware Enterprise PKS on vSphere with NSX-T.

Overview

The NSX Manager CA certificate is an IP-based, self-signed certificate that you create and register with the NSX Manager. The NSX Manager CA certificate is used to authenticate Enterprise PKS with NSX Manager during Enterprise PKS installation on vSphere with NSX-T.

Both the BOSH Director and Enterprise PKS tiles require the NSX Manager CA certificate when on vSphere with NSX-T. Ops Manager requires strict certificate validation and requires the subject and issuer of a self-signed certificate to be either the IP address or fully qualified domain name (FQDN) of the NSX Manager.

> **Note**: By default, the NSX Manager includes a self-signed API certificate with both the subject and issuer populated with hostname instead of IP address or FQDN. You must generate a valid self-signed certificate.

You can use either of the following methods to generate your NSX Manager CA certificate:

- Generate the NSX-T Root CA Certificate using the Cluster Virtual IP
- Generate the NSX-T Root CA Certificate using the Load Balancer IP

You can then register the generated certificate with the NSX Manager using the NSX API:

- Register the Cluster Virtual IP NSX-T Root CA Certificate
- Register the Load Balancer IP NSX-T Root CA Certificate

> **Note**: The following instructions are specific to NSX-T v2.4.1.

Prerequisites

Before you generate and register an NSX Manager CA certificate, ensure that you have successfully completed all of the following steps:

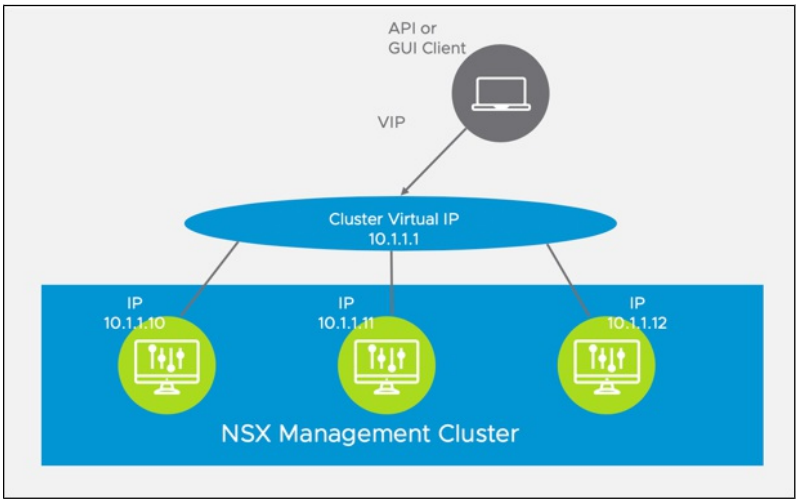
- Preparing to Install Enterprise PKS on vSphere with NSX-T Data Center
- Hardware Requirements for Enterprise PKS on vSphere with NSX-T
- Creating the Enterprise PKS Management Plane
- Creating Enterprise PKS Compute Plane
- Deploying Ops Manager with NSX-T for Enterprise PKS

Before configuring your BOSH and PKS tiles with your NSX Manager CA certificate, ensure that you have successfully completed the steps in Install Enterprise PKS in *Installing Enterprise PKS on vSphere with NSX-T*.

For configuration information, see the Networking section of *Installing Enterprise PKS on vSphere with NSX-T*.

Generate the NSX-T Root CA Certificate using the Cluster Virtual IP Address

The NSX-T Management Cluster is comprised of three NSX-T Manager nodes. You can configure the NSX-T Manager UI and API to support a single access point by assigning a virtual IP address (VIP) to the NSX-T Management Cluster.



To generate an NSX-T Management Cluster root CA certificate using your NSX Management Cluster VIP, complete the following steps:

- Configure a VIP for the NSX-T Management Cluster
- Define the Certificate Signing Request Using the VIP
- Generate the VIP Certificate and Private Key

Configure a VIP for the NSX-T Management Cluster

The NSX-T Management Cluster VIP IP address is used to configure the NSX-T Management Cluster certificate.

1. To support a single access point for the NSX Manager API, complete the steps in Assign a Virtual IP Address to the NSX-T Management Cluster

Define the Certificate Signing Request Using the VIP

To define the Certificate Signing Request, complete the following steps:

1. Create a new Certificate Signing Request (CSR) file named `nsx-cert.cnf`.
2. Complete the CSR file using the following template:

```
[ req ]
default_bits = 2048
distinguished_name = req_distinguished_name
req_extensions = req_ext
prompt = no
[ req_distinguished_name ]
countryName = COUNTRY-INIT
stateOrProvinceName = STATE
localityName = STATE-INIT
organizationName = NSX
commonName = IP-ADDRESS
[ req_ext ]
subjectAltName = @alt_names
[alt_names]
DNS.1 = IP-ADDRESS
```

Where:

- `COUNTRY-INIT` are the initials for your corporation's country of origin.
- `STATE` is your corporation's state or province.
- `STATE-INIT` are the initials for your corporation's state or province.
- `IP-ADDRESS` is your Management Cluster VIP 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 = CA
organizationName = NSX
commonName = 10.40.206.5
[ req_ext ]
subjectAltName = @alt_names
[alt_names]
DNS.1 = IP-ADDRESS
```

Note: The Cluster VIP IP address must be used as the `commonName` attribute value because the certificate will be registered as a Cluster CA certificate, not as a Node CA certificate.

3. Copy the completed `nsx-cert.cnf` file to a Linux-based VM that is on the same network as the PKS Management Plane.

Generate the VIP Certificate and Private Key

To use your new CSR to generate a certificate and private key, complete the following commands:

1. To export the `NSX_MANAGER_IP_ADDRESS` and `NSX_MANAGER_COMMONNAME` environment variables, run the following command:

```
export NSX_MANAGER_IP_ADDRESS=IP-ADDRESS
export NSX_MANAGER_COMMONNAME=IP-ADDRESS
```

Where `IP-ADDRESS` is your Management Cluster VIP IP Address.
For example:

```
$ export NSX_MANAGER_IP_ADDRESS=10.40.206.5
$ export NSX_MANAGER_COMMONNAME=10.40.206.5
```

2. To use the above CSR to generate the certificate and private key files, `nsx.crt` and `nsx.key`, 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
```

3. To verify the certificate, run the following command:

```
openssl x509 -in nsx.crt -text -noout
```

Register the Cluster Virtual IP Certificate

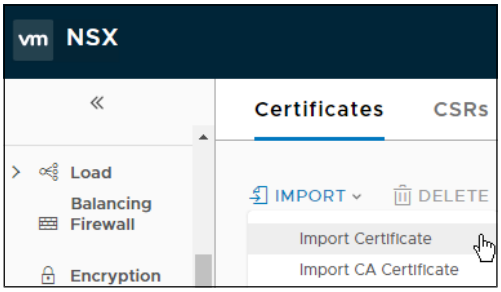
To register the Cluster Virtual IP NSX-T Root CA certificate, complete the following:

- Import the VIP Certificate to NSX Manager
- Register the VIP Certificate Using the Cluster Certificate API
- Verify the VIP Certificate
- Add the VIP Certificate to BOSH and PKS Tiles

Import the VIP Certificate to NSX Manager

To import the certificate to the NSX Manager, complete the following steps:

1. Log in to the NSX Manager UI using the VIP IP address.
2. Navigate to **System** > **Certificates**.

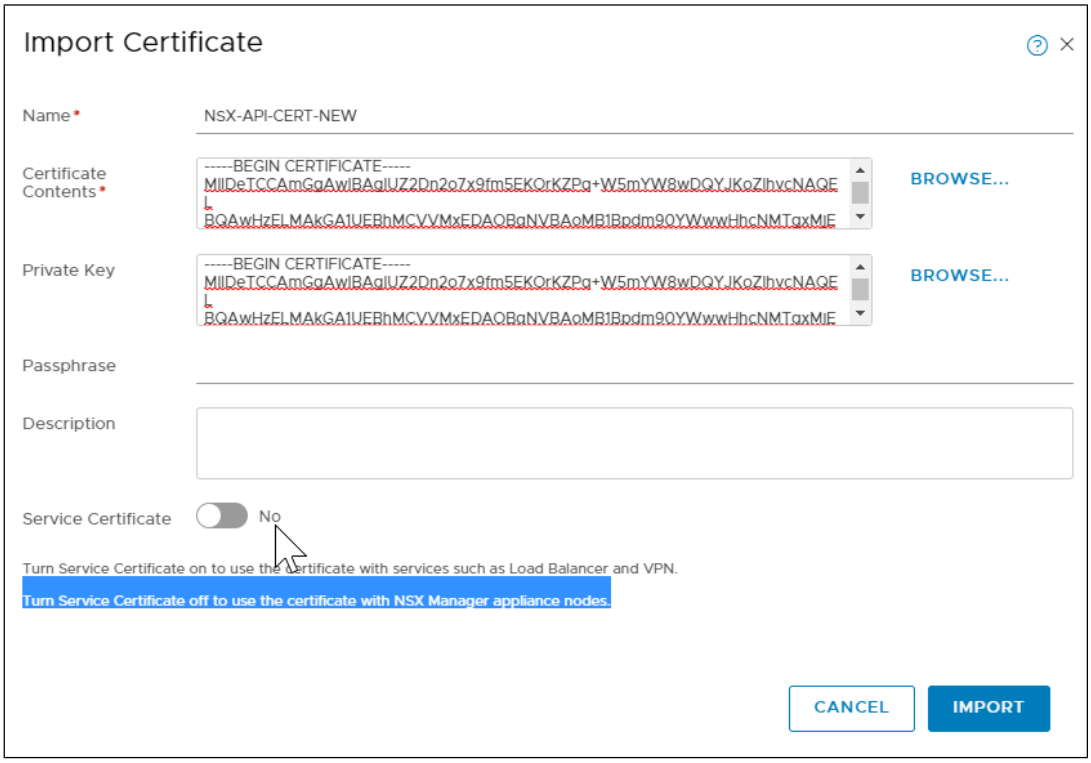


3. Click **Import** > **Import Certificate**. The **Import Certificate** screen is displayed.

> **Note**: Ensure that you select **Import Certificate** and not **Import CA Certificate**.

4. In the **Name** field, enter a unique name for the certificate, such as `NSX-VIP-CERT`.

> **Note**: The certificate name must be unique. The default NSX Manager CA certificate is typically named `NSX-API-CERT`.

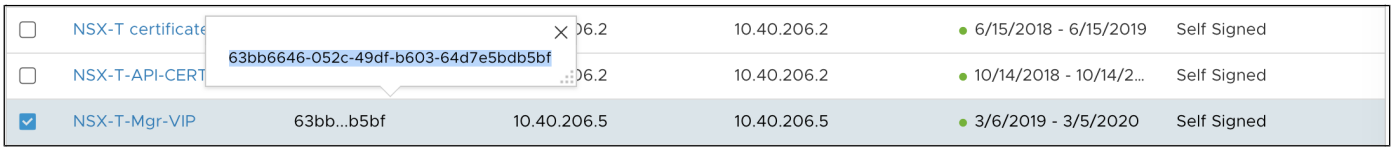


5. In the **Certificate Contents** field, enter the certificate string. You can locate the certificate string in the `nsx.crt` certificate file that you previously generated. Copy and paste the certificate contents of the certificate file into the field.
6. In the **Private Key** field, enter your certificate's private key. You can locate the key string in the `nsx.key` private key file that you previously generated. Copy and paste the key content of the key file into the field.
7. Verify that the **Service Certificate** option is set to **No**. The Service Certificate setting should be "off" because you are using the certificate with NSX Manager appliance nodes.
8. Click **Import**.
9. To verify that your configuration has replicated to all NSX-T Manager instances, perform the following steps:
   a. Log in to each individual NSX Manager node.
   b. On each node, navigate to the **System** > **Certificates** screen.
   c. Confirm that the certificate has replicated to the NSX-T Manager instance.

Register the VIP Certificate Using the Cluster Certificate API

To register the imported VIP certificate with the NSX Management Cluster Certificate API, complete the following steps:

1. To retrieve the certificate UUID, open the NSX Manager **System** > **Certificates** screen that you used to import the certificate and copy the UUID.



2. To create `NSX_MANAGER_IP_ADDRESS` and `CERTIFICATE_ID` environment variables, run the following commands:

```
export NSX_MANAGER_IP_ADDRESS=IP-ADDRESS
export CERTIFICATE_ID="CERTIFICATE-ID"
```

   Where:

   - `IP-ADDRESS` is the VIP IP address.
   - `CERTIFICATE-ID` is the certificate UUID.

For example:

```
$ export NSX_MANAGER_IP_ADDRESS=10.40.206.5
$ export CERTIFICATE_ID="63cd6646-057b-48bf-b603-64d7e5bdb5bf"
```

3. To register the NSX-T Manager CA certificate, run the following cURL request to the Cluster Certificate API:

```
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 the password for NSX Manager.
For example:

```
$ curl --insecure -u admin:'P1A2S3S4W56O7R8D!' -X POST "https://$NSX_MANAGER_IP_ADDRESS/api/v1/cluster/api-certificate?action=set_cluster_certificate&certificate_id=$CERTIFICATE_ID"
{
 "certificate_id": "63bb6646-052c-49df-b603-64d7e5bdb5bf"
}
```

Note: The certificate chain must be in the industry standard order of `certificate - intermediate - root`.

4. Verify that the API request response contains the certificate UUID.

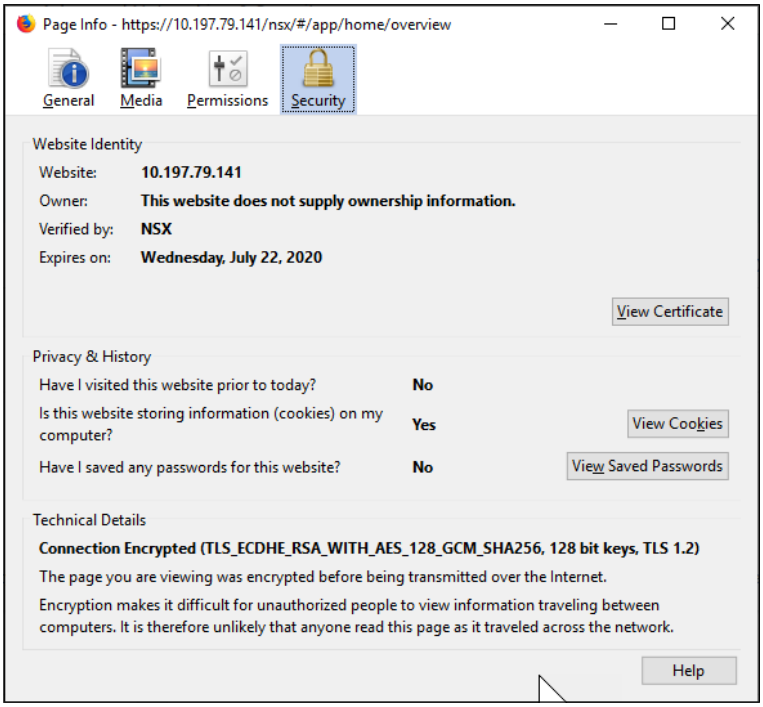
Verify the VIP Certificate

To verify that the VIP certificate has been registered with the NSX Manager, perform the following steps:

1. Open a browser to the VIP IP address of the NSX Manager.

2. Log in to the NSX Manager.

3. Confirm that the new certificate is used by the site.



4. SSH to each NSX Manager host and run the following commands:

```
get certificate api
get certificate cluster
```

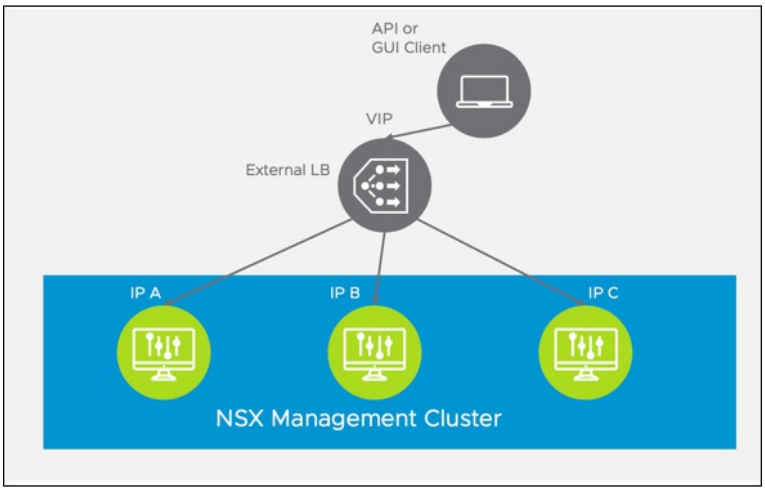
5. Confirm that all returned certificates are the certificates that you generated when performing the steps above.

Add the VIP Certificate to BOSH and PKS Tiles

To add the VIP Certificate to BOSH and PKS Tiles, complete the steps in Add or Update the Certificate for the BOSH and PKS tiles below.

Generate the NSX-T Root CA Certificate using the Load Balancer IP Address

The NSX-T Management Cluster is comprised of three NSX-T Manager Nodes.



The IP address used to configure the NSX-T Management Cluster certificate above is the IP address to use for configuring the load balancer.

To generate the NSX-T root CA certificate using the load balancer IP, complete the following:

- Configure a Load Balancer for the NSX Management Cluster
- Define a CSR Using the Load Balancer IP
- Generate NSX Root CA Certificate and Private Key Using the Load Balancer IP Address

Configure a Load Balancer for the NSX Management Cluster

To scale your load balancer, complete the following steps:

1. Scale your load balancer by completing the steps in Provision a Load Balancer for the NSX-T Management Cluster.

Define a CSR Using the Load Balancer IP Address

To define the Certificate Signing Request for a load balancer in front of the management cluster, complete the following steps:

1. Create a new Certificate Signing Request (CSR) file named `nsx-cert.cnf`.
2. Complete the CSR file using the following template:

```
[ req ]
default_bits = 2048
distinguished_name = req_distinguished_name
req_extensions = req_ext
prompt = no
[ req_distinguished_name ]
countryName = COUNTRY-INIT
stateOrProvinceName = STATE
localityName = STATE-INIT
organizationName = NSX
commonName = IP-ADDRESS
[ req_ext ]
subjectAltName = @alt_names
[alt_names]
DNS.1 = IP-ADDRESS
```

   Where:

   - `COUNTRY-INIT` are the initials for your corporation's country of origin.
   - `STATE` is your corporation's state or province.
   - `STATE-INIT` are the initials for your corporation's state or province.

- `IP-ADDRESS` is your load balancer's 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 = CA
organizationName = NSX
commonName = 10.40.14.250
[ req_ext ]
subjectAltName = @alt_names
[alt_names]
DNS.1 = 10.40.14.250
```

Generate NSX Root CA Certificate and Private Key Using the Load Balancer IP Address

1. To create `NSX_MANAGER_IP_ADDRESS` and `CERTIFICATE_ID` environment variables, run the following commands:

   ```
   export NSX_MANAGER_IP_ADDRESS=IP-ADDRESS
   export CERTIFICATE_ID="CERTIFICATE-ID"
   ```

   Where:

   - `IP-ADDRESS` is the VIP IP address.
   - `CERTIFICATE-ID` is the certificate UUID.

   For example:

   ```
   $ export NSX_MANAGER_IP_ADDRESS=10.40.14.250
   $ export NSX_MANAGER_COMMONNAME=10.40.14.250
   ```

2. To use the above CSR to generate the certificate and private key files, `nsx.crt` and `nsx.key`, run the following commands:

   ```
   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
   ```

3. To verify the certificate, run the following command:

   ```
   openssl x509 -in nsx.crt -text -noout
   ```

Register the Load Balancer IP NSX-T Root CA Certificate

To register the load balancer IP NSX-T root CA certificate, complete the following:

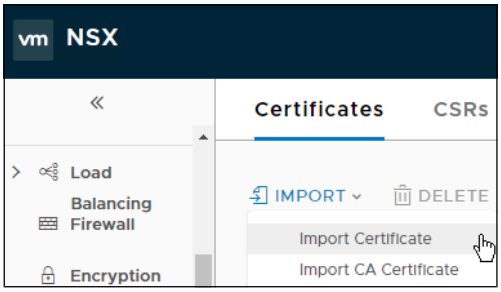
- Import CA Certificate to All Three NSX Managers
- Register Load Balancer Certificate with All Three NSX-T Manager Appliances
- Verify Load Balancer Certificate
- Add VIP Certificate to BOSH and PKS Tiles

Import CA Certificate to All Three NSX Managers

Complete the following steps to import the certificate to the NSX Manager:

1. Log in to the NSX Manager UI using the VIP IP address.

2. Navigate to **System** > **Certificates**.



3. Click **Import** > **Import Certificate**. The **Import Certificate** screen is displayed.

   > **Note**: Ensure that you select **Import Certificate** and not **Import CA Certificate**.

4. In the **Name** field, enter a unique name for the certificate, such as `NSX-VIP-CERT`.

   > **Note**: The certificate name must be unique. The default NSX Manager CA certificate is typically named `NSX-API-CERT`.

5. In the **Certificate Contents** field, enter the certificate string. You can locate the certificate string in the `nsx.crt` certificate file that you previously generated. Copy and paste the certificate contents of the certificate file into the field.

6. In the **Private Key** field, enter your certificate's private key. You can find the key string in the `nsx.key` private key file you previously generated. Copy and paste the key contents of the key key into the field.

7. Verify that the **Service Certificate** option is set to **No**. The Service Certificate setting should be "off" because you are using the certificate with NSX Manager appliance nodes.

8. Click **Import**.

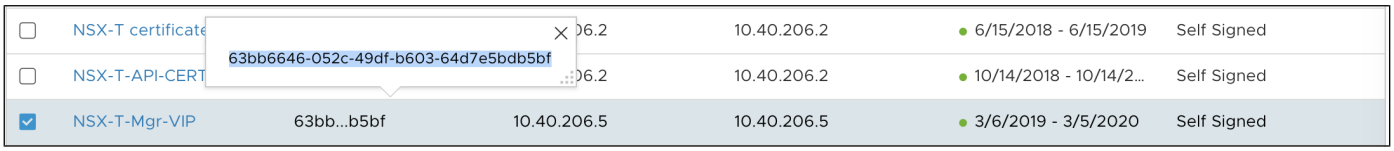
9. To verify that your configuration has replicated to all NSX-T Manager instances, perform the following:

   a. Log in to each individual NSX Manager node.
   b. On each node, navigate to the **System** > **Certificates** screen.
   c. Confirm that the certificate has replicated to the NSX-T Manager instance.

Register Load Balancer Certificate with All Three NSX-T Manager Appliances

To register the imported load balancer certificate with the NSX Management Cluster Certificate API, complete the following steps:

1. To retrieve the certificate UUID, open the NSX Manager **System** > **Certificates** screen that you used to import the certificate and copy the UUID.



2. To create `NSX_MANAGER_IP_ADDRESS` and `CERTIFICATE_ID` environment variables, run the following commands:

```
export NSX_MANAGER_IP_ADDRESS=IP-ADDRESS
export CERTIFICATE_ID="CERTIFICATE-ID"
```

   Where:

   - `IP-ADDRESS` is the IP address of one of the NSX-Manager nodes.
   - `CERTIFICATE-ID` is the certificate UUID.

   For example:

```
$ export NSX_MANAGER_IP_ADDRESS=10.40.206.1
$ export CERTIFICATE_ID="63bb6646-052c-49df-b603-64d7e5bdb5bf"
```

   In this example, there are three NSX Manager nodes, `10.40.206.1`, `10.40.206.2` and `10.40.206.3`, where `10.40.206.1` is the IP address of NSX Manager 1.

3. To register the NSX-T Manager CA certificate, run the following cURL request to the NSX API:

```
curl --insecure -u admin:'PASSWORD' -X POST "https://$NSX_MANAGER_IP_ADDRESS/api/v1/node/services/http?action=apply_certificate&certificate_id=$CERTIFICATE_ID"
```

Where `PASSWORD` is the password for NSX Manager.
For example:

```
$ curl --insecure -u admin:'P1A2S3S4W5O6R7D!' -X POST "https://$NSX_MANAGER_IP_ADDRESS/api/v1/node/services/http?action=apply_certificate&certificate_id=$CERTIFICATE_ID"
{
 "certificate_id": "63bb6646-052c-49df-b603-64d7e5bdb5bf"
}
```

Note: The certificate chain must be in the industry standard order of `certificate - intermediate - root`.

4. Verify that the API request response contains the certificate UUID.

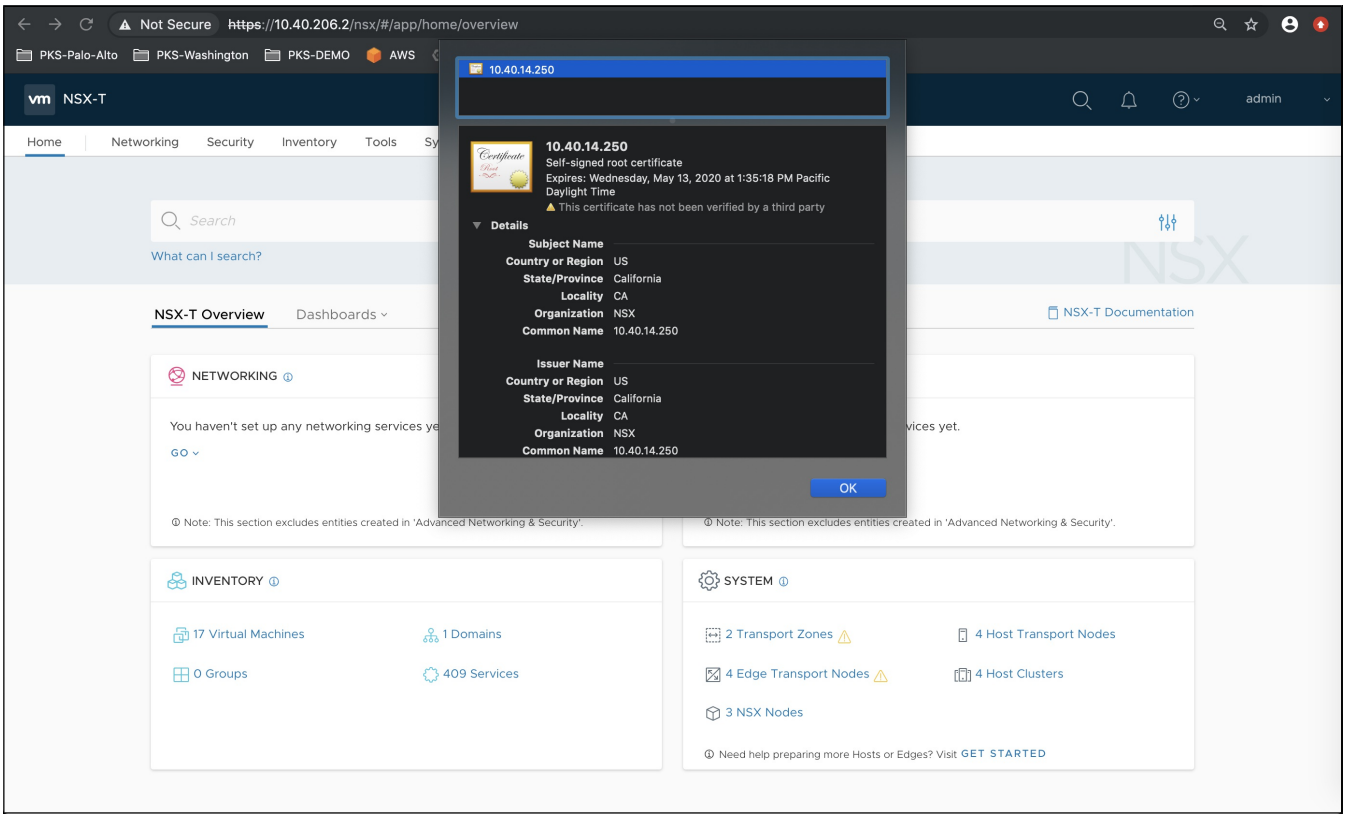
5. Repeat the preceding registration steps for the other two NSX Manager nodes.

Note: When repeating the steps for the other NSX Manager nodes, `NSX_MANAGER_IP_ADDRESS` should be different for each node. The `CERTIFICATE_ID` should be identical for all of the NSX Manager nodes.

Verify Load Balancer Certificate

1. Log in to each NSX Manager node.

   a. For each site, confirm that the site's load balancer IP address is used on the site's certificate **Common Name** field.



2. SSH to each NSX Manager host.

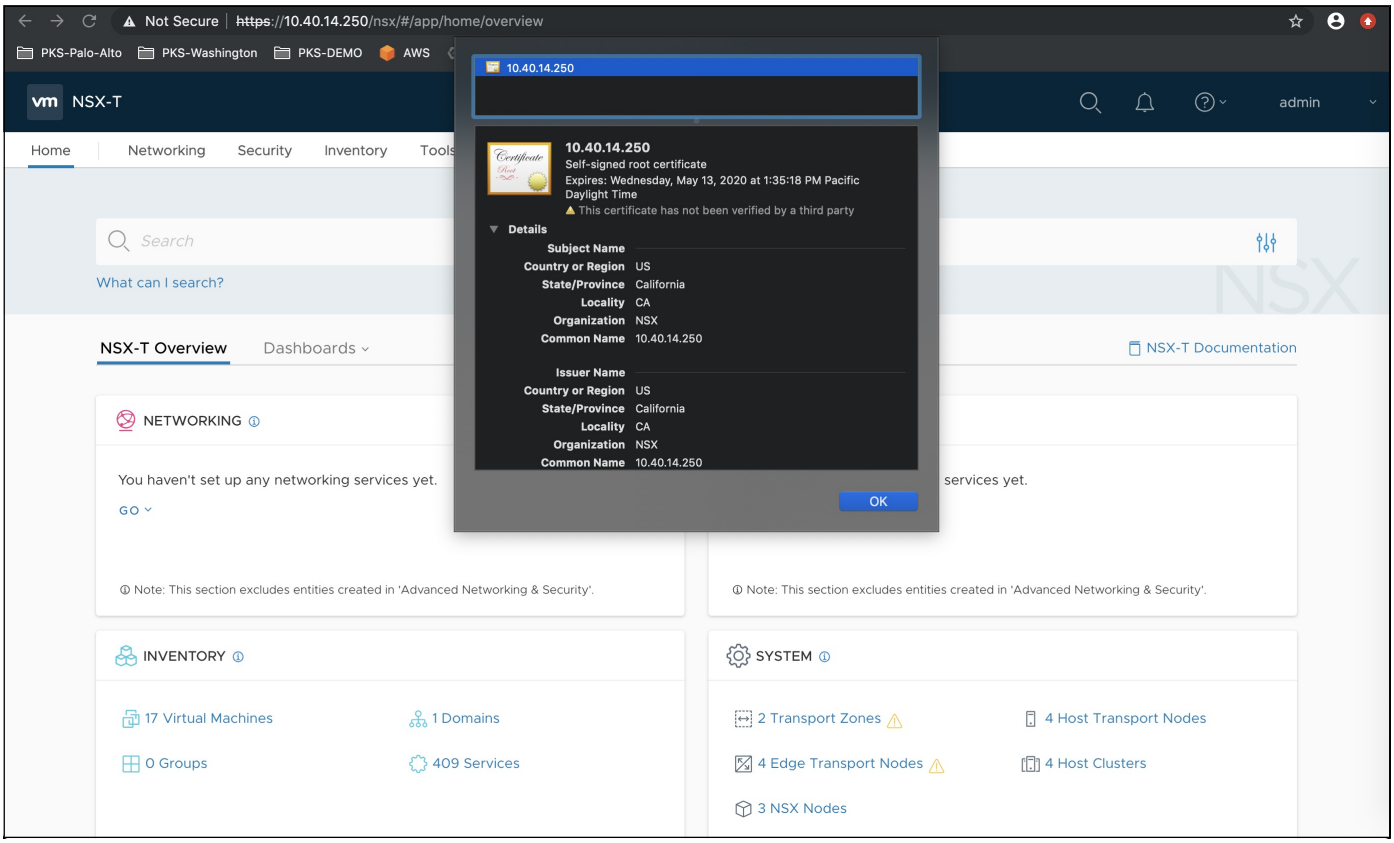
   a. For each host, run the following command:

```
get certificate cluster
```

   b. Confirm that the returned certificate is the certificate that you generated when performing the steps above.

3. Access the NSX-T Manager using the NSX-T Load Balancer IP address.

4. Confirm that the returned certificate is the certificate that you generated when performing the steps above.



Add the VIP Certificate to BOSH and PKS Tiles

To add the VIP Certificate to BOSH and PKS Tiles, complete the steps in Add or Update the Certificate for the BOSH and PKS tiles

Add or Update the Certificate for the BOSH and PKS Tiles

To create, delete, and modify NSX-T networking resources, Pivotal Platform tiles and APIs use a VMware NSX Manager account with the Enterprise Administrator role and permissions.

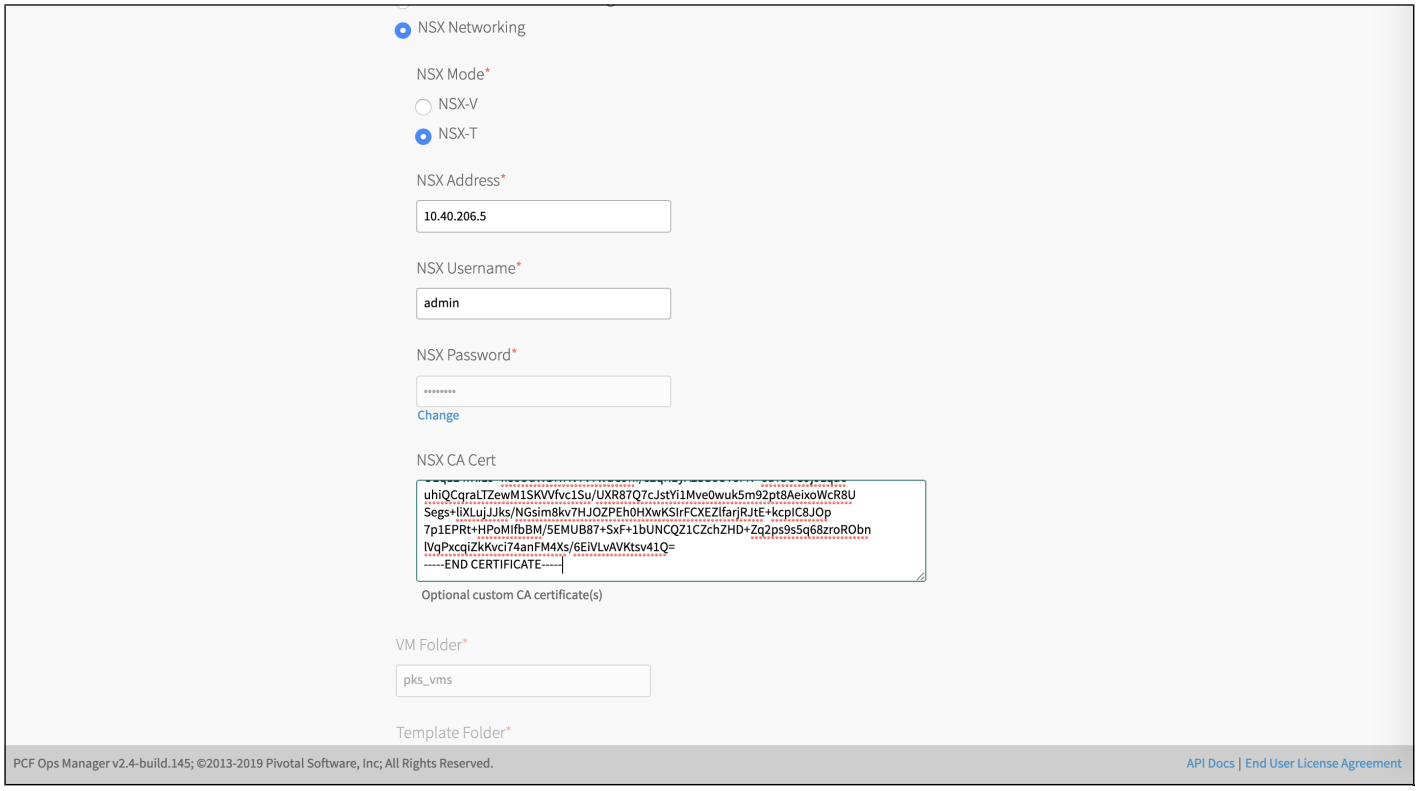
Users configure Pivotal Platform to authenticate to NSX Manager for different purposes in different tiles:

- **Enterprise PKS tile**
  The Enterprise PKS tile uses NSX Manager to create load balancers, providing a Kubernetes service described in the Create an External Load Balancer section of the Kubernetes documentation.

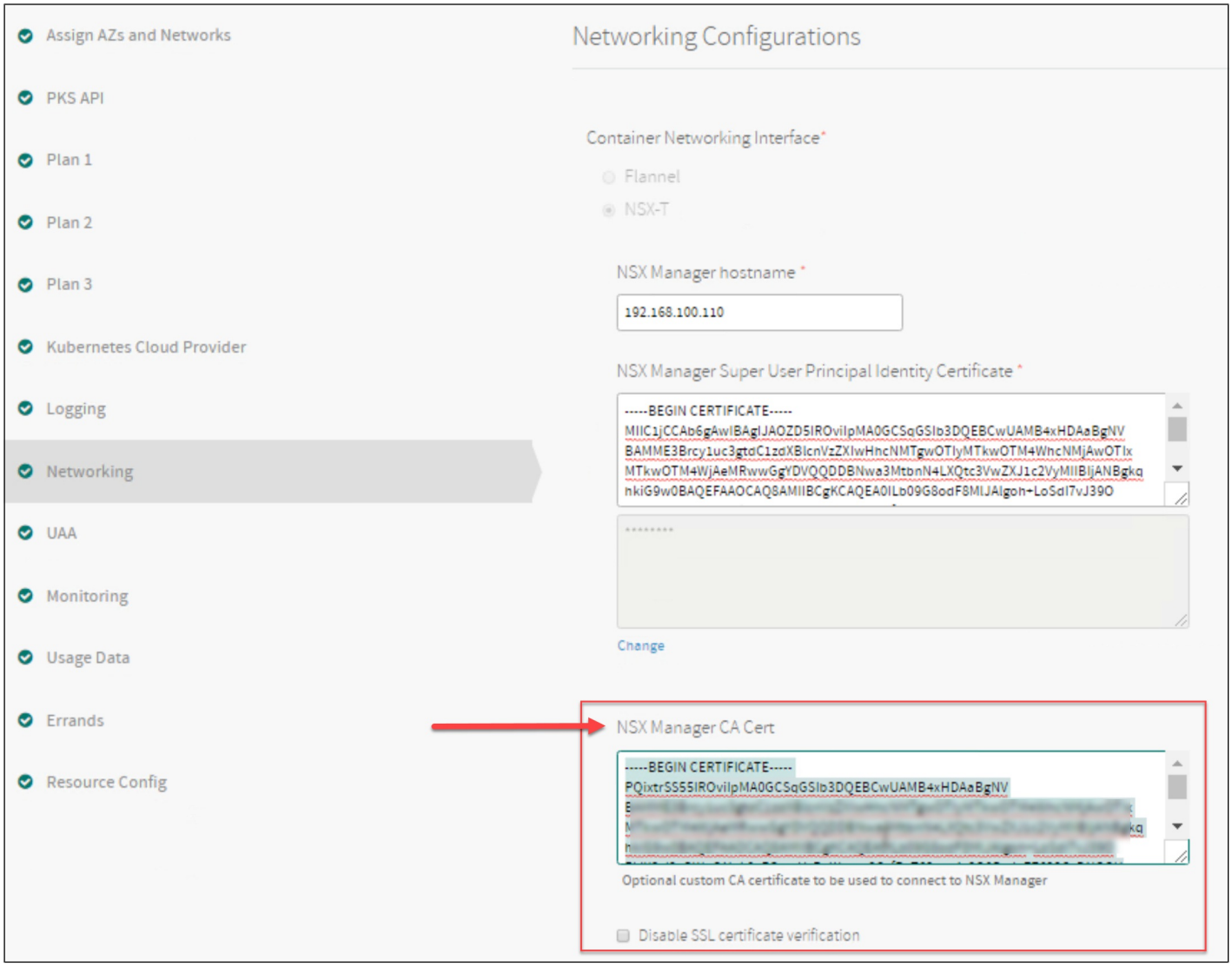
- **BOSH Director for vSphere tile**
  The **BOSH Director for vSphere** tile uses NSX Manager to configure networking and security for external-facing Pivotal Platform component VMs, such as Pivotal Application Service routers.

Both the BOSH Director and PKS tiles must be configured with the NSX Manager CA certificate during Enterprise PKS installation on vSphere with NSX-T. The following are examples of properly configured BOSH Director and PKS tile certificate settings.

- BOSH Director tile NSX Manager CA certificate configuration:



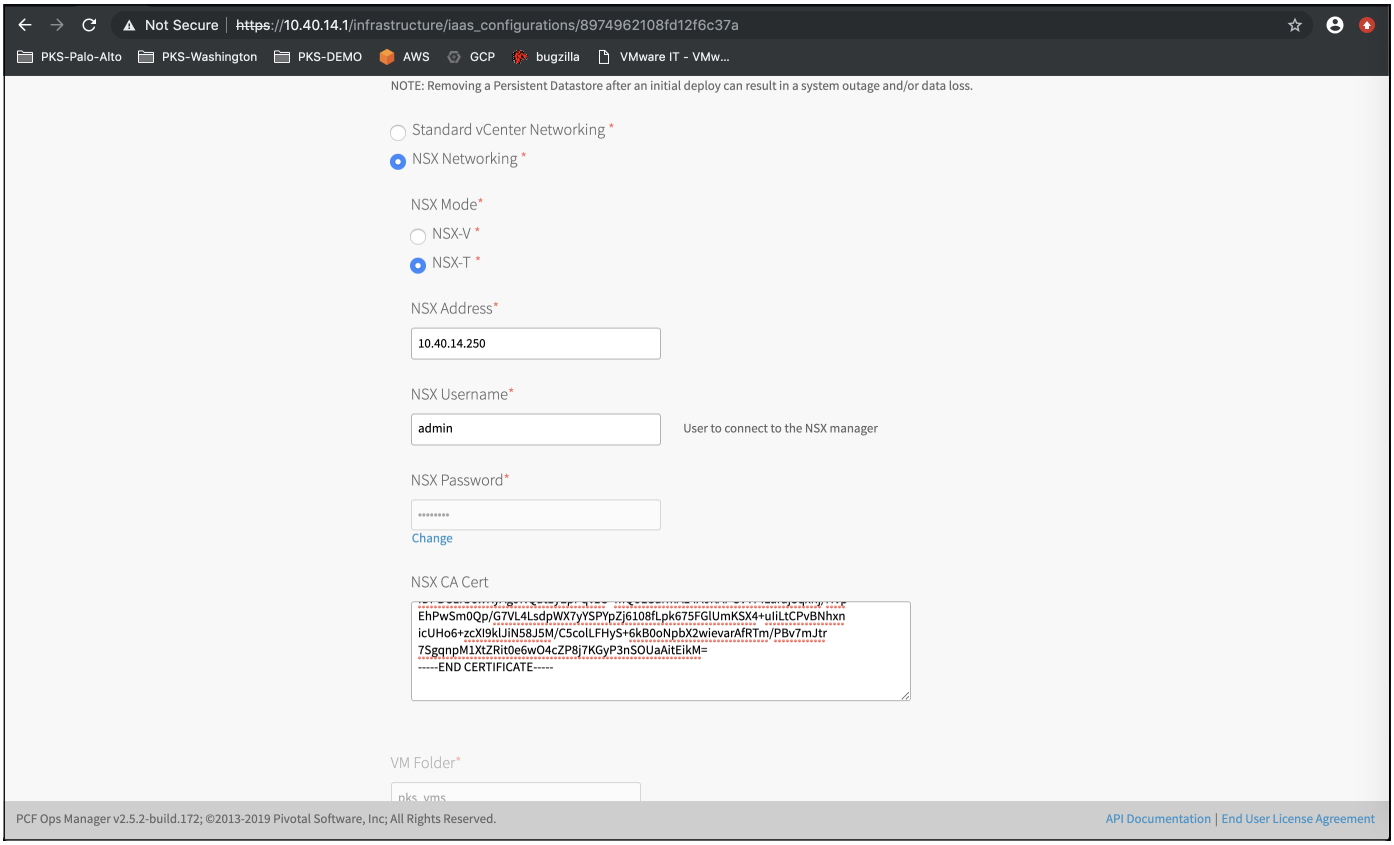
- PKS tile NSX Manager CA certificate configuration:



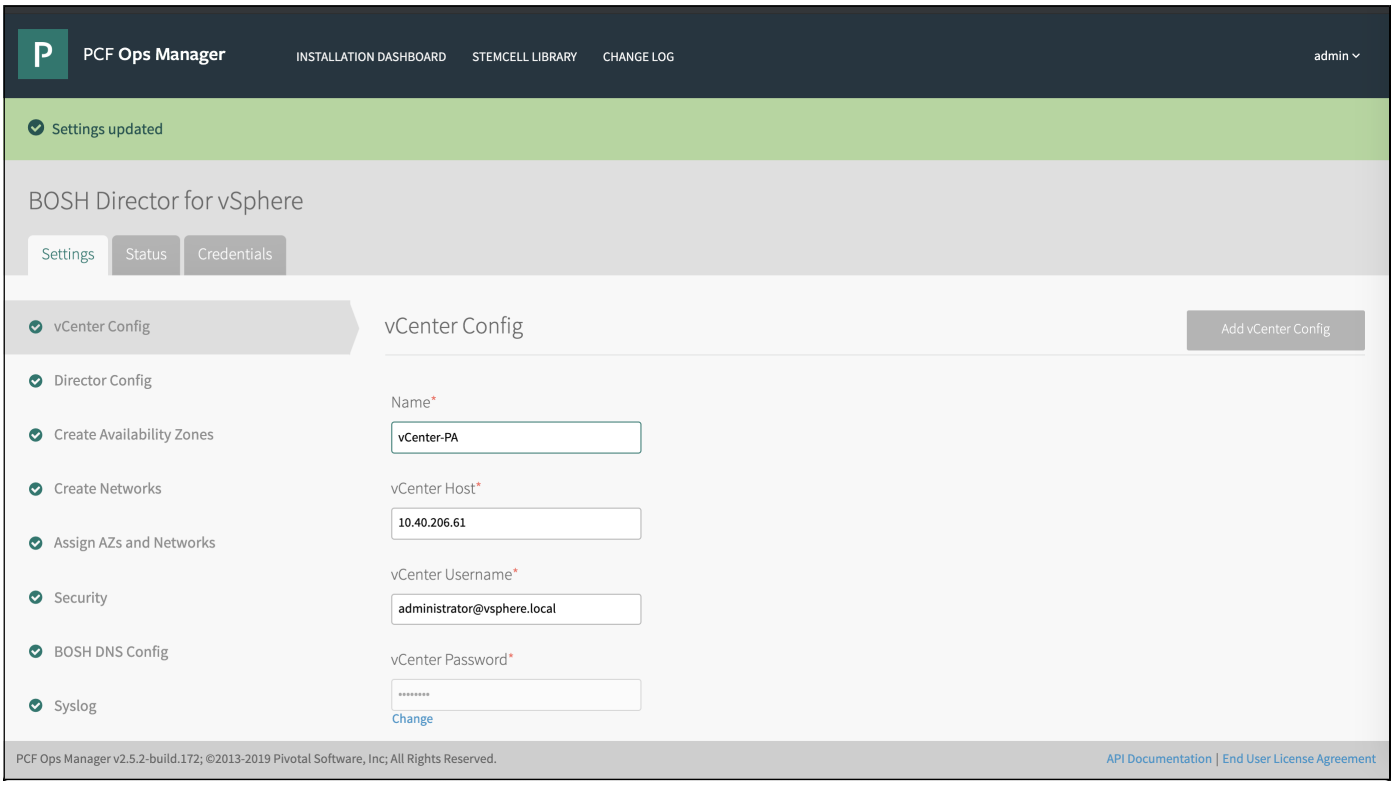
Note: The **Disable SSL certificate verification** option disables NSX Manager CA certificate validation. If you disable TLS certificate verification, unsecured mode takes precedence regardless of whether the **NSX Manager CA Cert** field is populated. Select the **Disable SSL certificate verification** option for testing purposes only.

Update the NSX-T Manager IP and Certificate for BOSH

1. Log in to Ops Manager.
2. Select the BOSH Tile.
3. Select the **vCenter config**.

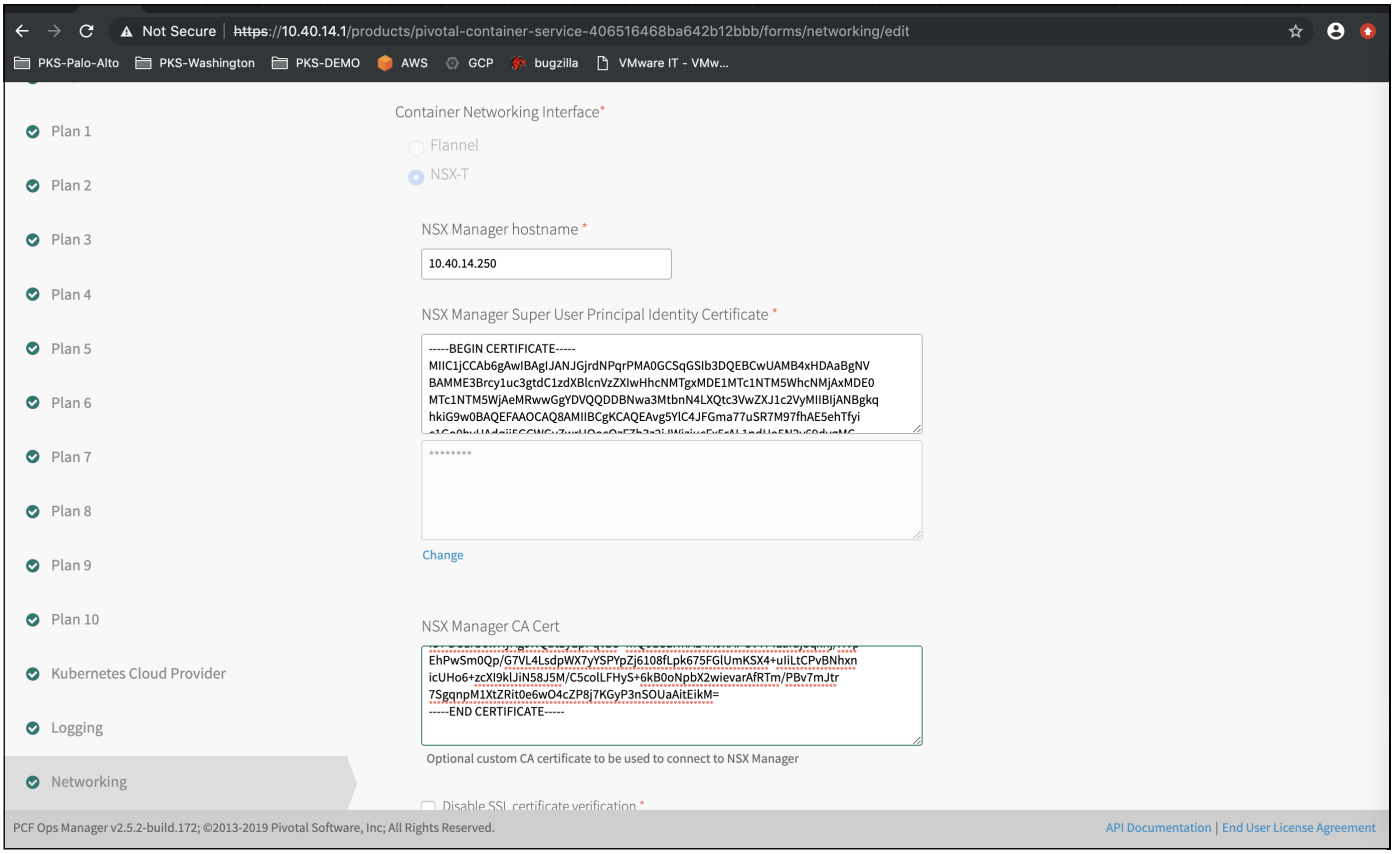


4. Update the **NSX Address** field with the load balancer VIP IP address.
5. Update the **NSX CA Cert** field with the newly generated certificate.
6. Click **Save**.

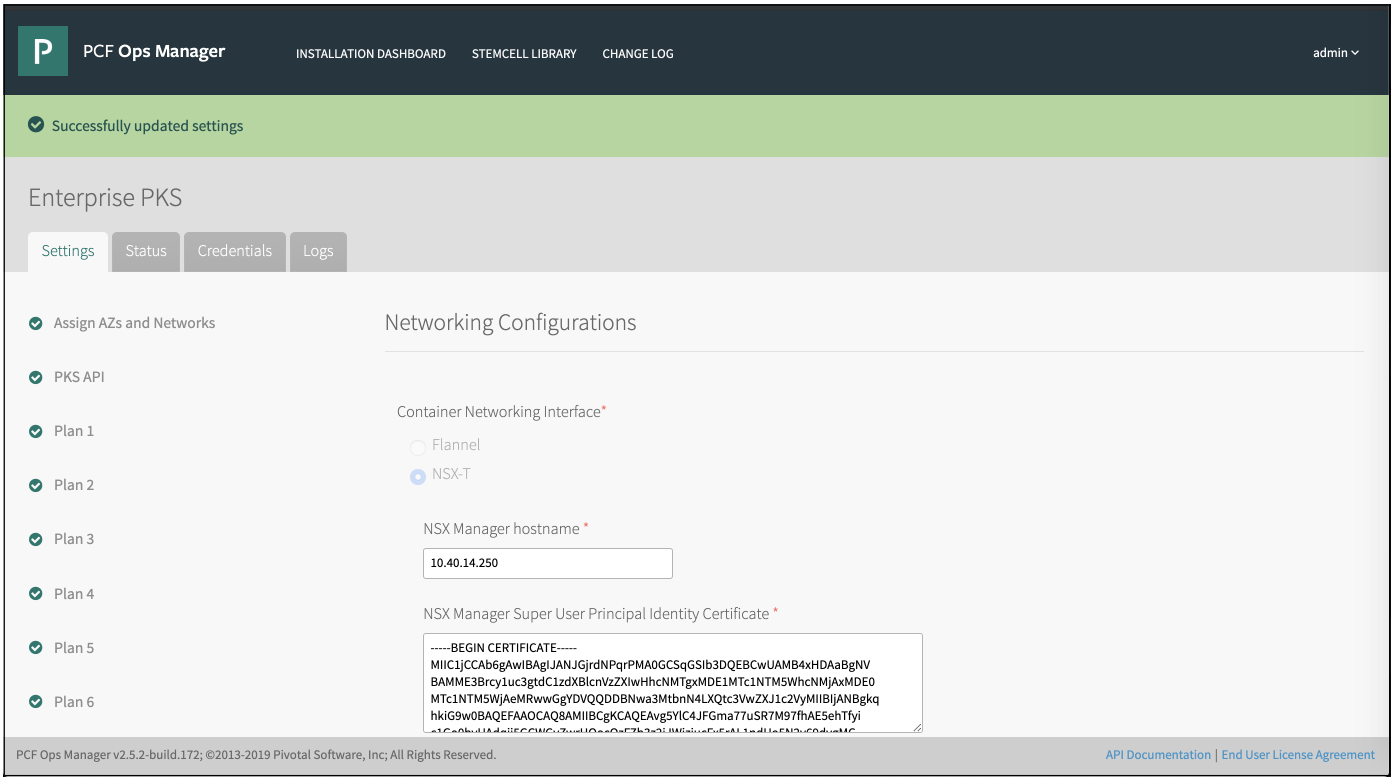


Update the NSX-T Manager IP and Certificate for PKS

1. Log in to Ops Manager.
2. Select the PKS tile.
3. Select the **Networking**.



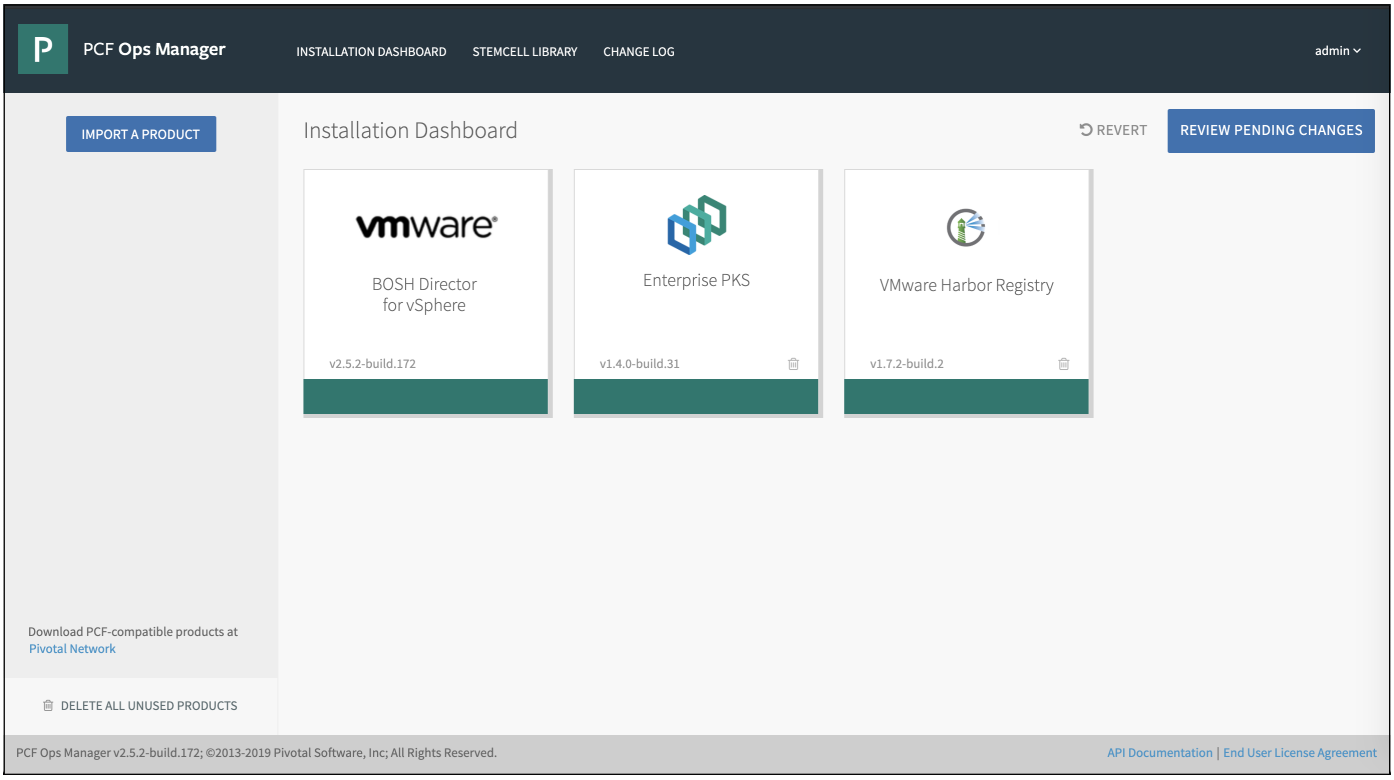
4. Update the **NSX Manager hostname** field with the load balancer VIP IP address.
5. Update the **NSX Manager CA cert** field with the newly generated certificate.
6. Click **Save**.



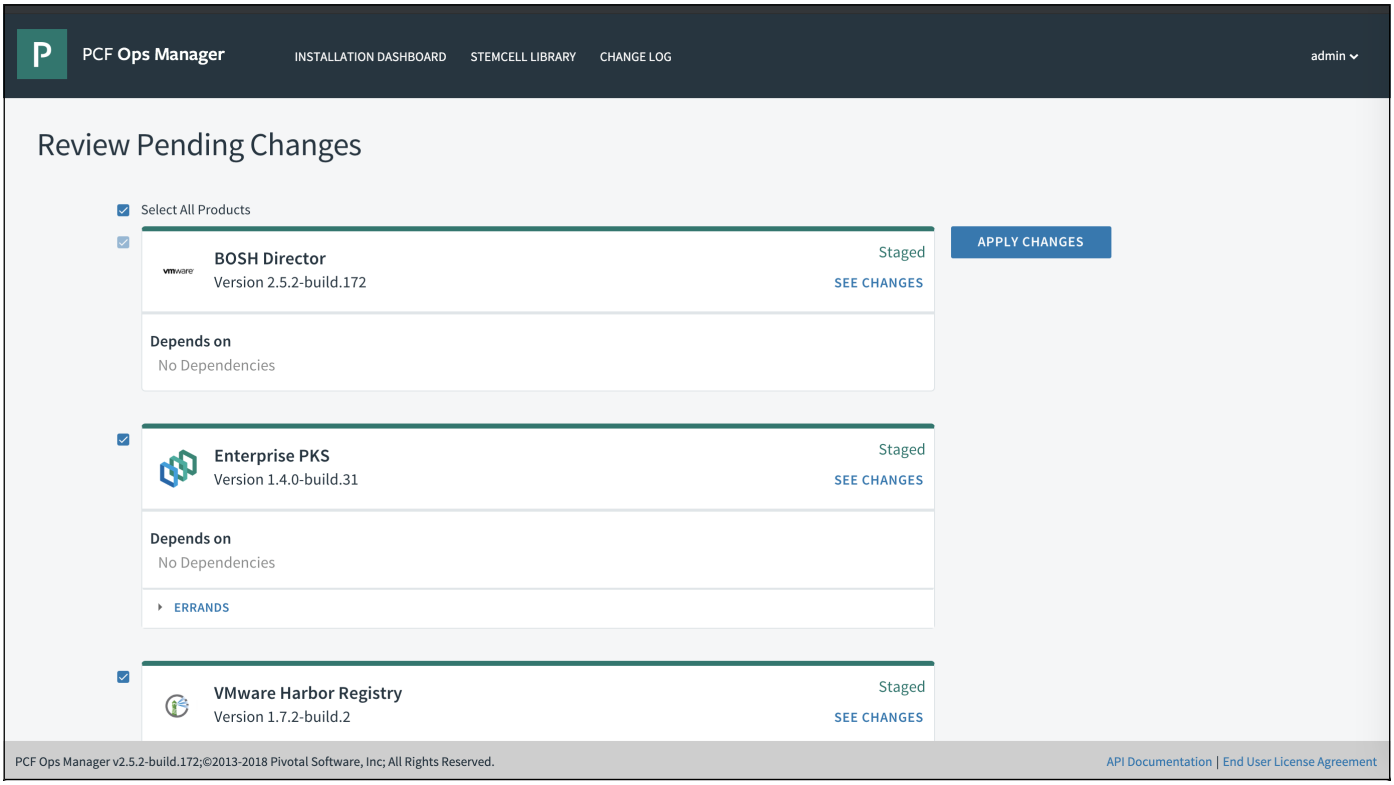
Deploy Enterprise PKS

To update the certificate IP address while upgrading Enterprise PKS, complete the following steps:

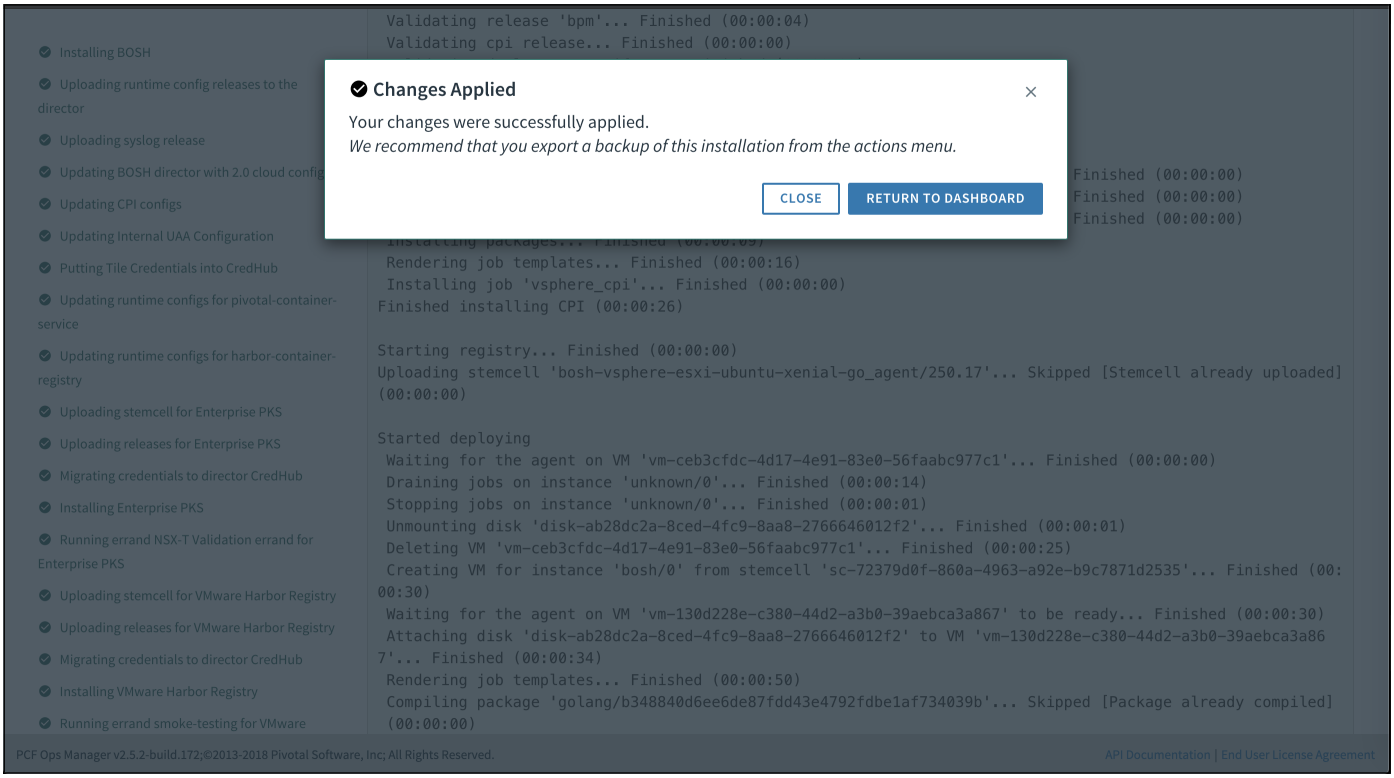
1. At the Ops Manager **Installation Dashboard**, click **Review Pending Changes**.



2. Verify that the **Update all clusters errand** is enabled for Enterprise PKS.



3. Click **Apply Changes**.



Deploying Enterprise PKS with the **Upgrade all clusters errand** selected allows NCP configurations on all Kubernetes clusters to be updated with the NSX-T Management Cluster VIP, instead of only the individual NSX-T Manager node.

Next Step

Configure BOSH Director with NSX-T for Enterprise PKS

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

Configuring BOSH Director with NSX-T for Enterprise PKS

In this topic

Page last updated:

This topic describes how to configure BOSH Director for vSphere with NSX-T integration for VMware Enterprise PKS.

Prerequisites

Before you begin this procedure, ensure that you have successfully completed all preceding steps for installing Enterprise PKS on vSphere with NSX-T, including:

- Preparing to Install Enterprise PKS on vSphere with NSX-T Data Center
- Installing and Configuring NSX-T for Enterprise PKS
- Creating the Enterprise PKS Management Plane
- Create Enterprise PKS Compute Plane
- Deploying Ops Manager with NSX-T for Enterprise PKS
- Generating and Registering the NSX Manager Certificate for Enterprise PKS

How Pivotal Platform Accesses NSX Manager

To create, delete, and modify NSX-T networking resources, Pivotal Platform tiles and APIs use a VMware NSX Manager account with the Enterprise Administrator role and permissions.

Users configure Pivotal Platform to authenticate to NSX Manager for different purposes in different tiles:

- **Enterprise PKS tile**

The Enterprise PKS tile uses NSX Manager to create load balancers, providing a Kubernetes service described in the Create an External Load Balancer section of the Kubernetes documentation.

To configure the **Enterprise PKS** tile's authentication to NSX Manager, see the topic Generating and Registering the NSX Manager Superuser Principal Identity Certificate and Key.

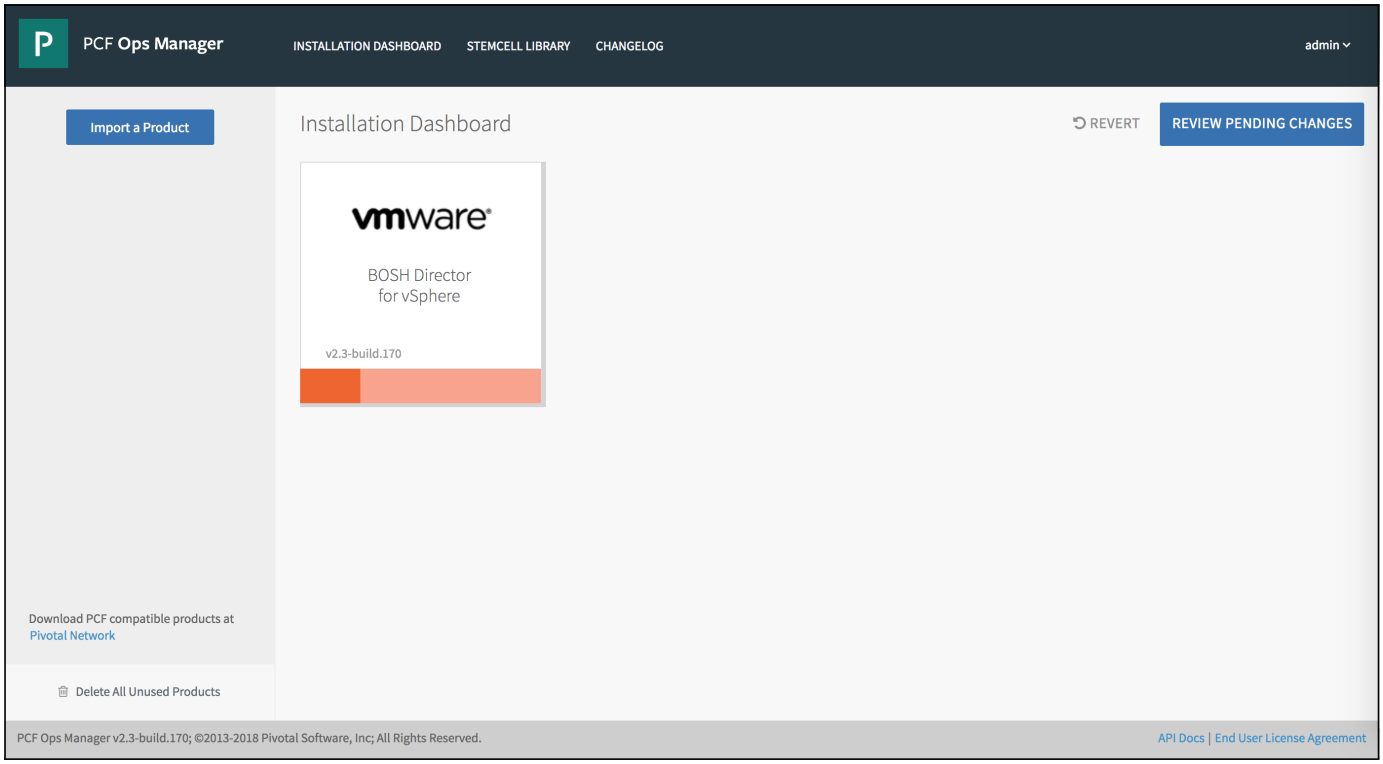
- **BOSH Director for vSphere tile**
  The **BOSH Director for vSphere** tile uses NSX Manager to configure networking and security for external-facing Pivotal Platform component VMs, such as Pivotal Application Service routers.

  To configure the **BOSH Director for vSphere** tile's authentication to NSX Manager, see Configure vCenter for Enterprise PKS, below.

Step 1: Log in to Ops Manager

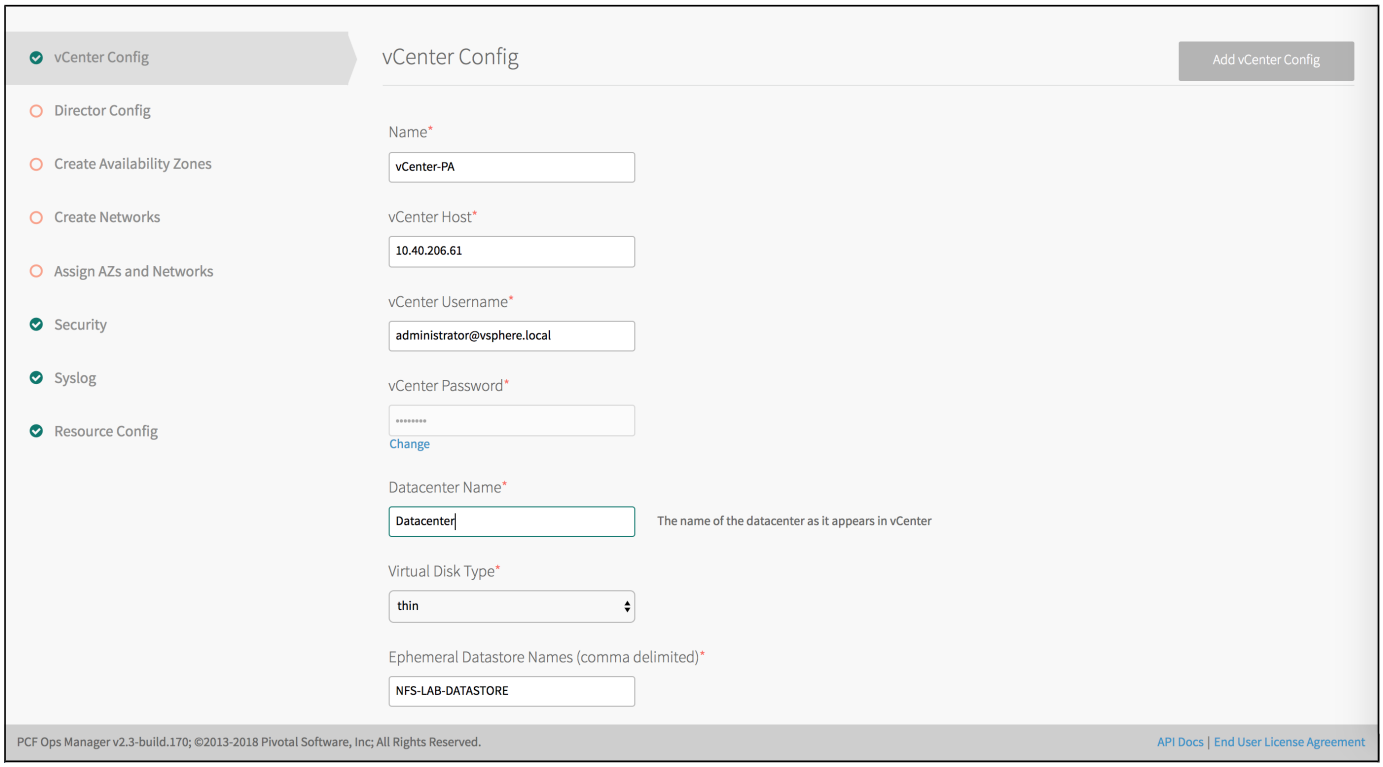
1. Log in to Ops Manager with the username and password credentials that you set up in Configure Ops Manager for Enterprise PKS.

2. Click the **BOSH Director for vSphere** tile.



Step 2: Configure vCenter for Enterprise PKS

1. Select **vCenter Config**.

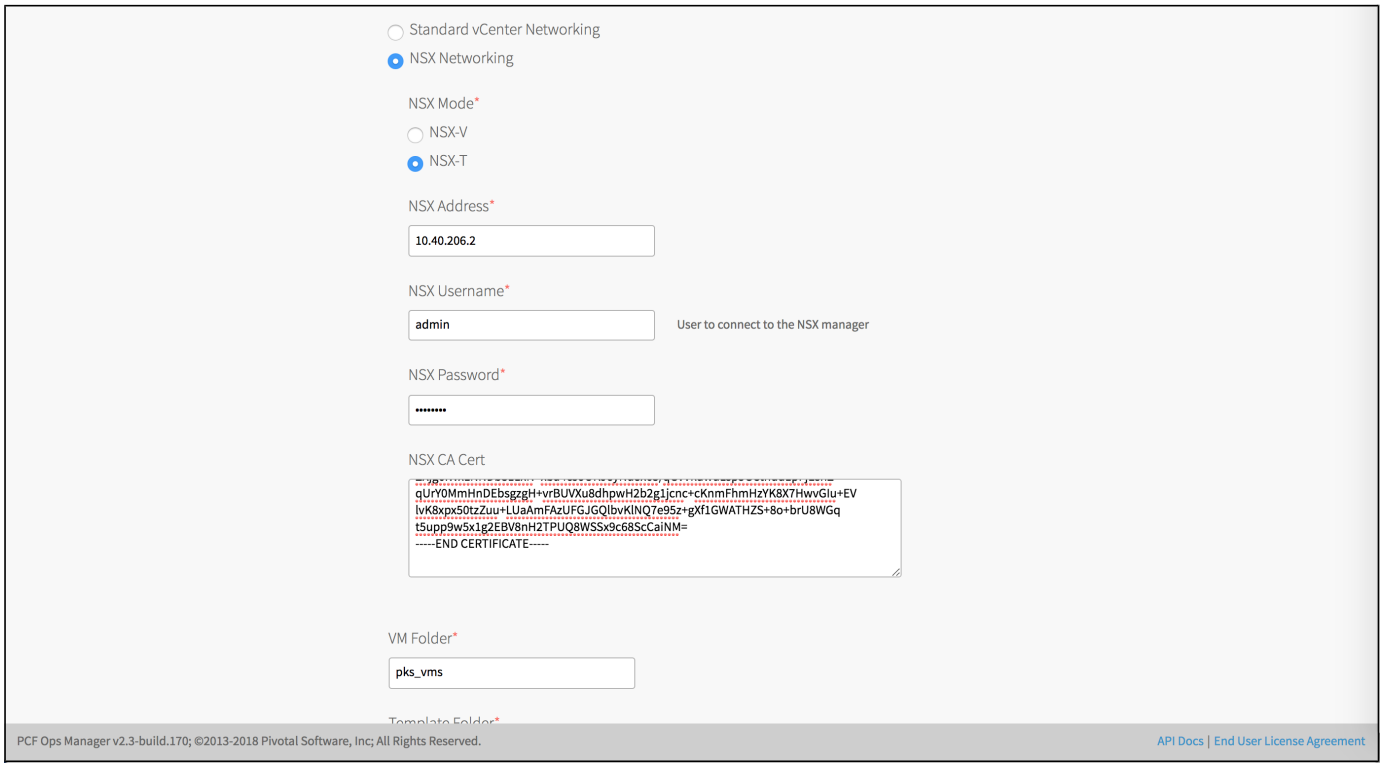


2. Enter the following information:

   - **Name**: A name that you provide for your vCenter configuration. This field is used to identify the datacenter configuration in Ops Manager if you are configuring multiple datacenters.
   - **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 vSphere Virtual Disk Types.
   - **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.

> **Note**: The vSphere datastore type must be Datastore. Enterprise PKS does not support the use of vSphere Datastore Clusters with or without Storage DRS. For more information, see Datastores and Datastore Clusters in the vSphere documentation.

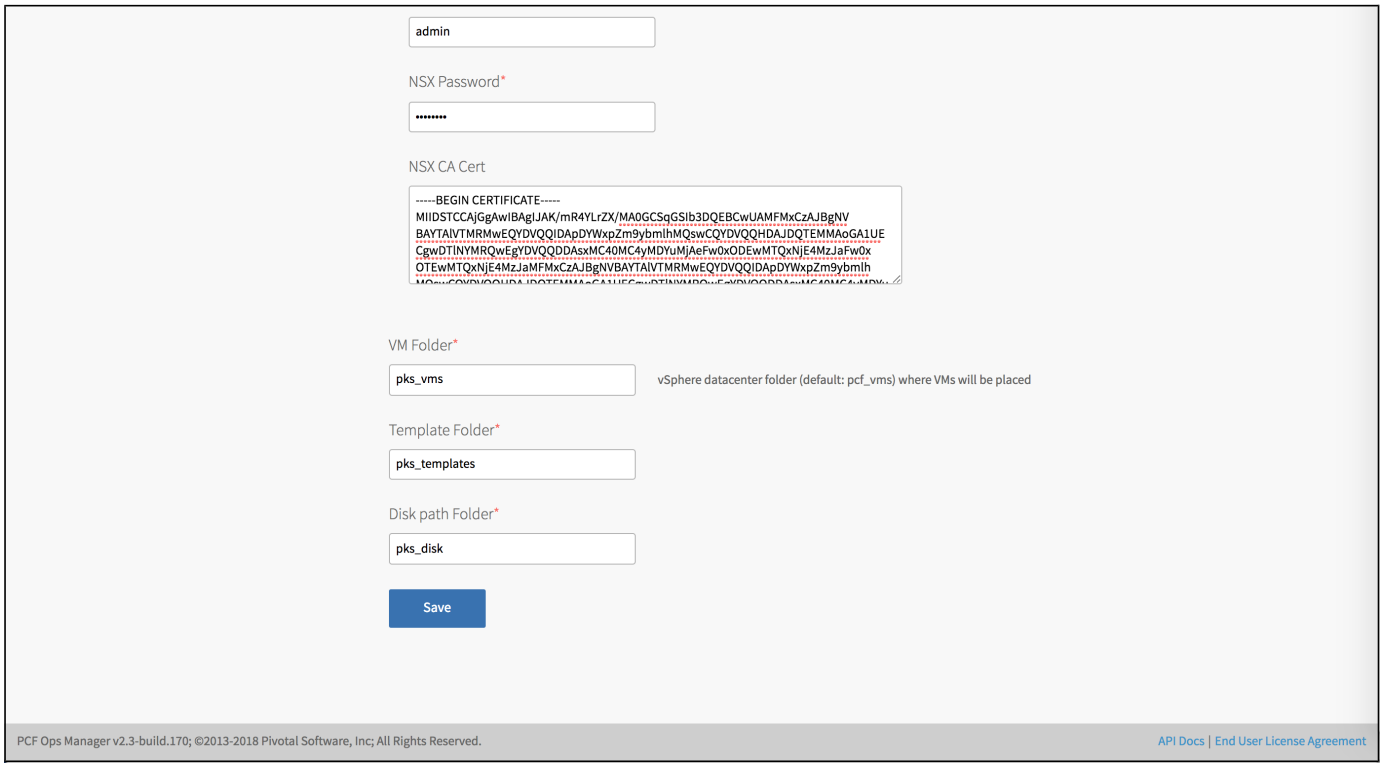
3. Select **NSX Networking**, then select **NSX-T**.



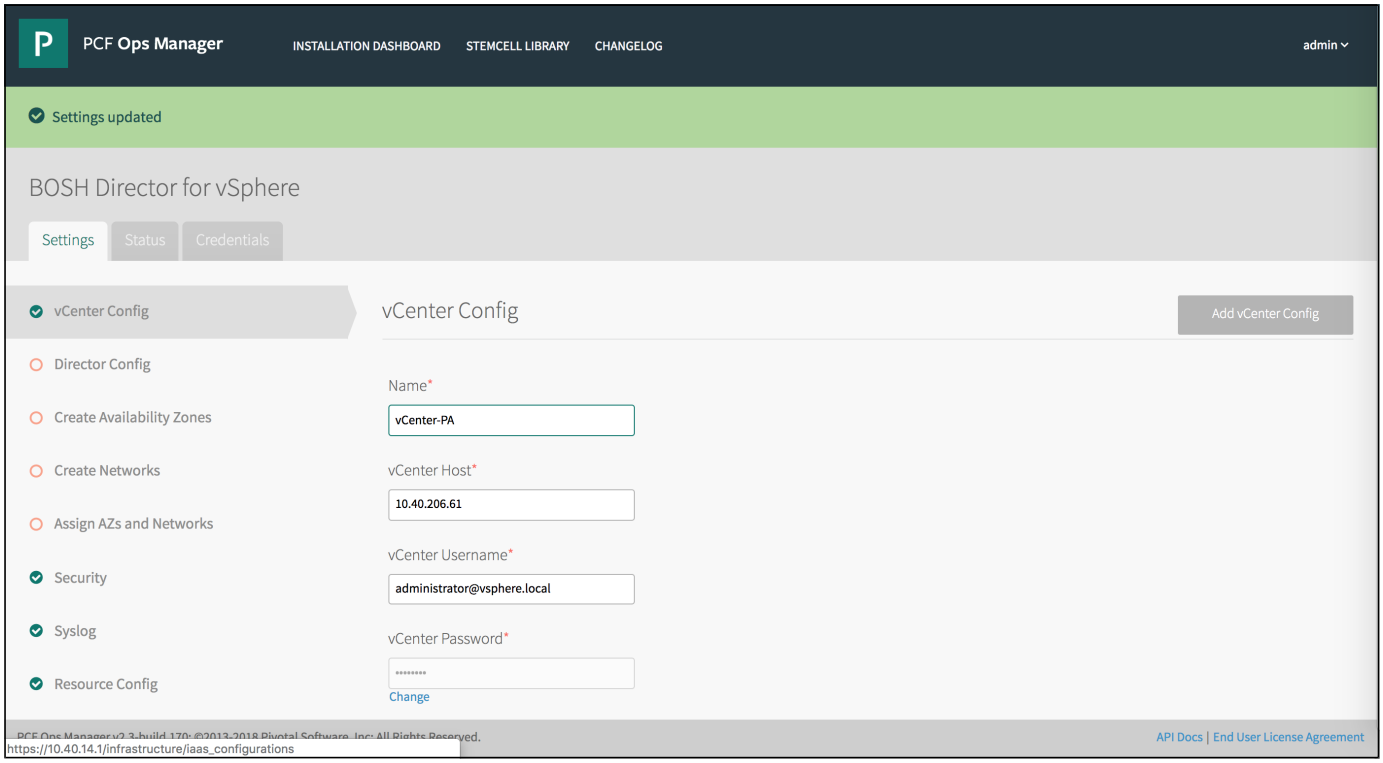
4. Configure NSX-T networking as follows:
   - **NSX Address**: Enter the IP address of the NSX Manager host.
   - **NSX Username** and **NSX Password**: Enter the NSX Manager username and password.
   - **NSX CA Cert**: Provide the CA certificate in PEM format that authenticates to the NSX server. Open 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.

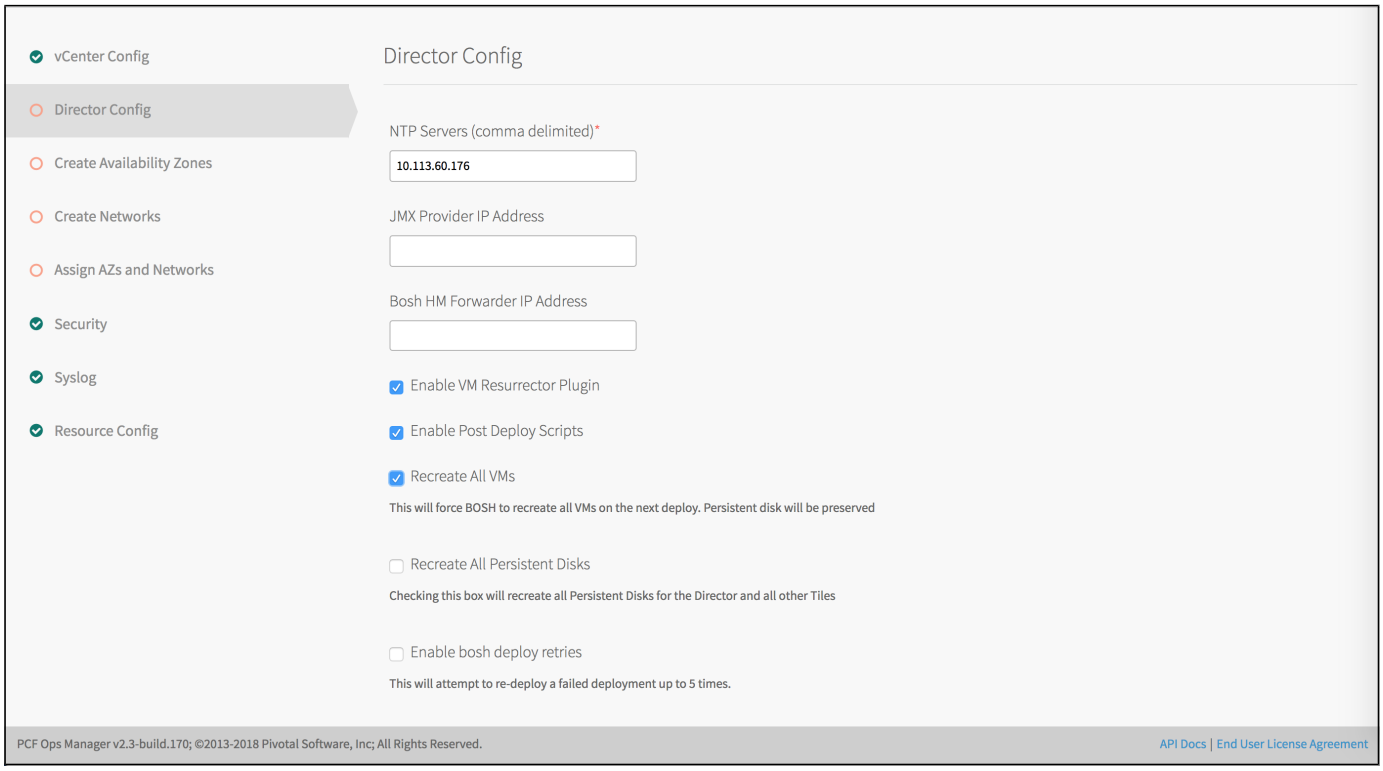


6. Click **Save**.



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.

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

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.

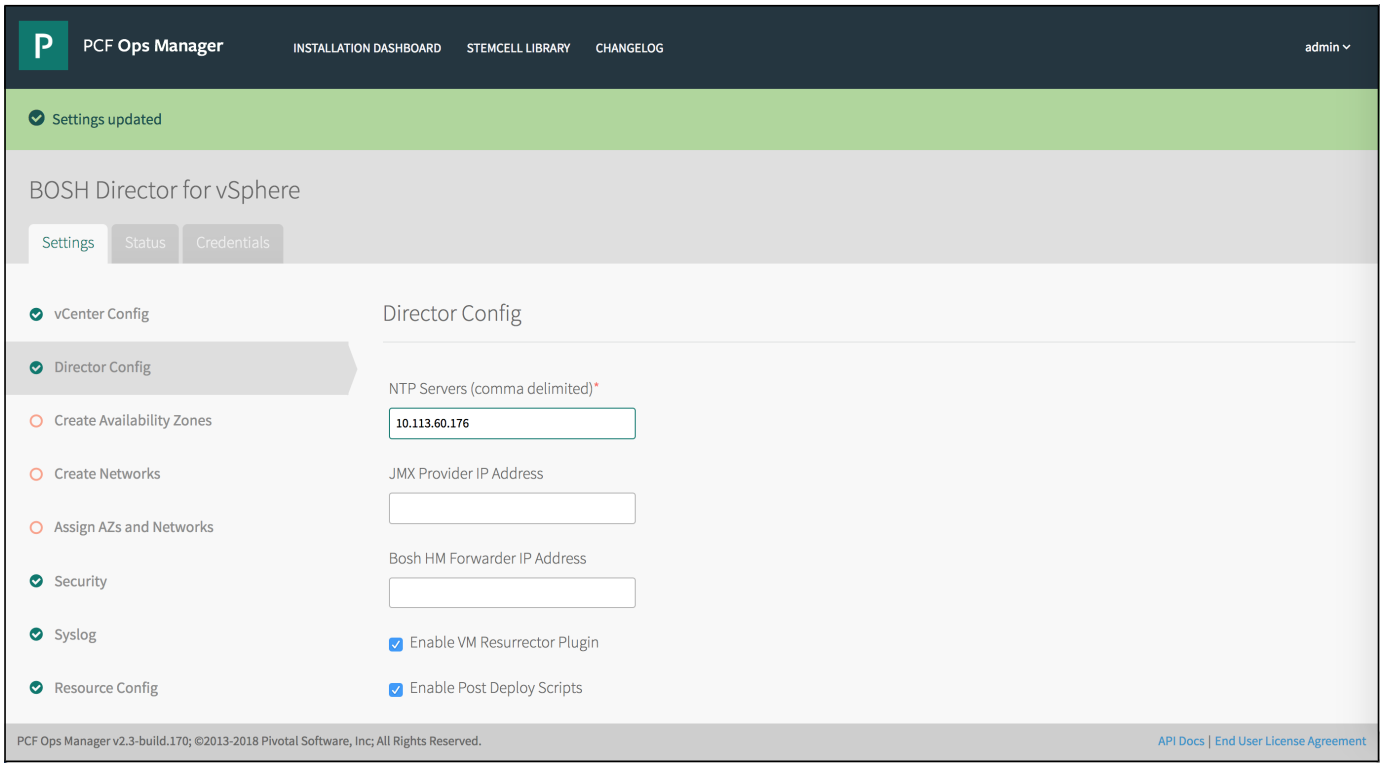
> **Note**: You must enable post-deploy scripts to install Enterprise PKS.

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 Enterprise 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 Pane in *Configuring BOSH Director on vSphere* in the Pivotal Platform 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 Enterprise PKS with NSX-T in a NAT topology, you must provide the **Director Hostname** for BOSH at the time of installation. See Director Config Pane in *Configuring BOSH Director on vSphere* in the Pivotal Platform documentation for details.

9. Click **Save**.



Step 4: Create Availability Zones

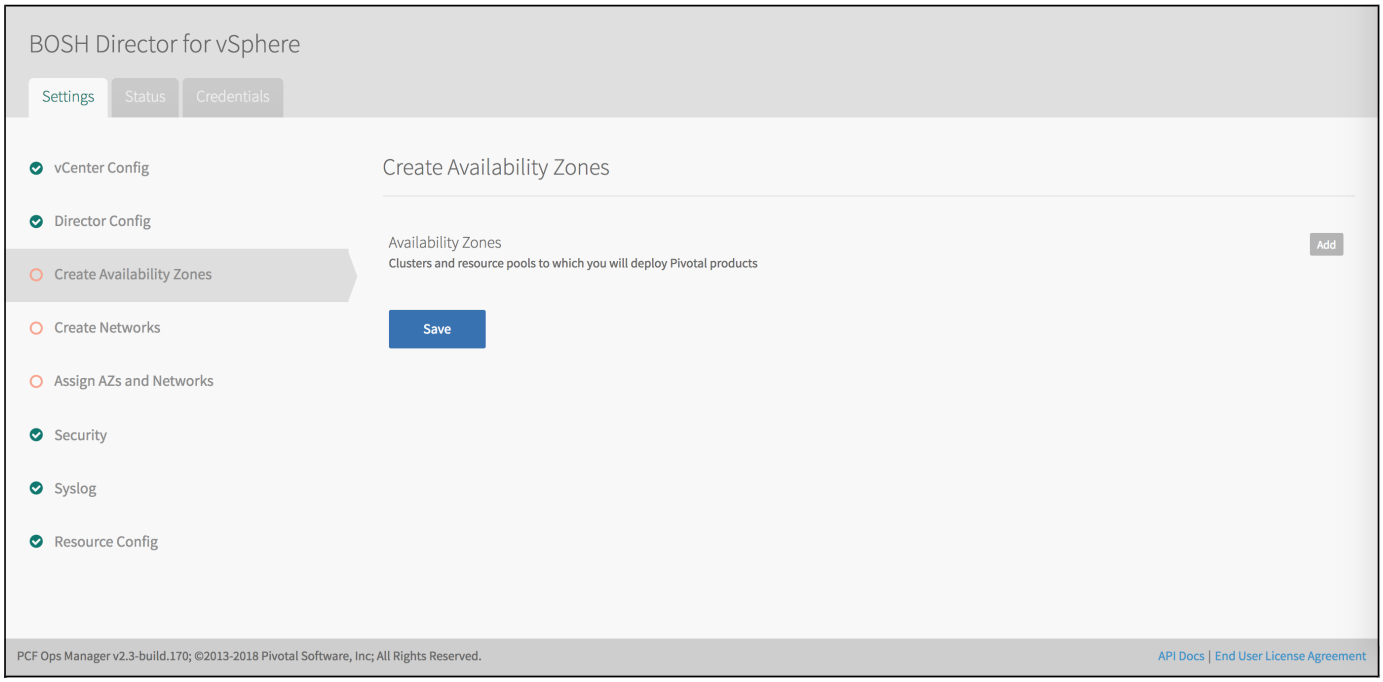
Ops Manager Availability Zones correspond to your vCenter clusters and resource pools.

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.

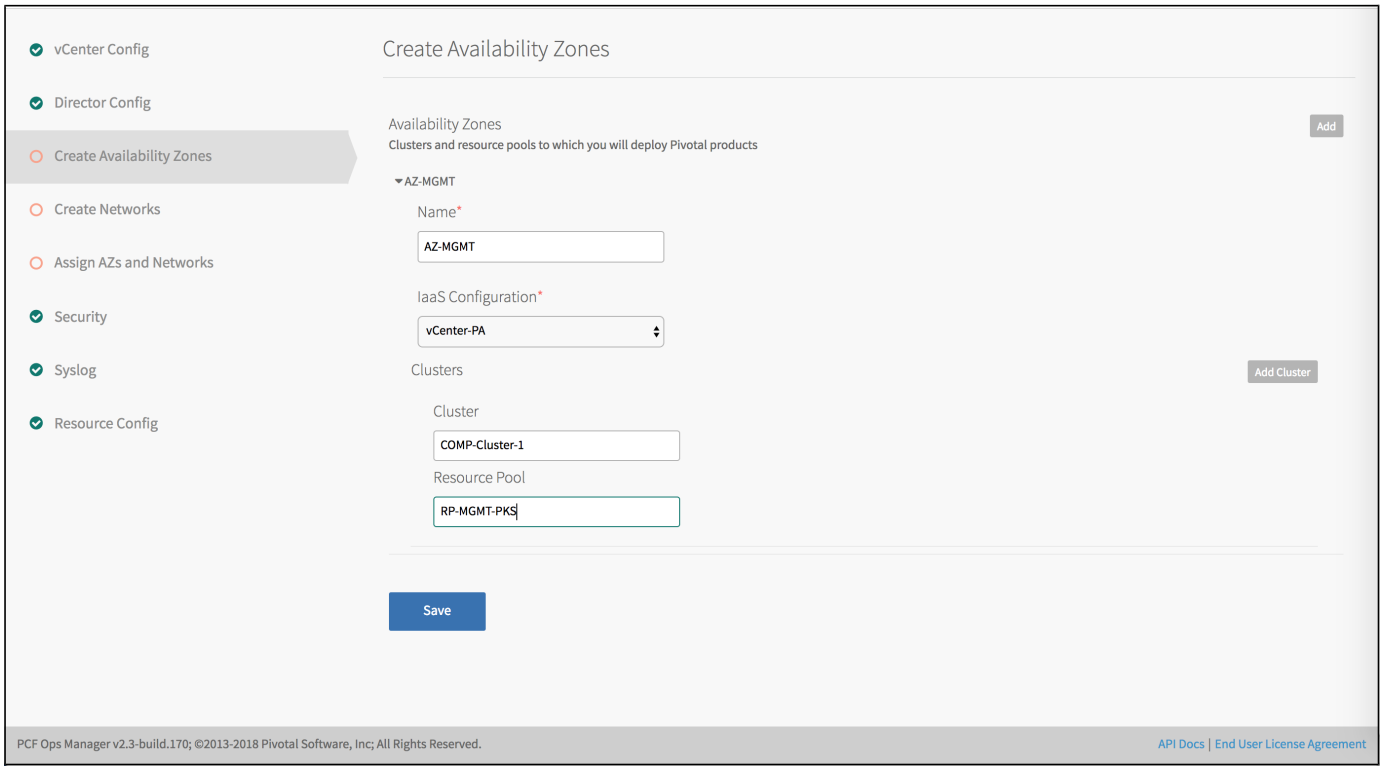
For a highly available installation of your chosen runtime, use at least three availability zones.

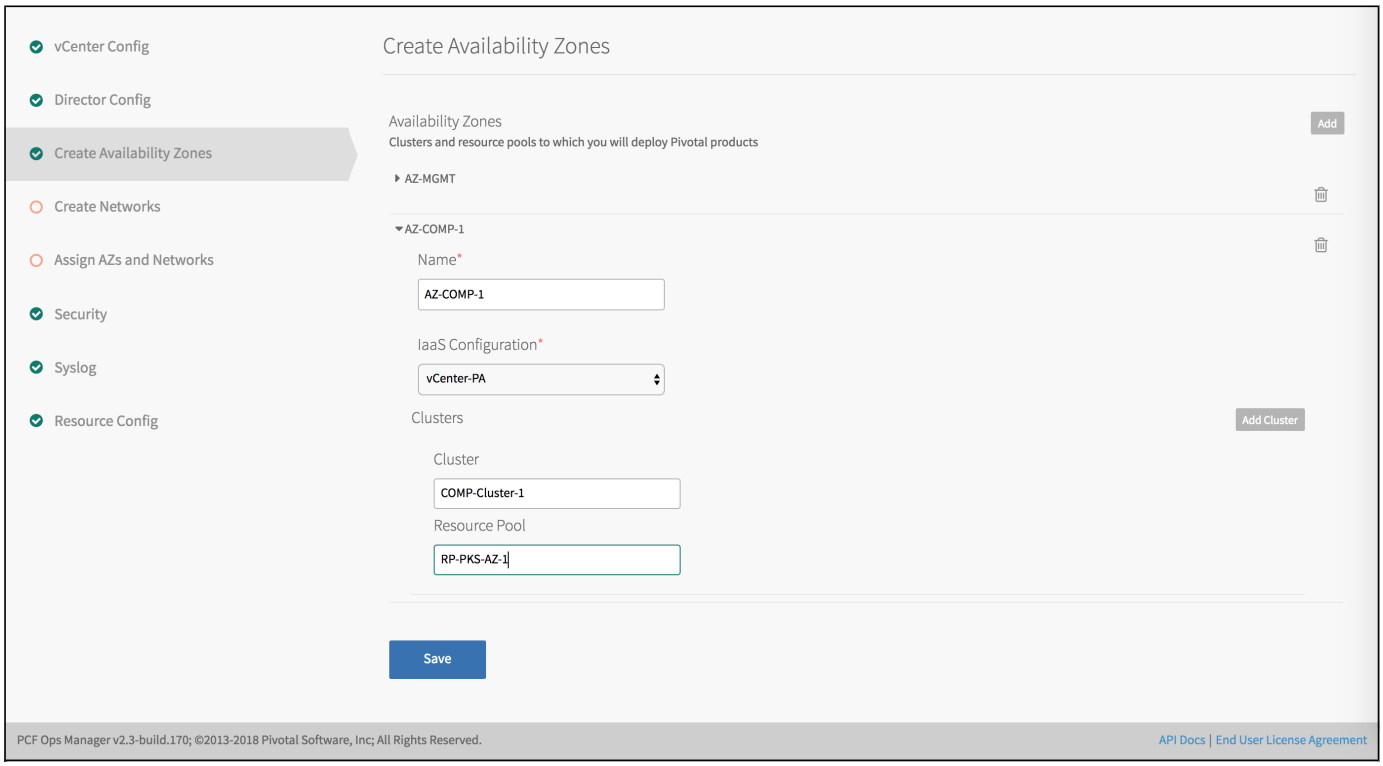
> **Note**: For more information about AZs and high availability in vSphere, see Compute and HA Considerations in *vSphere Reference Architecture*.

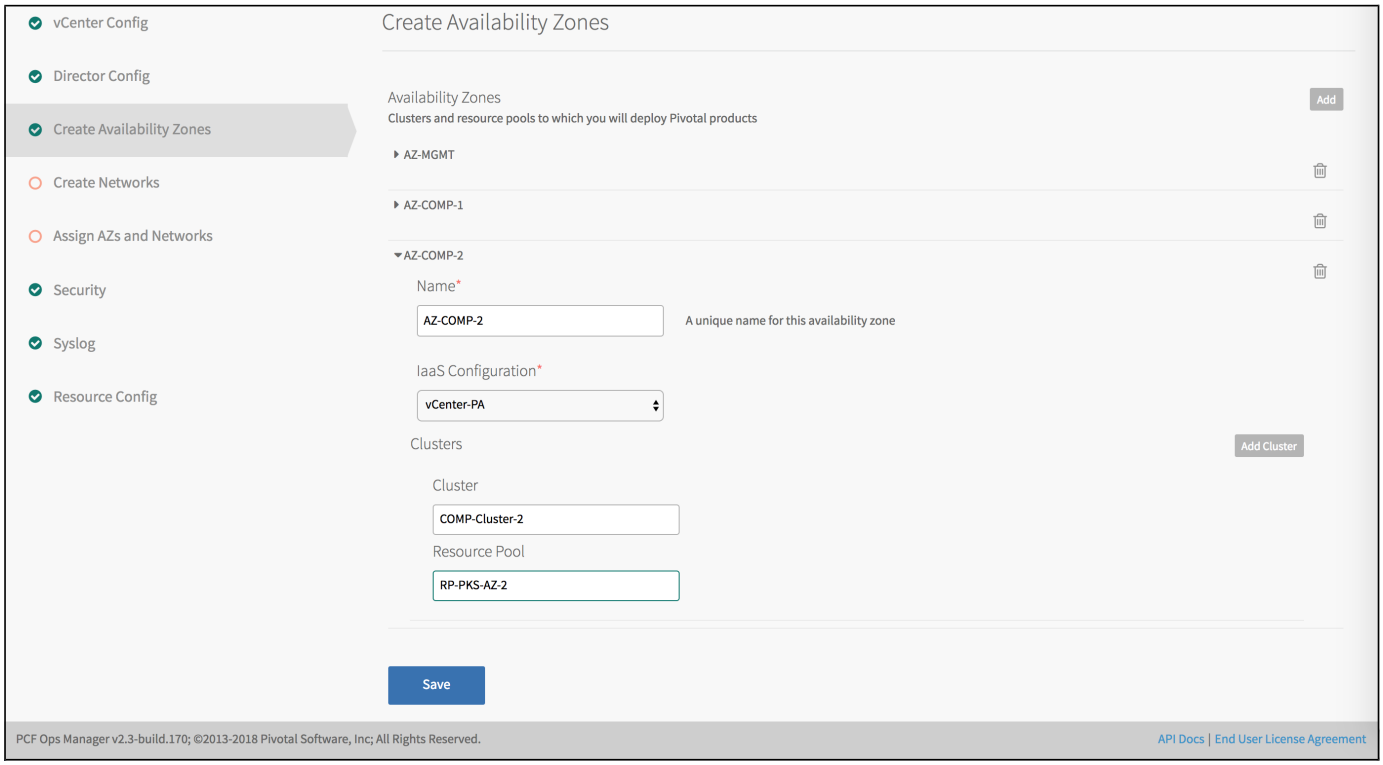
1. Select **Create Availability Zones**.

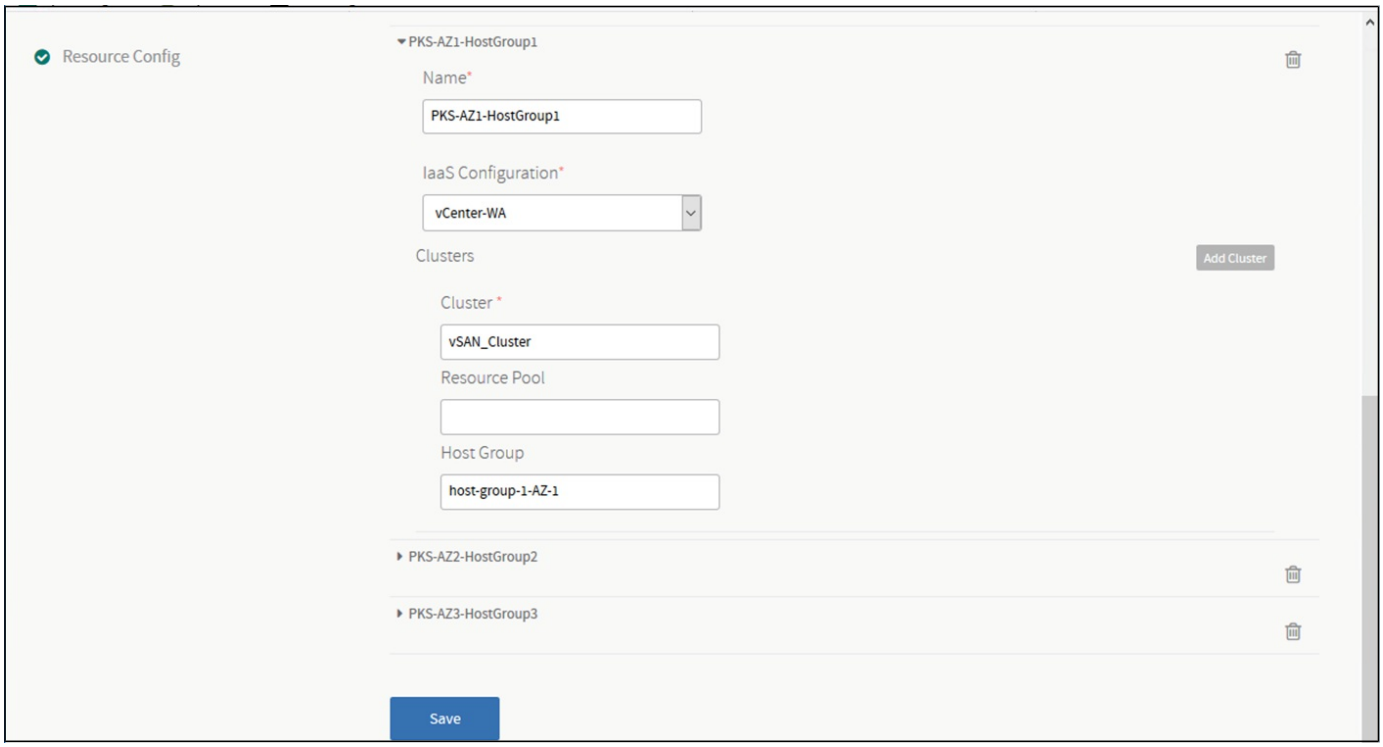


2. Use the following steps to create one or more Availability Zones for Enterprise PKS to use:
   - Click **Add** and create the Enterprise PKS Management AZ.
   - Enter a unique **Name** for the Availability Zone, such as `AZ-MGMT`.
   - Select the IaaS configuration (vSphere/vCenter).
   - Enter the name of an existing vCenter **Cluster** to use as an Availability Zone, such as `COMP-Cluster-1`.
   - Enter the name of the Enterprise 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 Enterprise PKS Compute AZ.
   - Specify the **Cluster** and the **Resource Pool**, such as `RP-PKS-AZ`. Alternatively, specify the **Cluster** and the **Host Group**. See Using vSphere Host Group for more information.
   - Add additional clusters as necessary. Click the trash icon to delete a cluster. The first cluster cannot be deleted.

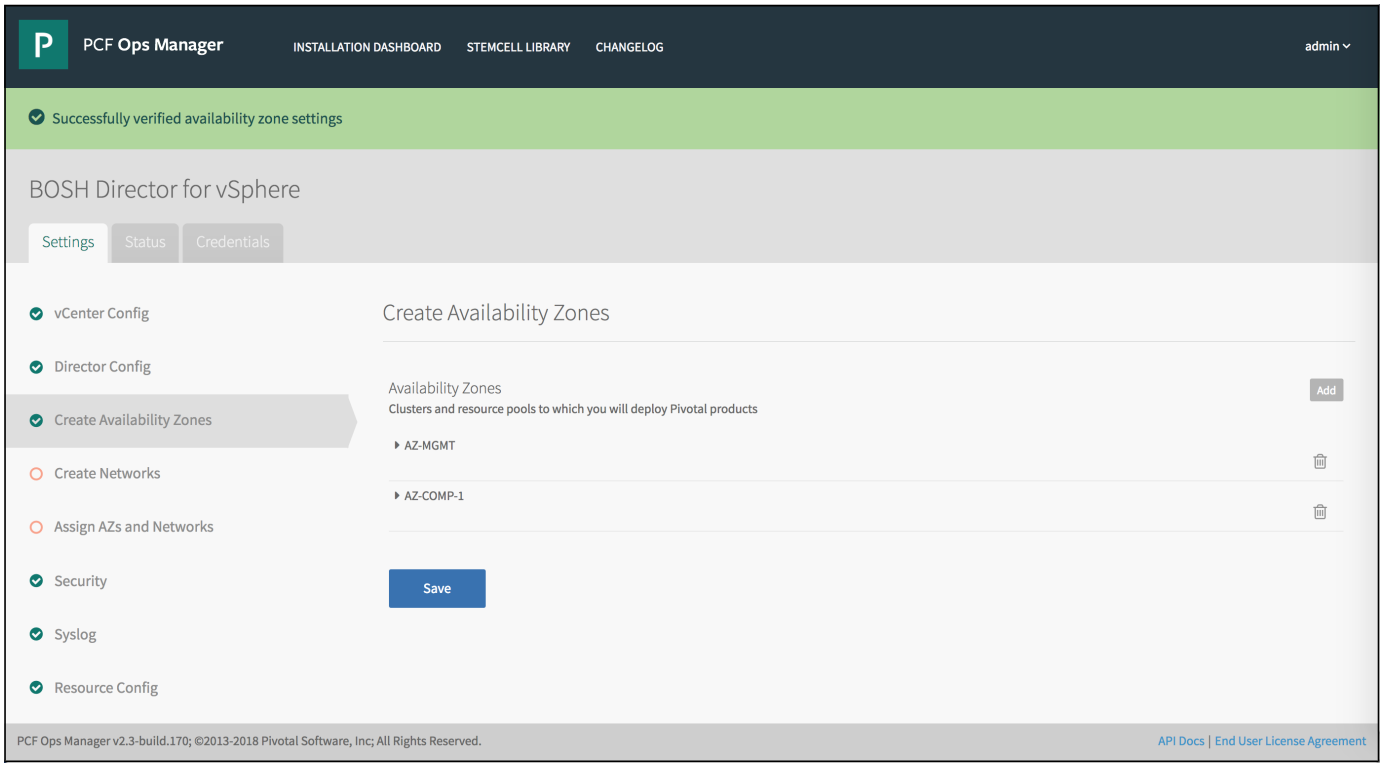

vCenter Config
Director Config
Create Availability Zones
Create Networks
Assign AZs and Networks
Security
Syslog
Resource Config
Create Availability Zones
Availability Zones
Clusters and resource pools to which you will deploy Pivotal products
Add
AZ-MGMT
Name*
AZ-MGMT
IaaS Configuration*
vCenter-PA
Clusters
Add Cluster
Cluster
COMP-Cluster-1
Resource Pool
RP-MGMT-PKS
Save
PCF Ops Manager v2.3-build.170; ©2013-2018 Pivotal Software, Inc; All Rights Reserved.
API Docs | End User License Agreement


vCenter Config
Director Config
Create Availability Zones
Create Networks
Assign AZs and Networks
Security
Syslog
Resource Config
Create Availability Zones
Availability Zones
Clusters and resource pools to which you will deploy Pivotal products
Add
AZ-MGMT
AZ-COMP-1
Name*
AZ-COMP-1
IaaS Configuration*
vCenter-PA
Clusters
Add Cluster
Cluster
COMP-Cluster-1
Resource Pool
RP-PKS-AZ-1
Save
PCF Ops Manager v2.3-build.170; ©2013-2018 Pivotal Software, Inc; All Rights Reserved.
API Docs | End User License Agreement


vCenter Config
Director Config
Create Availability Zones
Create Networks
Assign AZs and Networks
Security
Syslog
Resource Config
Create Availability Zones
Availability Zones
Clusters and resource pools to which you will deploy Pivotal products
Add
AZ-MGMT
AZ-COMP-1
AZ-COMP-2
Name*
AZ-COMP-2
A unique name for this availability zone
IaaS Configuration*
vCenter-PA
Clusters
Add Cluster
Cluster
COMP-Cluster-2
Resource Pool
RP-PKS-AZ-2
Save
PCF Ops Manager v2.3-build.170; ©2013-2018 Pivotal Software, Inc; All Rights Reserved.
API Docs | End User License Agreement

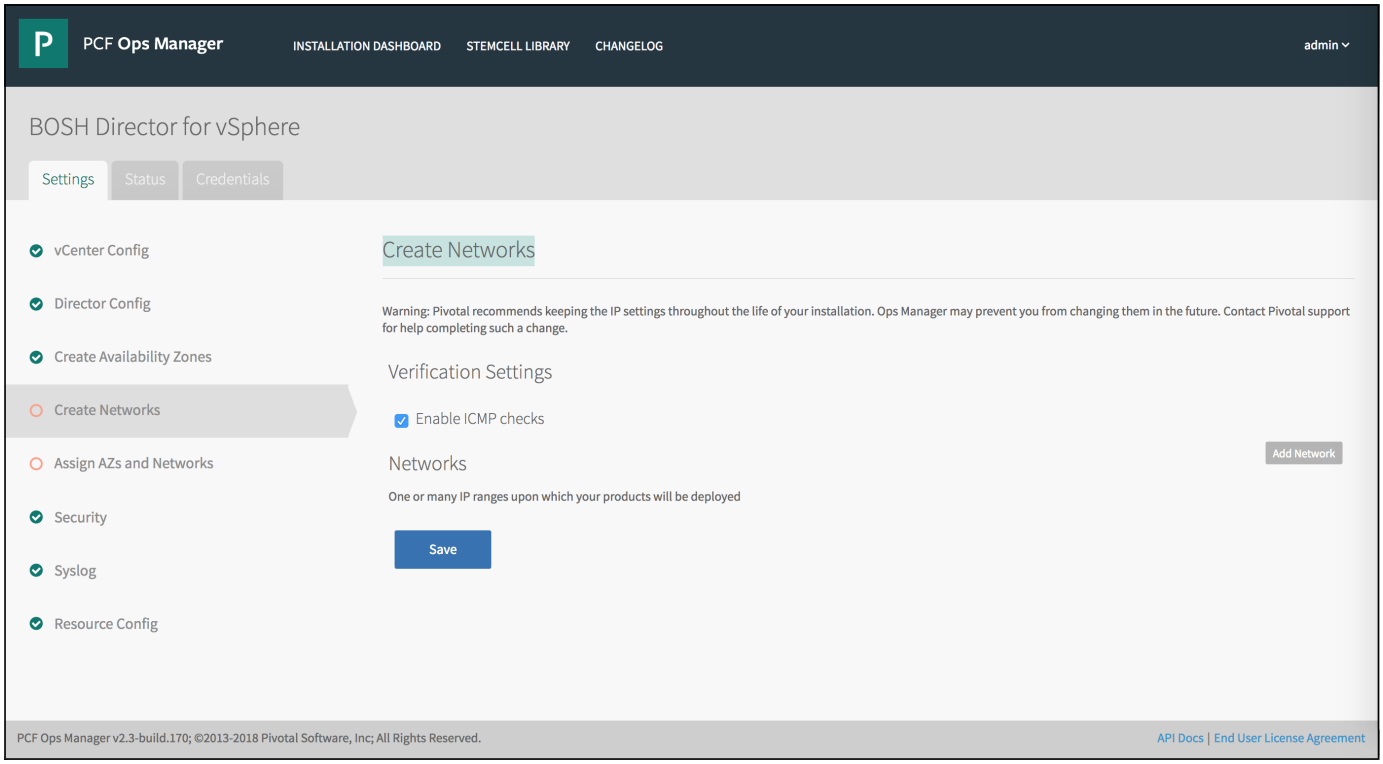

Resource Config
PKS-AZ1-HostGroup1
Name*
PKS-AZ1-HostGroup1
IaaS Configuration*
vCenter-WA
Clusters
Add Cluster
Cluster *
vSAN_Cluster
Resource Pool
Host Group
host-group-1-AZ-1
PKS-AZ2-HostGroup2
PKS-AZ3-HostGroup3
Save

3. Click **Save**.



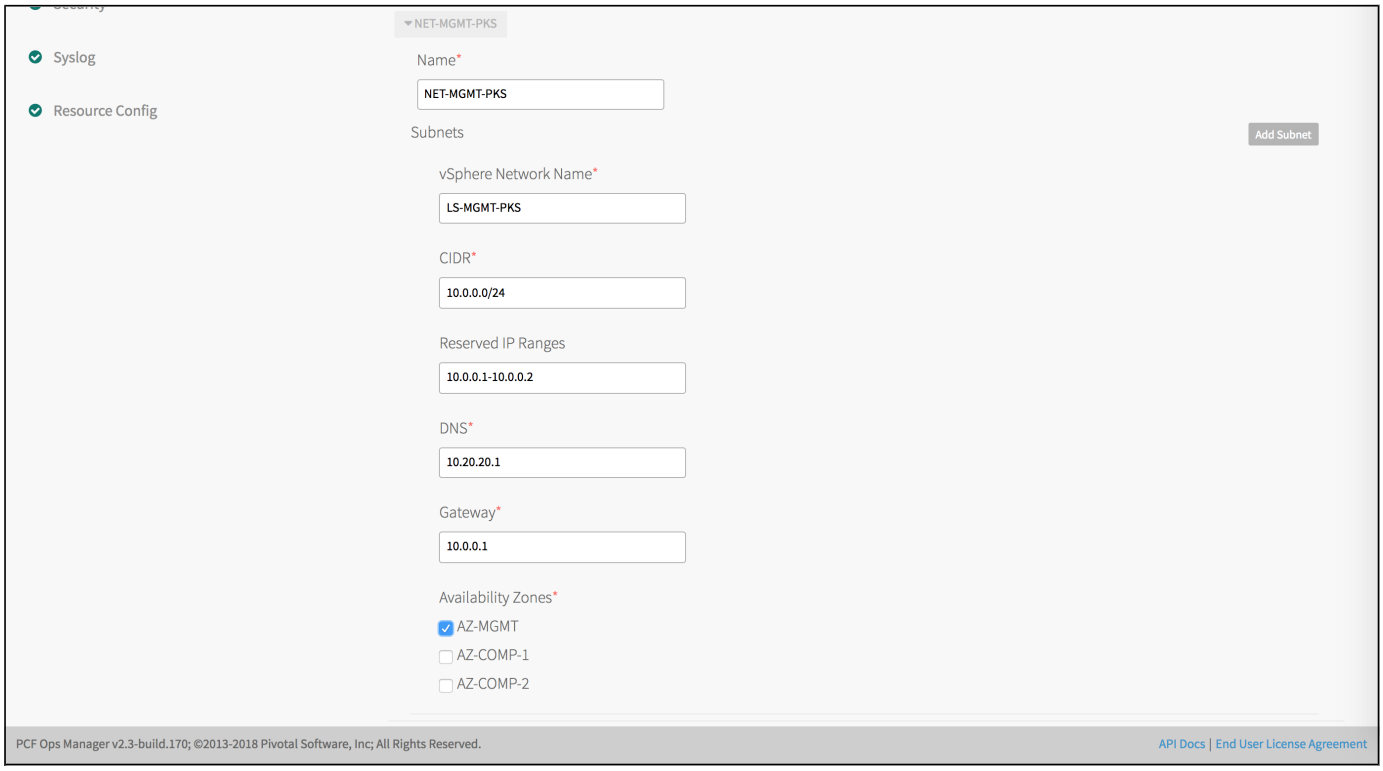
Step 5: Create Networks

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 Enterprise PKS Management Network. See Creating Enterprise 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 Enterprise PKS. You should not manually create this network.

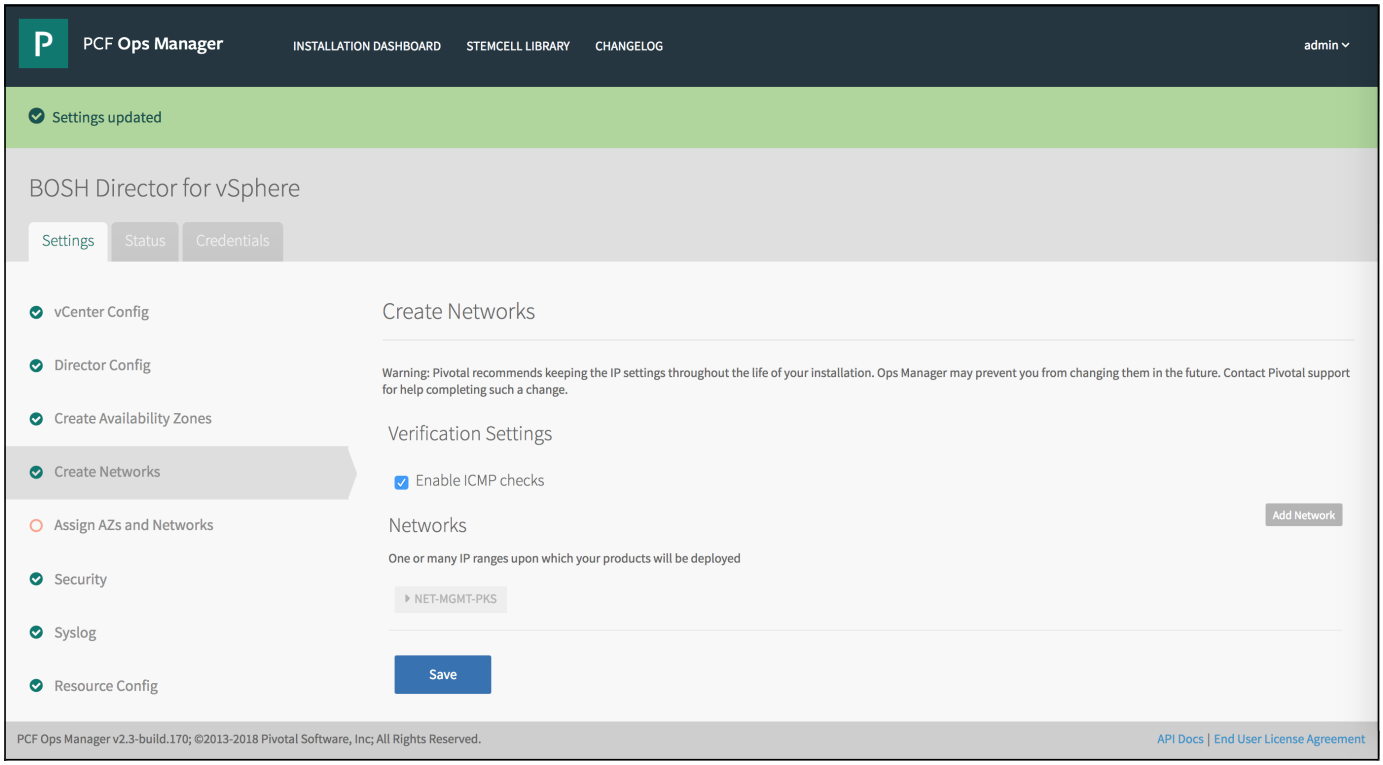
Use the following values as a guide when you define the network in BOSH. Replace the IP addresses with ranges you defined for the Enterprise PKS Management Network.. Reserve any IP addresses from the subnet that are already in use, such as the IP for Ops Manager and subnet gateway.

	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`

5. Select the **AZ-MGMT** Availability Zone to use with the `NET-MGMT-PKS` network.

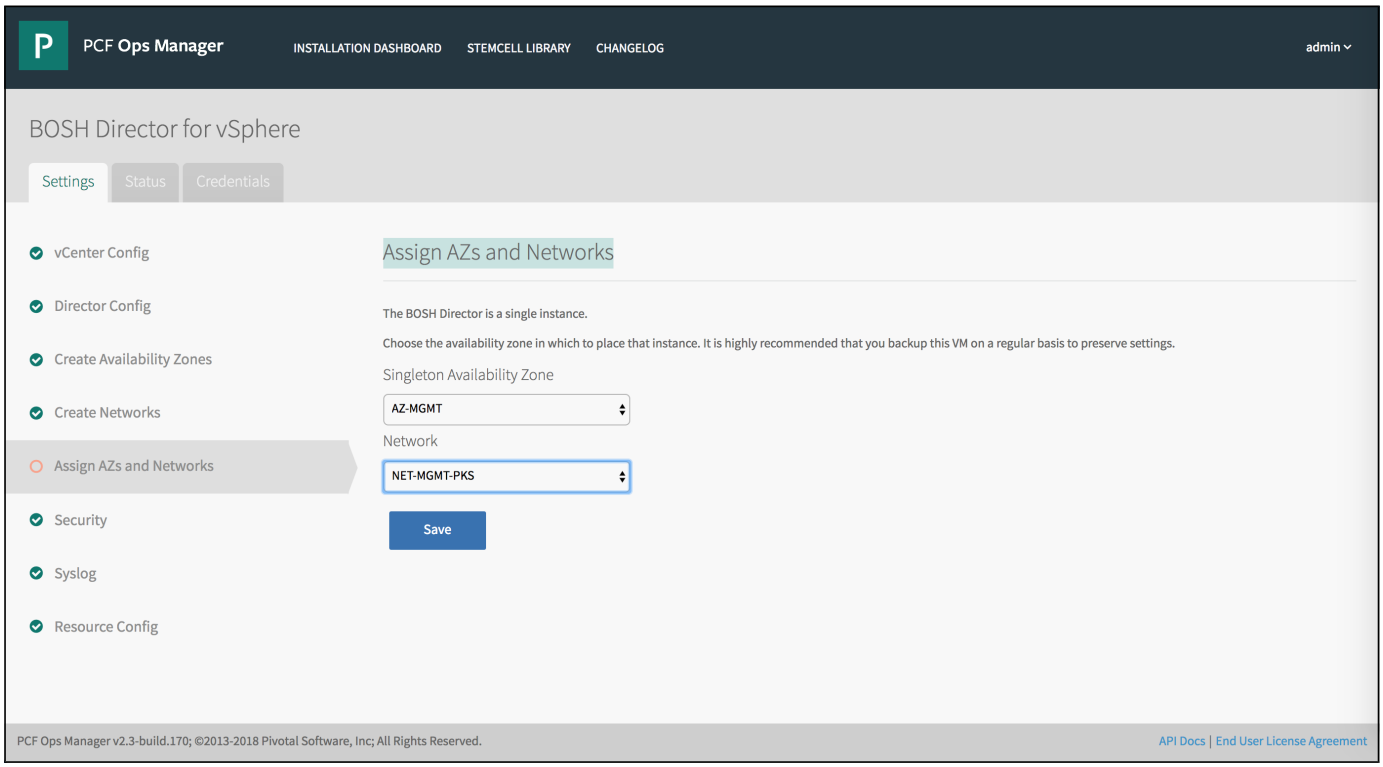
> **Note**: Do not select the COMPUTE network at this point in the configuration. It will be configured at the end of the procedure.

6. Click **Save**.



Step 6: Assign AZs and Networks

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

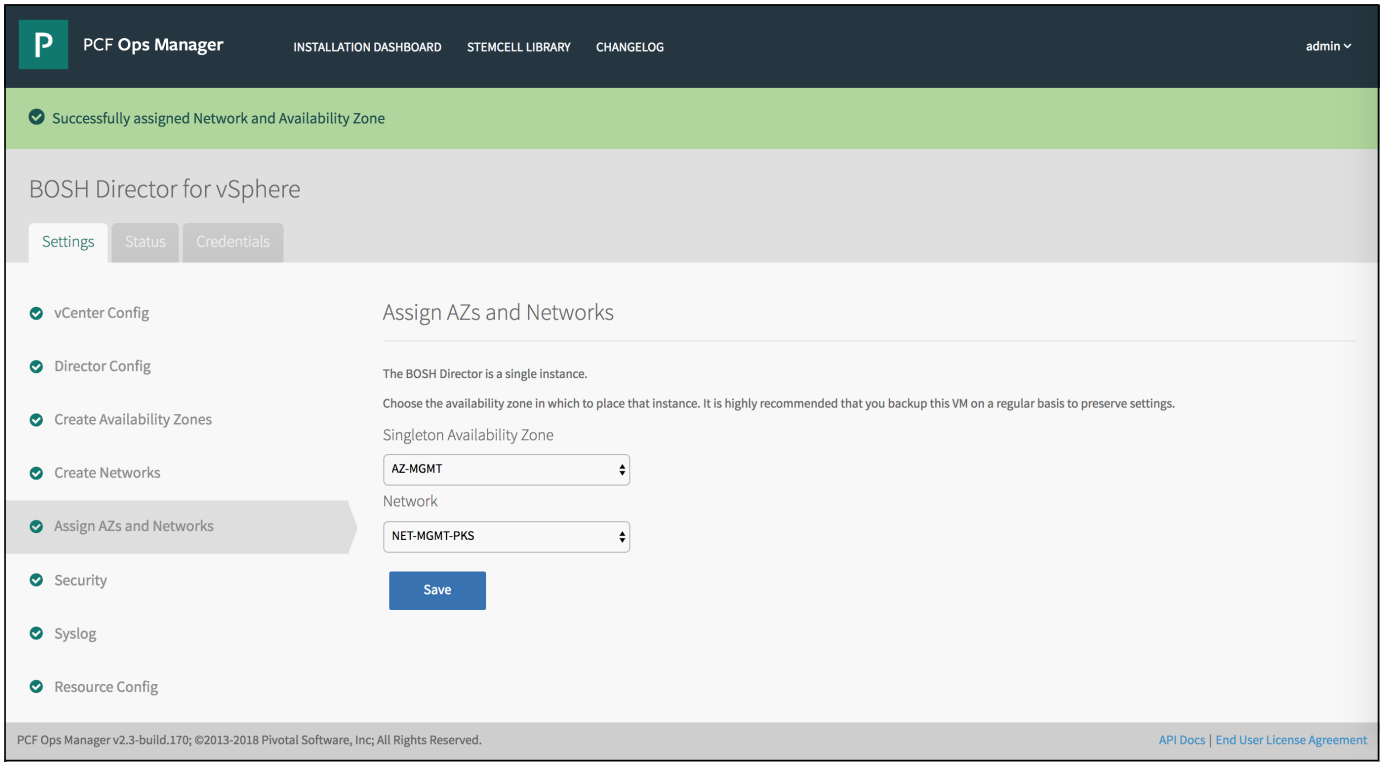


2. Use the drop-down menu to select a **Singleton Availability Zone**. The Ops Manager Director installs in this Availability Zone. For Enterprise PKS, this will be the `AZ-MGMT` availability zone.

3. Use the drop-down menu to select a **Network** for BOSH Director. BOSH Director runs on the Enterprise 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 feature 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 Enterprise PKS for details.

3. Choose **Generate passwords** or **Use default BOSH password**. Use the **Generate passwords** option for increased security.

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

Step 8: Configure BOSH DNS

1. Select **BOSH DNS Config**.

2. (Optional) In **Excluded Recursors**, enter a list of prohibited recursor addresses.

3. (Optional) In **Recursor Timeout**, enter a time limit for contacting the connected recursors. This includes dialing, writing, and reading from the recursor. If any of these actions exceeds the time limit you set, the action fails.

   Note: This time limit must include one of the Go parse duration time units. For example, entering `5s` sets the timeout limit to five seconds. For more information about supported time units, see func ParseDuration in the Go

> Programming Language documentation.

4. (Optional) In **Handlers**, enter a list of custom domain handlers in JSON format.
5. Click **Save**.

Step 9: 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) Select the **Enable TLS** checkbox to send encrypted logs to remote server with TLS. After you select the checkbox, perform the following steps:
   a. Enter either the name or SHA1 fingerprint of the remote peer in **Permitted Peer**.
   b. Enter the SSL certificate for the remote server in **SSL Certificate**.

   > **Note**: For an optimal security configuration, enable TLS encryption when you are forwarding logs. Logs can contain sensitive information, such as cloud provider credentials.

7. (Optional) Enter an integer in **Queue Size**. This value specifies the number of log messages held in the buffer. The default value is 100,000.
8. (Optional) Select the checkbox to **Forward Debug Logs** to an external source. This option is deselected by default. If you select it, you may generate a large amount of log data.
9. (Optional) Enter configuration details for rsyslog in the **Custom rsyslog Configuration** field. This field requires the rainerscript syntax.
10. Click **Save Syslog Settings**.

Step 10: 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 11: (Optional) Add Custom VM Extensions

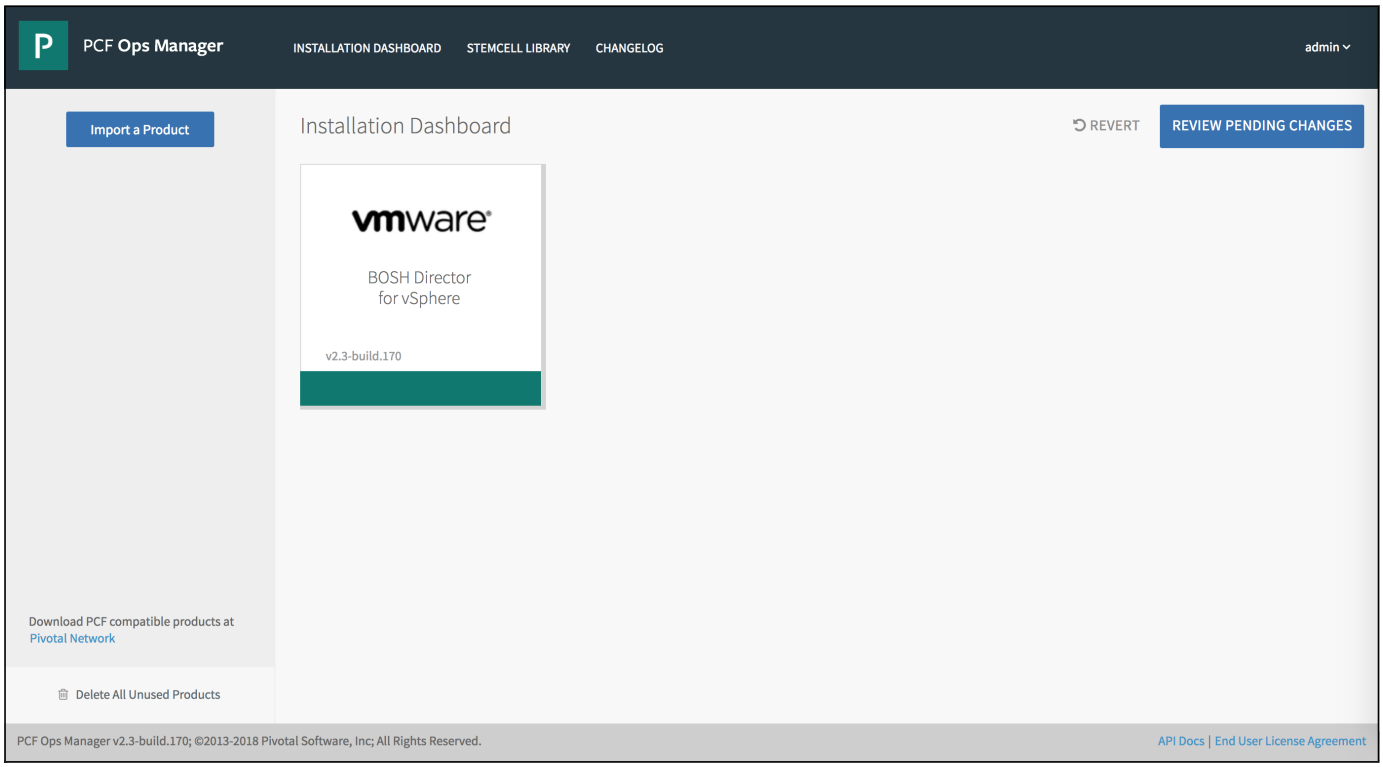
Use the Ops Manager API to add custom properties to your VMs such as associated security groups and load balancers.

For more information, see Managing Custom VM Extensions.

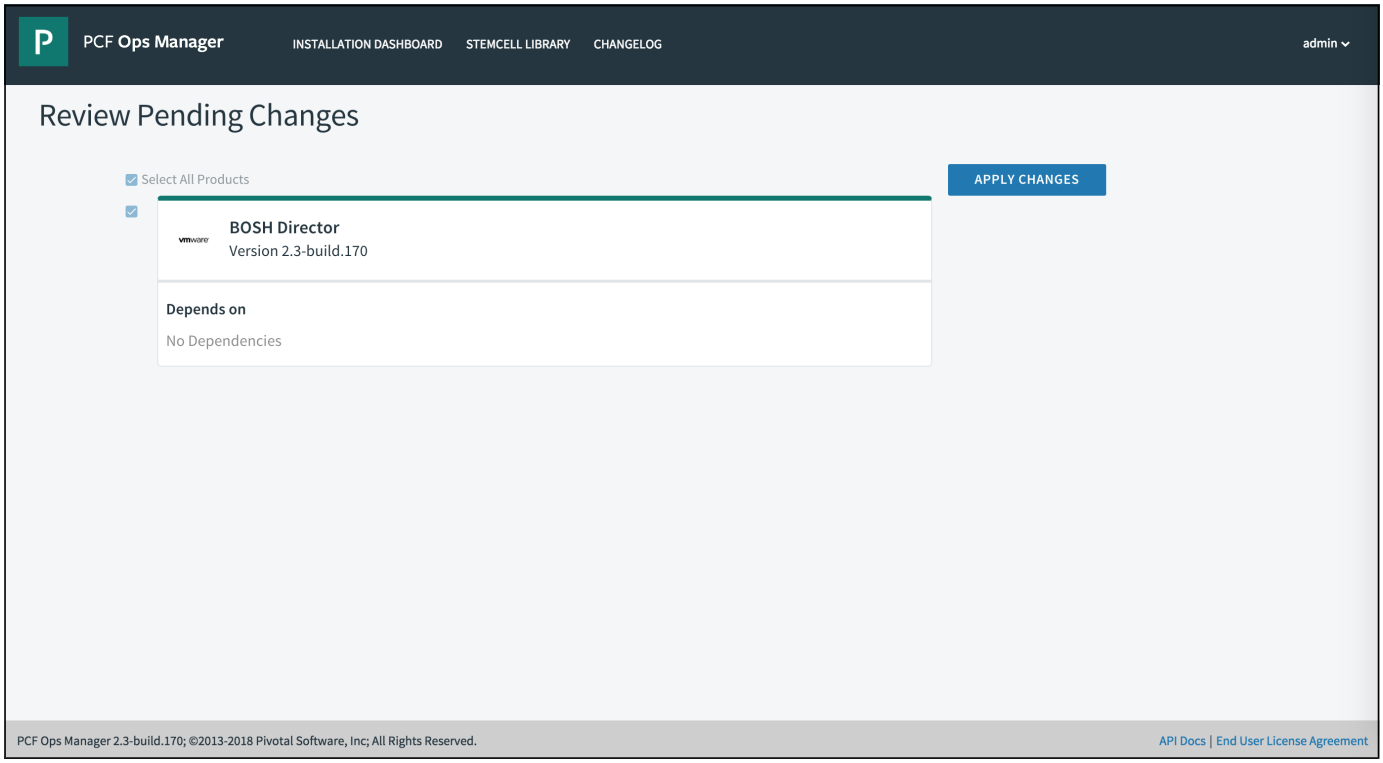
Step 12: Deploy BOSH

Follow the steps below to deploy BOSH:

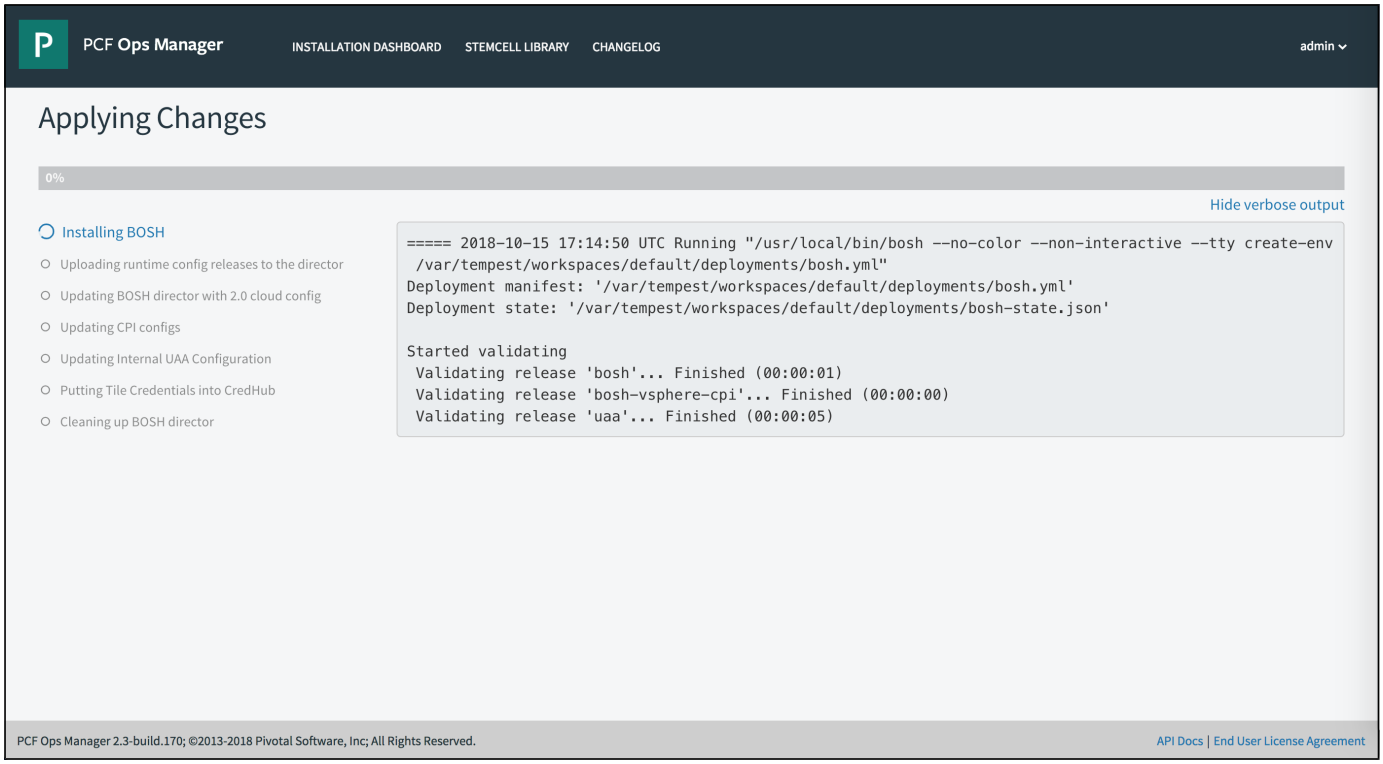
1. Go to the Ops Manager **Installation Dashboard**.



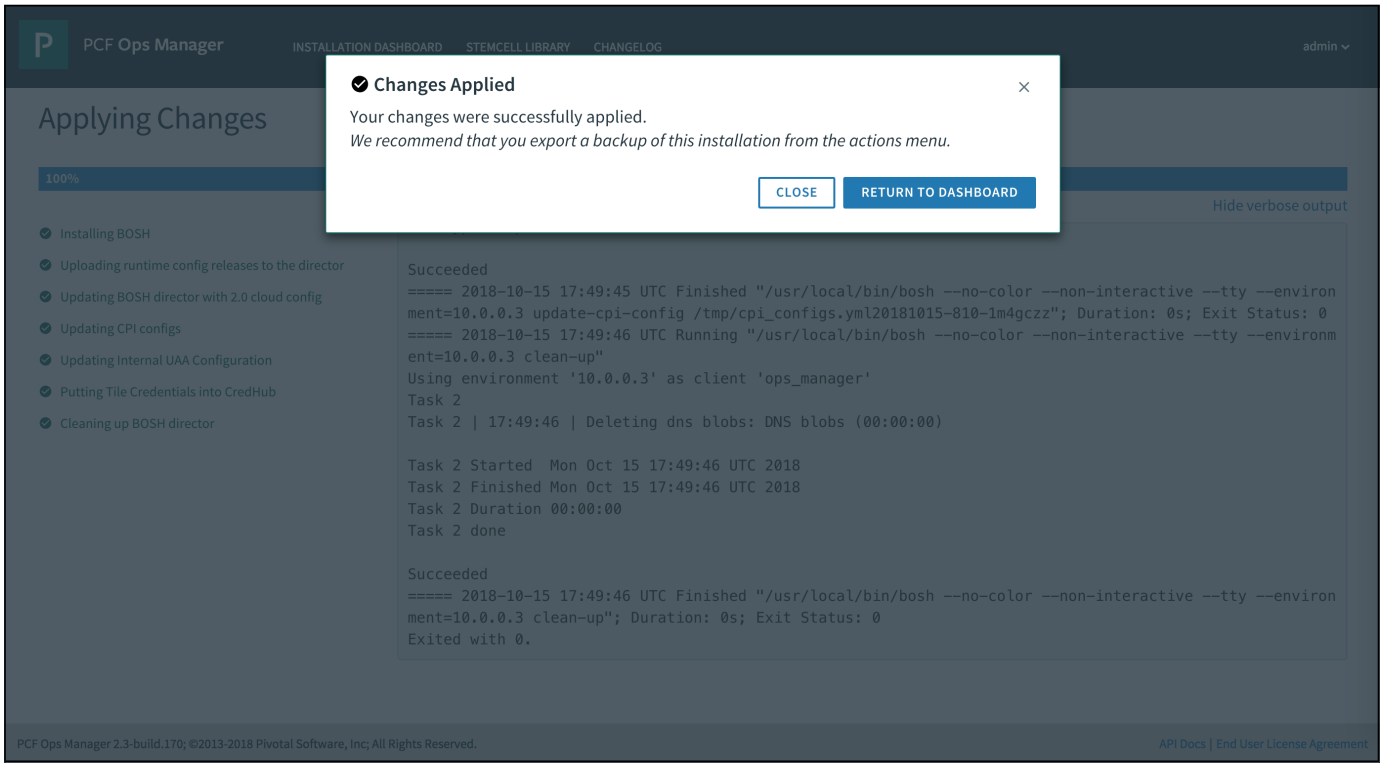
2. Click **Review Pending Changes**.



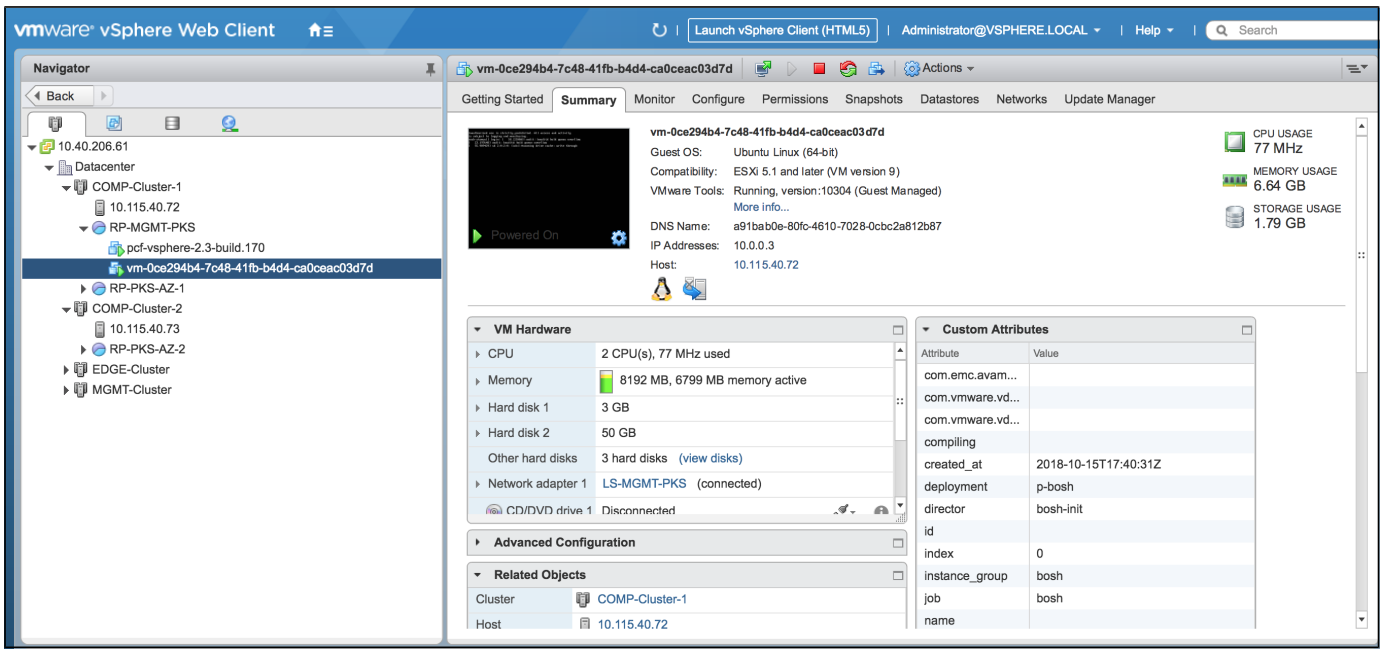
3. Click **Apply Changes**.



4. Confirm changes applied successfully.



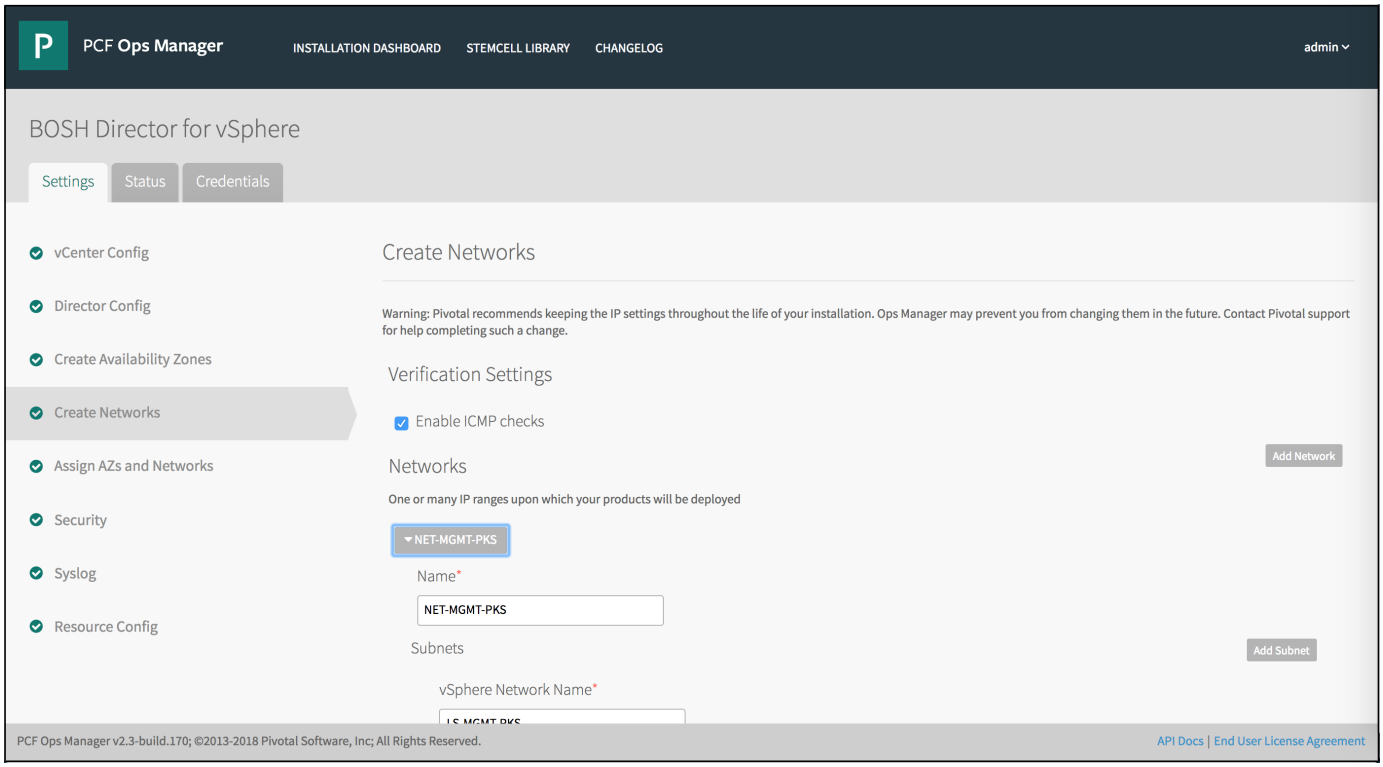
5. Check BOSH VM. Log in to vCenter and check for the `p-bosh` VM deployment in the Enterprise PKS Management resource pool.



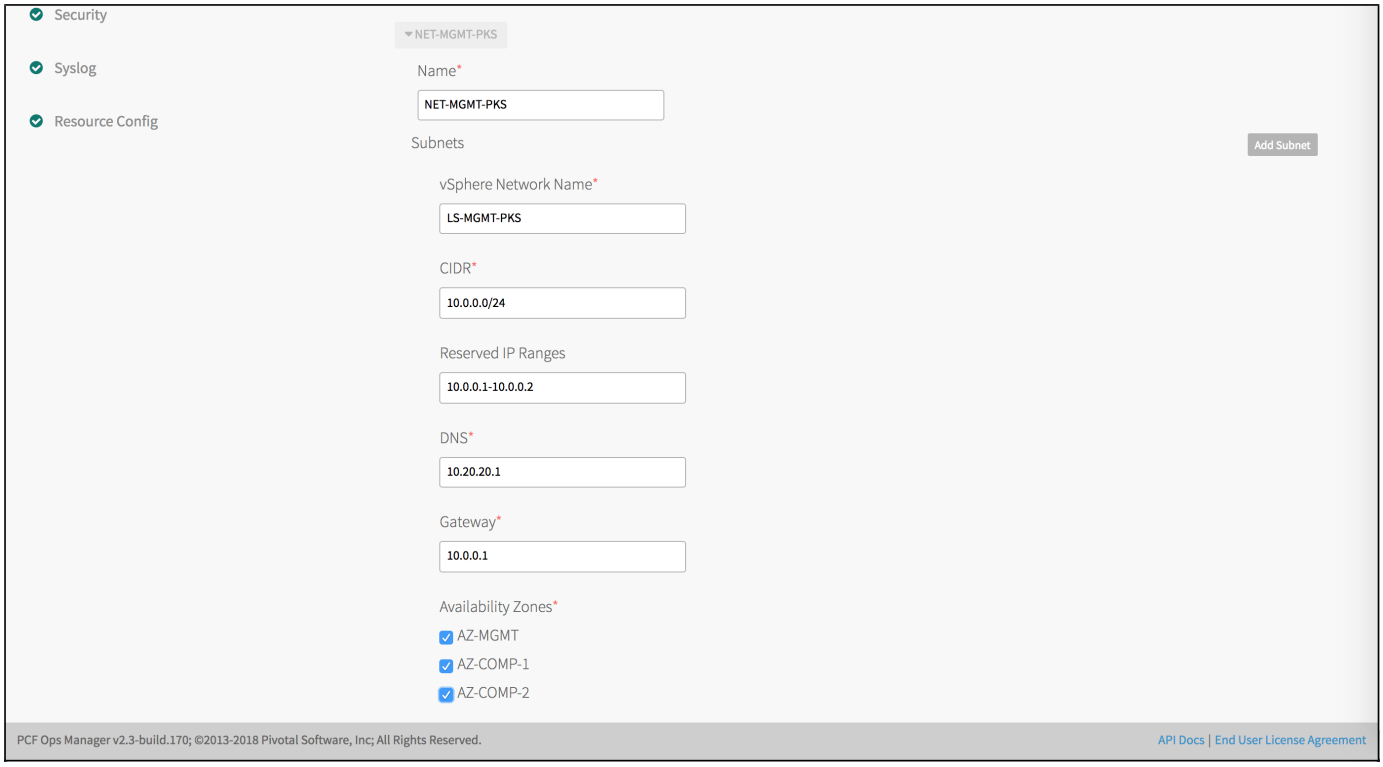
Step 12: Update Network Availability Zones

Ater BOSH is successfully deployed, update the network you defined above (`NET-MGMT-PKS`) to include each of the COMPUTE AZs that you defined. This ensures that both the Management AZ and the Compute AZs appear in the Enterprise PKS tile for the Plans.

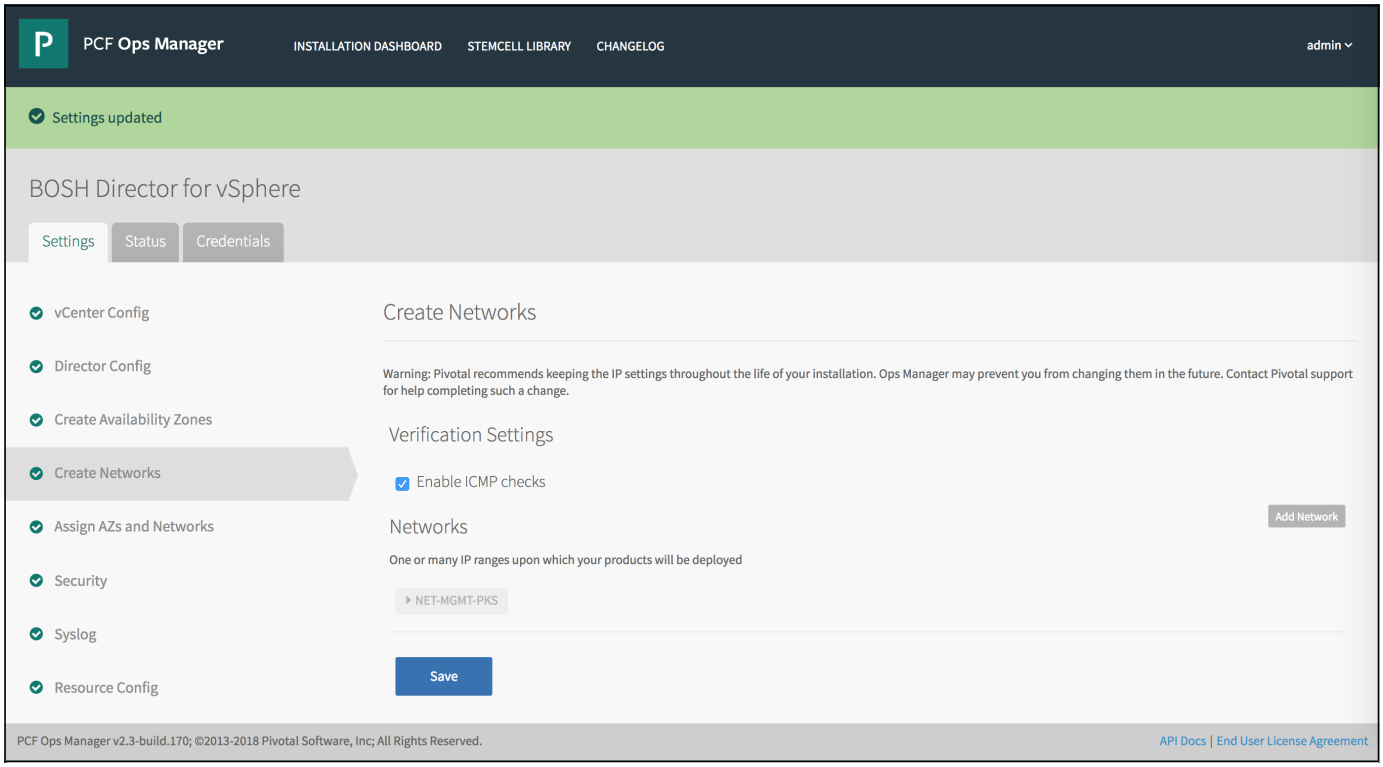
1. Return to the BOSH tile and click **Create Networks**.



2. Edit the network (NET-MGMT-PKS) and each COMPUTE AZ.



3. Click Save.



4. Review pending changes, and click **Apply Changes** to redeploy BOSH.

Next Step

Generate and Register the NSX Manager Superuser Principal Identity Certificate and Key for Enterprise PKS

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

Generating and Registering the NSX Manager Superuser Principal Identity Certificate and Key

In this topic

Page last updated:

This topic describes how to generate and register the NSX Manager superuser principal identity certificate and key in preparation for installing VMware Enterprise PKS on vSphere with NSX-T.

The NSX Manager superuser for PKS has the Enterprise Administrator role and permissions. See Role-Based Access Control in the VMware documentation for more information.

Prerequisites

Before you begin this procedure, ensure that you have successfully completed all preceding steps for installing Enterprise PKS on vSphere with NSX-T, including the following:

- Preparing to Install Enterprise PKS on vSphere with NSX-T Data Center
- Installing and Configuring NSX-T for Enterprise PKS
- Creating the Enterprise PKS Management Plane
- Create Enterprise PKS Compute Plane
- Deploying Ops Manager with NSX-T for Enterprise PKS
- Generating and Registering the NSX Manager Certificate for Enterprise PKS
- Configuring BOSH Director with NSX-T for Enterprise PKS

How Pivotal Platform Accesses NSX Manager

To create, delete, and modify NSX-T networking resources, Pivotal Platform tiles and APIs use a VMware NSX Manager account with the Enterprise Administrator role and permissions.

Users configure Pivotal Platform to authenticate to NSX Manager for different purposes in different tiles:

- **Enterprise PKS tile**
  The Enterprise PKS tile uses NSX Manager to create load balancers, providing a Kubernetes service described in the Create an External Load Balancer section of the Kubernetes documentation.

  To configure the **Enterprise PKS** tile's authentication to NSX Manager, see About the NSX Manager Superuser Principal Identity, below.

- **BOSH Director for vSphere tile**
  The **BOSH Director for vSphere** tile uses NSX Manager to configure networking and security for external-facing Pivotal Platform component VMs, such as Pivotal Application Service routers.

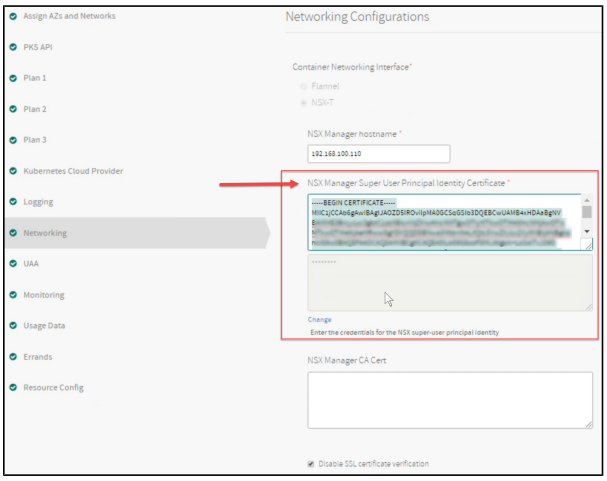
  To configure the **BOSH Director for vSphere** tile's authentication to NSX Manager, see Configure vCenter for Enterprise PKS in *Configuring BOSH Director with NSX-T for Enterprise PKS*.

About the NSX Manager Super User Principal Identity

The PKS API accesses the NSX Manager through an Enterprise Administrator account. This superuser account lets PKS use NSX-T to create, delete, and modify networking resources for Kubernetes cluster nodes.

When you configure Enterprise PKS with NSX-T as the container networking interface, you must provide the certificate and private key for the NSX Manager Enterprise Administrator account in the **Networking** pane of the Enterprise PKS tile.

See the **NSX Manager Super User Principal Identity Certificate** field in the following screenshot:



View a larger version of this image.

For more information, see the Networking section of *Installing Enterprise 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: Run a script on a Linux host with OpenSSL installed that generates the certificate and private key. For more information, see Option A: Generate and Register the Certificate and Key Using Scripts below.
- Option B: Use the automatic **Generate RSA Certificate** option in the Enterprise PKS tile. For more information, see Option B: Generate and Register the Certificate and Key Using the Enterprise PKS Tile below.

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 Scripts

This option uses Bash shell scripts to generate and register the NSX Manager superuser principal identity certificate and key. When you configure Enterprise 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 Enterprise PKS tile.

> **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 Enterprise PKS environment.
2. Create an empty file using `vi create_certificate.sh` or `nano create_certificate.sh`.
3. Modify the file you created to have the following script contents:

```
#!/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"
```

Where:

- `NSX-MANAGER-IP` is the IP address of the NSX Management Cluster VIP or NSX Management Load Balancer IP.
- `NSX-MANAGER-USERNAME` is the Username for NSX Manager.

4. Save the `create_certificate.sh` file.
5. Run the script using `bash create_certificate.sh`.
6. When prompted, enter the `NSX_MANAGER_PASSWORD` for the NSX-T user you specified in the script.
7. Verify results:
   - 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-ID` value is returned to the console.

- The certificate is uploaded to the NSX-T Manager node in the **System** > **Certificates** screen.

8. Copy the UUID that is returned or from the NSX-T UI. You need it 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-T 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. Create an empty file for the script using `vi create_pi.sh` or `nano create_pi.sh`.

2. Copy the script contents into the `create_pi.sh` file you created.

3. Modify the file you created to have the following script contents:

```
#!/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"
```

Where:

- `NSX-MANAGER-IP` is the IP address of the NSX Management Cluster VIP or NSX Management Load Balancer IP.

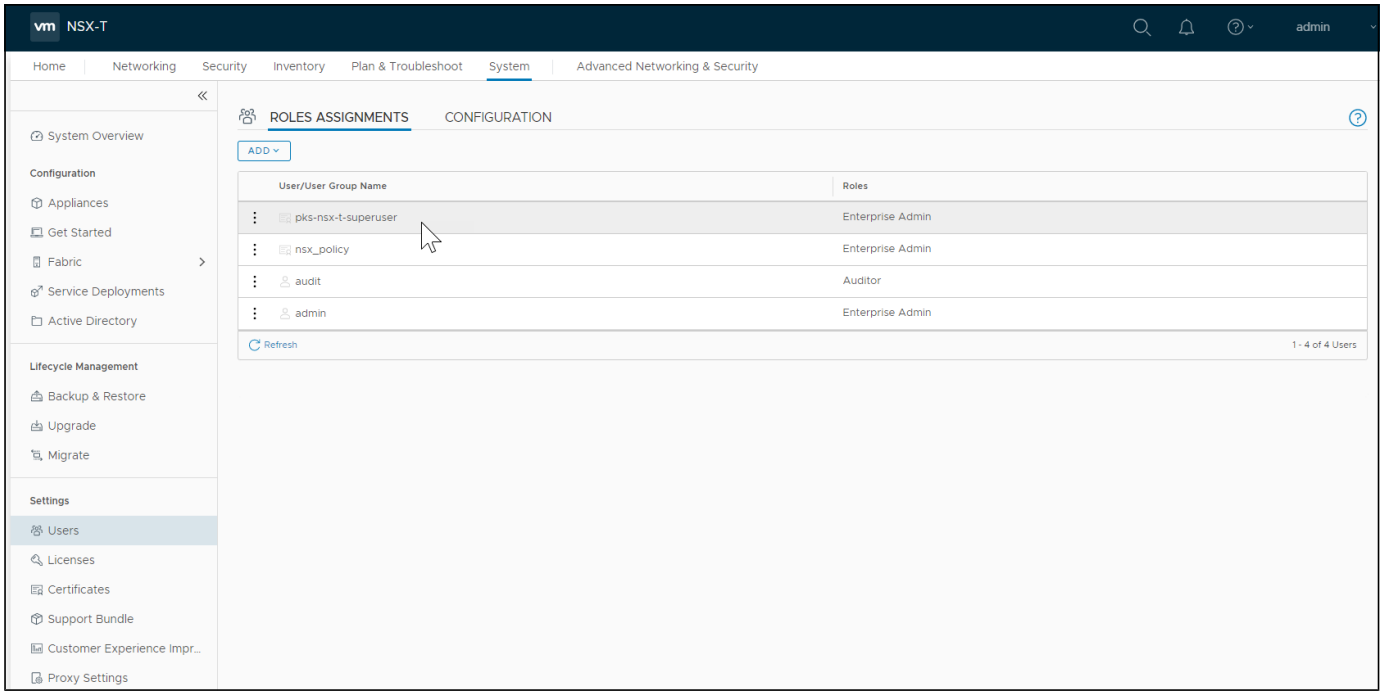
- `NSX-MANAGER-USERNAME` is the Username for NSX Manager.
- `CERTIFICATE-ID` is the response from the `create_certificate.sh` script.

4. Save the changes to the `create_pi.sh` script.

5. Run the script using `bash create_pi.sh`.

6. When prompted, enter the `NSX_MANAGER_PASSWORD` for the NSX user you specified in the script.

7. Verify results: Review the NSX-T Manager **System > Users > Role Assignments**screen. Confirm the principal identity `pks-nsx-t-superuser` is registered with the role `Enterprise Admin`.



View a larger version of this image.

Option B: Generate and Register the Certificate and Key Using the Enterprise PKS Tile

Step 1: Generate the Certificate and Key

To generate the certificate and key automatically in the **Networking** pane in the Enterprise PKS tile, follow the steps below:

1. Navigate to the **Networking** pane in the Enterprise PKS tile. For more information, see Networking in *Installing Enterprise 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. Use the NSX Management Cluster VIP or load balancer for the `NSX_MANAGER_IP`.

```
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"
```

Next Step

After you complete this procedure, follow the instructions in Installing Enterprise PKS on vSphere with NSX-T.

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

Installing Enterprise PKS on vSphere with NSX-T

In this topic

Page last updated:

This topic describes how to install and configure VMware Enterprise PKS on vSphere with NSX-T integration.

Prerequisites

Before you begin this procedure, ensure that you have successfully completed all preceding steps for installing Enterprise PKS on vSphere with NSX-T, including:

- Preparing to Install Enterprise PKS on vSphere with NSX-T Data Center
- Installing and Configuring NSX-T for Enterprise PKS
- Creating the Enterprise PKS Management Plane
- Creating the Enterprise PKS Compute Plane
- Deploying Ops Manager with NSX-T for Enterprise PKS
- Generating and Registering the NSX Manager Certificate for Enterprise PKS
- Configuring BOSH Director with NSX-T for Enterprise PKS
- Generating and Registering the NSX Manager Superuser Principal Identity Certificate and Key for Enterprise PKS

Step 1: Install Enterprise PKS

To install Enterprise 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 **Enterprise PKS** in the left column, click the plus sign to add this product to your staging area.

Step 2: Configure Enterprise PKS

Click the orange **Enterprise PKS** tile to start the configuration process.

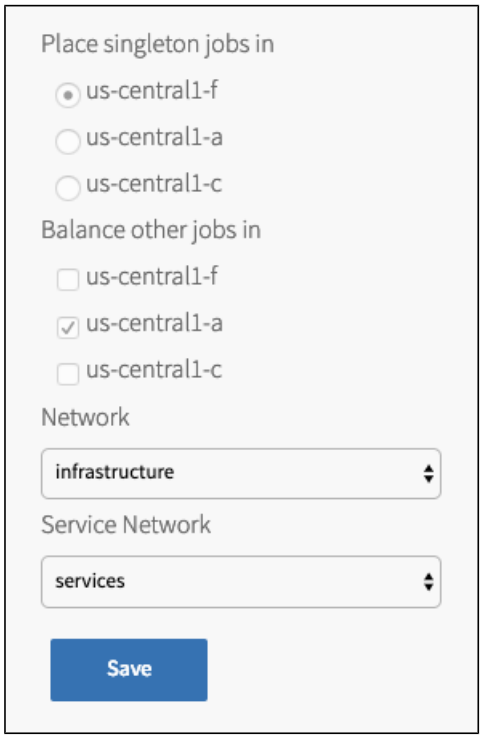
Note: Configuration of NSX-T or Flannel **cannot** be changed after initial installation and configuration of Enterprise PKS.

warning: When you configure the Enterprise 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 Enterprise PKS fails.

Assign AZs and Networks

To configure the availability zones (AZs) and networks used by the Enterprise PKS control plane:

1. Click **Assign AZs and Networks**.
2. Under **Place singleton jobs in**, select the availability zone (AZ) where you want to deploy the PKS API VM as a singleton job.



3. Under **Balance other jobs in**, select the AZ for balancing other Enterprise PKS control plane jobs.

> **Note**: You must specify the **Balance other jobs in** AZ, but the selection has no effect in the current version of Enterprise PKS.

4. 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 Enterprise PKS*. This will provide network placement for the PKS API VM.

5. Under **Service Network**, your selection depends on whether you are installing a new Enterprise PKS deployment or upgrading from a previous version of Enterprise PKS.

   - If you are deploying Enterprise PKS with NSX-T for the first time, select the PKS Management Network that you specified in the **Network** field. You do not need to create or define a service network because Enterprise PKS creates the service network for you during the installation process.
   - If you are upgrading from a previous version of Enterprise PKS, then select the **Service Network** linked to the `ls-pks-service` NSX-T logical switch that Enterprise 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 Enterprise PKS broker.

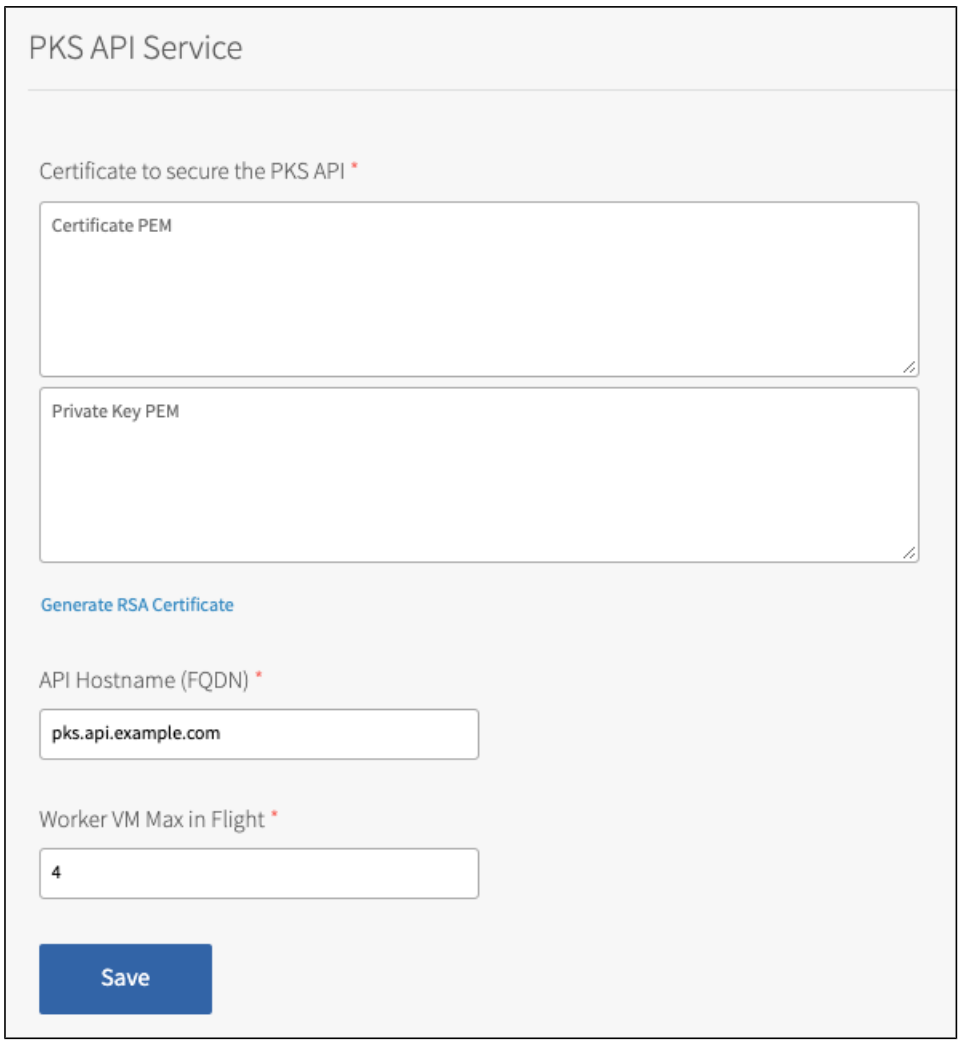
6. Click **Save**.

PKS API

Perform the following steps:

1. Click **PKS API**.

2. Under **Certificate to secure the PKS API**, provide a certificate and private key pair.

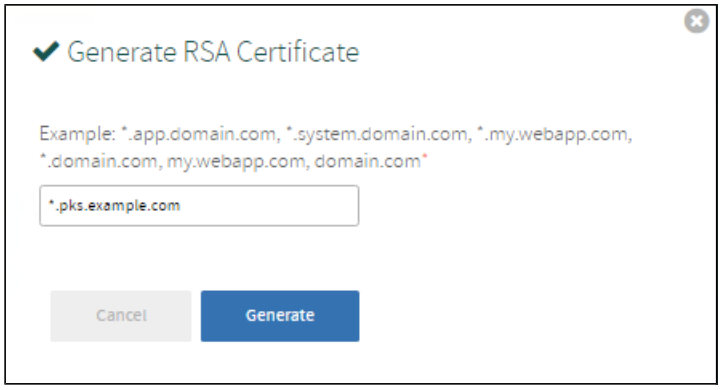


The certificate that you supply should cover the specific subdomain that routes to the PKS API VM with TLS termination on the ingress.

> **warning:** TLS certificates generated for wildcard DNS records only work for a single domain level. For example, a certificate generated for `*.pks.EXAMPLE.com` does not permit communication to `*.api.pks.EXAMPLE.com`. If the certificate does not contain the correct FQDN for the PKS API, calls to the API will fail.

You can enter your own certificate and private key pair, or have Ops Manager generate one for you.
To generate a certificate using Ops Manager:

a. If the **Generate RSA Certificate** option is not available, click **Change**.
b. Click **Generate RSA Certificate**.
c. Enter the domain for your API hostname. This can be a standard FQDN or a wildcard domain.
d. 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 value. When you create or resize a cluster, the `max_in_flight` value limits the number of component instances that can be created or started simultaneously. By default, the `max_in_flight` value is set to `4`, which means that up to four component instances are simultaneously created or started at a time.

5. Click **Save**.

Plans

A plan defines a set of resource types used for deploying a cluster.

Activate a Plan

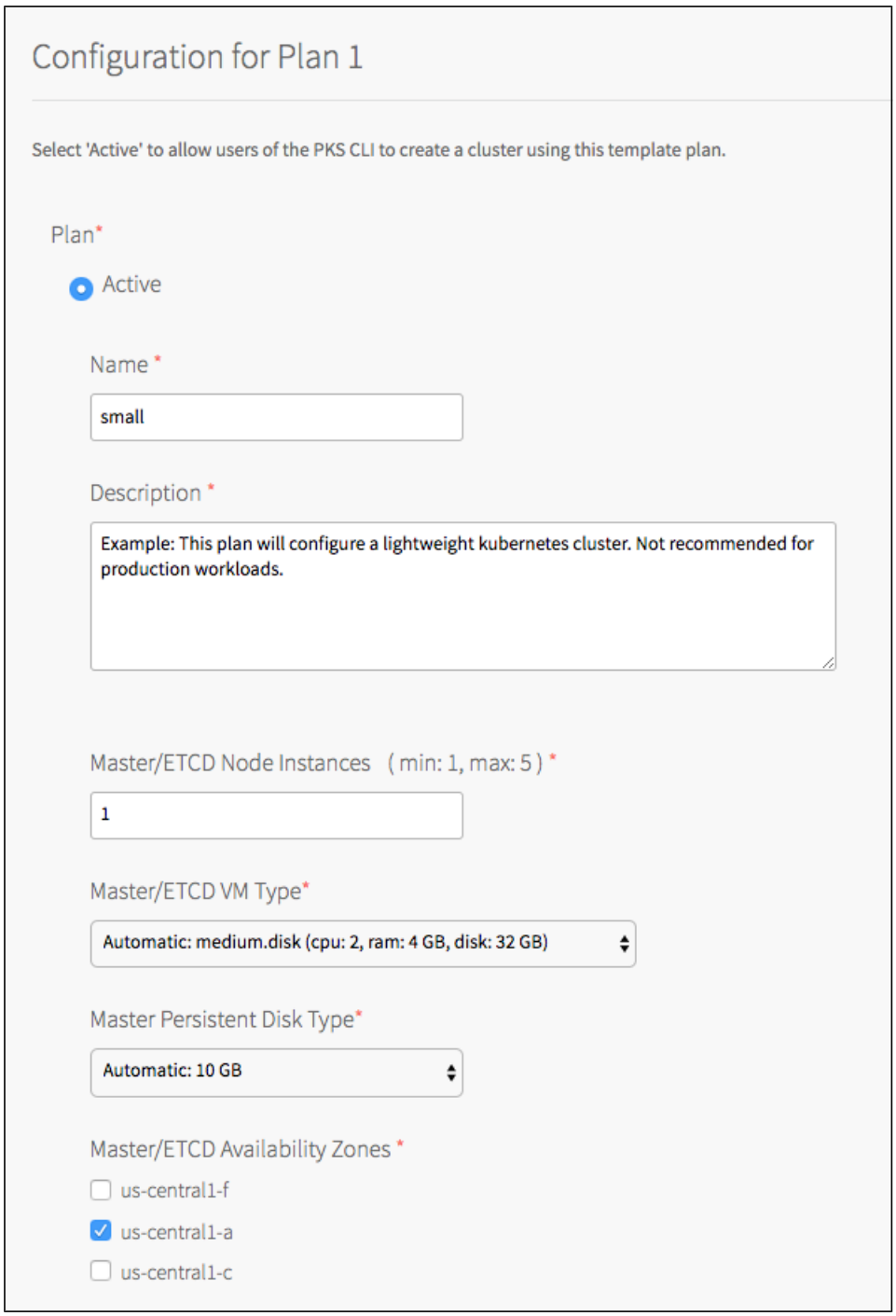
You must first activate and configure **Plan 1**, and afterwards you can optionally activate **Plan 2** through **Plan 10**.

To activate and configure a plan, perform the following steps:

1. Click the plan that you want to activate.

   > **Note**: Plans 11, 12 and 13 support only Windows worker-based Kubernetes clusters, on vSphere with Flannel.

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 the 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 `1`, `3`, or `5`.

> **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.

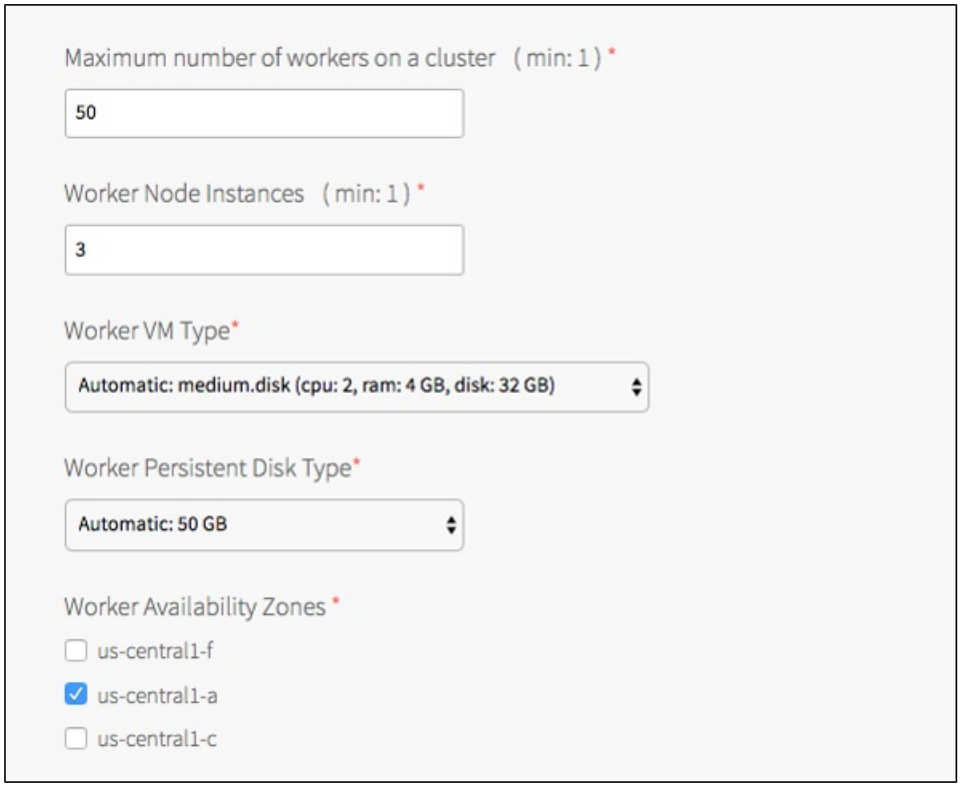
warning: To change the number of master/etcd nodes for a plan, you must ensure that no existing clusters use the plan. Enterprise PKS does not support changing the number of master/etcd nodes for plans with existing clusters.

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 Enterprise 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 Enterprise PKS. If you select more than one AZ, Enterprise PKS deploys the master VM in the first AZ and the worker VMs across the remaining AZs. If you are using multiple masters, Enterprise PKS deploys the master and worker VMs across the AZs in round-robin fashion.

9. Under **Maximum number of workers on a cluster**, set the maximum number of Kubernetes worker node VMs that Enterprise 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.

For more information about creating clusters, see Creating Clusters.

> **Note**: Changing a plan's **Worker Node Instances** setting does not alter the number of worker nodes on existing clusters. For information about scaling an existing cluster, see Scale Horizontally by Changing the Number of Worker Nodes Using the PKS CLI in *Scaling Existing Clusters*.

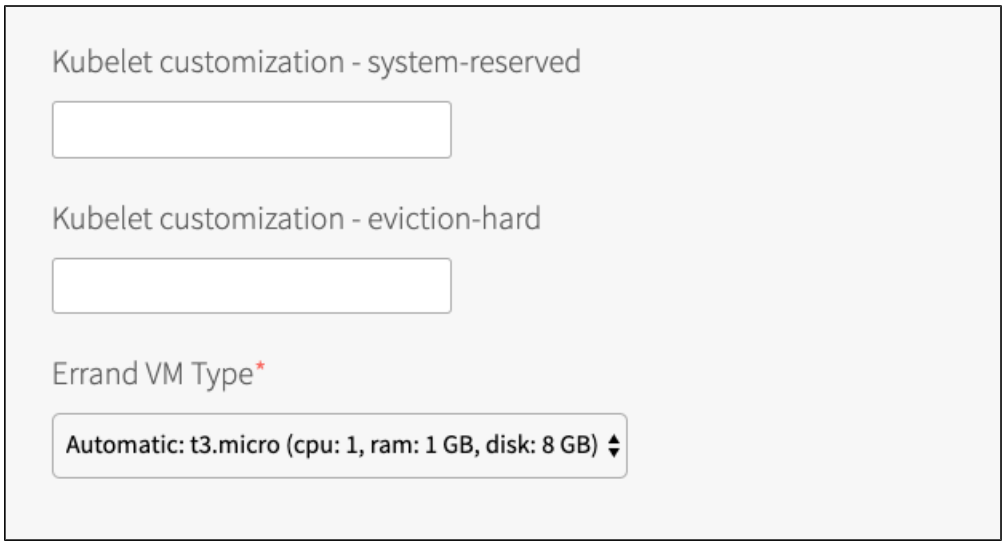
11. Under **Worker VM Type**, select the type of VM to use for Kubernetes worker node VMs. For more information, including worker node VM customization options, see the Worker Node VM Number and Size section of *VM Sizing for Enterprise PKS Clusters*.

> **Note**: If you install Enterprise 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 Enterprise 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. Enterprise 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.

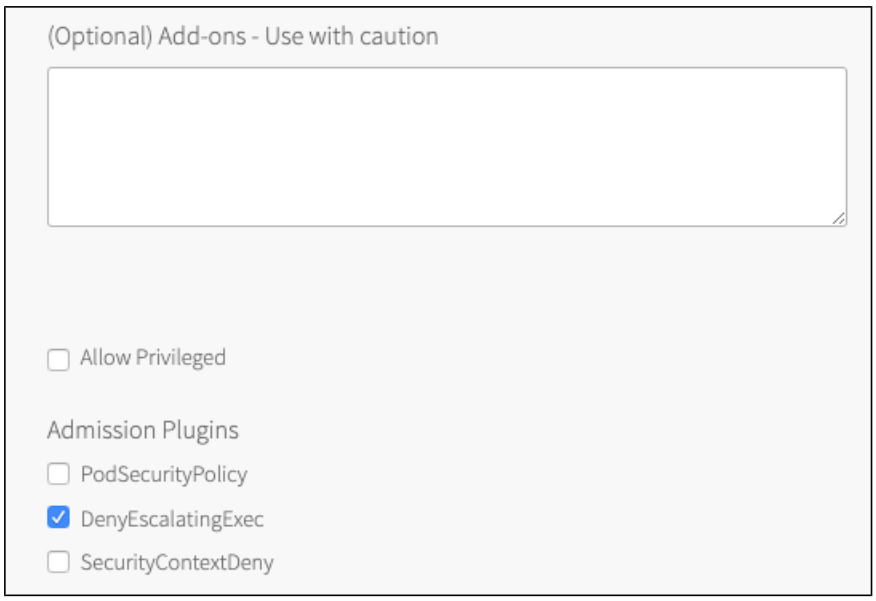


15. Under **Kubelet customization - eviction-hard**, enter threshold limits that Kubelet can use to evict pods when they exceed the limit. Enter limits in the format `EVICTION-SIGNAL=QUANTITY`. For example, `memory.available=100Mi, nodefs.available=10%, nodefs.inodesFree=5%`. For more information about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



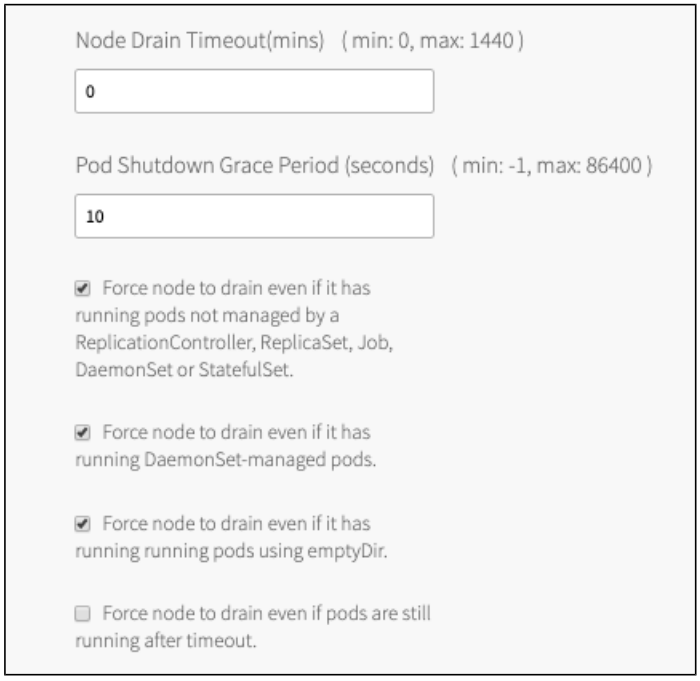
Linux Workloads.

18. (Optional) To allow users to create pods with privileged containers, select the **Allow Privileged** option. For more information, see Pods in the Kubernetes documentation.

> **Note:** Enabling the `Allow Privileged` option means that all containers in the cluster will run in privileged mode. Pod Security Policy provides a privileged parameter that can be used to enable or disable Pods running in privileged mode. As a best practice, if you enable `Allow Privileged`, define PSP to limit which Pods run in privileged mode. If you are implementing PSP for privileged pods, you must enable `Allow Privileged` mode.

19. (Optional) Enable or disable one or more admission controller plugins: **PodSecurityPolicy**, **DenyEscalatingExec**, and **SecurityContextDeny**. See Admission Plugins for more information.

20. (Optional) Under **Node Drain Timeout(mins)**, enter the timeout in minutes for the node to drain pods. If you set this value to



`0`, the node drain does not terminate.

21. (Optional) Under **Pod Shutdown Grace Period (seconds)**, enter a timeout in seconds for the node to wait before it forces the pod to terminate. If you set this value to `-1`, the default timeout is set to the one specified by the pod.

22. (Optional) To configure when the node drains, enable the following:
    - **Force node to drain even if it has running pods not managed by a ReplicationController, ReplicaSet, Job, DaemonSet or StatefulSet.**
    - **Force node to drain even if it has running DaemonSet-managed pods.**
    - **Force node to drain even if it has running running pods using emptyDir.**
    - **Force node to drain even if pods are still running after timeout.**

    > **warning:** If you select **Force node to drain even if pods are still running after timeout**, the node kills all running workloads on pods. Before enabling this configuration, set **Node Drain Timeout** to a value greater than `0`.

    For more information about configuring default node drain behavior, see Worker Node Hangs Indefinitely in *Troubleshooting*.

23. Click **Save**.

Deactivate a Plan

To deactivate a plan, perform the following steps:

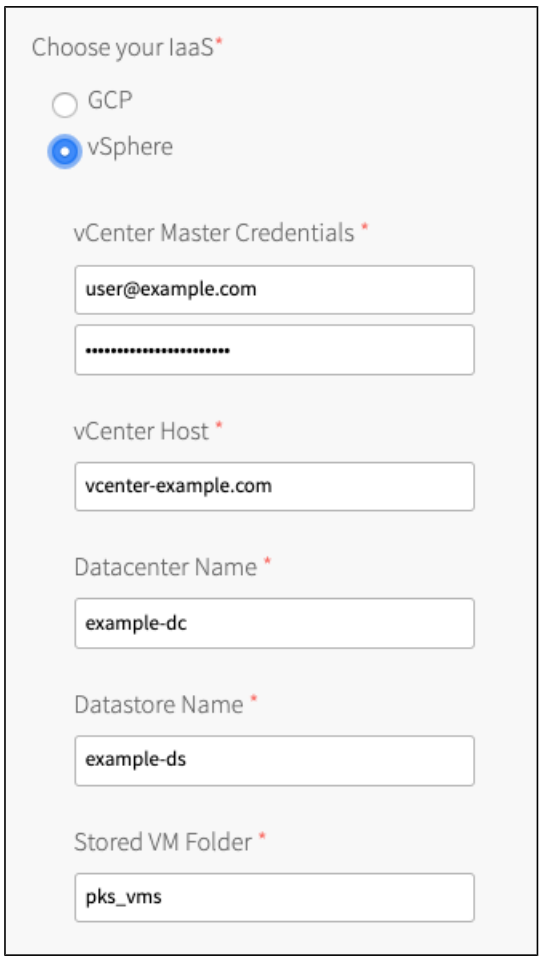
1. Click the plan that you want to deactivate.
2. Select **Inactive**.
3. Click **Save**.

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 Enterprise 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 vSphere Before Deploying Enterprise PKS.
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`. Populate **Datastore Name** with the Persistent Datastore name configured in your **BOSH Director** tile under **vCenter Config** > **Persistent Datastore Names**. The **Datastore Name** field should contain a single Persistent datastore.

> **Note**: The vSphere datastore type must be Datastore. Enterprise PKS does not support the use of vSphere Datastore Clusters with or without Storage DRS. For more information, see Datastores and Datastore Clusters in the vSphere documentation.

> **Note**: The **Datastore Name** is the default datastore used if the Kubernetes cluster `StorageClass` does not define a `StoragePolicy`. Do not enter a datastore that is a list of BOSH Job/VMDK datastores. For more information, see PersistentVolume Storage Options on vSphere.

> **Note**: For multi-AZ and multi-cluster environments, your **Datastore Name** should be a shared Persistent datastore available to each vSphere cluster. Do not enter a datastore that is local to a single cluster. For more information, see PersistentVolume Storage Options on vSphere.

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

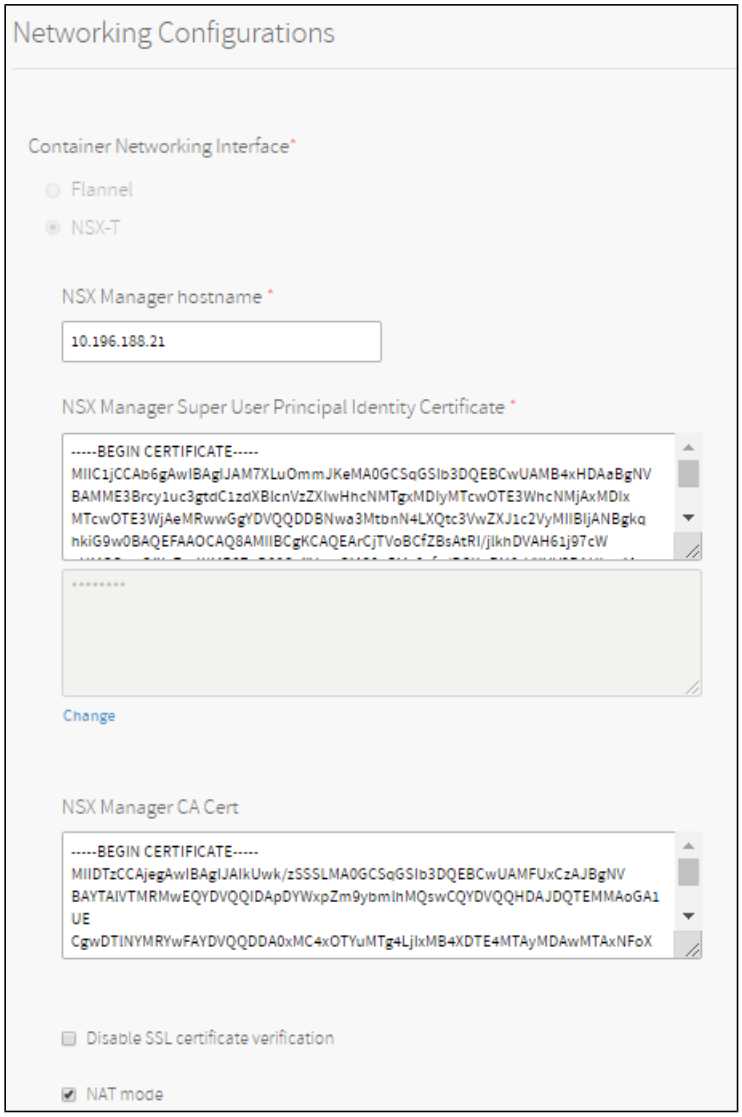
`pks_vms` .

f. Click **Save**.

Networking

To configure networking, do the following:

1. Click **Networking**.



2. Under **Container Networking Interface**, select **NSX-T**.

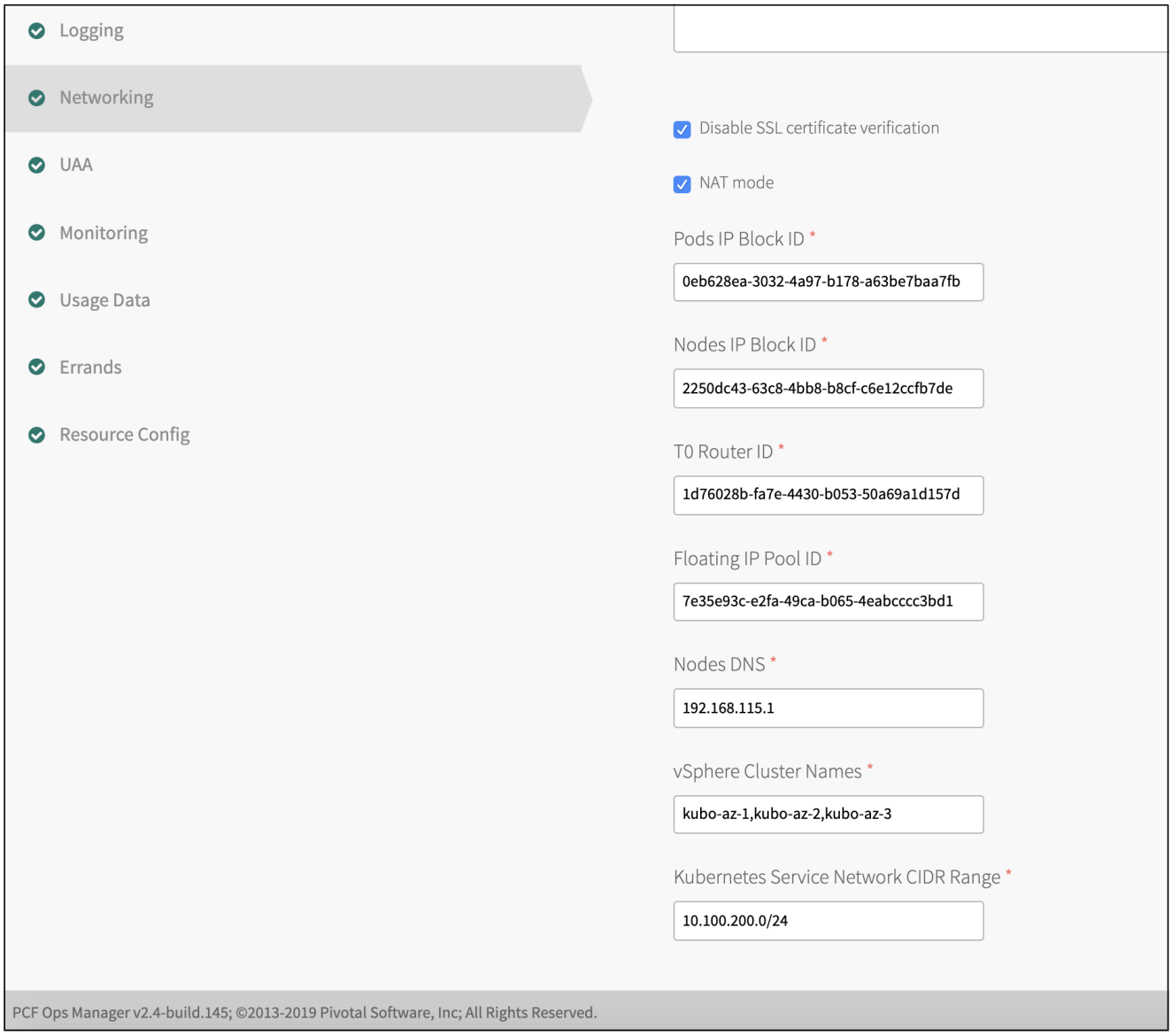
   a. For **NSX Manager hostname**, enter the hostname or IP address of your NSX Manager.
   b. For **NSX Manager Super User Principal Identify Certificate**, copy and paste the contents and private key of the Principal Identity certificate you created in Generating and Registering the NSX Manager Superuser 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:



View a larger version of this image.

- **Pods IP Block ID**: Enter the UUID of the IP block to be used for Kubernetes pods. Enterprise 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. Enterprise 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 Enterprise 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. Enterprise 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`.
k. For **Kubernetes Service Network CIDR Range**, specify an IP address and subnet size depending on the number of Kubernetes services that you plan to deploy within a single Kubernetes cluster, for example: `10.100.200.0/24`. The IP address used here is internal to the cluster and can be anything, such as `10.100.200.0`. A `/24` subnet provides 256 IPs. If you have a cluster that requires more than 256 IPs, define a larger subnet, such as /20.

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 Enterprise PKS on NSX-Tfor 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.

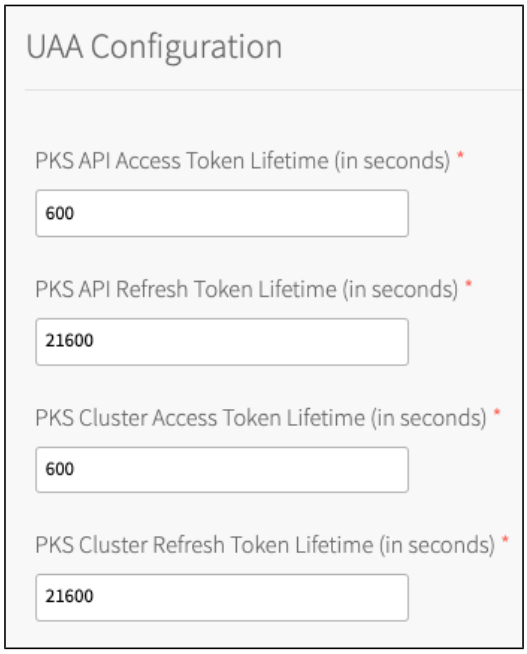
5. Click **Save**.

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. This field defaults to `600`.



3. Under **PKS API Refresh Token Lifetime**, enter a time in seconds for the PKS API refresh token lifetime. This field defaults to `21600`.

4. Under **PKS Cluster Access Token Lifetime**, enter a time in seconds for the cluster access token lifetime. This field defaults

to `600`.

5. Under **PKS Cluster Refresh Token Lifetime**, enter a time in seconds for the cluster refresh token lifetime. This field defaults to `21600`.

> **Note**: Pivotal recommends using the default UAA token timeout values. By default, access tokens expire after ten minutes and refresh tokens expire after six hours. If you want to customize your token timeout values, see Token Management in *UAA Overview*.

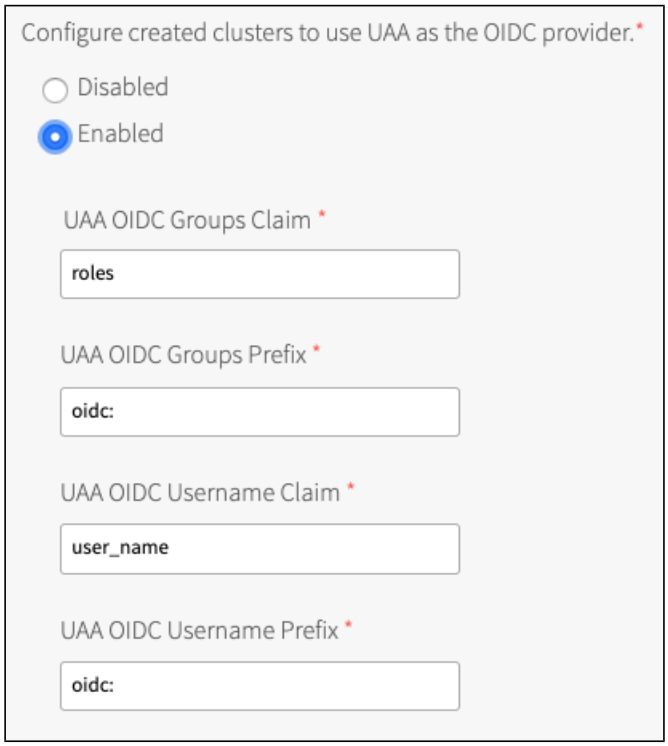
6. Under **Configure created clusters to use UAA as the OIDC provider**, select **Enabled** or **Disabled**. If you click **Enabled**, Kubernetes verifies end-user identities based on authentication executed by UAA. For more information, see the table below.

Option	Description
Disabled	If you do not enable UAA as the OpenID Connect (OIDC) provider, Kubernetes authenticates users against its internal user management system.
Enabled	If you enable UAA as the OIDC provider, Kubernetes authenticates users as follows: ◦ If you select **Internal UAA** in the next step, Kubernetes authenticates users against the internal UAA authentication mechanism. ◦ If you select **LDAP Server** in the next step, Kubernetes authenticates users against the LDAP server. ◦ If you select **SAML Identity Provider** in the next step, Kubernetes authenticates users against the SAML identity provider.

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

To configure Enterprise PKS to use UAA as the OIDC provider, do the following:

a. Under **Configure created clusters to use UAA as the OIDC provider**, select **Enabled**.



b. For **UAA OIDC Groups Claim**, enter the name of your groups claim. This is used to set a user's group in the JSON Web Token (JWT) claim. The default value is `roles`.
c. For **UAA OIDC Groups Prefix**, enter a prefix for your groups claim. This prevents conflicts with existing names. For example, if you enter the prefix `oidc:`, UAA creates a group name like `oidc:developers`. If you are configuring a new Enterprise PKS installation, the default is `oidc:`. If you are upgrading to Enterprise PKS v1.5, the default is `-`.
d. For **UAA OIDC Username Claim**, enter the name of your username claim. This is used to set a user's username in the JWT claim. The default value is `user_name`. Depending on your provider, admins can enter claims besides `user_name`, like `email` or `name`.
e. For **UAA OIDC Username Prefix**, enter a prefix for your username claim. This prevents conflicts with existing names. For example, if you enter the prefix `oidc:`, UAA creates a username like `oidc:admin`. If you are configuring a new Enterprise PKS installation, the default is `oidc:`. If you are upgrading to Enterprise PKS v1.5, the default is `-`.

> **Note:** Pivotal recommends adding OIDC prefixes to prevent OIDC users and groups from gaining unintended cluster privileges. When you upgrade to Enterprise PKS v1.5, if you do not change the values for **UAA OIDC Groups Prefix** or **UAA OIDC Username Prefix**, Enterprise PKS does not add prefixes.

> **warning:** If you change the above values for a pre-existing Enterprise PKS installation, you must change any existing role bindings that bind to a username or group. If you do not change your role bindings, developers cannot access Kubernetes clusters. For instructions about creating role bindings, see Managing Cluster Access and Permissions.

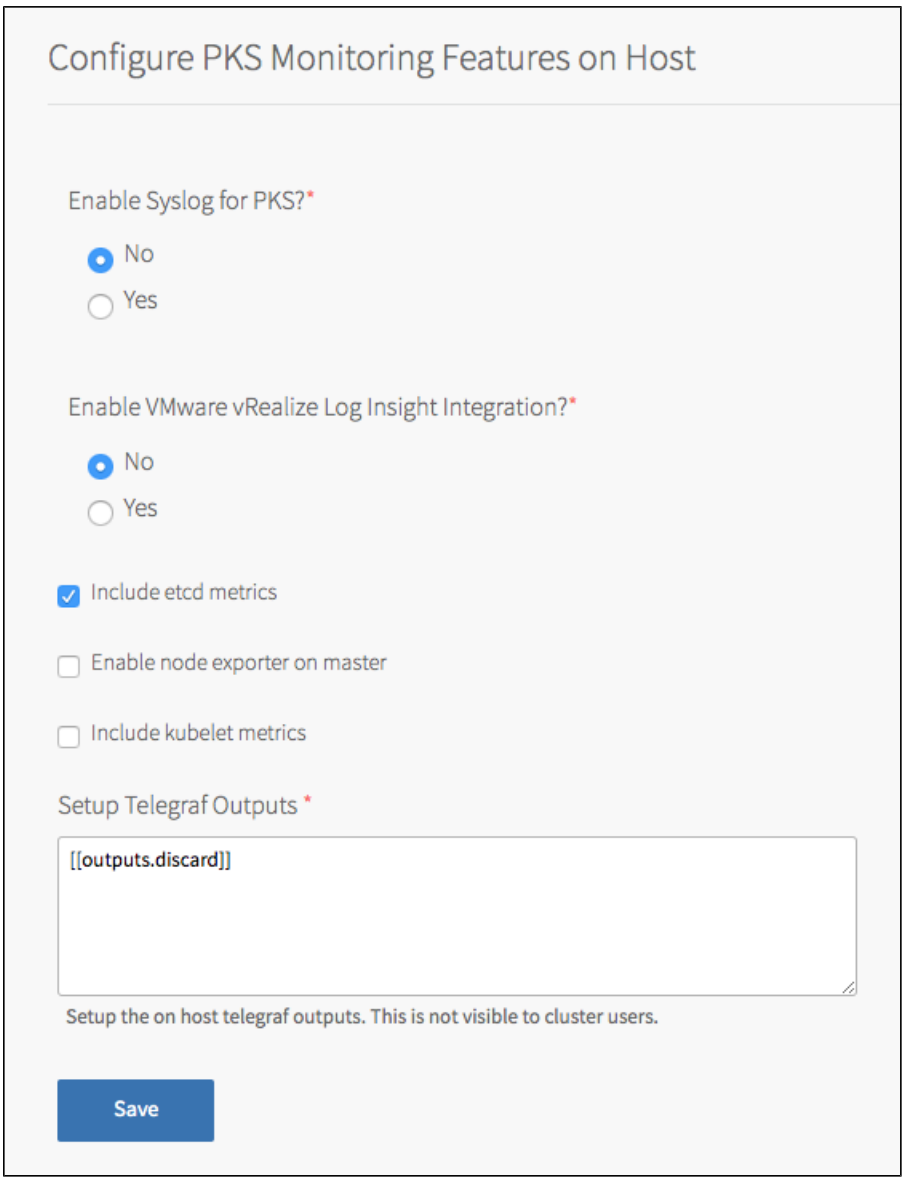
7. Select one of the following options:
   - To use an internal user account store for UAA, select **Internal UAA**. Click **Save** and continue to (Optional) Host Monitoring.
   - To use LDAP for UAA, select **LDAP Server** and continue to Connecting Enterprise PKS to an LDAP Server.
   - To use SAML for UAA, select **SAML Identity Provider** and continue to Connecting Enterprise PKS to a SAML Identity Provider.

(Optional) Host Monitoring

In **Host Monitoring**, you can configure one or more of the following:

- To configure Syslog for BOSH-deployed VMs, see Syslog. Syslog forwards log messages from all BOSH-deployed VMs to a syslog endpoint.
- To configure VMware vRealize Log Insight (vRLI) Integration, see VMware vRealize Log Insight Integration. The vRLI 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`.
- To configure the Telegraf agent, see Telegraf. The Telegraf agent sends metrics from master node and worker node VMs to a monitoring service, such as Wavefront or Datadog.

For more information about these components, see Monitoring PKS and PKS-Provisioned Clusters.



Syslog

To configure Syslog for all BOSH-deployed VMs in Enterprise PKS:

1. Click **Host Monitoring**.
2. Under **Enable Syslog for PKS**, select **Yes**.
3. Under **Address**, enter the destination syslog endpoint.
4. Under **Port**, enter the destination syslog port.
5. Under **Transport Protocol**, select a transport protocol for log forwarding.
6. (Optional) To enable TLS encryption during log forwarding, complete the following steps:
   a. Ensure **Enable TLS** is selected.

> **Note**: Logs may contain sensitive information, such as cloud provider credentials. Pivotal strongly recommends that you enable TLS encryption for log forwarding.

b. Under **Permitted Peer**, provide the accepted fingerprint (SHA1) or name of remote peer. For example, `*.YOUR-LOGGING-SYSTEM.com`.

c. 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.

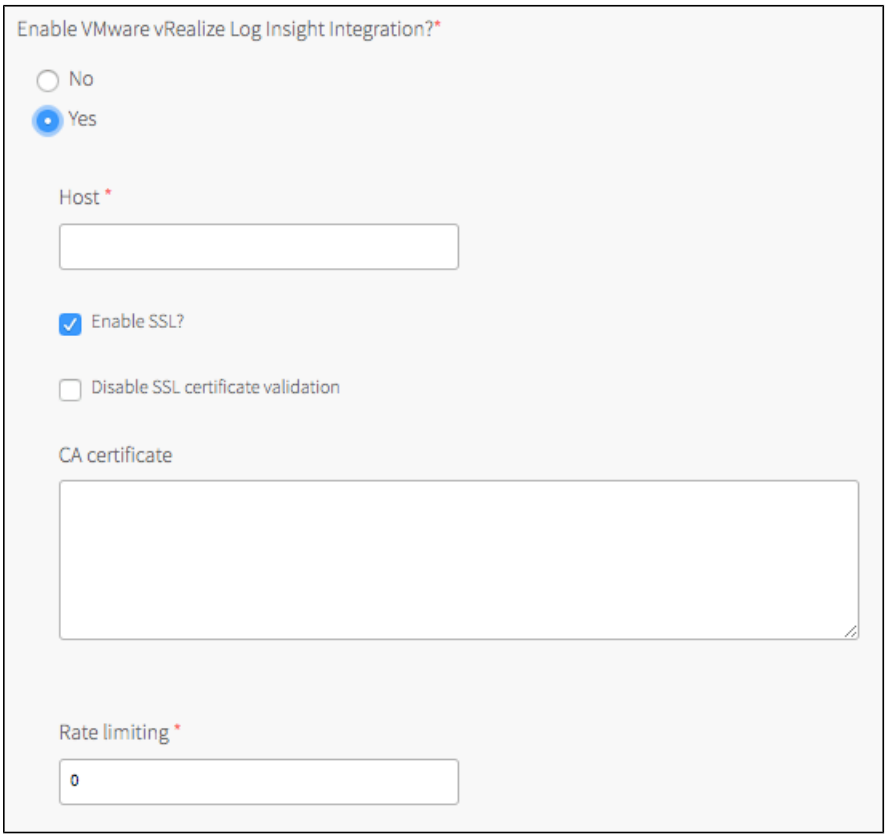
7. (Optional) Under **Max Message Size**, enter a maximum message size for logs that are forwarded to a syslog endpoint. By default, the **Max Message Size** field is 10,000 characters.

8. Click **Save**.

VMware vRealize Log Insight Integration

> **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.

To configure vRLI logging, under **Enable VMware vRealize Log Insight Integration?**, select **Yes** and then perform the following steps:



1. Under **Host**, enter the IP address or FQDN of the vRLI host.

2. (Optional) Select the **Enable SSL?** checkbox to encrypt the logs being sent to vRLI using SSL.

3. 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.
4. (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.
5. 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 that 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 value, such as `10`, then tuning to optimize for your deployment. A large number might result in dropping too many log entries.

6. 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 Enterprise PKS tile does not validate your vRLI configuration settings. To verify your setup, look for log entries in vRLI.

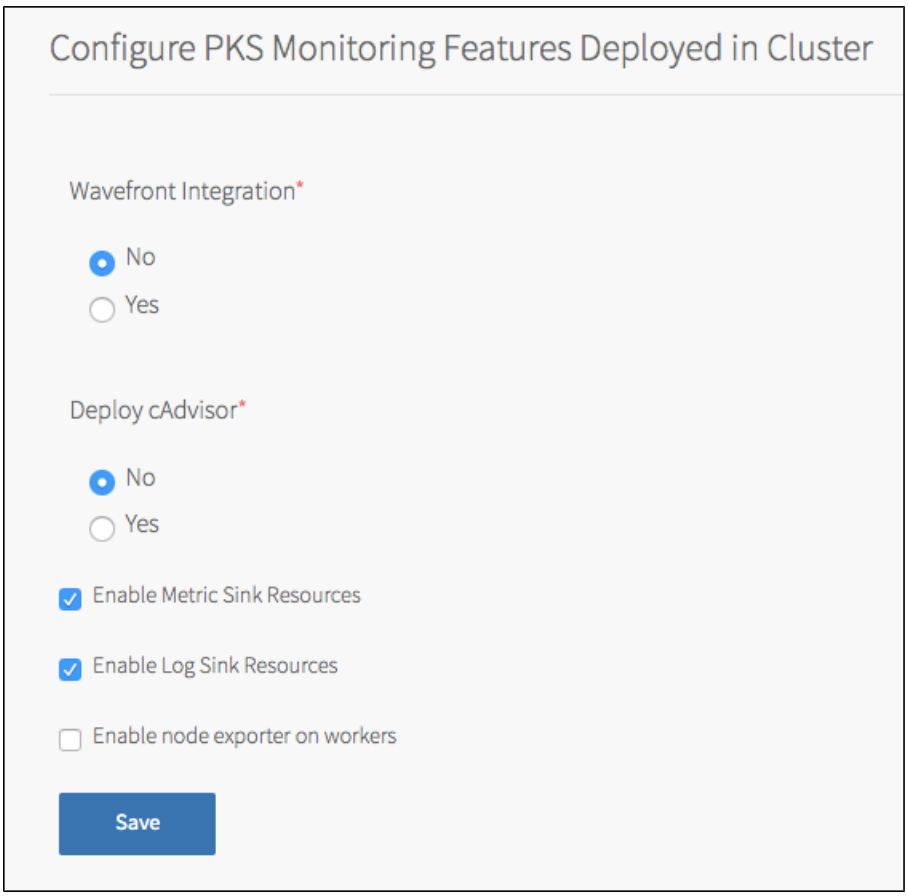
Telegraf

To configure Enterprise PKS to use Telegraf for metric collection:

1. Create a configuration file for your monitoring service. For instructions, see Create a Configuration File.
2. Select one or more of the following checkboxes:
   - **Include etcd metrics**: This includes etcd server and debug metrics.
   - **Enable node exporter on master**: This enables Node Exporter on the *localhost* of each master node VM.
   - **Include kubelet metrics**: This includes all workload metrics across your Kubernetes clusters. Enabling **Include kubelet metrics** generates a high volume of metrics.
3. In **Setup Telegraf Outputs**, replace the default value `[[outputs.discard]]` with the contents of the configuration file that you created above.
4. Click **Save**.

(Optional) In-Cluster Monitoring

In **In-Cluster Monitoring**, you can configure one or more observability components and integrations that run in Kubernetes clusters and capture logs and metrics about your workloads. For more information, see Monitoring Workers and Workloads.



To configure in-cluster monitoring:

- To configure Wavefront, see Wavefront.
- To configure cAdvisor, see VMware vRealize Operations Management Pack for Container Monitoring.
- To configure sink resources, see:
  - Metric Sink Resources
  - Log Sink Resources

  You can enable both log and metric sink resources or only one of them.

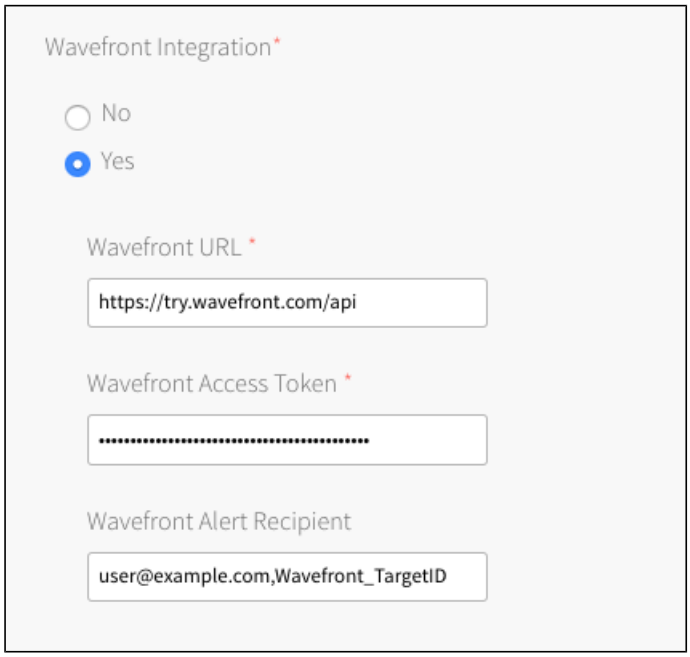
Wavefront

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 the Wavefront documentation.

To enable and configure Wavefront monitoring:

1. In the Enterprise PKS tile, select **In-Cluster Monitoring**.



2. 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**, using the following syntax:

```
USER-EMAIL,WAVEFRONT-TARGETID_001,WAVEFRONT-TARGETID_002
```

   Where:

   - `USER-EMAIL` is the alert recipient's email address.
   - `WAVEFRONT-TARGETID_001` and `WAVEFRONT-TARGETID_002` are your comma-delimited Wavefront Target IDs.

   For example:

```
randomuser@example.com,51n6psdj933ozdjf
```

6. Click **Save**.

To create alerts, you must enable errands in Enterprise PKS.

1. In the Enterprise PKS tile, select **Errands**.
2. On the **Errands** pane, enable **Create pre-defined Wavefront alerts errand**.
3. Enable **Delete pre-defined Wavefront alerts errand**.
4. Click **Save**. Your settings apply to any clusters created after you have saved these configuration settings and clicked **Apply Changes**.

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

VMware vRealize Operations Management Pack for Container Monitoring

You can monitor Enterprise PKS Kubernetes clusters with VMware vRealize Operations Management Pack for Container Monitoring.

To integrate Enterprise PKS with VMware vRealize Operations Management Pack for Container Monitoring, you must deploy a container running cAdvisor in your PKS deployment.

cAdvisor is an open source tool that provides monitoring and statistics for Kubernetes clusters.

To deploy a cAdvisor container:

1. Select **In-Cluster Monitoring**.
2. Under **Deploy cAdvisor**, select **Yes**.
3. Click **Save**.

For more information about integrating this type of monitoring with PKS, see the VMware vRealize Operations Management Pack for Container Monitoring User Guide and Release Notes in the VMware documentation.

Metric Sink Resources

You can configure PKS-provisioned clusters to send Kubernetes node metrics and pod metrics to metric sinks. For more information about metric sink resources and what to do after you enable them in the tile, see Sink Resources in *Monitoring Workers and Workloads*.

To enable clusters to send Kubernetes node metrics and pod metrics to metric sinks:

1. In **In-Cluster Monitoring**, select **Enable Metric Sink Resources**. If you enable this checkbox, Enterprise PKS deploys Telegraf as a `DaemonSet`, a pod that runs on each worker node in all your Kubernetes clusters.
2. (Optional) To enable Node Exporter to send worker node metrics to metric sinks of kind `ClusterMetricSink`, select **Enable node exporter on workers**. If you enable this checkbox, Enterprise PKS deploys Node Exporter as a `DaemonSet`, a pod that runs on each worker node in all your Kubernetes clusters.
   For instructions on how to create a metric sink of kind `ClusterMetricSink` for Node Exporter metrics, see Create a ClusterMetricSink Resource for Node Exporter Metrics in *Creating and Managing Sink Resources*.
3. Click **Save**.

Log Sink Resources

You can configure PKS-provisioned clusters to send Kubernetes API events and pod logs to log sinks. For more information about log sink resources and what to do after you enable them in the tile, see Sink Resources in *Monitoring Workers and Workloads*.

To enable clusters to send Kubernetes API events and pod logs to log sinks:

1. Select **Enable Log Sink Resources**. If you enable this checkbox, Enterprise PKS deploys Fluent Bit as a `DaemonSet`, a pod that runs on each worker node in all your Kubernetes clusters.

2. Click **Save**.

Tanzu Mission Control (Experimental)

Participants in the VMware Tanzu Mission Control beta program can use the **Tanzu Mission Control (Experimental)** pane of the Enterprise PKS tile to integrate their Enterprise PKS deployment with Tanzu Mission Control.

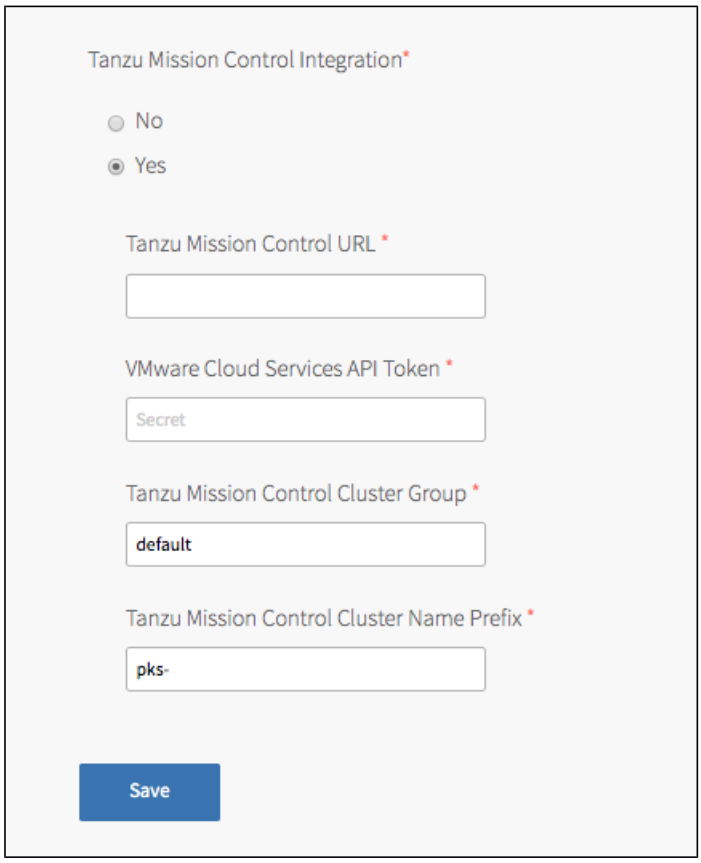
Tanzu Mission Control integration lets you monitor and manage Enterprise PKS clusters from the Tanzu Mission Control console, which makes the Tanzu Mission Control console a single point of control for all Kubernetes clusters.

> **warning:** VMware Tanzu Mission Control is currently experimental beta software and is intended for evaluation and test purposes only. For more information about Tanzu Mission Control, see the VMware Tanzu Mission Control home page.

To integrate Enterprise PKS with Tanzu Mission Control:

1. Confirm that the PKS Control Plane VM has internet access and can connect to `*.tmc.cloud.vmware.com` and the other outbound URLs listed in the What Happens When You Attach a Cluster section of the Tanzu Mission Control documentation.

2. Navigate to the **Enterprise PKS** tile > the **Tanzu Mission Control (Experimental)** pane and select **Yes** under **Tanzu Mission Control Integration**.



3. Configure the fields below:

   - **Tanzu Mission Control URL**: Enter the Org URL of your Tanzu Mission Control subscription, without a trailing slash `/`). For example, `YOUR-ORG.tmc.cloud.vmware.com`.
   - **VMware Cloud Services API token**: Enter your API token to authenticate with VMware Cloud Services APIs. You can retrieve this token by logging in to VMware Cloud Services and viewing your account information.

- **Tanzu Mission Control Cluster Group**: Enter the name of a Tanzu Mission Control cluster group.

  The name can be `default` or another value, depending on your role and access policy:

  - `Org Member` users in VMware cloud services have a `service.admin` role in Tanzu Mission Control. These users:
    - By default, can create and attach clusters only in the `default` cluster group.
    - Can create and attach clusters to other cluster groups after an `organization.admin` user grants them the `clustergroup.admin` or `clustergroup.edit` role for those groups.
  - `Org Owner` users in VMware cloud services have `organization.admin` permissions in Tanzu Mission Control. These users:
    - Can create cluster groups.
    - Can grant `clustergroup` roles to `service.admin` users through the Tanzu Mission Control Access Policy view.

  For more information about role and access policy, see Access Control in the *VMware Tanzu Mission Control Product Documentation*.

- **Tanzu Mission Control Cluster Name Prefix**: Enter a name prefix for identifying the Enterprise PKS clusters in Tanzu Mission Control.

4. Click **Save**.

warning: After the Enterprise PKS tile is deployed with a configured cluster group, the cluster group cannot be updated.

Note: When you upgrade your Kubernetes clusters and have Tanzu Mission Control integration enabled, existing clusters will be attached to Tanzu Mission Control.

CEIP and Telemetry

To configure VMware's Customer Experience Improvement Program (CEIP) and the Pivotal Telemetry Program (Telemetry), do the following:

1. Click **CEIP and Telemetry**.

2. Review the information about the CEIP and Telemetry.

CEIP and Telemetry

About the CEIP and Telemetry Program

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.

Customers who participate (at the enhanced tier) are eligible for several benefits, including Proactive Support.

Additional information regarding the data collected through CEIP or Telemetry, and the purposes for which it is used by VMware is set forth in the Trust & Assurance Center and for Pivotal on the Pivotal Telemetry pages. If you prefer not to participate in CEIP and Telemetry for PKS, you should not join below. You may join or leave CEIP and Telemetry for PKS at any time.

Levels of Participation

No personally identifiable information (PII) is collected at either level of participation. Please refer to the data dictionary for more information on the data we collect.

Standard: This participation tier is anonymous. Your data will be used to improve PKS, but is not identifiable to your organization

Enhanced: This participation tier allows us to identify your organization so we may provide proactive support and other benefits. See sample reports for more details.

Please Note

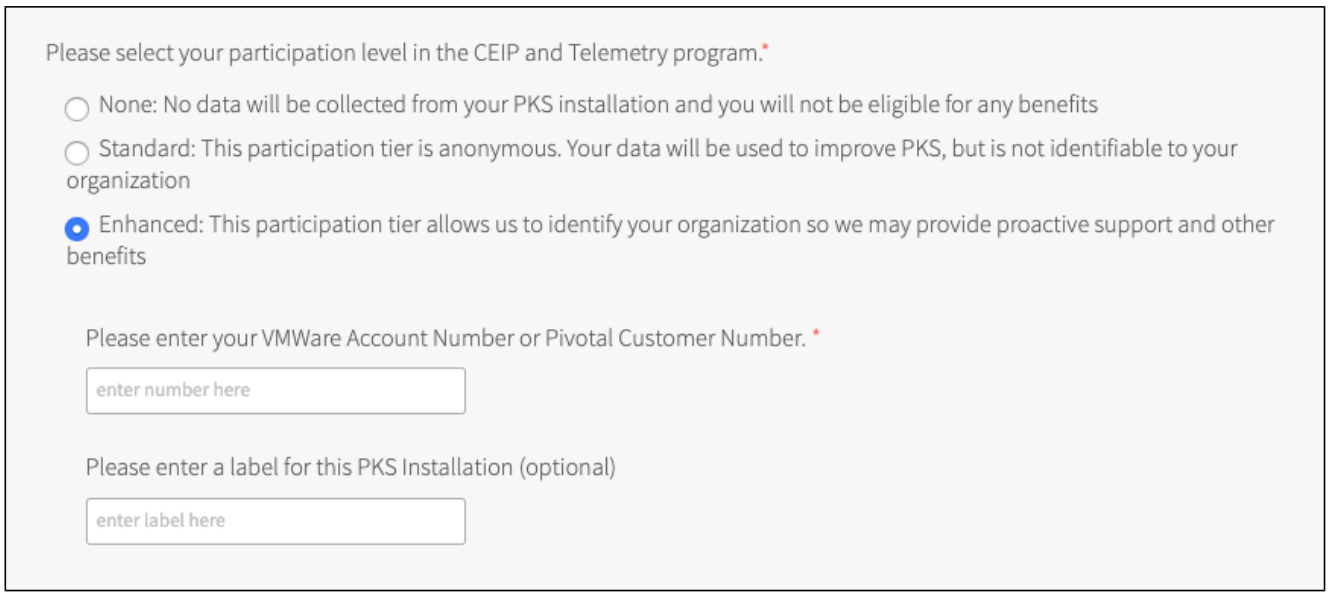
- If you are opting in on behalf of an organization (and not for you as an individual), you represent and warrant that you have legal authority to bind that organization, and you hereby join CEIP/Telemetry on behalf of your organization.
- If you are opting in to the enhanced tier, in the event a term or condition of CEIP or the Telemetry program conflicts with a term or condition of a previously executed license procurement agreement between you and Licensor (Pivotal or VMware), the CEIP or Telemetry program terms supersede solely for purposes of CEIP and Telemetry
- If you are running PKS on a private network, you will need to enable outgoing internet access by opening your firewall to allow traffic to `https://vcsa.vmware.com/ph` on port `443`

Resources

- Data Dictionary
- Participation Benefits
- Sample Reports
- Trust and Assurance Center
- Pivotal Telemetry

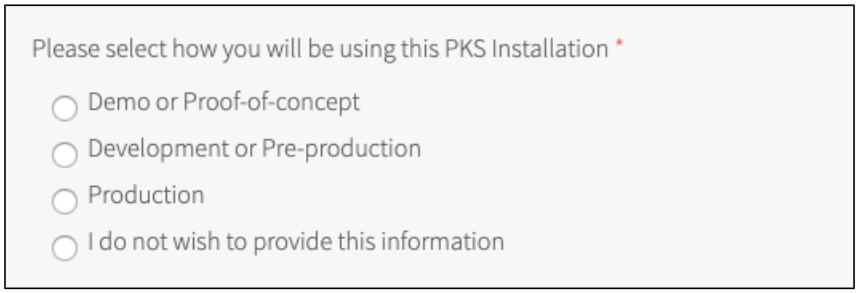
View a larger version of this image.

3. To specify your level of participation in the CEIP and Telemetry program, select one of the **Participation Level** options:
   - **None:** If you select this option, data is not collected from your Enterprise PKS installation.
   - (Default) **Standard:** If you select this option, data is collected from your Enterprise PKS installation to improve Enterprise PKS. This participation level is anonymous and does not permit the CEIP and Telemetry to identify your organization.
   - **Enhanced:** If you select this option, data is collected from your Enterprise PKS installation to provide you proactive support and other benefits. This participation level permits the CEIP and Telemetry to identify your organization.



For more information about the CEIP and Telemetry participation levels, see Participation Levels in *Telemetry*.

4. If you selected the **Enhanced** participation level, complete the following:
   - Enter your VMware account number or Pivotal customer number in the **VMware Account Number or Pivotal Customer Number** field. If you are a VMware customer, you can find your VMware Account Number in your **Account Summary** on my.vmware.com. If you are a Pivotal customer, you can find your Pivotal Customer Number in your Pivotal Order Confirmation email.
   - (Optional) Enter a descriptive name for your PKS installation in the **PKS Installation Label** field. The label you assign to this installation will be used in telemetry reports to identify the environment.
5. To provide information about the purpose for this installation, select an option in the **PKS Installation Type** list.



6. Click **Save**.

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

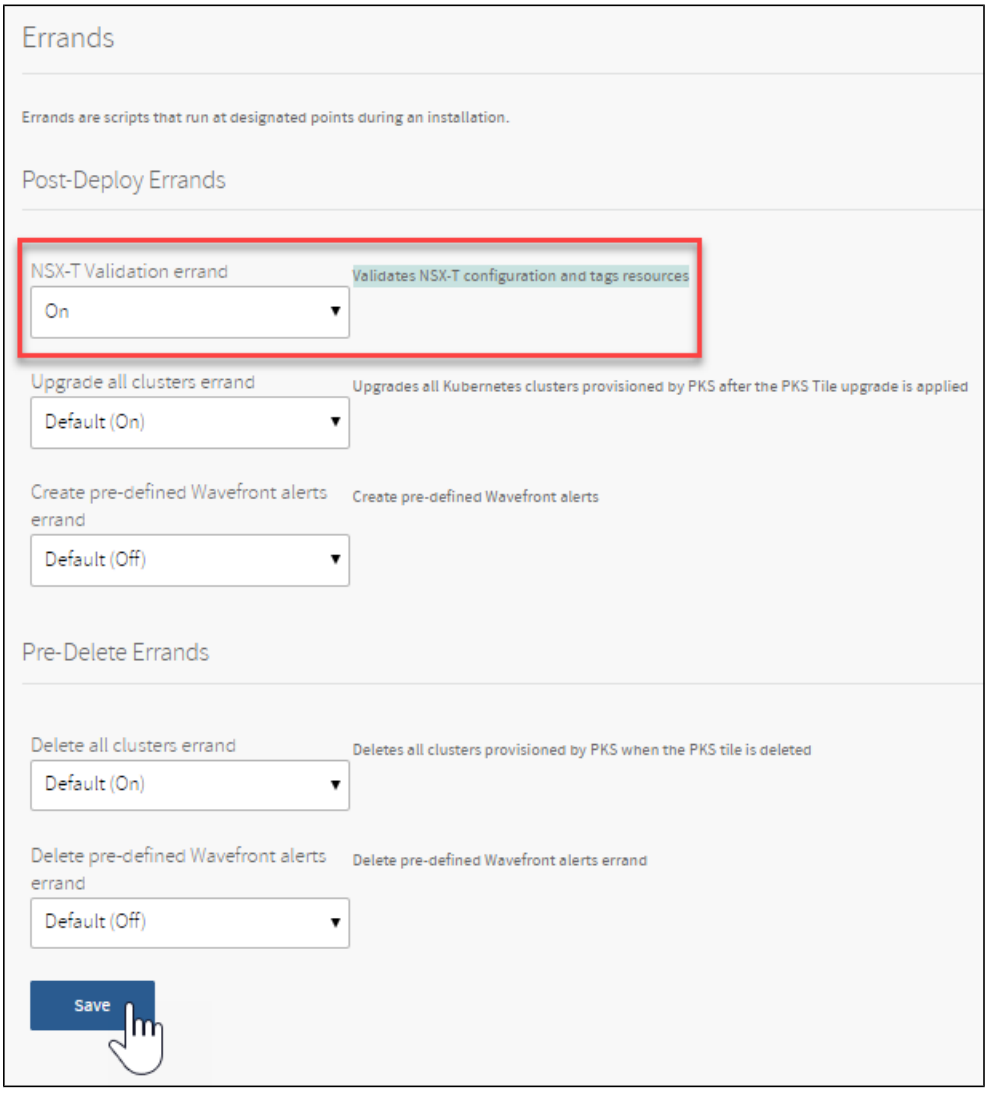
Note: Even if you select **None**, Enterprise PKS-provisioned clusters send usage data to the PKS control plane. However, this data is not sent to VMware or Pivotal and remains on your Enterprise PKS installation.

Errands

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

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

> **warning**: You must enable the NSX-T Validation errand to verify NSX-T objects.



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

> **warning:** If **Upgrade all clusters errand** is enabled, updating the Enterprise PKS tile with a new Linux stemcell triggers the rolling of every Linux VM in each Kubernetes cluster. Similarly, updating the Enterprise PKS tile with a new Windows stemcell triggers the rolling of every Windows VM in your Kubernetes clusters. This automatic rolling ensures that all your VMs are patched. To avoid workload downtime, use the resource configuration recommended in What Happens During Enterprise PKS Upgrades and Maintaining Workload Uptime.

(Optional) Resource Config

To modify the resource usage of Enterprise PKS and specify your PKS API load balancer, follow the steps below:

1. Select **Resource Config**.
2. (Optional) Edit other resources used by the **Pivotal Container Service** job. The **Pivotal Container Service** job requires a VM

with the following minimum resources:

CPU	Memory	Disk Space
2	8 GB	29 GB

Note: The automatic **VM Type** value matches the minimum recommended size for the **Pivotal Container Service** job. If you experience timeouts or slowness when interacting with the PKS API, select a **VM Type** with greater CPU and memory resources.

Step 3: Apply Changes

After configuring the Enterprise 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 CLI and the Kubernetes CLI help you interact with your Enterprise 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 Enterprise PKS Management Plane. See Creating the Enterprise PKS Management Planefor 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. For details, see Creating the PKS Compute Plane.

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 Enterprise 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 Authentication for Enterprise PKS

Follow the procedures in Setting Up Enterprise PKS Admin Users on vSpherein *Installing Enterprise PKS > vSphere*.

Next Steps

After installing Enterprise PKS on vSphere with NSX-T integration, complete the following tasks:

- Integrate VMware Harbor with Enterprise PKS to store and manage container images. For more information, see Integrating VMware Harbor Registry with Enterprise PKS.
- Setting Up Enterprise PKS Admin Users on vSphere.
- Creating an Enterprise PKS Cluster.

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

Advanced Configurations for Enterprise PKS on vSphere with NSX-T Data Center

Page last updated:

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

Post-Installation NSX-T Configurations

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

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

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

Using Proxies with Enterprise PKS on NSX-T

In this topic

Page last updated:

This topic describes how to use proxies with VMware Enterprise PKS with NSX-T.

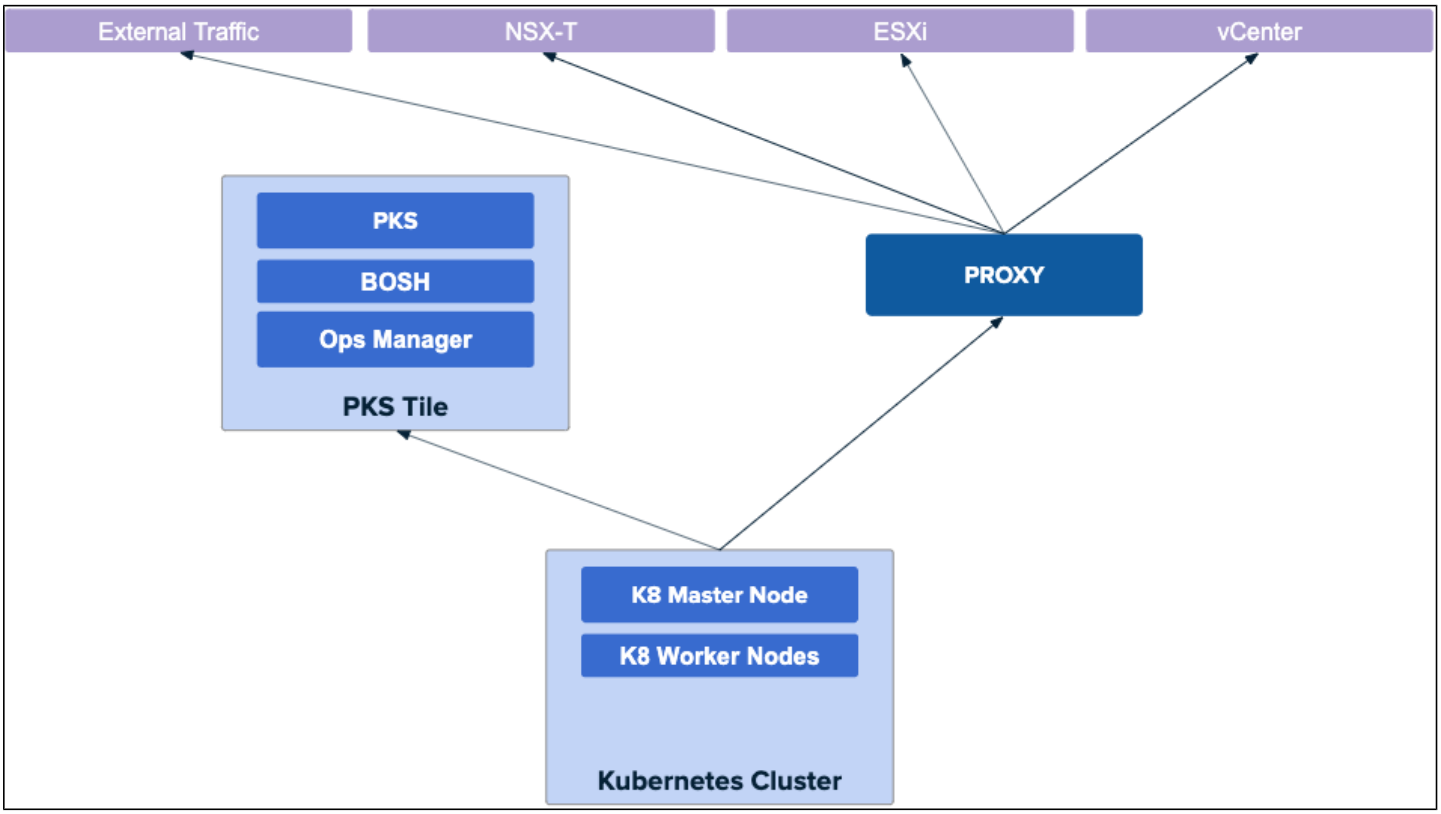
Overview

If your environment includes HTTP proxies, you can configure Enterprise PKS with NSX-T to use these proxies so that Enterprise PKS-deployed Kubernetes master and worker nodes access public Internet services and other internal services through a proxy.

In addition, Enterprise PKS proxy settings apply to the PKS API instance. When an Enterprise 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 Enterprise PKS components use the same proxy service.

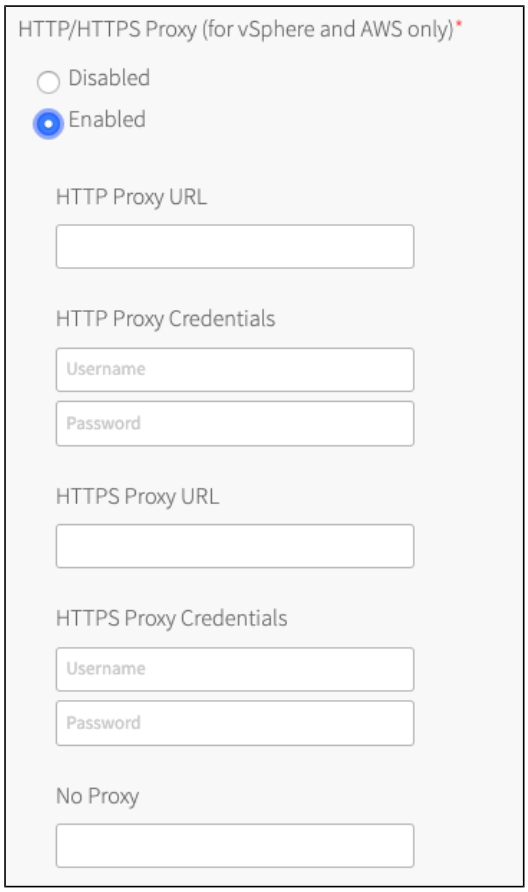
The following diagram illustrates the network architecture:



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 **Enterprise 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.



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

Configuring Enterprise PKS to use these proxies allows Enterprise PKS-deployed Kubernetes nodes to access public Internet services and other internal services.

> **Note**: This setting will not set the proxy for running Kubernetes workloads or pods.

To complete your global proxy configuration for all outgoing HTTP/HTTPS traffic from your Kubernetes clusters, perform the following steps:

a. To proxy outgoing HTTP traffic, enter the URL of your HTTP proxy endpoint under **HTTP Proxy URL**. For example, `http://myproxy.com:1234`.
b. (Optional) If your outgoing HTTP proxy uses basic authentication, enter the username and password in the **HTTP Proxy Credentials** fields.

c. To proxy outgoing HTTPS traffic, enter the URL of your HTTP proxy endpoint under **HTTPS Proxy URL**. For example, `http://myproxy.com:1234`.

> **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.

d. (Optional) If your HTTPS proxy uses basic authentication, enter the username and password in the **HTTPS Proxy Credentials** fields.

e. Under **No Proxy**, enter the comma-separated list of IP addresses that must bypass the proxy to allow for internal Enterprise PKS communication.

   The list should include `127.0.0.1` and `localhost`.

   Also include:

   - Your Enterprise PKS environment's CIDRs, such as the service network CIDR where your Enterprise PKS cluster is deployed, the deployment network CIDR, the node network IP block CIDR, and the pod network IP block CIDR.
   - If a component is communicating with Enterprise PKS or a registry using a hostname, such as the Harbor API FQDN, instead of an IP address you will need to add the corresponding FQDN to the **No Proxy** list.
   - 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 Enterprise PKS to IaaS traffic from going through the proxy: NSX Manager, vCenter Server, and all ESXi hosts.
   - 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:

```
127.0.0.1,localhost,
*.example1.com,
.example2.com,
example3.com,
198.51.100.0/24,
203.0.113.0/24,
192.0.2.0/24
```

> **Note**: By default the `10.100.0.0/8` and `10.200.0.0/8` IP address ranges, `.internal`, `.svc`, `.svc.cluster.local`, `.svc.cluster`, and your Enterprise PKS FQDN are not proxied. This allows internal Enterprise 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. Save the changes to the Enterprise PKS tile.

6. Proceed with any remaining Enterprise PKS tile configurations and deploy Enterprise PKS. See Installing Enterprise PKS on vSphere with NSX-T.

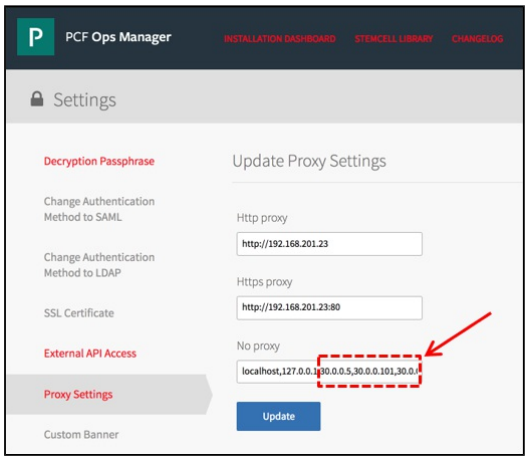
Enable Ops Manager and BOSH Proxy

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.

Configuring Multiple Tier-0 Routers for Tenant Isolation

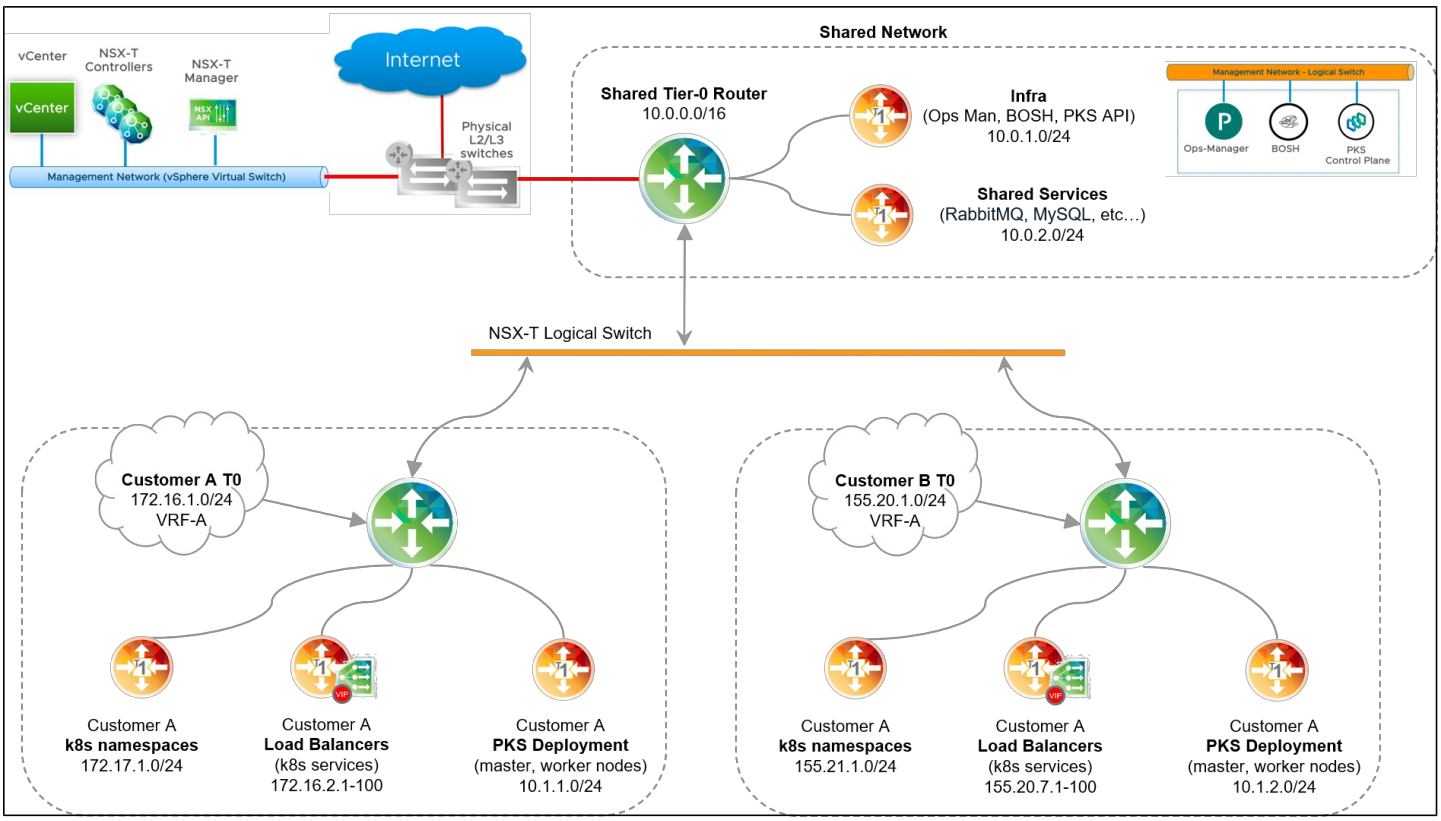
In this topic

Page last updated:

This topic describes how to create multiple NSX-T Tier-0 (T0) logical routers for use with VMware Enterprise PKS multi-tenant environments.

About Multi-T0 Router for Tenant Isolation

Enterprise 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

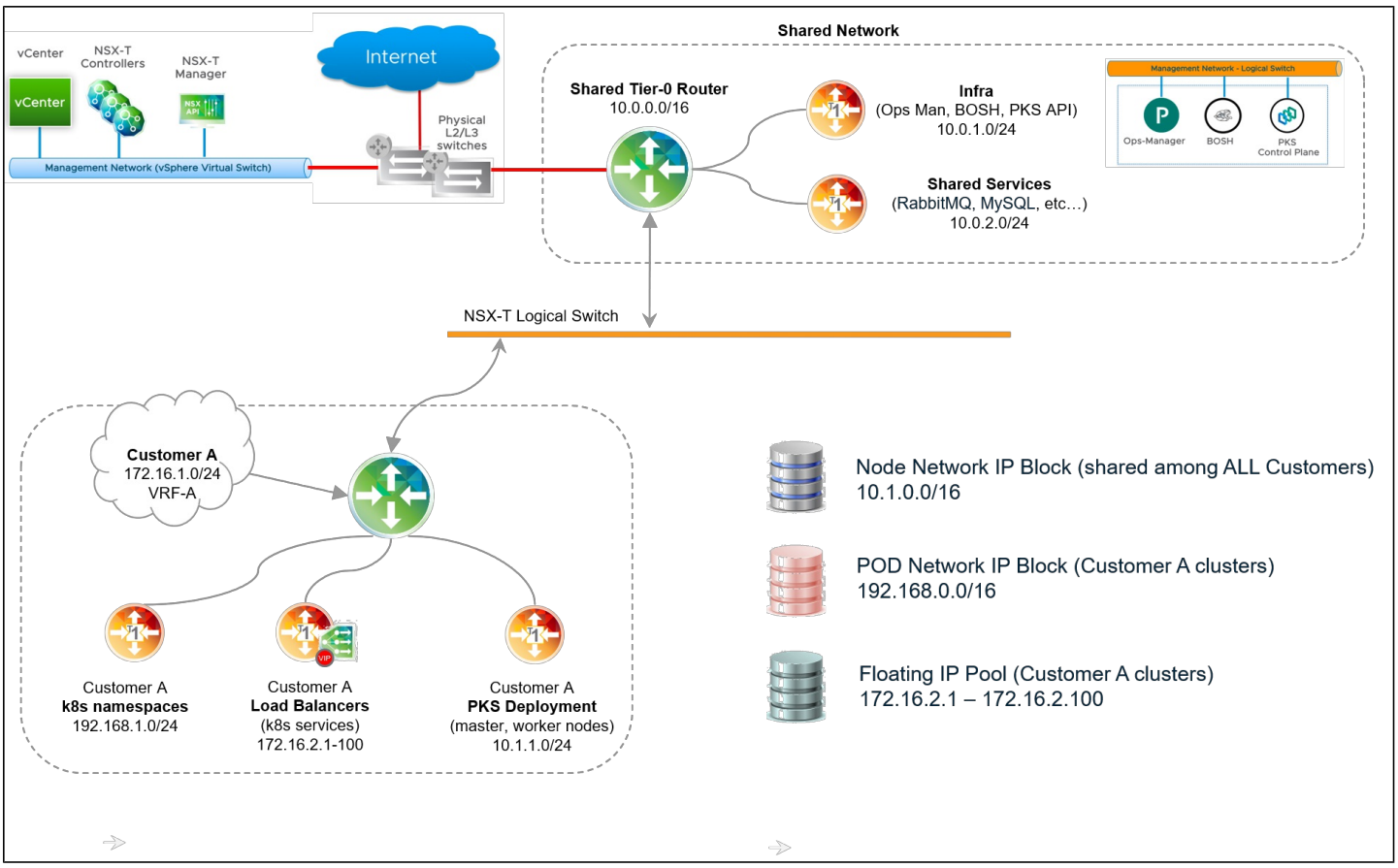
To implement Multi-T0, verify the following prerequisites:

- 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 Installing and Configuring NSX-T for Enterprise 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

Connect all NSX-T Edge Nodes using an overlay logical switch. This overlay network is used to transport traffic between the T0 routers.

Plan to allocate a network of sufficient size to accommodate all Tier-0 router interfaces that need to be connected to this 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. See Installing and Configuring NSX-T for Enterprise 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 Create T0 Logical 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 Logical 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 10.40.206.9/25 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 Rou	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 Create T0 Logical Router.

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 :

tier0-shared

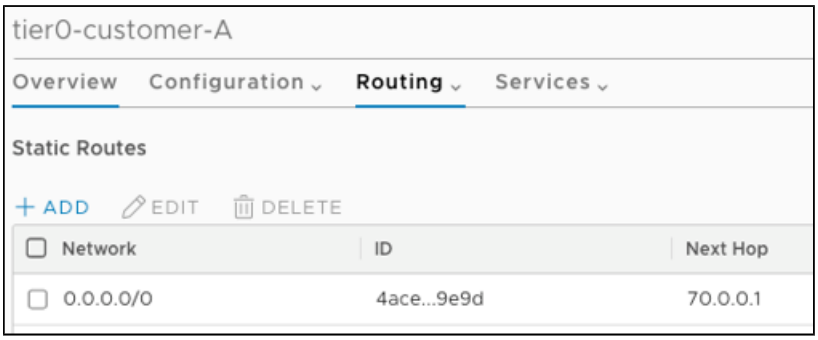
Overview Configuration Routing Services

Static Routes

+ ADD EDIT DELETE

Network	ID	Next Hop
0.0.0.0/0	0eaa...9e7c	90.0.0.1
192.168.201.0/24	d495...b030	90.0.0.1

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 Enterprise 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 Enterprise PKS deployments where NAT mode is configured on the Shared Tier-0 router cannot be re-purposed 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.

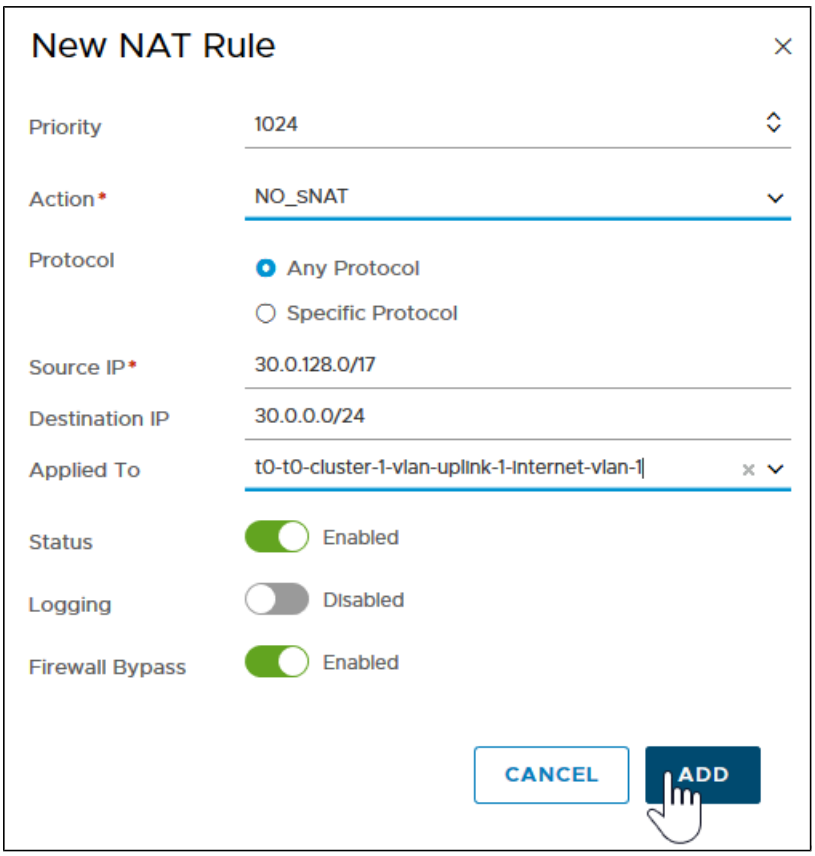
Note: NAT mode for Tenant Tier-0 routers is enabled by defining a non-routable custom Pods IP Block using a Network Profile. For more information, see Defining Network Profiles.

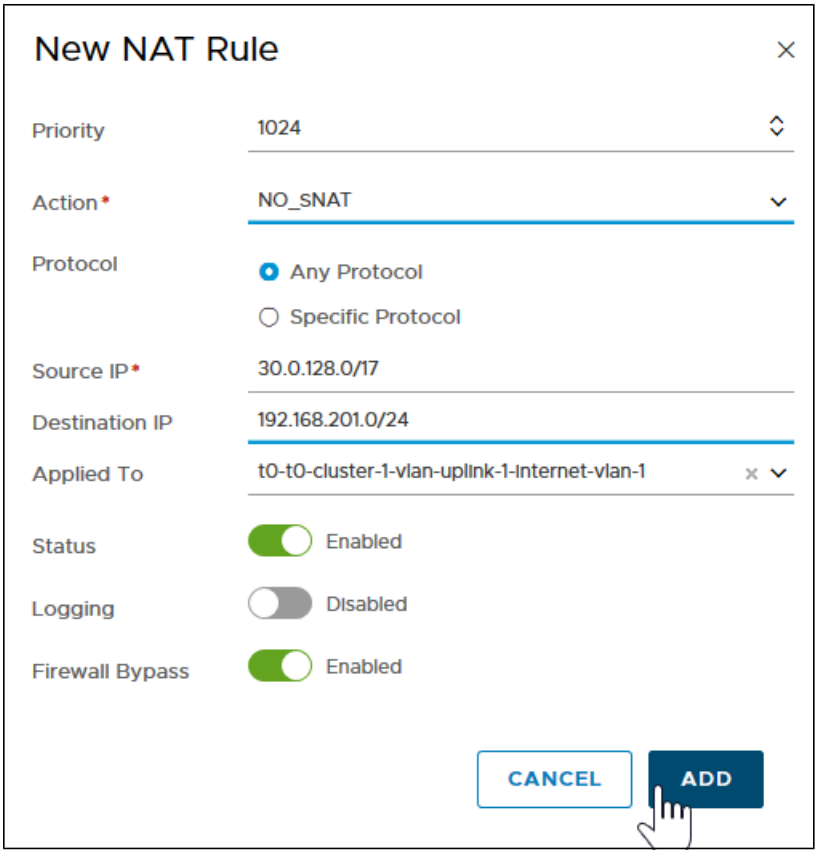
In a Multi-T0 environment with NAT mode, traffic on the Tenant Tier-0 network going from Kubernetes cluster nodes to PKS management components residing on the Shared Tier-0 router must bypass NAT rules. This is required because PKS-managed components such as BOSH Director connect to Kubernetes nodes 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.

tier0-customer-A

Overview | Configuration | Routing | Services

NAT | REFRESH

Total Rule Statistics | Last Updated: 11/12/2018, 3:51:22 PM

4 Active sessions 11177882 Packet count 14 GB Data

+ ADD EDIT DELETE

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	

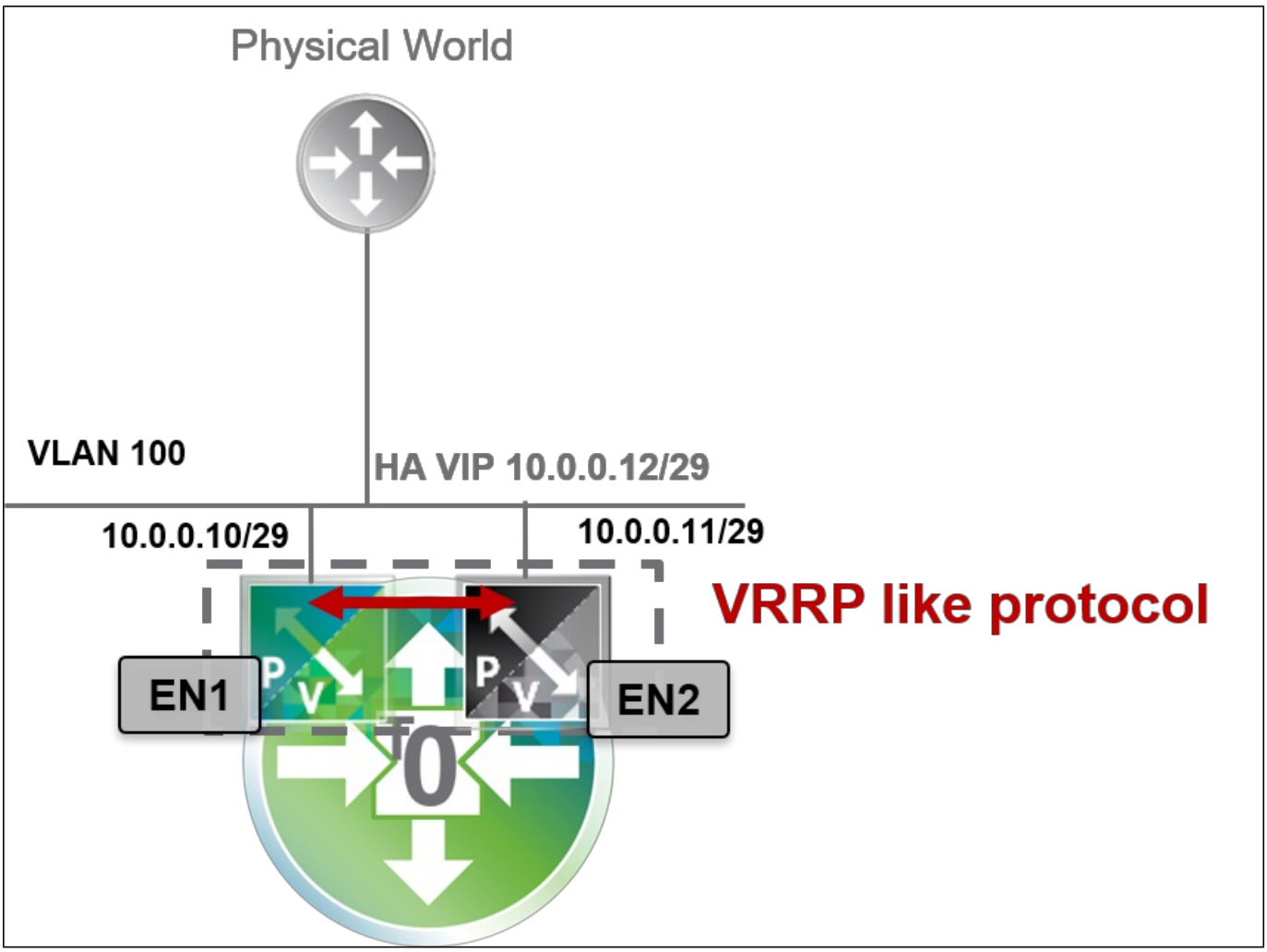
Step 10: Configure BGP on Each Tenant Tier-0 Router

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.

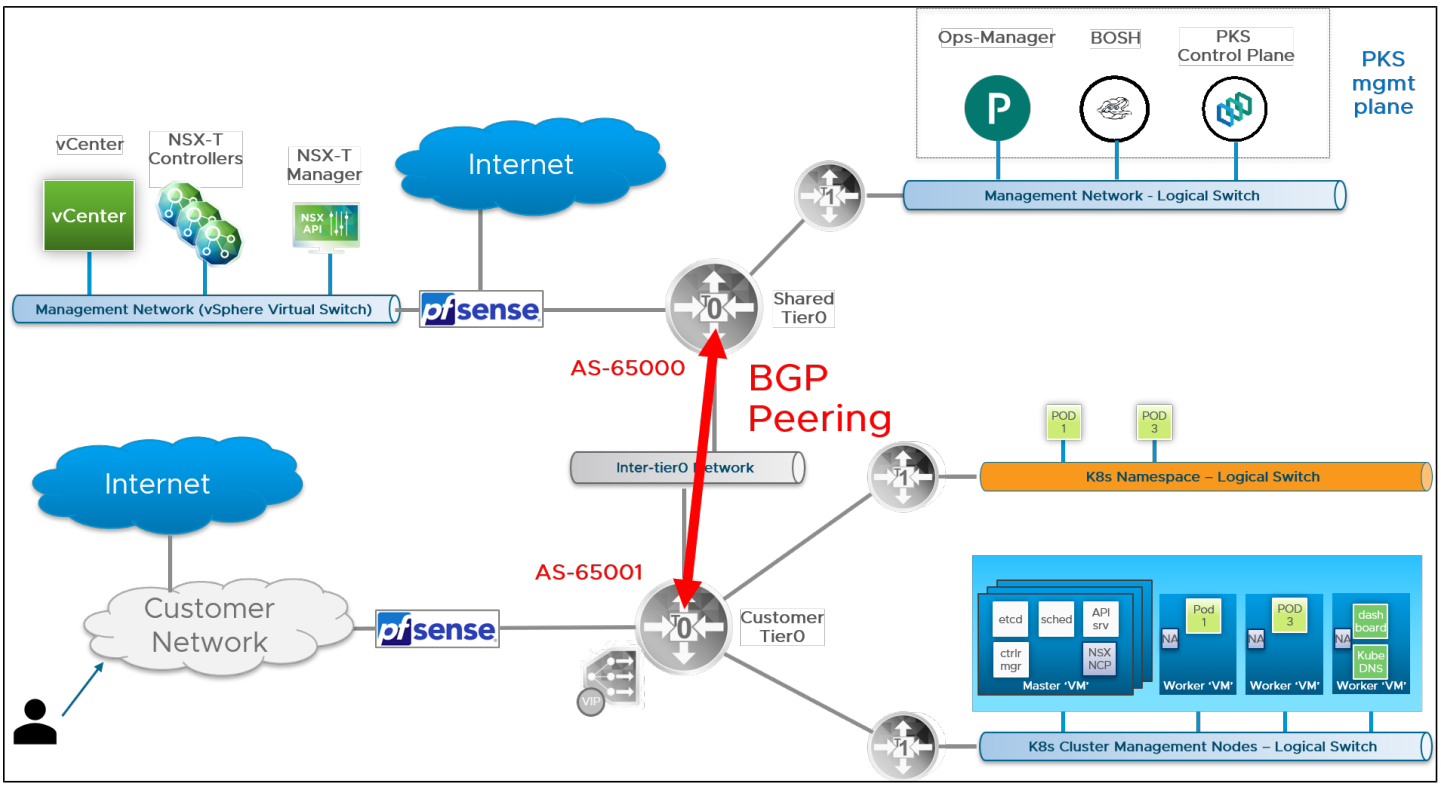
Failover of a logical router is triggered when the router is losing all of its BGP sessions. If multiple BGP sessions are established across different uplink interfaces of a Tier-0 router, failover will only occur if **all** such sessions are lost. Thus, to ensure high availability on the Shared and 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

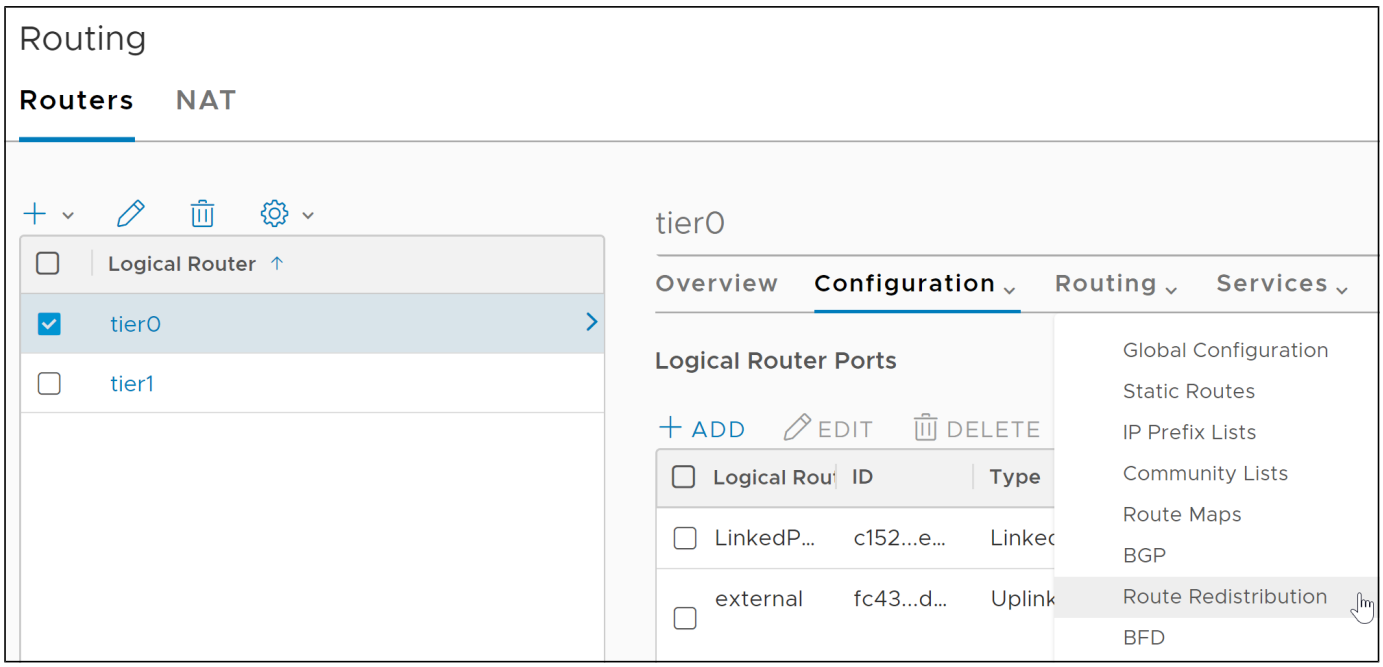
Once you have chosen the AS number for the Tenant Tier-0 router, configure BGP with the chosen AS number as follows:

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.

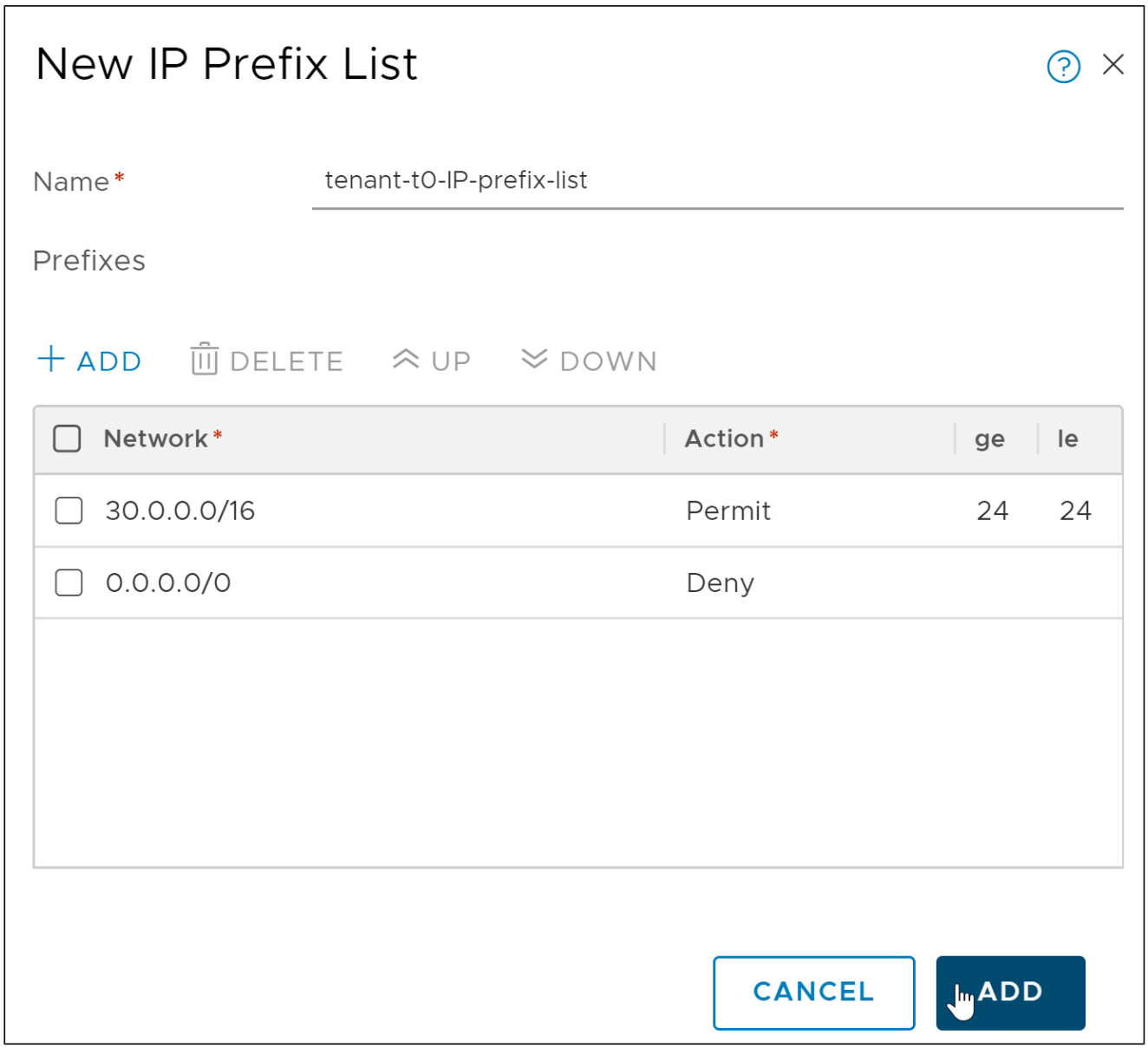
To configure an IP Prefix List for each Tenant Tier-0 router, follow the steps below:

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.

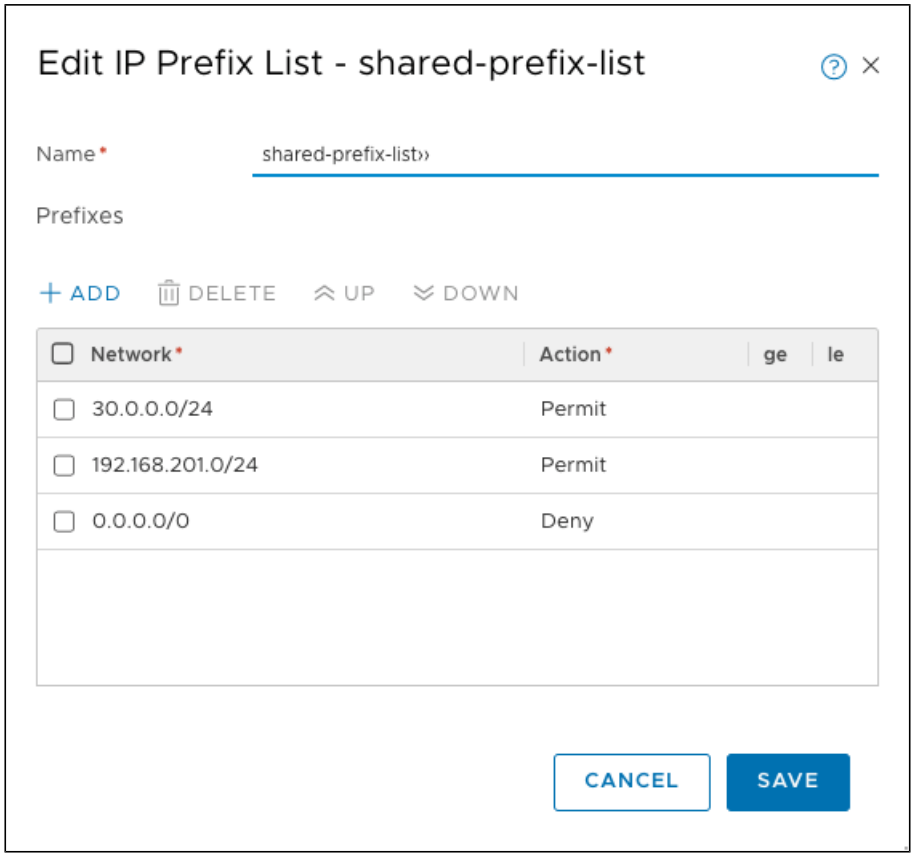
Configure BGP Route Distribution

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.

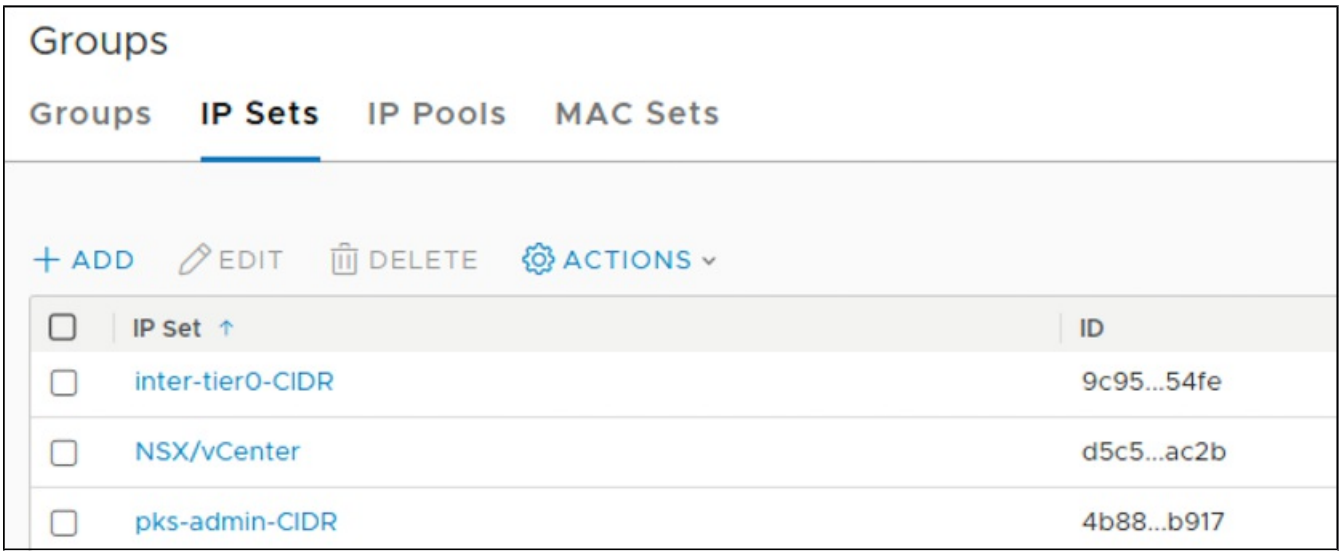
Secure Inter-Tenant Communications

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.

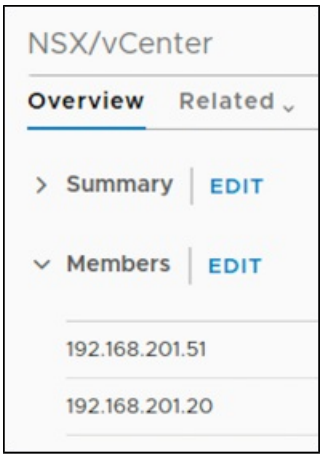
Step 1: Define IP Sets

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.

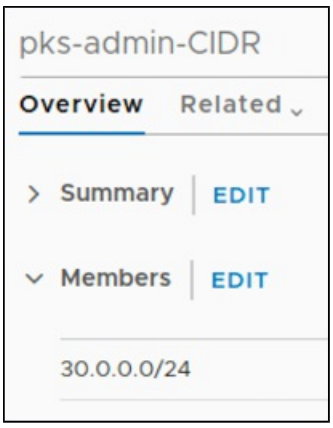
The image below shows a summary of the three required IP Sets you will need to create for securing Multi-T0 deployments:



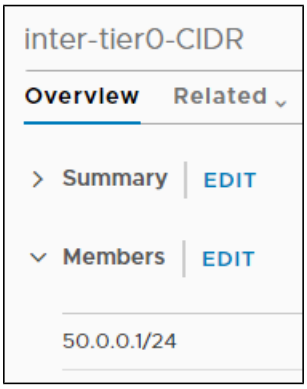
First, define an IP Set that includes the IP addresses for the NSX Manager and vCenter hosts. In the following IP Set example, `192.168.201.51` is the IP address for NSX and `192.168.201.20` is the IP address for vCenter.



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 Inter-T0 CIDR created during the base configuration.



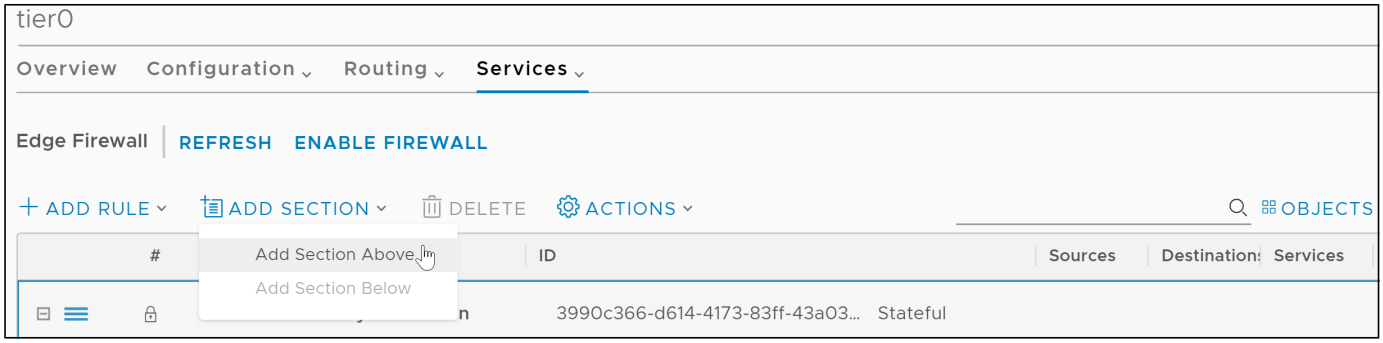
Note: These are the minimum IP Sets you need to create. You may want to define additional IP Sets for convenience.

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.

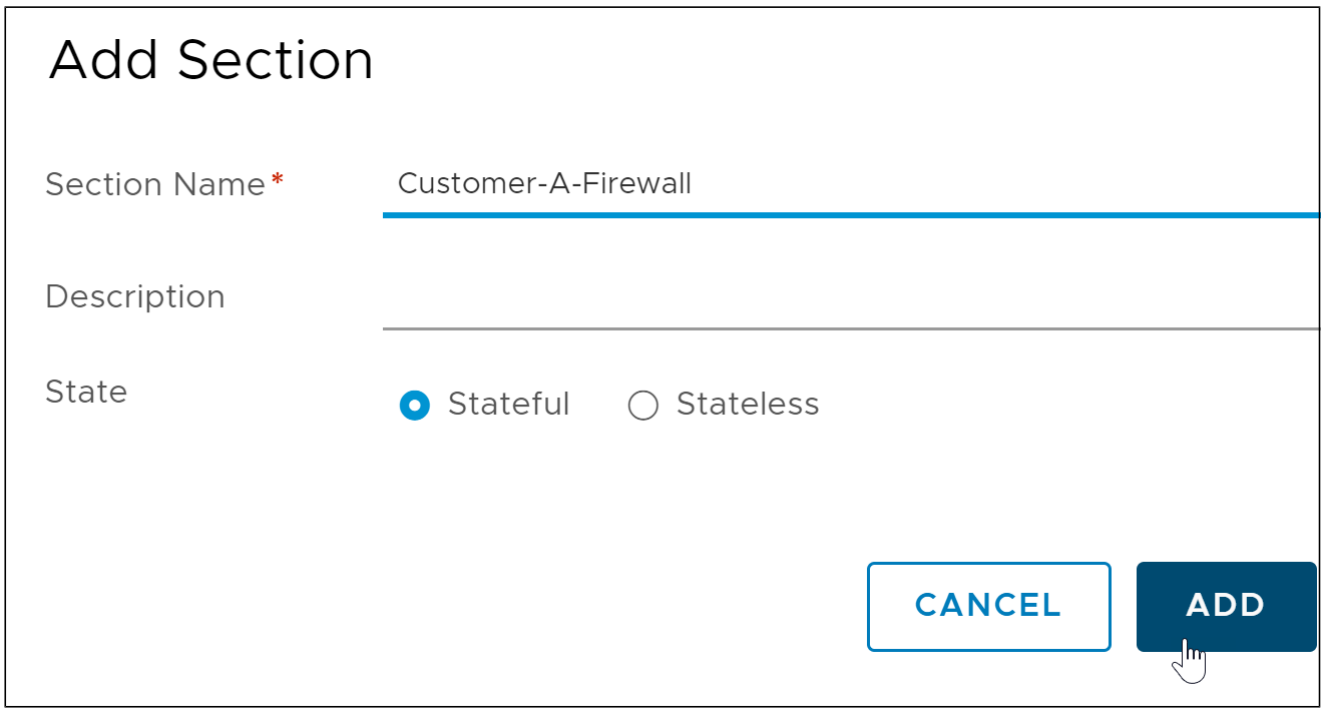
For each Tenant Tier-0 router, create an Edge Firewall and section as follows:

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:

tier0-customer-A

Overview | Configuration | Routing | Services

Edge Firewall | REFRESH | DISABLE FIREWALL

+ ADD RULE | ADD SECTION | DELETE | ACTIONS | OBJECTS

#	Name	ID	Direction	Sources	Destinations	Services	Action	Applied To
	InterTier0 PKS Firewall Rules	bd7edeb2-fb1b-4413-be27-1881b...	Stateful					
1	BGP	3159	IN_OUT	inter-tier0...	inter-ti...	Any	ALLOW	inter-tier0
2	ClusterA Masters to NSX/vCenter	3157	OUT	lb-pks-d3...	NSX/v...	Any	ALLOW	inter-tier0
3	k8s Nodes to BOSH	3158	OUT	all-pks-no...	BOSH	Any	ALLOW	inter-tier0
4	PKS to k8s Nodes	3154	IN	pks-admi...	all-pks...	Any	ALLOW	inter-tier0
5	Deny All	3153	IN_OUT	Any	Any	Any	DROP	inter-tier0

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`.

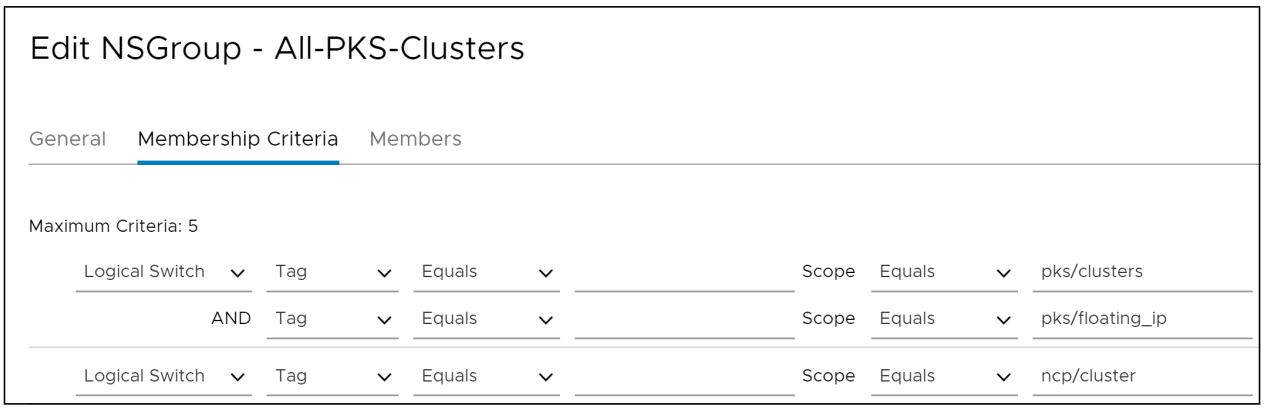
Secure Intra-Tenant Communications

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.

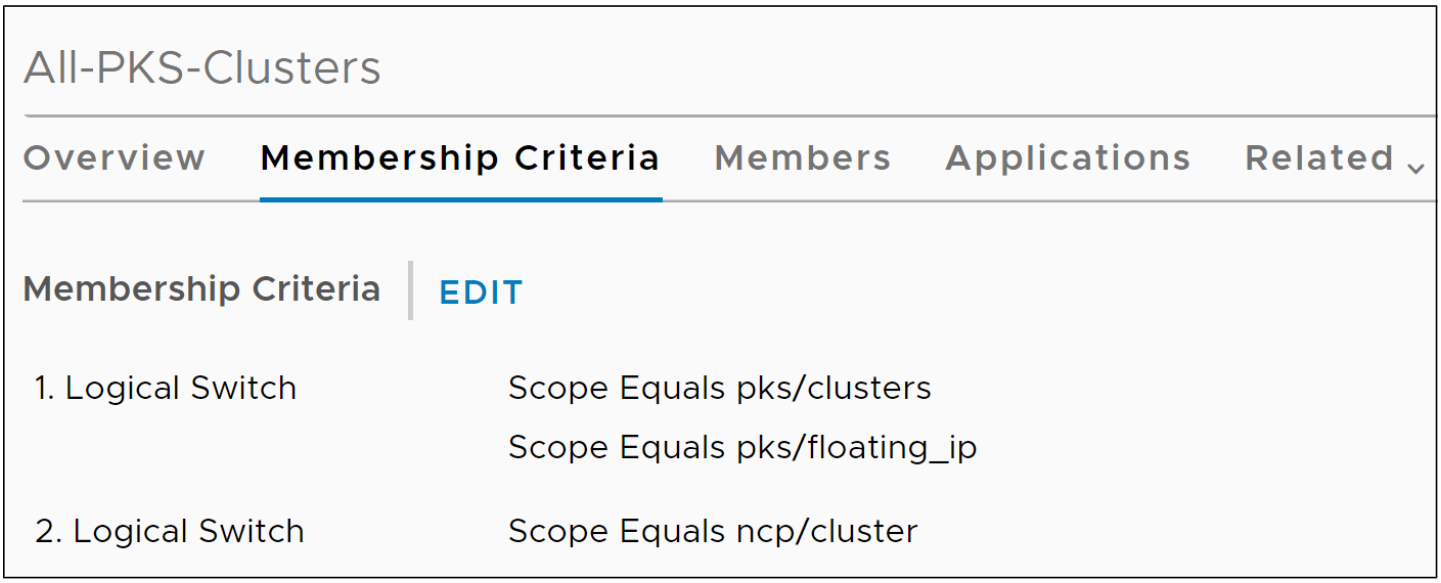
Step 1: Create NS Group for All Enterprise PKS Clusters

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.

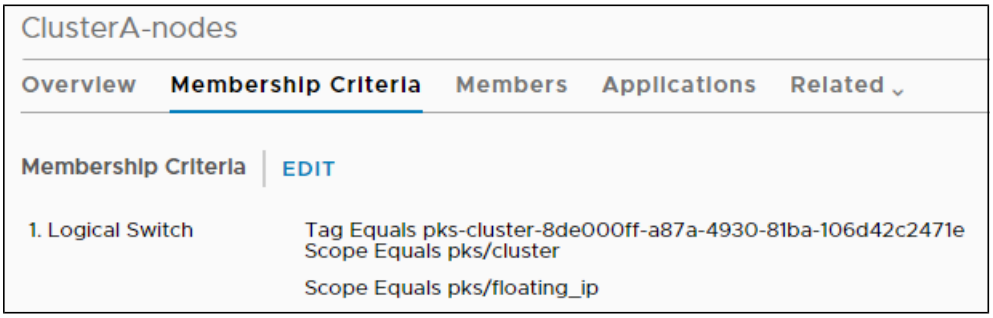
Create NS Group for Cluster Nodes

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

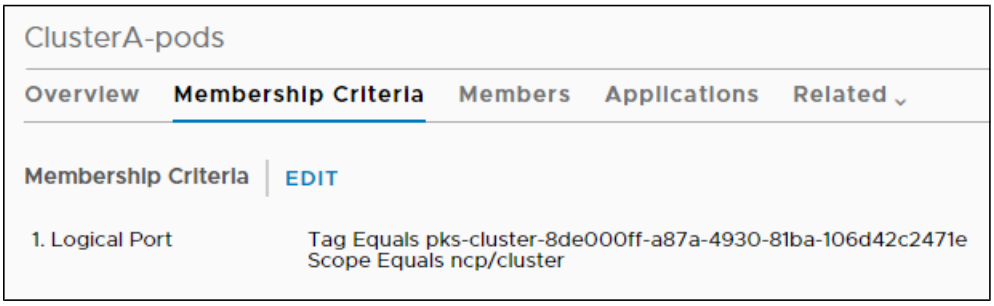
1. In NSX Manager, navigate to **Inventory > Groups > Groups** and click **Add new group**.

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:



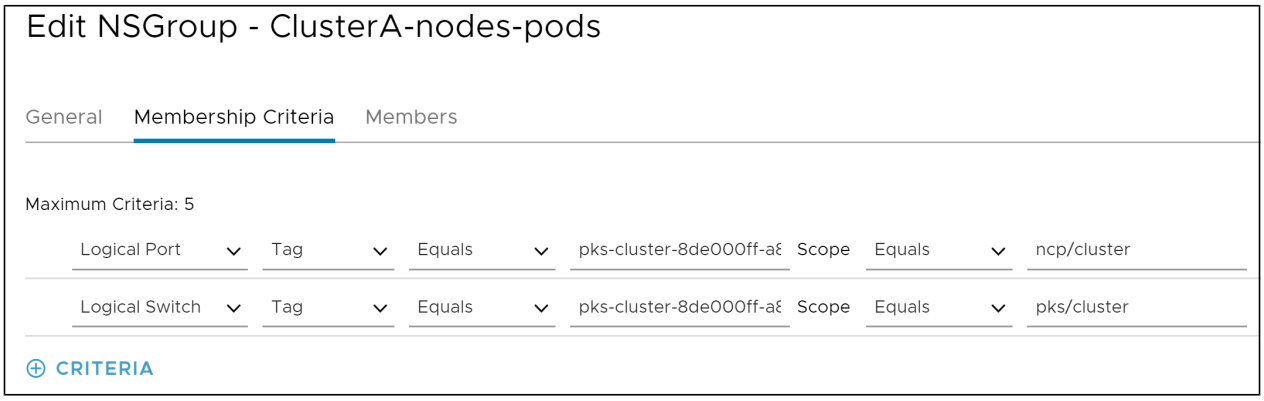
Create NS Group for Cluster Nodes and Pods

1. In NSX Manager, navigate to **Inventory > Groups > Groups** and click **Add new group**.

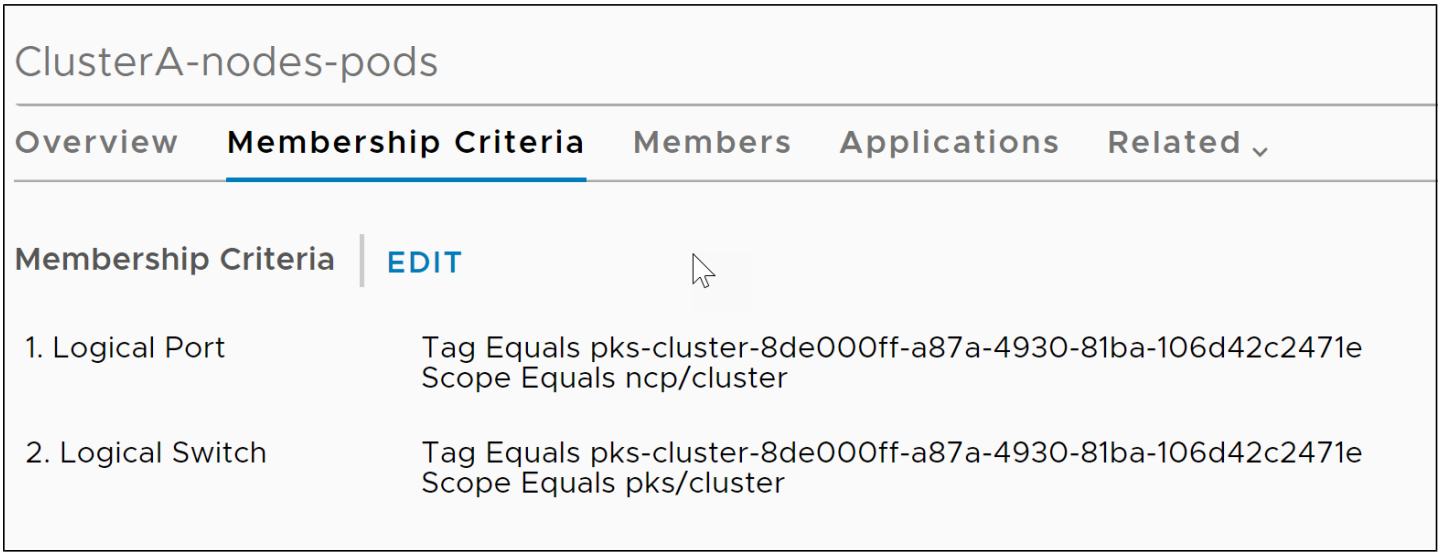
2. Configure the new NSGroup as follows:

   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.

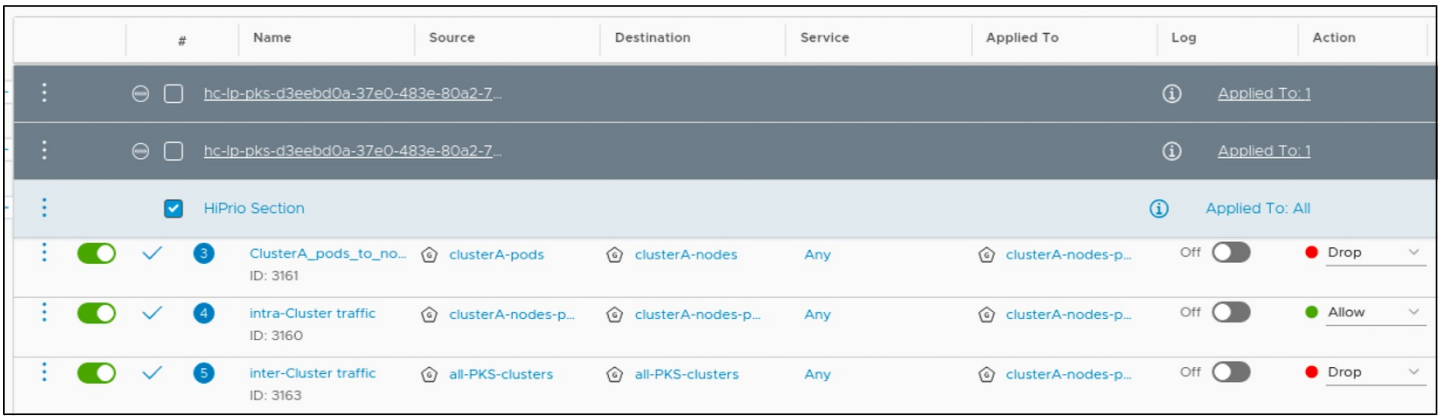
5. For **Service**, select **Any**.
6. For **Apply To**, select the `YOUR-CLUSTER-UUID-nodes-pods` NSGroup.
7. For **Action**, select **Drop**.

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.

Implementing a Multi-Foundation Enterprise PKS Deployment

In this topic

Page last updated:

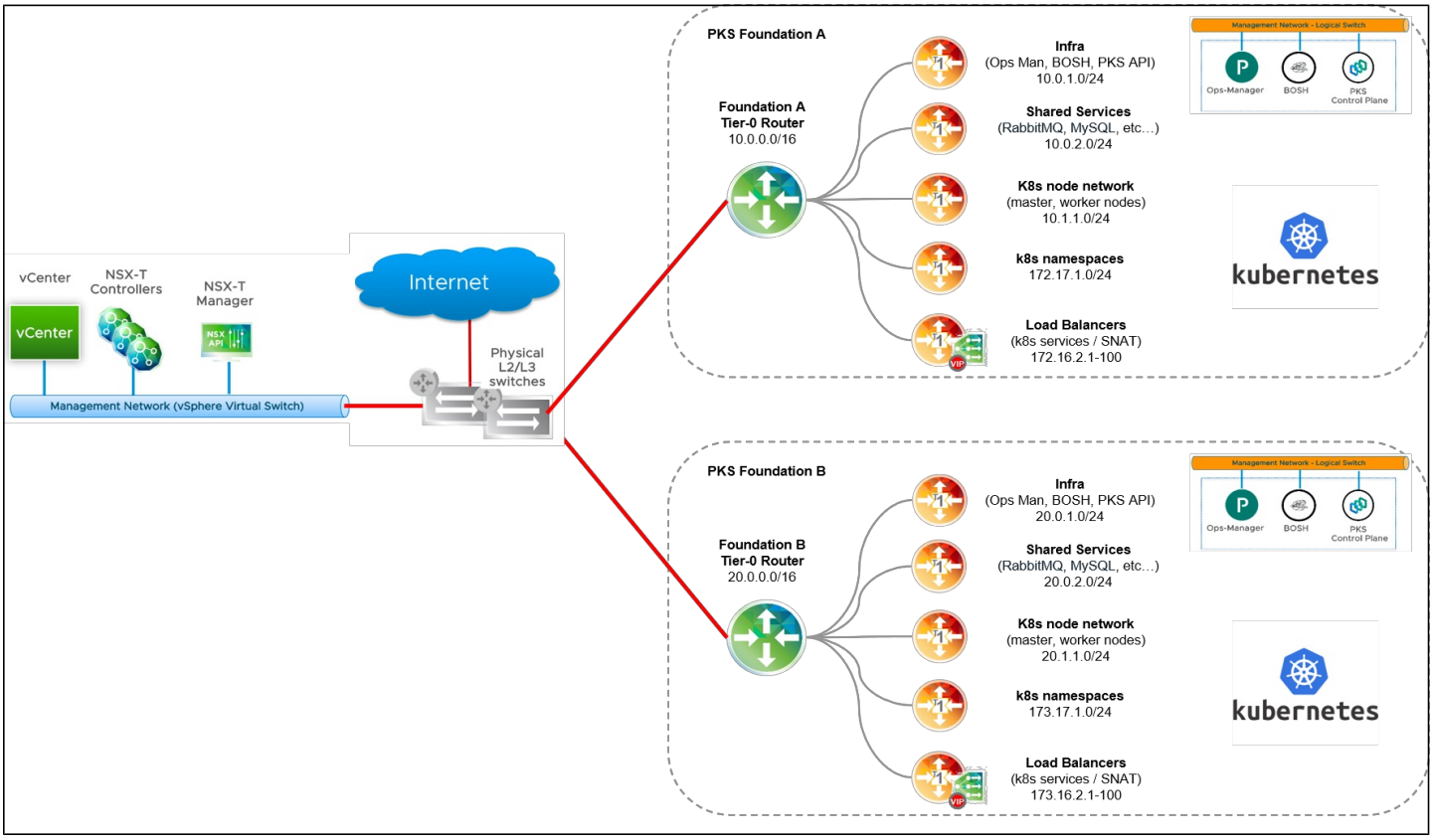
This topic describes how to deploy multiple instances of VMware Enterprise PKS on vSphere with NSX-T infrastructure.

About Multi-Foundation Enterprise PKS

A multi-foundation deployment of Enterprise PKS lets you install and run multiple instances of Enterprise PKS. The purpose of a multi-foundation deployment of Enterprise 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 Enterprise 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 Enterprise 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 Enterprise PKS instance.



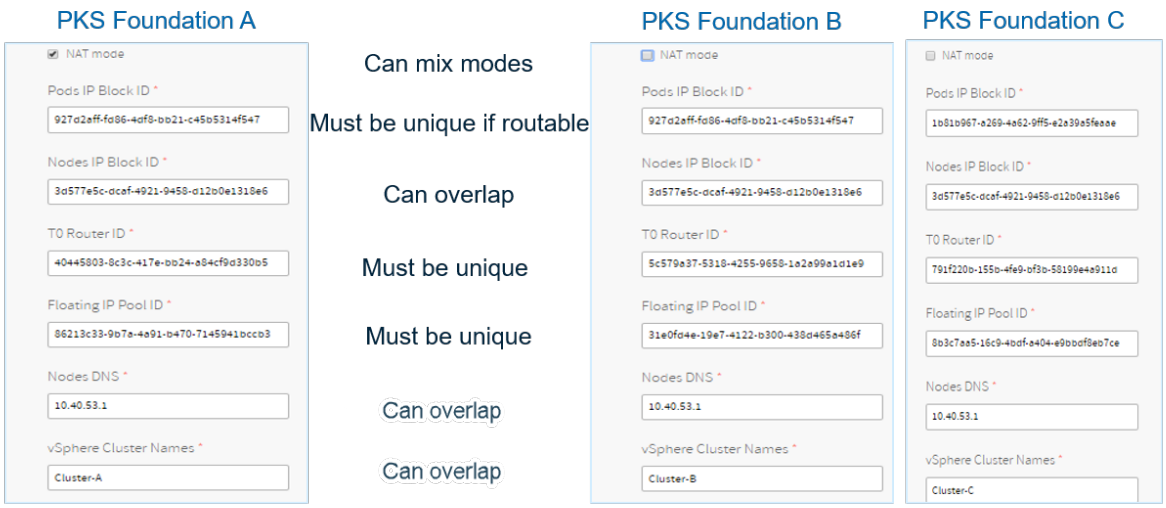
Requirements

To implement a multi-foundation Enterprise PKS topology, adhere to the following requirements:

- One Tier-0 router for each Enterprise 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.
- Enterprise PKS instances can be deployed in NAT and no-NAT mode. If more than one Enterprise 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 Enterprise PKS installations across three Tier-0 foundations. Key considerations to keep in mind with a multi-foundation Enterprise 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.

Verifying Enterprise PKS Installation on vSphere with NSX-T

In this topic

Page last updated:

This topic provides general instructions for verifying the installation of an NSX−T VM, either an NSX−T Manager VM or an NSX−T Edge Node VM, that you have deployed for use with Enterprise PKS.

Prerequisites

The following procedures must be executed from an SSH session into your Ops Manager. For information about creating an SSH keypair, see one of the following:

- Generate an SSH keypair for installing Ops Manager v2.6 on vSphere in the Pivotal Knowledge Base
- How to generate an SSH key pair for installing Ops Manager v2.6 on vSphere (71143) in the VMware Support Knowledge Base

Install the PKS and Kubectl CLIs

You will manage your environment using the Pivotal Container Service Command Line Interface (PKS CLI) and the Kubernetes CLI (kubectl). Complete the following steps to deploy the PKS and Kubernetes CLIs.

1. To copy CLIs to your Ops Manager using SCP, run the following PSCP commands:

   ```
   pscp c:\temp\pks-linux-amd64-PKS-VERSION ubuntu@IP-ADDRESS:/home/ubuntu/pks
   pscp c:\temp\kubectl-linux-amd64-STEMCELL-VERSION ubuntu@IP-ADDRESS:/home/ubuntu/pks
   ```

   Where:

   - `PKS-VERSION` is your PKS build ID.
   - `STEMCELL-VERSION` is your stemcell version ID.
   - `IP-ADDRESS` is the IP Address for Ops Manager host.

   For example:

   ```
   $ pscp c:\temp\pks\pks-linux-amd64-1.5.0-build.291 ubuntu@10.197.79.152:/home/ubuntu/pks
   $ pscp c:\temp\pks\kubectl-linux-amd64-1.14.5 ubuntu@10.197.79.152:/home/ubuntu/pks
   ```

2. To make the CLIs executable, run the following command:

   ```
   chmod +x pks-linux-amd64-VERSION
   chmod +x kubectl-linux-amd64-VERSION
   ```

   Where:

   - `PKS-VERSION` is your PKS build ID.
   - `STEMCELL-VERSION` is your stemcell version ID.

   For example:

   ```
   $ chmod +x pks-linux-amd64-1.5.0-build.291
   $ chmod +x kubectl-linux-amd64-1.14.5
   ```

3. To rename and move the files to your PATH, run the following commands:

   ```
   sudo mv pks-linux-amd64-VERSION /usr/local/bin/pks
   sudo mv kubectl-linux-amd64-VERSION /usr/local/bin/kubectl
   ```

   Where:

   - `PKS-VERSION` is your PKS build ID.
   - `STEMCELL-VERSION` is your stemcell version ID.

   For example:

   ```
   $ sudo mv pks-linux-amd64-1.5.0-build.291 /usr/local/bin/pks
   $ sudo mv kubectl-linux-amd64-1.14.5 /usr/local/bin/kubectl
   ```

Manage User Accounts Using UAA

You will manage accounts and authentication using User Account and Authentication (UAA) and the UAA Command Line Interface (UAAC).

Install UAAC and Configure UAA

Complete the following steps to deploy UAAC and UAA.

1. To install UAAC, run the following commands:

```
cd /usr/local/bin
sudo apt -y install ruby ruby-dev gcc build-essential g++
sudo gem install cf-uaac
```

2. To verify installation, run the following command:

```
uaac -v
```

3. To add PKS API hostname and IP address to a DNS server, complete the following steps:

   a. Retrieve the DNS server IP address from vSphere.
   b. Retrieve the PKS API Hostname from the **PKS tile** > **PKS API** > **API Hostname (FQDN)**.
   c. Add the DNS server IP address and the PKS API Hostname to the `/etc/hosts` file using the following format:

   ```
   IP-ADDRESS HOSTNAME
   ```

   Where:

   - `IP-ADDRESS` is the DNS Server IP Address.
   - `PKS-HOSTNAME` is the PKS API Hostname.

   For example:

   ```
   $ vi /etc/hosts
   127.0.0.1 localhost
   ...
   10.0.0.7 api.pks.vsphere.local
   ```

4. To target PKS, run the following UAAC command:

```
uaac target https://PKS-HOSTNAME:8443 --skip-ssl-validation
```

Where `PKS-HOSTNAME` is the PKS API Hostname.
For example:

```
$ uaac target https://api.comet.pks.local:8443 --skip-ssl-validation
Unknown key: Max-Age = 86400
Target: https://api.comet.pks.local:8443
```

5. To retrieve the UAA admin secret, complete the following steps:

   a. Retrieve the token from Ops Manager: **PKS Tile** > **Credentials** > **Pks Uaa Management Admin Client**
   b. Click **Link to Credential**.
   c. Copy the `secret` value.

6. To fetch the UAA admin token, run the following command:

```
uaac token client get admin -s SECRET
```

Where `SECRET` is the secret copied in the preceding step.
For example:

```
$ uaac token client get admin -s syMZA-FUlvdEXgXrgI-SDmkUVuP5rlzA
Unknown key: Max-Age = 86400
Successfully fetched token via client credentials grant.
Target: https://api.comet.pks.local:8443
Context: admin, from client admin
```

Add PKS Admin Users to UAA

Complete the following steps to add the pks-admin and k8s-admin admin accounts to UAA.

1. To add the PKS Admin user to UAA, run the following commands:

```
uaac user add pks-admin --emails ADMIN-EMAIL-ADDRESS -p PASSWORD
uaac member add pks.clusters.admin pks-admin
```

Where:

- `ADMIN-EMAIL-ADDRESS` is the email address for the admin user account.
- `PASSWORD` is the password for the user account.

For example:

```
$ uaac user add pks-admin --emails pks-admin@example.com -p aZ1by2cX3d4!
user account successfully added


$ uaac member add pks.clusters.admin pks-admin
success
```

2. To add the PKS Manage user to UAA, run the following commands:

```
uaac user add k8s-admin --emails MANAGE-EMAIL-ADDRESS -p PASSWORD
uaac member add pks.clusters.manage k8s-admin
```

Where:

- `MANAGE-EMAIL-ADDRESS` is the email address for the PKS Manage user account.
- `PASSWORD` is the password for the user account.

For example:

```
$ uaac user add k8s-admin --emails k8s-admin@example.com -p aZ1by2cX3d4!
user account successfully added


$ uaac member add pks.clusters.manage k8s-admin
success
```

Log In to PKS

Validate the new UAA admin account by logging in to PKS.

1. To log in to PKS as `pks-admin` run the following command:

```
pks login -a "PKS-HOSTNAME" -u "pks-admin" -p "PASSWORD" -k
```

Where:

- `PKS-HOSTNAME` is the PKS API Hostname.
- `PASSWORD` is the password for the user account.

For example:

```
$ pks login -a "api.pks.vsphere.local" -u "pks-admin" -p "aE1ry2cJ3d2!" -k


API Endpoint: api.pks.vsphere.local
User: pks-admin
Login successful.
```

2. To log in to PKS as `k8s-admin` run the following command:

```
pks login -a "PKS-HOSTNAME" -u "k8s-admin" -p "PASSWORD" -k
```

Where:

- `PKS-HOSTNAME` is the PKS API Hostname.
- `PASSWORD` is the password for the user account.

For example:

```
$ pks login -a "api.comet.pks.local" -u "k8s-admin" -p "rtXare1!" -k


API Endpoint: api.comet.pks.local
User: k8s-admin
Login successful.
```

Manage PKS Using BOSH

Create a BOSH Alias

1. To create a BOSH alias, complete one of the following:
   - To create a BOSH alias using the `alias` command, complete the following:
     1. Go to **BOSH Director tile** > **Credentials** > **Bosh Commandline Credentials**.
     2. Click **Link to Credential**.
     3. Copy the `credential` string, including the surrounding double quotes.
     4. To create a BOSH alias run the following command:

```
alias bosh=CREDENTIAL-STRING
```

Where:

- `CREDENTIAL-STRING` is the entire `credential` string from the **Bosh Commandline Credentials**, including the surrounding double quotes.

For example:

```
$ alias bosh="BOSH_CLIENT=ops_manager BOSH_CLIENT_SECRET=sNc1itad BOSH_CA_CERT=/var/tempest/workspaces/default/root_ca_certificate BOSH_ENVIRONMENT=ip-from-bosh_commandline_
```

- To create a BOSH alias using the `source` command, complete the following:

  1. Go to **BOSH Director tile** > **Credentials** > **Bosh Commandline Credentials**.
  2. Click **Link to Credential**.
  3. Copy the `credential` string content.
  4. Create a file named `bosh_evn.sh` with the following modified `credential` string content:

     ```
     export BOSH_CLIENT=BOSH-CLIENT
     export BOSH_CLIENT_SECRET=BOSH-CLIENT-SECRET
     export BOSH_CA_CERT=BOSH-CA-CERT
     export BOSH_ENVIRONMENT=BOSH-ENVIRONMENT-IP
     alias om=om-linux
     ```

     Where:

     - `BOSH-CLIENT` is the `BOSH_CLIENT` value from the BOSH command line credentials.
     - `BOSH-CLIENT-SECRET` is the `BOSH_CLIENT_SECRET` value from the BOSH command line credentials.
     - `BOSH-CA-CERT` is the `BOSH_CA_CERT` path value from the BOSH command line credentials.
     - `IP-ADDRESS-OF-BOSH` is the `BOSH_ENVIRONMENT` IP Address value from the BOSH command line credentials.

     For example:

     ```
     export BOSH_CLIENT=ops_manager
     export BOSH_CLIENT_SECRET=sgFGsM_cOPrw5lhrIeIX1JDFxPR6LcgR
     export BOSH_CA_CERT=/var/tempest/workspaces/default/root_ca_certificate
     export BOSH_ENVIRONMENT=10.0.16.5
     ```

  5. Run the following command:

     ```
     source bosh_env.sh
     ```

List BOSH Deployments

1. To retrieve a list of BOSH deployments, run the following command:

   ```
   bosh deployments
   ```

List BOSH VMs

1. To retrieve a list of BOSH VMs, run the following command:

   ```
   bosh vms
   ```

List BOSH Components

1. To retrieve a list of bosh components. run the following command:

   ```
   bosh releases
   ```

Use BOSH to SSH to a Kubernetes Node

1. To SSH to a Kubernetes node, run the following command:

   ```
   bosh ssh -d 'SERVICE-ID' DESTINATION-ID
   ```

   Where:

   - `SERVICE-ID` is your node's service instance id.
   - `DESTINATION-ID` is your destination host's ID. For example, you can provide an ID such as a master host, UAA host or Diego Cell ID as a destination. For more information about `bosh ssh`, see SSH in the *Cloud Foundry BOSH Commands* documentation.

   For example:

   ```
   $ bosh ssh -d 'service-instance_9dd62330-fcea-469f-a50d-27c746622662' master/b1f99643-6905-4427-a01c-c26fc9101d40
   ```

Use BOSH to Delete a Kubernetes Cluster

If you want to delete a Kubernetes cluster that failed to deploy, and you cannot delete the cluster using the PKS CLI, you can delete the cluster using BOSH.

1. To delete a Kubernetes cluster, run the following command:

   ```
   bosh -e pks delete-deployment -d SERVICE-ID
   ```

   Where `SERVICE-ID` is your node's service instance id.
   For example:

   ```
   $ bosh -e pks delete-deployment -d service-instance_3bd9069f-aafd-4525-a145-f3e48570c055
   ```

Manage a Kubernetes Cluster using PKS and Kubernetes

You can manage your Kubernetes clusters using both PKS and Kubernetes command line commands.

Retrieve Cluster Credentials

1. To retrieve your Kubernetes cluster credentials, run the following:

   ```
   pks get-credentials CLUSTER-NAME
   ```

   Where `CLUSTER-NAME` is the name of your cluster.

Validate the Kubernetes Configuration File

1. To verify that the Kubernetes configuration file exists, run the following command:

   ```
   ls -l ~/.kube/config
   ```

Connect to Kubernetes Cluster

To manage a Kubernetes cluster using Kubernetes CLI commands, you must first connect to the cluster.

1. Retrieve your Kubernetes cluster credentials. For more information, see Retrieve Cluster Credentials above.
2. Verify that the Kubernetes configuration file exists. For more information, see Validate the Kubernetes Configuration File above.
3. To create a DNS record for the Kubernetes cluster and master node IP address, complete the following steps:
   a. Run the following command to get the information:

      ```
      pks cluster CLUSTER-NAME
      ```

   b. Locate the "Kubernetes Master Host" and "Kubernetes Master IP(s)" values.
   c. To create a DNS record, create or modify the `/etc/hosts` file to have a new line using these values:

      ```
      MASTER-IP MASTER-HOST
      ```

      Where:

      - `MASTER-IP` is the Kubernetes Master IP(s) collected in the preceding step.
      - `MASTER-HOST` is the Kubernetes Master Host collected in the preceding step.

      For example:

      ```
      $ vi /etc/hosts
      10.197.100.130 tea-cluster
      ```

4. To connect to the Kubernetes cluster, run the following command:

   ```
   kubectl config use-context CLUSTER-NAME
   ```

5. To confirm your connection list your cluster nodes. For more information see List Kubernetes Cluster Nodes below.

List Kubernetes Cluster Nodes

1. To list the cluster nodes of the currently connected Kubernetes cluster, run the following command:

   ```
   kubectl get nodes
   ```


Google Cloud Platform (GCP)

Page last updated:

This topic lists the procedures to follow to install VMware Enterprise PKS on Google Cloud Platform (GCP).

Install Enterprise PKS on GCP

To install Enterprise PKS on GCP, follow the instructions below:

- Prerequisites and Resource Requirements
- Installing and Configuring Ops Manager on GCP
- Creating Service Accounts in GCP for Enterprise PKS
- Creating a GCP Load Balancer for the PKS API
- Installing Enterprise PKS on GCP
- Setting Up Enterprise PKS Admin Users on GCP

Install the PKS and Kubernetes CLIs

The PKS CLI and Kubernetes CLI help you interact with your Enterprise 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

In this topic

Page last updated:

This topic describes the prerequisites and resource requirements for installing VMware Enterprise PKS on Google Cloud Platform (GCP).

Prerequisites

You can install Enterprise PKS on GCP manually or by using Terraform.

- Prerequisites for Installing Enterprise PKS Manually
- Prerequisites for Installing Enterprise PKS Using Terraform

Installing Enterprise PKS Manually

If you are installing Enterprise PKS manually, do the following before deploying Enterprise PKS:

1. Review Resource Requirements below.
2. Install and configure Ops Manager. To install and configure Ops Manager, follow the instructions in Installing and Configuring Ops Manager on GCP.
3. Create service accounts for Kubernetes master and worker nodes. To create the service accounts, follow the instructions in Creating Service Accounts in GCP for Enterprise PKS.

   Note: Perform this step after you install and configure Ops Manager.

4. Create a load balancer to access the PKS API from outside the network and run `pks` commands from your local workstation. To create a load balancer in GCP, follow the instructions in Creating a GCP Load Balancer for the PKS API.

   Note: Perform this step before you install Enterprise PKS. After you install Enterprise 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.

Installing Enterprise PKS Using Terraform

If you are installing Enterprise PKS using Terraform, do the following before deploying Enterprise PKS:

1. Review Resource Requirements below.
2. Install and configure Ops Manager. To install and configure Ops Manager, follow the instructions in Installing and Configuring

Ops Manager on GCP.

Resource Requirements

Installing Ops Manager and Enterprise 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 Pivotal Container Service VM as follows:

Number of Pods	Storage (Persistent Disk) Requirements
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 Enterprise 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 and Configuring Ops Manager on GCP

In this topic

Page last updated:

This topic describes how to install and configure Ops Manager before deploying VMware Enterprise PKS on Google Cloud Platform (GCP).

Prerequisites

You use Ops Manager to install and configure Enterprise PKS. Before you install Ops Manager, review GCP Prerequisites and Resource Requirements.

Install and Configure Ops Manager

Multiple versions of Ops Manager are compatible with each version of Enterprise PKS. To confirm that your Ops Manager version is compatible with the version of Enterprise PKS that you want to install, see the Enterprise PKS Release Notes.

To install and configure Ops Manager, follow either the manual or Terraform instructions in the table below:

Version	Manual Instructions	Terraform Instructions
Ops Manager v2.6	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 v2.7	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

Next Installation Step

- If you installed Ops Manager manually, proceed to Creating Service Accounts in GCP for Enterprise PKS.
- If you installed Ops Manager using Terraform, proceed to Installing Enterprise PKS on GCP.

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

Creating Service Accounts in GCP for Enterprise PKS

In this topic

Page last updated:

This topic describes the steps required to create service accounts for VMware Enterprise 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**.

Next Installation Step

To create a load balancer in GCP, follow the instructions in Creating a GCP Load Balancer for the PKS API

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

Creating a GCP Load Balancer for the PKS API

In this topic

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 VMware Enterprise 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 Enterprise 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 Enterprise 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 Enterprise 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 to Deploy Ops Manager on GCP Manually*.
- 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 Enterprise PKS

Follow the instructions in Installing Enterprise PKS on GCP to deploy Enterprise PKS. After you finish installing Enterprise 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 Enterprise PKS on GCP

In this topic

Page last updated:

This topic describes how to install and configure VMware Enterprise 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 Enterprise 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 Enterprise PKS

To install Enterprise 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 **Enterprise PKS** in the left column, click the plus sign to add this product to your staging area.

Step 2: Configure Enterprise PKS

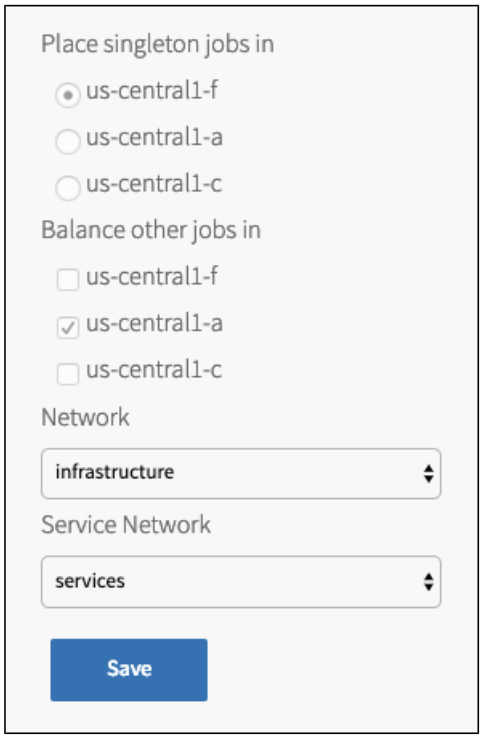
Click the orange **Enterprise PKS** tile to start the configuration process.

warning: When you configure the Enterprise 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 Enterprise PKS fails.

Assign AZs and Networks

To configure the availability zones (AZs) and networks used by the Enterprise PKS control plane:

1. Click **Assign AZs and Networks**.
2. Under **Place singleton jobs in**, select the availability zone (AZ) where you want to deploy the PKS API VM as a singleton job.



3. Under **Balance other jobs in**, select the AZ for balancing other Enterprise PKS control plane jobs.

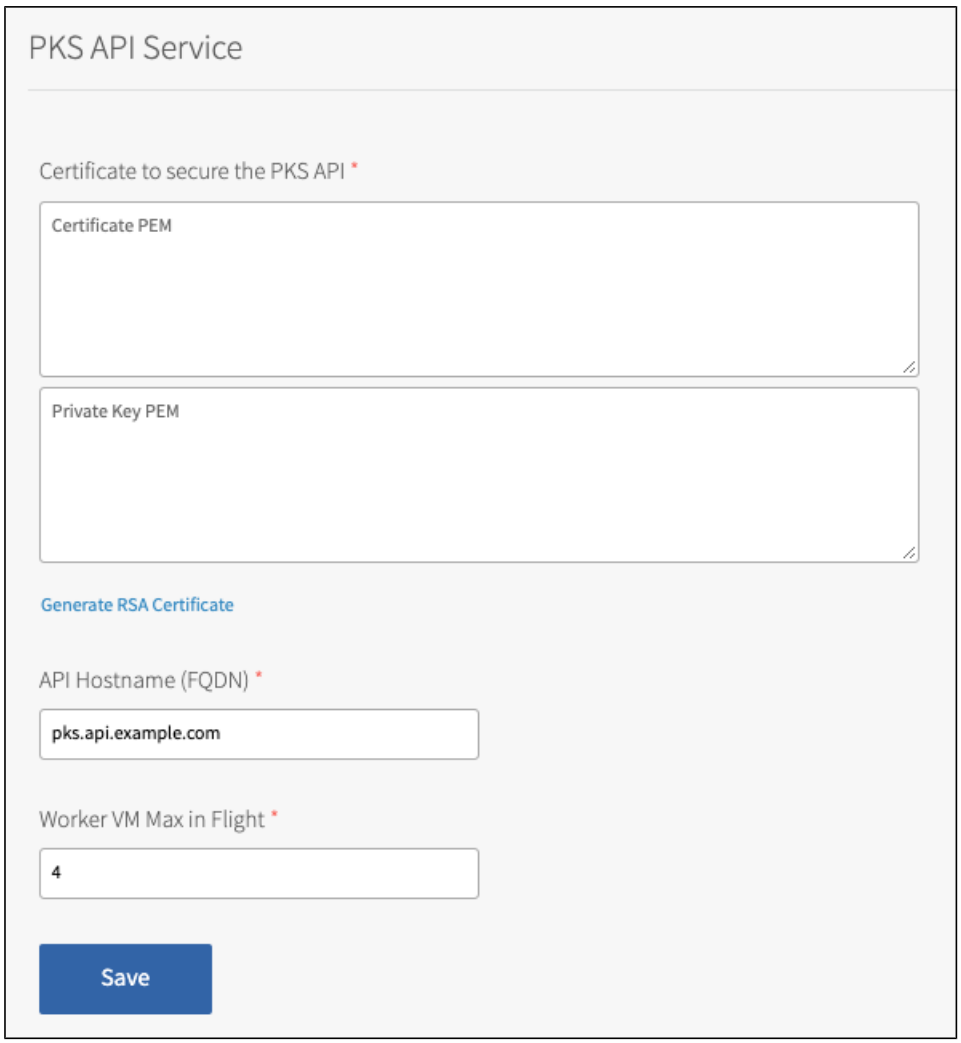
> **Note**: You must specify the **Balance other jobs in** AZ, but the selection has no effect in the current version of Enterprise PKS.

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

PKS API

Perform the following steps:

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

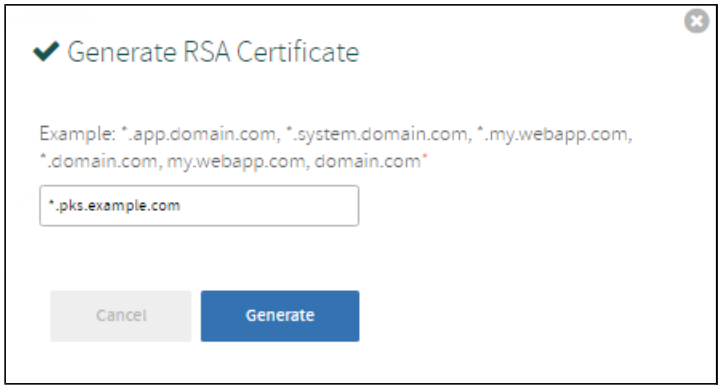


The certificate that you supply should cover the specific subdomain that routes to the PKS API VM with TLS termination on the ingress.

> **warning:** TLS certificates generated for wildcard DNS records only work for a single domain level. For example, a certificate generated for `*.pks.EXAMPLE.com` does not permit communication to `*.api.pks.EXAMPLE.com`. If the certificate does not contain the correct FQDN for the PKS API, calls to the API will fail.

You can enter your own certificate and private key pair, or have Ops Manager generate one for you.
To generate a certificate using Ops Manager:

a. If the **Generate RSA Certificate** option is not available, click **Change**.
b. Click **Generate RSA Certificate**.
c. Enter the domain for your API hostname. This can be a standard FQDN or a wildcard domain.
d. 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 value. When you create or resize a cluster, the `max_in_flight` value limits the number of component instances that can be created or started simultaneously. By default, the `max_in_flight` value is set to `4`, which means that up to four component instances are simultaneously created or started at a time.

5. Click **Save**.

Plans

A plan defines a set of resource types used for deploying a cluster.

Activate a Plan

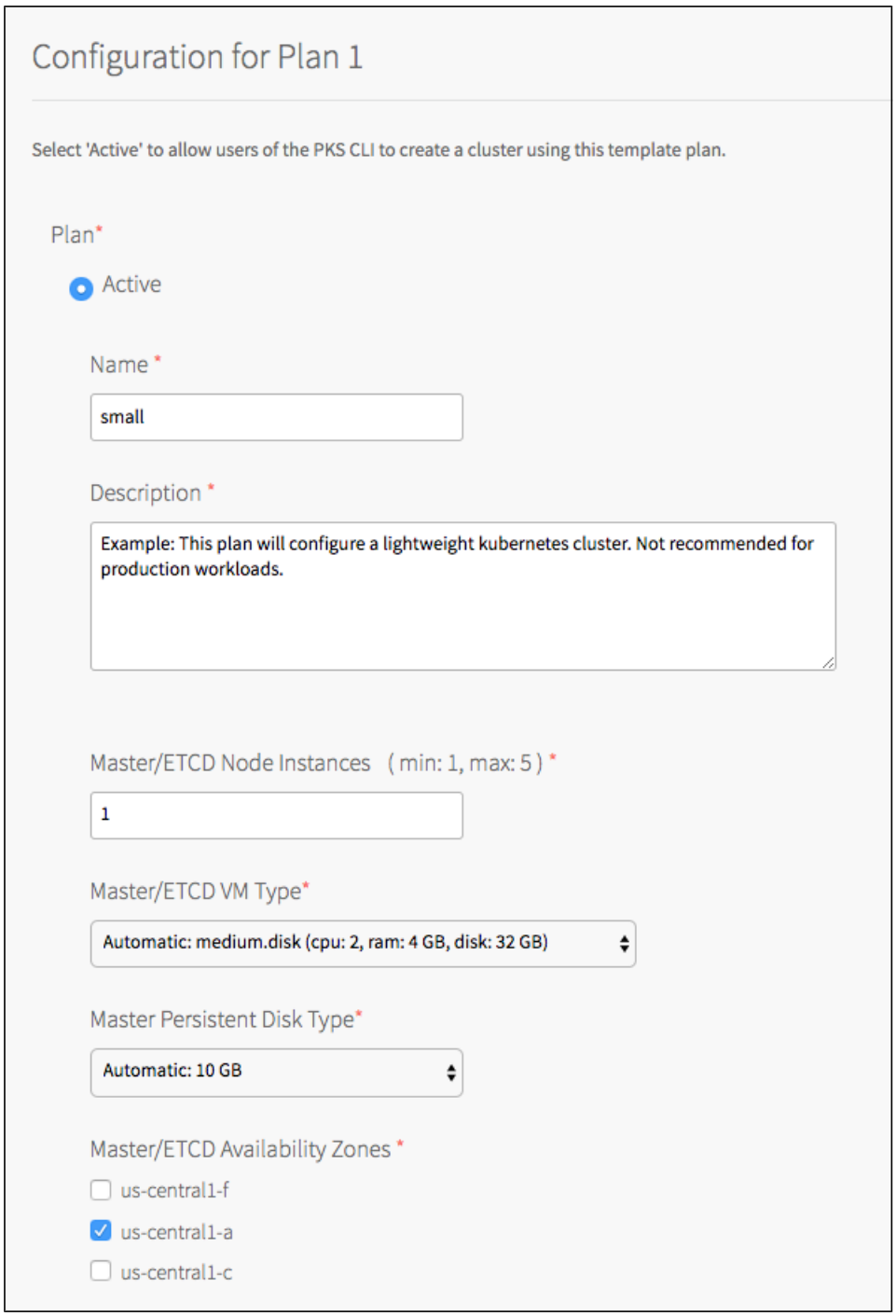
You must first activate and configure **Plan 1**, and afterwards you can optionally activate **Plan 2** through **Plan 10**.

To activate and configure a plan, perform the following steps:

1. Click the plan that you want to activate.

   > **Note**: Plans 11, 12 and 13 support only Windows worker-based Kubernetes clusters, on vSphere with Flannel.

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 the 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 `1`, `3`, or `5`.

> **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.

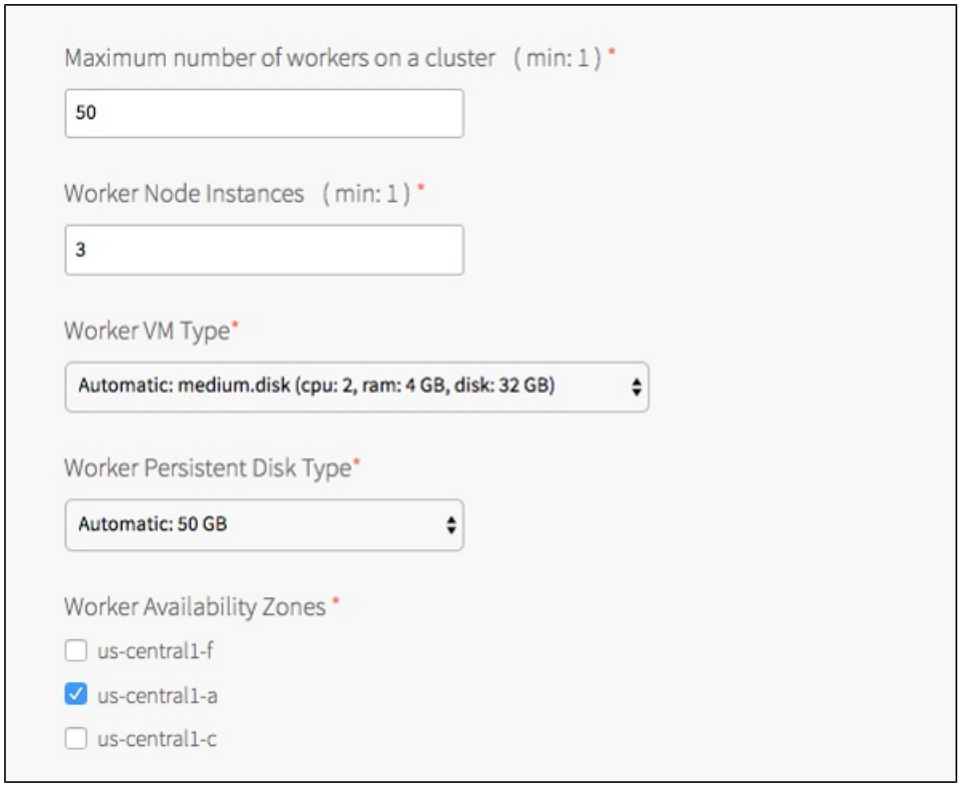
warning: To change the number of master/etcd nodes for a plan, you must ensure that no existing clusters use the plan. Enterprise PKS does not support changing the number of master/etcd nodes for plans with existing clusters.

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 Enterprise 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 Enterprise PKS. If you select more than one AZ, Enterprise PKS deploys the master VM in the first AZ and the worker VMs across the remaining AZs. If you are using multiple masters, Enterprise PKS deploys the master and worker VMs across the AZs in round-robin fashion.

9. Under **Maximum number of workers on a cluster**, set the maximum number of Kubernetes worker node VMs that Enterprise 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.

For more information about creating clusters, see Creating Clusters.

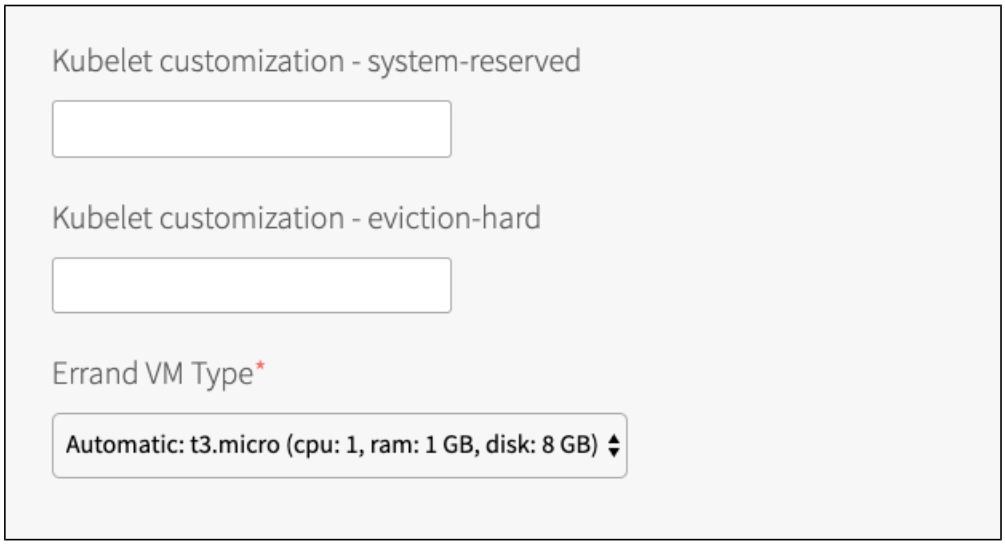
> **Note**: Changing a plan's **Worker Node Instances** setting does not alter the number of worker nodes on existing clusters. For information about scaling an existing cluster, see Scale Horizontally by Changing the Number of Worker Nodes Using the PKS CLI in *Scaling Existing Clusters*.

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 Enterprise PKS Clusters*.

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. Enterprise 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.

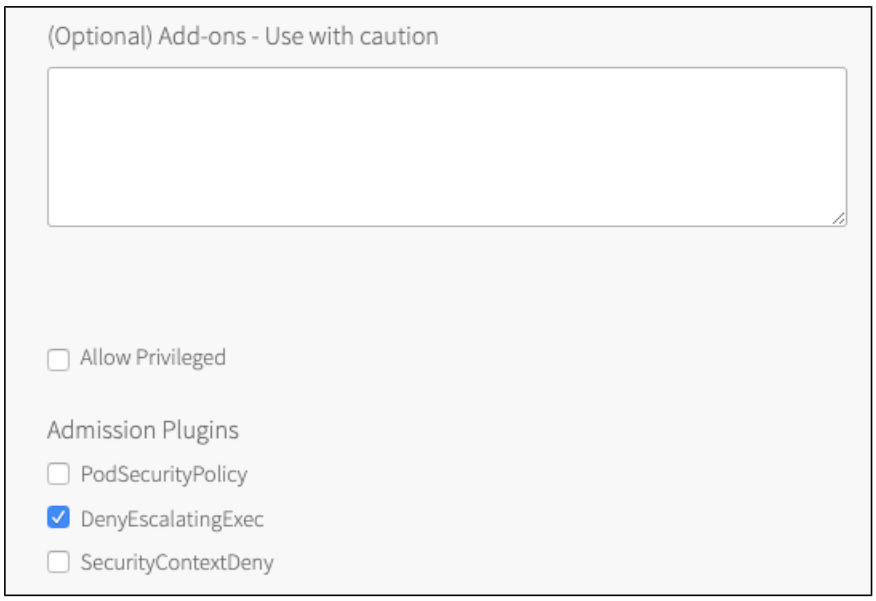


15. Under **Kubelet customization - eviction-hard**, enter threshold limits that Kubelet can use to evict pods when they exceed the limit. Enter limits in the format `EVICTION-SIGNAL=QUANTITY`. For example, `memory.available=100Mi, nodefs.available=10%, nodefs.inodesFree=5%`. For more information about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



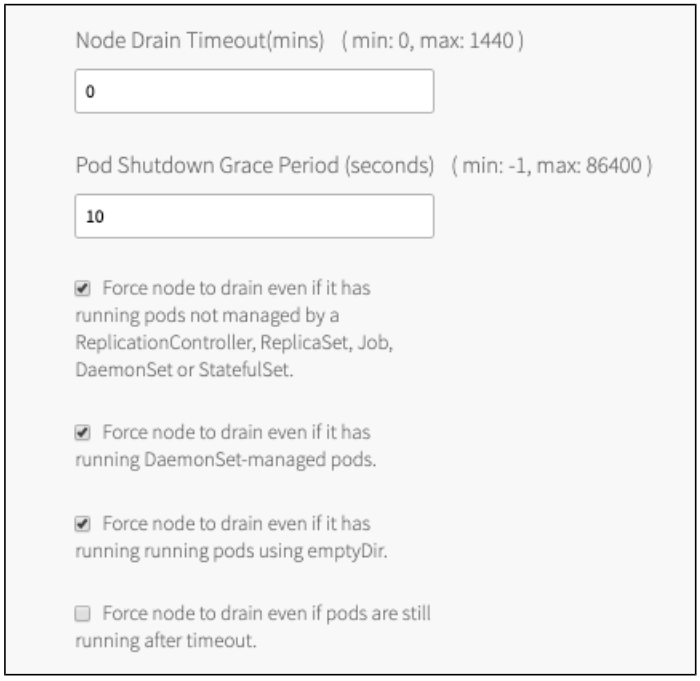
Linux Workloads.

18. (Optional) To allow users to create pods with privileged containers, select the **Allow Privileged** option. For more information, see Pods in the Kubernetes documentation.

> **Note:** Enabling the `Allow Privileged` option means that all containers in the cluster will run in privileged mode. Pod Security Policy provides a privileged parameter that can be used to enable or disable Pods running in privileged mode. As a best practice, if you enable `Allow Privileged`, define PSP to limit which Pods run in privileged mode. If you are implementing PSP for privileged pods, you must enable `Allow Privileged` mode.

19. (Optional) Enable or disable one or more admission controller plugins: **PodSecurityPolicy**, **DenyEscalatingExec**, and **SecurityContextDeny**. See Admission Plugins for more information.

20. (Optional) Under **Node Drain Timeout(mins)**, enter the timeout in minutes for the node to drain pods. If you set this value to



`0`, the node drain does not terminate.

21. (Optional) Under **Pod Shutdown Grace Period (seconds)**, enter a timeout in seconds for the node to wait before it forces the pod to terminate. If you set this value to `-1`, the default timeout is set to the one specified by the pod.

22. (Optional) To configure when the node drains, enable the following:
    - **Force node to drain even if it has running pods not managed by a ReplicationController, ReplicaSet, Job, DaemonSet or StatefulSet.**
    - **Force node to drain even if it has running DaemonSet-managed pods.**
    - **Force node to drain even if it has running running pods using emptyDir.**
    - **Force node to drain even if pods are still running after timeout.**

    > **warning:** If you select **Force node to drain even if pods are still running after timeout**, the node kills all running workloads on pods. Before enabling this configuration, set **Node Drain Timeout** to a value greater than `0`.

    For more information about configuring default node drain behavior, see Worker Node Hangs Indefinitely in *Troubleshooting*.

23. Click **Save**.

Deactivate a Plan

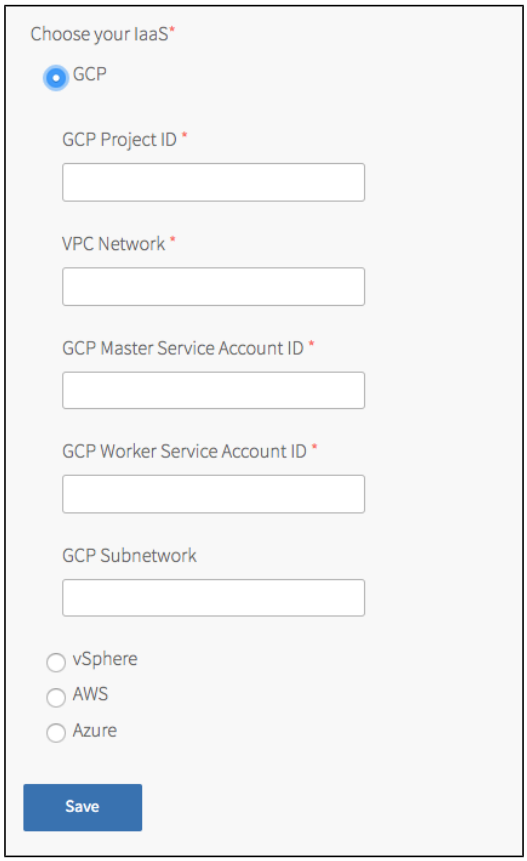
To deactivate a plan, perform the following steps:

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 Enterprise PKS manually:** You configured the master node service account in Create the Master Node Service Account in *Creating Service Accounts in GCP for Enterprise PKS*.
   - **If you are installing Enterprise 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 Enterprise PKS manually:** You configured the worker node service account in Create the Worker Node Service Account in *Creating Service Accounts in GCP for Enterprise PKS*.
   - **If you are installing Enterprise 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`.

e. (Optional) Enter your **GCP Subnetwork**. This is the name of the services subnetwork that you created for Kubernetes cluster VMs in GCP.

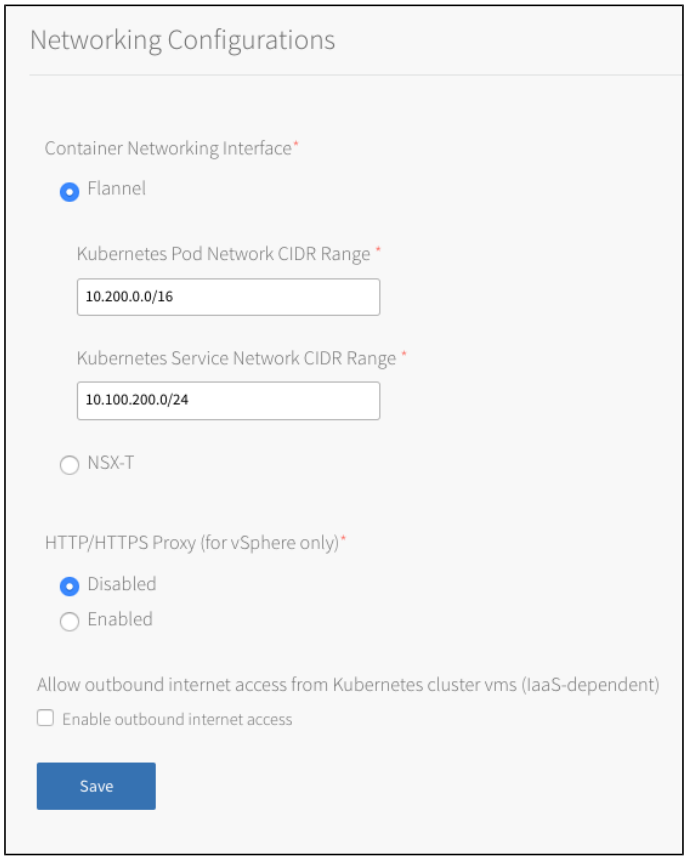
   > **Note:** If you want to create GCP internal load balancers through Services of type `LoadBalancer`, you must configure the **GCP Subnetwork** field.

4. Click **Save**.

Networking

To configure networking, do the following:

1. Click **Networking**.

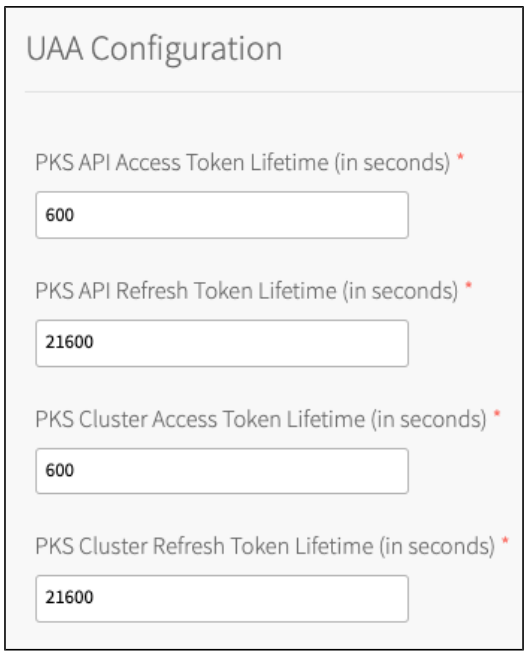


2. Under **Container Networking Interface**, select **Flannel**.
3. (Optional) Enter values for **Kubernetes Pod Network CIDR Range** and **Kubernetes Service Network CIDR Range**.
   - Ensure that the CIDR ranges do not overlap and have sufficient space for your deployed services.
   - Ensure that the 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. This field defaults to `600`.



3. Under **PKS API Refresh Token Lifetime**, enter a time in seconds for the PKS API refresh token lifetime. This field defaults to `21600`.

4. Under **PKS Cluster Access Token Lifetime**, enter a time in seconds for the cluster access token lifetime. This field defaults to `600`.

5. Under **PKS Cluster Refresh Token Lifetime**, enter a time in seconds for the cluster refresh token lifetime. This field defaults to `21600`.

> **Note**: Pivotal recommends using the default UAA token timeout values. By default, access tokens expire after ten minutes and refresh tokens expire after six hours. If you want to customize your token timeout values, see Token Management in *UAA Overview*.

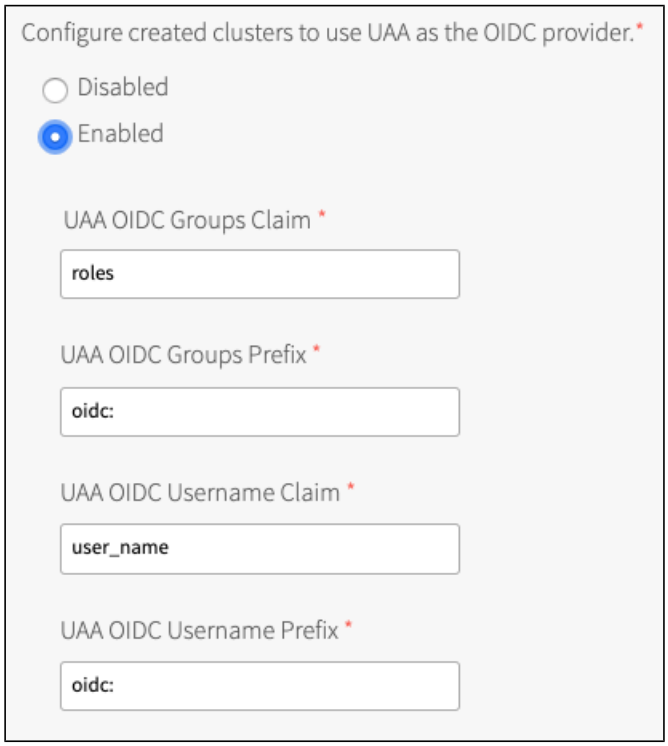
6. Under **Configure created clusters to use UAA as the OIDC provider**, select **Enabled** or **Disabled**. If you click **Enabled**, Kubernetes verifies end-user identities based on authentication executed by UAA. For more information, see the table below.

Option	Description
Disabled	If you do not enable UAA as the OpenID Connect (OIDC) provider, Kubernetes authenticates users against its internal user management system.
Enabled	If you enable UAA as the OIDC provider, Kubernetes authenticates users as follows: ○ If you select **Internal UAA** in the next step, Kubernetes authenticates users against the internal UAA authentication mechanism. ○ If you select **LDAP Server** in the next step, Kubernetes authenticates users against the LDAP server. ○ If you select **SAML Identity Provider** in the next step, Kubernetes authenticates users against the SAML identity provider.

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

To configure Enterprise PKS to use UAA as the OIDC provider, do the following:

a. Under **Configure created clusters to use UAA as the OIDC provider**, select **Enabled**.



b. For **UAA OIDC Groups Claim**, enter the name of your groups claim. This is used to set a user's group in the JSON Web Token (JWT) claim. The default value is `roles`.
c. For **UAA OIDC Groups Prefix**, enter a prefix for your groups claim. This prevents conflicts with existing names. For example, if you enter the prefix `oidc:`, UAA creates a group name like `oidc:developers`. If you are configuring a new Enterprise PKS installation, the default is `oidc:`. If you are upgrading to Enterprise PKS v1.5, the default is `-`.
d. For **UAA OIDC Username Claim**, enter the name of your username claim. This is used to set a user's username in the JWT claim. The default value is `user_name`. Depending on your provider, admins can enter claims besides `user_name`, like `email` or `name`.
e. For **UAA OIDC Username Prefix**, enter a prefix for your username claim. This prevents conflicts with existing names. For example, if you enter the prefix `oidc:`, UAA creates a username like `oidc:admin`. If you are configuring a new Enterprise PKS installation, the default is `oidc:`. If you are upgrading to Enterprise PKS v1.5, the default is `-`.

> **Note:** Pivotal recommends adding OIDC prefixes to prevent OIDC users and groups from gaining unintended cluster privileges. When you upgrade to Enterprise PKS v1.5, if you do not change the values for **UAA OIDC Groups Prefix** or **UAA OIDC Username Prefix**, Enterprise PKS does not add prefixes.

> **warning:** If you change the above values for a pre-existing Enterprise PKS installation, you must change any existing role bindings that bind to a username or group. If you do not change your role bindings, developers cannot access Kubernetes clusters. For instructions about creating role bindings, see Managing Cluster Access and Permissions.

7. Select one of the following options:

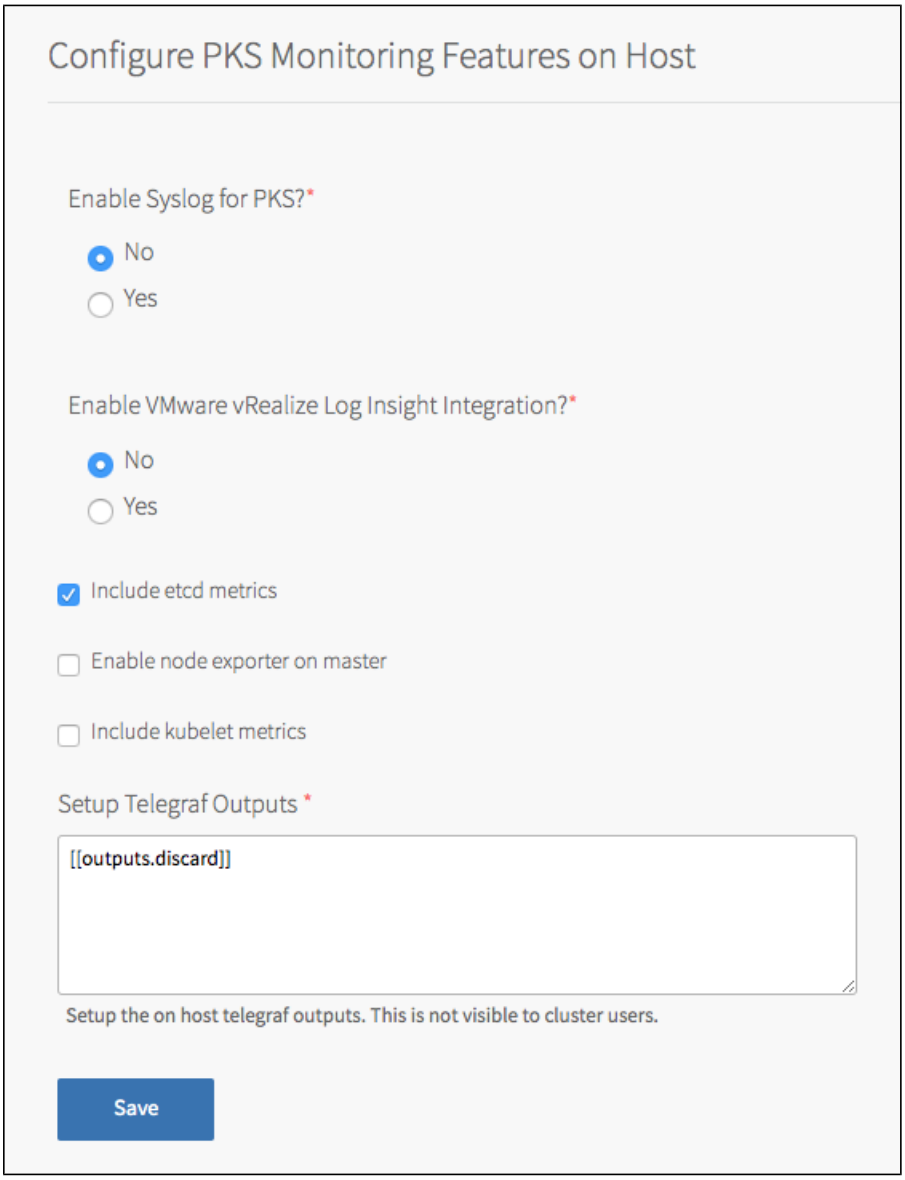
- To use an internal user account store for UAA, select **Internal UAA**. Click **Save** and continue to (Optional) Host Monitoring.
- To use LDAP for UAA, select **LDAP Server** and continue to Connecting Enterprise PKS to an LDAP Server.
- To use SAML for UAA, select **SAML Identity Provider** and continue to Connecting Enterprise PKS to a SAML Identity Provider.

(Optional) Host Monitoring

In **Host Monitoring**, you can configure one or more of the following:

- To configure Syslog for BOSH-deployed VMs, see Syslog. Syslog forwards log messages from all BOSH-deployed VMs to a syslog endpoint.
- To configure the Telegraf agent, see Telegraf. The Telegraf agent sends metrics from master node and worker node VMs to a monitoring service, such as Wavefront or Datadog.

For more information about these components, see Monitoring PKS and PKS-Provisioned Clusters.



Syslog

To configure Syslog for all BOSH-deployed VMs in Enterprise PKS:

1. Click **Host Monitoring**.
2. Under **Enable Syslog for PKS**, select **Yes**.
3. Under **Address**, enter the destination syslog endpoint.

4. Under **Port**, enter the destination syslog port.

5. Under **Transport Protocol**, select a transport protocol for log forwarding.

6. (Optional) To enable TLS encryption during log forwarding, complete the following steps:

   a. Ensure **Enable TLS** is selected.

   > **Note**: Logs may contain sensitive information, such as cloud provider credentials. Pivotal strongly recommends that you enable TLS encryption for log forwarding.

   b. Under **Permitted Peer**, provide the accepted fingerprint (SHA1) or name of remote peer. For example, `*.YOUR-LOGGING-SYSTEM.com`.

   c. 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. (Optional) Under **Max Message Size**, enter a maximum message size for logs that are forwarded to a syslog endpoint. By default, the **Max Message Size** field is 10,000 characters.

8. Click **Save**.

Telegraf

To configure Enterprise PKS to use Telegraf for metric collection:

1. Create a configuration file for your monitoring service. For instructions, see Create a Configuration File.

2. Select one or more of the following checkboxes:

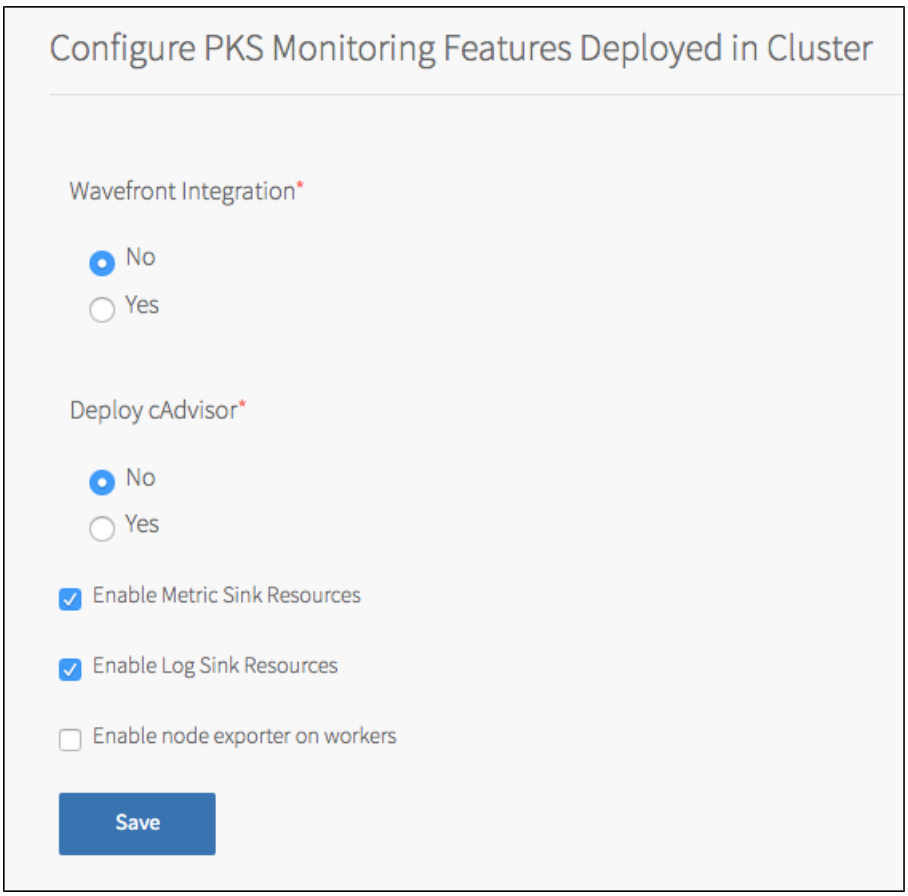
   - **Include etcd metrics**: This includes etcd server and debug metrics.
   - **Enable node exporter on master**: This enables Node Exporter on the *localhost* of each master node VM.
   - **Include kubelet metrics**: This includes all workload metrics across your Kubernetes clusters. Enabling **Include kubelet metrics** generates a high volume of metrics.

3. In **Setup Telegraf Outputs**, replace the default value `[[outputs.discard]]` with the contents of the configuration file that you created above.

4. Click **Save**.

(Optional) In-Cluster Monitoring

In **In-Cluster Monitoring**, you can configure one or more observability components and integrations that run in Kubernetes clusters and capture logs and metrics about your workloads. For more information, see Monitoring Workers and Workloads.



To configure in-cluster monitoring:

- To configure Wavefront, see Wavefront.
- To configure cAdvisor, see cAdvisor.
- To configure sink resources, see:
  - Metric Sink Resources
  - Log Sink Resources

  You can enable both log and metric sink resources or only one of them.

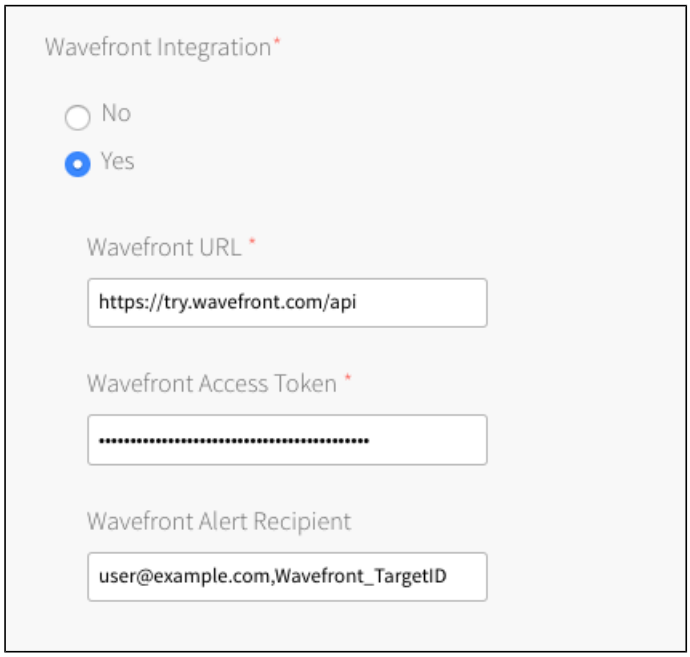
Wavefront

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 the Wavefront documentation.

To enable and configure Wavefront monitoring:

1. In the Enterprise PKS tile, select **In-Cluster Monitoring**.



2. 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**, using the following syntax:

```
USER-EMAIL,WAVEFRONT-TARGETID_001,WAVEFRONT-TARGETID_002
```

   Where:

   - `USER-EMAIL` is the alert recipient's email address.
   - `WAVEFRONT-TARGETID_001` and `WAVEFRONT-TARGETID_002` are your comma-delimited Wavefront Target IDs.

   For example:

```
randomuser@example.com,51n6psdj933ozdjf
```

6. Click **Save**.

To create alerts, you must enable errands in Enterprise PKS.

1. In the Enterprise PKS tile, select **Errands**.
2. On the **Errands** pane, enable **Create pre-defined Wavefront alerts errand**.
3. Enable **Delete pre-defined Wavefront alerts errand**.
4. Click **Save**. Your settings apply to any clusters created after you have saved these configuration settings and clicked **Apply Changes**.

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

cAdvisor

cAdvisor is an open source tool for monitoring, analyzing, and exposing Kubernetes container resource usage and performance statistics.

To deploy a cAdvisor container:

1. Select **In-Cluster Monitoring**.
2. Under **Deploy cAdvisor**, select **Yes**.
3. Click **Save**.

> **Note:** For information about configuring cAdvisor to monitor your running Kubernetes containers, see cAdvisor in the cAdvisor GitHub repository. For general information about Kubernetes cluster monitoring, see Tools for Monitoring Resources in the Kubernetes documentation.

Metric Sink Resources

You can configure PKS-provisioned clusters to send Kubernetes node metrics and pod metrics to metric sinks. For more information about metric sink resources and what to do after you enable them in the tile, see Sink Resources in *Monitoring Workers and Workloads*.

To enable clusters to send Kubernetes node metrics and pod metrics to metric sinks:

1. In **In-Cluster Monitoring**, select **Enable Metric Sink Resources**. If you enable this checkbox, Enterprise PKS deploys Telegraf as a `DaemonSet`, a pod that runs on each worker node in all your Kubernetes clusters.
2. (Optional) To enable Node Exporter to send worker node metrics to metric sinks of kind `ClusterMetricSink`, select **Enable node exporter on workers**. If you enable this checkbox, Enterprise PKS deploys Node Exporter as a `DaemonSet`, a pod that runs on each worker node in all your Kubernetes clusters.
   For instructions on how to create a metric sink of kind `ClusterMetricSink` for Node Exporter metrics, see Create a ClusterMetricSink Resource for Node Exporter Metrics in *Creating and Managing Sink Resources*.
3. Click **Save**.

Log Sink Resources

You can configure PKS-provisioned clusters to send Kubernetes API events and pod logs to log sinks. For more information about log sink resources and what to do after you enable them in the tile, see Sink Resources in *Monitoring Workers and Workloads*.

To enable clusters to send Kubernetes API events and pod logs to log sinks:

1. Select **Enable Log Sink Resources**. If you enable this checkbox, Enterprise PKS deploys Fluent Bit as a `DaemonSet`, a pod that runs on each worker node in all your Kubernetes clusters.
2. Click **Save**.

Tanzu Mission Control (Experimental)

Participants in the VMware Tanzu Mission Control beta program can use the **Tanzu Mission Control (Experimental)** pane of the Enterprise PKS tile to integrate their Enterprise PKS deployment with Tanzu Mission Control.

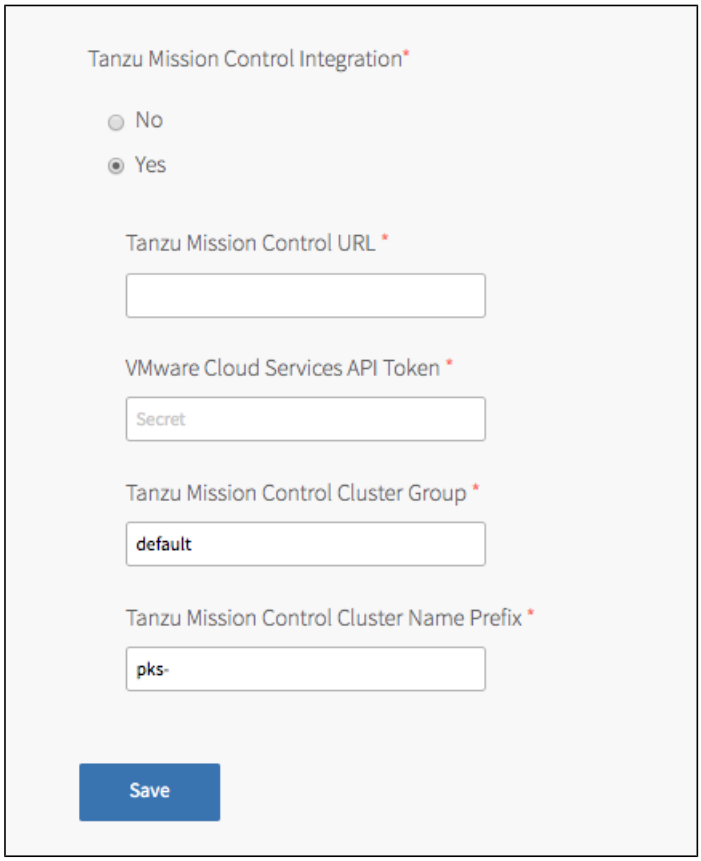
Tanzu Mission Control integration lets you monitor and manage Enterprise PKS clusters from the Tanzu Mission Control console, which makes the Tanzu Mission Control console a single point of control for all Kubernetes clusters.

> **warning:** VMware Tanzu Mission Control is currently experimental beta software and is intended for evaluation and test purposes only. For more information about Tanzu Mission Control, see the VMware Tanzu Mission Control home page.

To integrate Enterprise PKS with Tanzu Mission Control:

1. Confirm that the PKS Control Plane VM has internet access and can connect to `*.tmc.cloud.vmware.com` and the other outbound URLs listed in the What Happens When You Attach a Cluster section of the Tanzu Mission Control documentation.

2. Navigate to the **Enterprise PKS** tile > the **Tanzu Mission Control (Experimental)** pane and select **Yes** under **Tanzu Mission Control Integration**.



3. Configure the fields below:

   - **Tanzu Mission Control URL**: Enter the Org URL of your Tanzu Mission Control subscription, without a trailing slash `/`). For example, `YOUR-ORG.tmc.cloud.vmware.com`.
   - **VMware Cloud Services API token**: Enter your API token to authenticate with VMware Cloud Services APIs. You can retrieve this token by logging in to VMware Cloud Services and viewing your account information.
   - **Tanzu Mission Control Cluster Group**: Enter the name of a Tanzu Mission Control cluster group.

     The name can be `default` or another value, depending on your role and access policy:

- `Org Member` users in VMware cloud services have a `service.admin` role in Tanzu Mission Control. These users:
  - By default, can create and attach clusters only in the `default` cluster group.
  - Can create and attach clusters to other cluster groups after an `organization.admin` user grants them the `clustergroup.admin` or `clustergroup.edit` role for those groups.
- `Org Owner` users in VMware cloud services have `organization.admin` permissions in Tanzu Mission Control. These users:
  - Can create cluster groups.
  - Can grant `clustergroup` roles to `service.admin` users through the Tanzu Mission Control Access Policy view.

  For more information about role and access policy, see Access Control in the *VMware Tanzu Mission Control Product Documentation*.

- **Tanzu Mission Control Cluster Name Prefix**: Enter a name prefix for identifying the Enterprise PKS clusters in Tanzu Mission Control.

4. Click **Save**.

warning: After the Enterprise PKS tile is deployed with a configured cluster group, the cluster group cannot be updated.

Note: When you upgrade your Kubernetes clusters and have Tanzu Mission Control integration enabled, existing clusters will be attached to Tanzu Mission Control.

CEIP and Telemetry

To configure VMware's Customer Experience Improvement Program (CEIP) and the Pivotal Telemetry Program (Telemetry), do the following:

1. Click **CEIP and Telemetry**.

2. Review the information about the CEIP and Telemetry.

CEIP and Telemetry

About the CEIP and Telemetry Program

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.

Customers who participate (at the enhanced tier) are eligible for several benefits, including Proactive Support.

Additional information regarding the data collected through CEIP or Telemetry, and the purposes for which it is used by VMware is set forth in the Trust & Assurance Center and for Pivotal on the Pivotal Telemetry pages. If you prefer not to participate in CEIP and Telemetry for PKS, you should not join below. You may join or leave CEIP and Telemetry for PKS at any time.

Levels of Participation

No personally identifiable information (PII) is collected at either level of participation. Please refer to the data dictionary for more information on the data we collect.

Standard: This participation tier is anonymous. Your data will be used to improve PKS, but is not identifiable to your organization

Enhanced: This participation tier allows us to identify your organization so we may provide proactive support and other benefits. See sample reports for more details.

Please Note

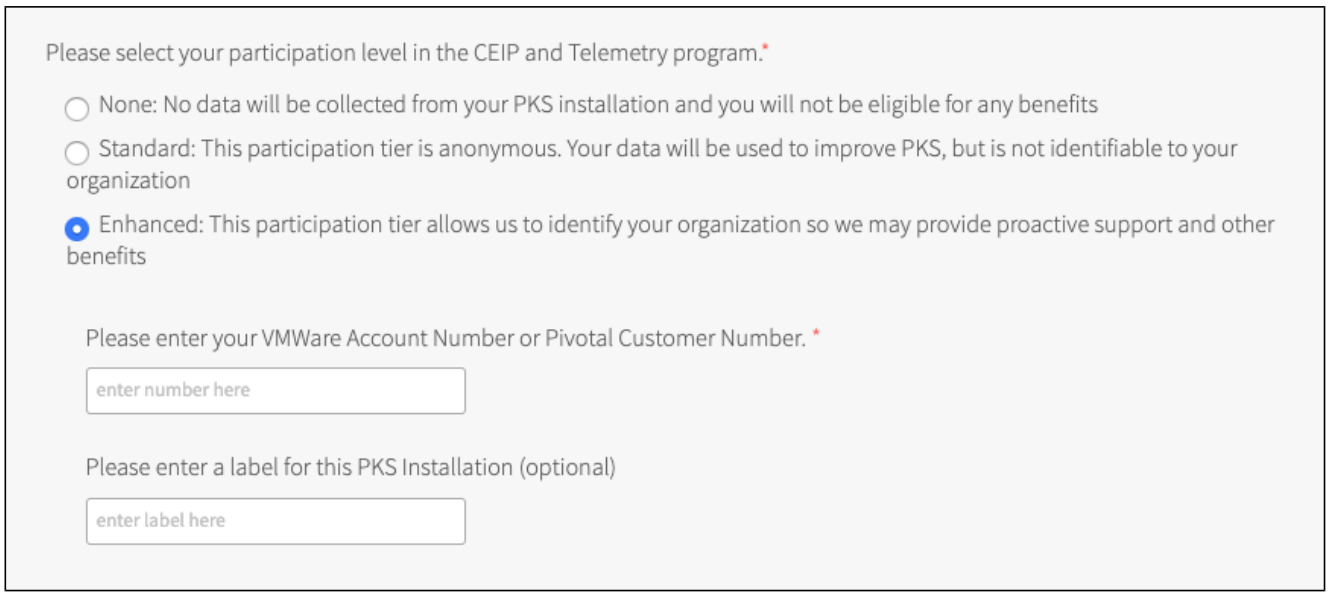
- If you are opting in on behalf of an organization (and not for you as an individual), you represent and warrant that you have legal authority to bind that organization, and you hereby join CEIP/Telemetry on behalf of your organization.
- If you are opting in to the enhanced tier, in the event a term or condition of CEIP or the Telemetry program conflicts with a term or condition of a previously executed license procurement agreement between you and Licensor (Pivotal or VMware), the CEIP or Telemetry program terms supersede solely for purposes of CEIP and Telemetry
- If you are running PKS on a private network, you will need to enable outgoing internet access by opening your firewall to allow traffic to `https://vcsa.vmware.com/ph` on port `443`

Resources

- Data Dictionary
- Participation Benefits
- Sample Reports
- Trust and Assurance Center
- Pivotal Telemetry

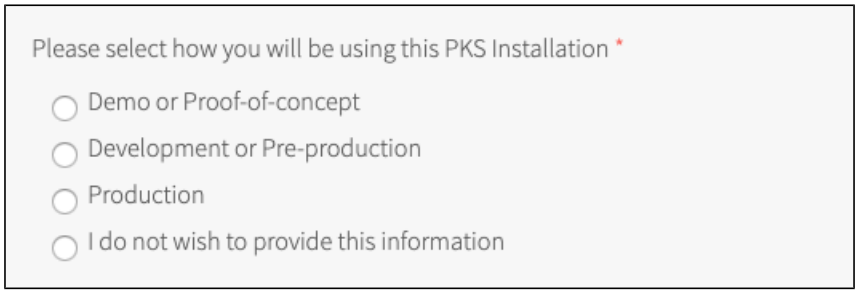
View a larger version of this image.

3. To specify your level of participation in the CEIP and Telemetry program, select one of the **Participation Level** options:
   - **None:** If you select this option, data is not collected from your Enterprise PKS installation.
   - (Default) **Standard:** If you select this option, data is collected from your Enterprise PKS installation to improve Enterprise PKS. This participation level is anonymous and does not permit the CEIP and Telemetry to identify your organization.
   - **Enhanced:** If you select this option, data is collected from your Enterprise PKS installation to provide you proactive support and other benefits. This participation level permits the CEIP and Telemetry to identify your organization.



For more information about the CEIP and Telemetry participation levels, see Participation Levels in *Telemetry*.

4. If you selected the **Enhanced** participation level, complete the following:
   - Enter your VMware account number or Pivotal customer number in the **VMware Account Number or Pivotal Customer Number** field. If you are a VMware customer, you can find your VMware Account Number in your **Account Summary** on my.vmware.com. If you are a Pivotal customer, you can find your Pivotal Customer Number in your Pivotal Order Confirmation email.
   - (Optional) Enter a descriptive name for your PKS installation in the **PKS Installation Label** field. The label you assign to this installation will be used in telemetry reports to identify the environment.
5. To provide information about the purpose for this installation, select an option in the **PKS Installation Type** list.



6. Click **Save**.

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

Note: Even if you select **None**, Enterprise PKS-provisioned clusters send usage data to the PKS control plane. However, this data is not sent to VMware or Pivotal and remains on your Enterprise PKS installation.

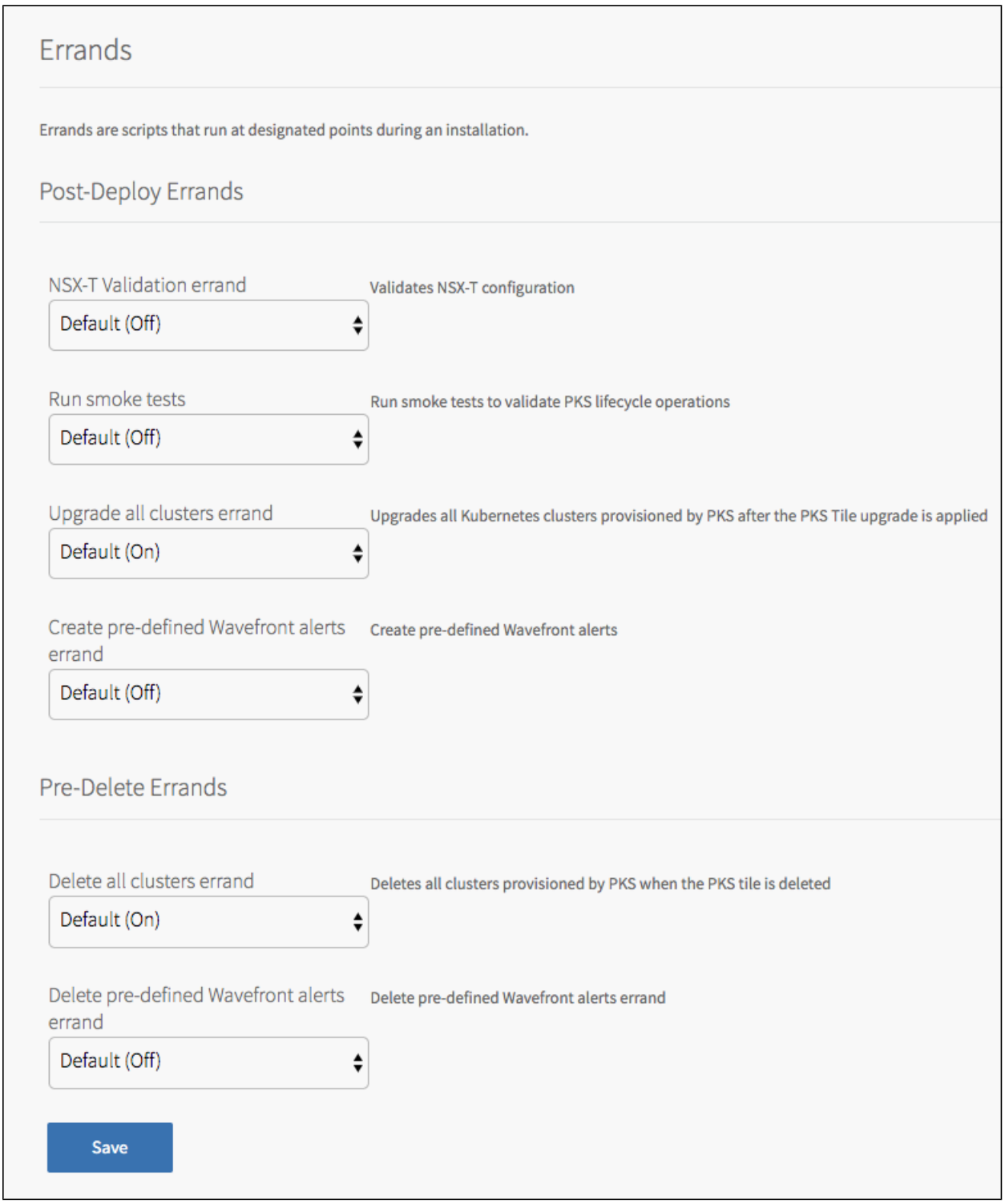
Errands

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

To configure when post-deploy and pre-delete errands for Enterprise 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 CLI to create a Kubernetes cluster and then delete it. If the creation or deletion fails, the errand fails and the installation of the Enterprise 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.

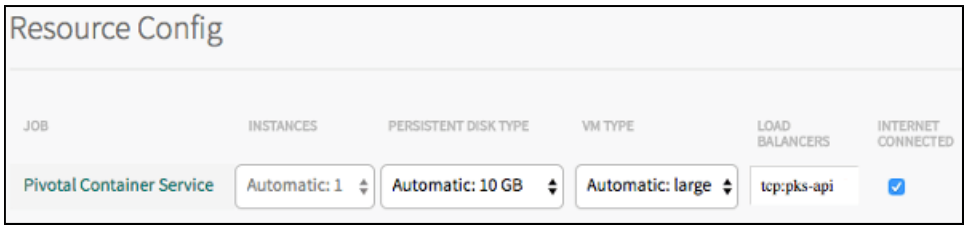
warning: If **Upgrade all clusters errand** is enabled, updating the Enterprise PKS tile with a new Linux stemcell triggers the

rolling of every Linux VM in each Kubernetes cluster. Similarly, updating the Enterprise PKS tile with a new Windows stemcell triggers the rolling of every Windows VM in your Kubernetes clusters. This automatic rolling ensures that all your VMs are patched. To avoid workload downtime, use the resource configuration recommended in What Happens During Enterprise PKS Upgrades and Maintaining Workload Uptime.

Resource Config

To modify the resource usage of Enterprise 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 Enterprise 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 Enterprise 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.

3. (Optional) Edit other resources used by the **Pivotal Container Service** job. The **Pivotal Container Service** job requires a VM with the following minimum resources:

CPU	Memory	Disk
2	8 GB	29 GB

   Note: The automatic **VM Type** value matches the minimum recommended size for the **Pivotal Container Service** job. If you experience timeouts or slowness when interacting with the PKS API, select a **VM Type** with greater CPU and memory resources.

4. (Optional) If you do not use a NAT instance, select **Internet Connected** to allow component instances direct access to the internet.

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 need to retrieve the PKS API endpoint to allow your organization to use the API to create, update, and delete Kubernetes clusters.

To retrieve the PKS API endpoint, do the following:

1. Navigate to the Ops Manager **Installation Dashboard**.
2. Click the **Enterprise PKS** 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 Enterprise 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 CLI and the Kubernetes CLI help you interact with your Enterprise 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 Authentication for Enterprise PKS

Follow the procedures in Setting Up Enterprise PKS Admin Users on GCP.

Next Steps

After installing Enterprise PKS on GCP, you may want to do one or more of the following:

- Create a load balancer for your Enterprise PKS clusters. For more information, see Creating and Configuring a GCP Load Balancer for Enterprise PKS Clusters.
- Create your first Enterprise PKS cluster. For more information, see Creating Clusters.

Setting Up Enterprise PKS Admin Users on GCP

In this topic

Page last updated:

This topic describes how to create admin users in VMware Enterprise PKS with User Account and Authentication (UAA). Creating at least one admin user is a necessary step during the initial set up of Enterprise PKS.

Overview

UAA is the identity management service for Enterprise PKS. Enterprise PKS includes a UAA server, which is hosted on the PKS control plane VM.

To interact with the UAA server, you can use the UAA Command Line Interface (UAAC). You can either run UAAC commands from the Ops Manager VM or install UAAC on your local workstation.

Prerequisites

Before setting up admin users for Enterprise PKS, you must have one of the following:

- SSH access to the Ops Manager VM
- A machine that can connect to your PKS control plane VM

Step 1: Connect to the PKS Control Plane VM

You can connect to the PKS control plane VM from the Ops Manager VM or from a different machine such as your local workstation.

Option 1: Connect through the Ops Manager VM

You can connect to the PKS control plane VM by logging in to the Ops Manager VM through SSH. To SSH into the Ops Manager VM on GCP, do the following:

1. Confirm that you have installed the gcloud Command Line Interface (CLI). For more information, see Downloading gcloud in the Google Cloud Platform (GCP) documentation.
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 pop-up window.
6. SSH into the Ops Manager VM by pasting the command into your terminal. For example:

```
$ gcloud compute ssh om-pcf-1a --zone us-central1-b
```

7. Switch to the `ubuntu` user by running the `sudo su - ubuntu` command.
8. Proceed to the Log In as a UAA Admin section to manage users with UAAC.

Option 2: Connect through a Non-Ops Manager Machine

To connect to the PKS control plane VM and run UAA commands, do the following:

1. Install UAAC on your machine. For example:

```
gem install cf-uaac
```

2. Download a copy of your Ops Manager root CA certificate to the machine. To download the certificate, do the following:
   a. In a web browser, navigate to the FQDN of Ops Manager and log in.
   b. In Ops Manager, navigate to **Settings** in the drop-down menu under your username.
   c. Click **Advanced Options**.
   d. On the **Advanced Options** configuration page, click **Download Root CA Cert**.
   e. Move the certificate to a secure location on your machine and record the path.
3. Proceed to the Log In as a UAA Admin section to create admin users with UAAC.

Step 2: Log In as a UAA Admin

Before creating PKS users, you must log in to the UAA server as a UAA admin. To log in to the UAA server, do the following:

1. Retrieve the UAA management admin client secret:
   a. In a web browser, navigate to the Ops Manager **Installation Dashboard** and click the **Enterprise PKS** tile.
   b. Click the **Credentials** tab.
   c. Click **Link to Credential** next to **Pks Uaa Management Admin Client** and copy the value of `secret`.
2. Target your UAA server by running the following command:

```
uaac target https://PKS-API:8443 --ca-cert CERTIFICATE-PATH
```

   Where:

   - `PKS-API` is the domain name of your PKS API server. You entered this domain name in the **Enterprise PKS** tile > **PKS API** > **API Hostname (FQDN)**.
   - `CERTIFICATE-PATH` is the path to your Ops Manager root CA certificate. Provide this certificate to validate the PKS API certificate with SSL.

- If you are logged in to the Ops Manager VM, specify `/var/tempest/workspaces/default/root_ca_certificate` as the path. This is the default location of the root certificate on the Ops Manager VM.
- If you downloaded the Ops Manager root CA certificate to your machine, specify the path where you stored the 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 ignore it because it has no impact.

3. Authenticate with UAA by running the following command:

```
uaac token client get admin -s ADMIN-CLIENT-SECRET
```

Where `ADMIN-CLIENT-SECRET` is your UAA management admin client secret that you retrieved in a previous step. The client username is `admin`.

Step 3: Assign Enterprise PKS Cluster Scopes

The `pks.clusters.manage` and `pks.clusters.admin` UAA scopes grant users the ability to create and manage Kubernetes clusters in Enterprise PKS. For information about UAA scopes in Enterprise PKS, see UAA Scopes for Enterprise PKS Users.

To create Enterprise PKS users with the `pks.clusters.manage` or `pks.clusters.admin` UAA scope, perform one or more of the following procedures based on the needs of your deployment:

- To assign PKS cluster scopes to an individual user, see Grant Enterprise PKS Access to an Individual User. Follow this procedure if you selected **Internal UAA** when you configured **UAA** in the Enterprise PKS tile. For more information, see Installing Enterprise PKS on GCP.
- To assign PKS cluster scopes to an LDAP group, see Grant Enterprise PKS Access to an External LDAP Group. Follow this procedure if you selected **LDAP Server** when you configured **UAA** in the Enterprise PKS tile. For more information, see Installing Enterprise PKS PKS on GCP.
- To assign PKS cluster scopes to a SAML group, see Grant Enterprise PKS Access to an External SAML Group. Follow this procedure if you selected **SAML Identity Provider** when you configured **UAA** in the Enterprise PKS tile. For more information, see Installing Enterprise PKS PKS on GCP.
- To assign PKS cluster scopes to a client, see Grant Enterprise PKS Access to a Client.

Next Step

After you create admin users in Enterprise PKS, the admin users can create and manage Kubernetes clusters in Enterprise PKS. For more information, see Managing Kubernetes Clusters and Workloads.

Amazon Web Services (AWS)

Page last updated:

This topic lists the procedures to follow to install VMware Enterprise PKS on Amazon Web Services (AWS).

Install Enterprise PKS on AWS

To install Enterprise PKS on AWS, follow the instructions below:

- AWS Prerequisites and Resource Requirements
- Installing and Configuring Ops Manager on AWS
- Installing Enterprise PKS on AWS
- Setting Up Enterprise PKS Admin Users on AWS

Install the PKS and Kubernetes CLIs

The PKS CLI and Kubernetes CLI help you interact with your Enterprise 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.

AWS Prerequisites and Resource Requirements

In this topic

Page last updated:

This topic describes the prerequisites and resource requirements for installing VMware Enterprise PKS on Amazon Web Services (AWS).

Prerequisites

Before installing Enterprise PKS:

1. Review the sections below.
2. Install and configure Ops Manager. To install Ops Manager, follow the instructions in Installing and Configuring Ops Manager on AWS.

Resource Requirements

Installing Ops Manager and Enterprise 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 Pivotal Container Service VM as follows:

Number of Pods	Storage (Persistent Disk) Requirements
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 Enterprise 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 and Configuring Ops Manager on AWS

In this topic

Page last updated:

This topic describes how to install and configure Ops Manager before deploying VMware Enterprise PKS on Amazon Web Services (AWS).

Prerequisites

You use Ops Manager to install and configure Enterprise PKS. Before you install Ops Manager, review AWS Prerequisites and Resource Requirements.

Install and Configure Ops Manager

Multiple versions of Ops Manager are compatible with each version of Enterprise PKS. To confirm that your Ops Manager version is compatible with the version of Enterprise PKS that you want to install, see the Enterprise PKS Release Notes.

To install and configure Ops Manager, follow the instructions in the table below:

Version	Instructions
Ops Manager v2.6	1. Deploying Ops Manager on AWS Using Terraform 2. Configuring BOSH Director on AWS Using Terraform
Ops Manager v2.7	1. Deploying Ops Manager on AWS Using Terraform 2. Configuring BOSH Director on AWS Using Terraform

Note: The topics above 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.

Next Installation Step

To install and configure Enterprise PKS, follow the instructions in Installing Enterprise PKS on AWS.

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

Installing Enterprise PKS on AWS

In this topic

Page last updated:

This topic describes how to install and configure VMware Enterprise 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 VMware Enterprise 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 in the Pivotal documentation.

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

To install Enterprise 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 **Enterprise PKS** in the left column, click the plus sign to add this product to your staging area.

Step 2: Configure Enterprise PKS

Click the orange **Enterprise PKS** tile to start the configuration process.

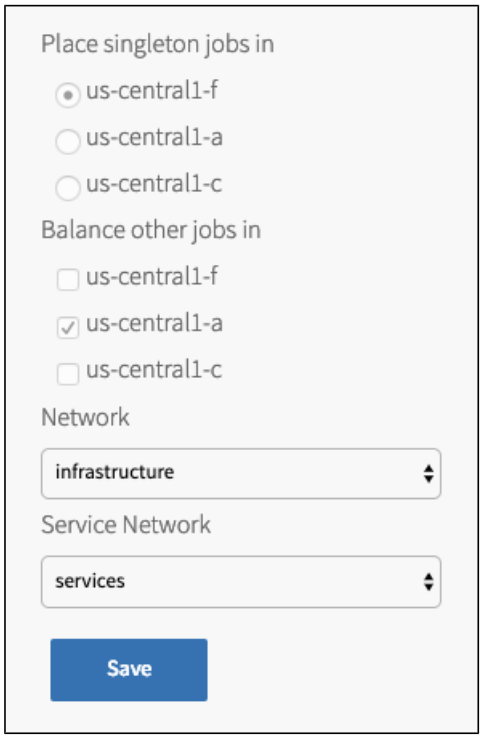
warning: When you configure the Enterprise PKS tile, do not use spaces in any field entries. This includes spaces between characters as well as leading and trailing spaces. If you use a space in any field entry, the deployment of Enterprise PKS fails.

Assign AZs and Networks

To configure the availability zones (AZs) and networks used by the Enterprise PKS control plane:

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

2. Under **Place singleton jobs in**, select the availability zone (AZ) where you want to deploy the PKS API VM as a singleton job.



3. Under **Balance other jobs in**, select the AZ for balancing other Enterprise PKS control plane jobs.

> **Note**: You must specify the **Balance other jobs in** AZ, but the selection has no effect in the current version of Enterprise PKS.

4. Under **Network**, select the infrastructure subnet that you created for the PKS API VM.

5. Under **Service Network**, select the services subnet that you created for Kubernetes cluster VMs.

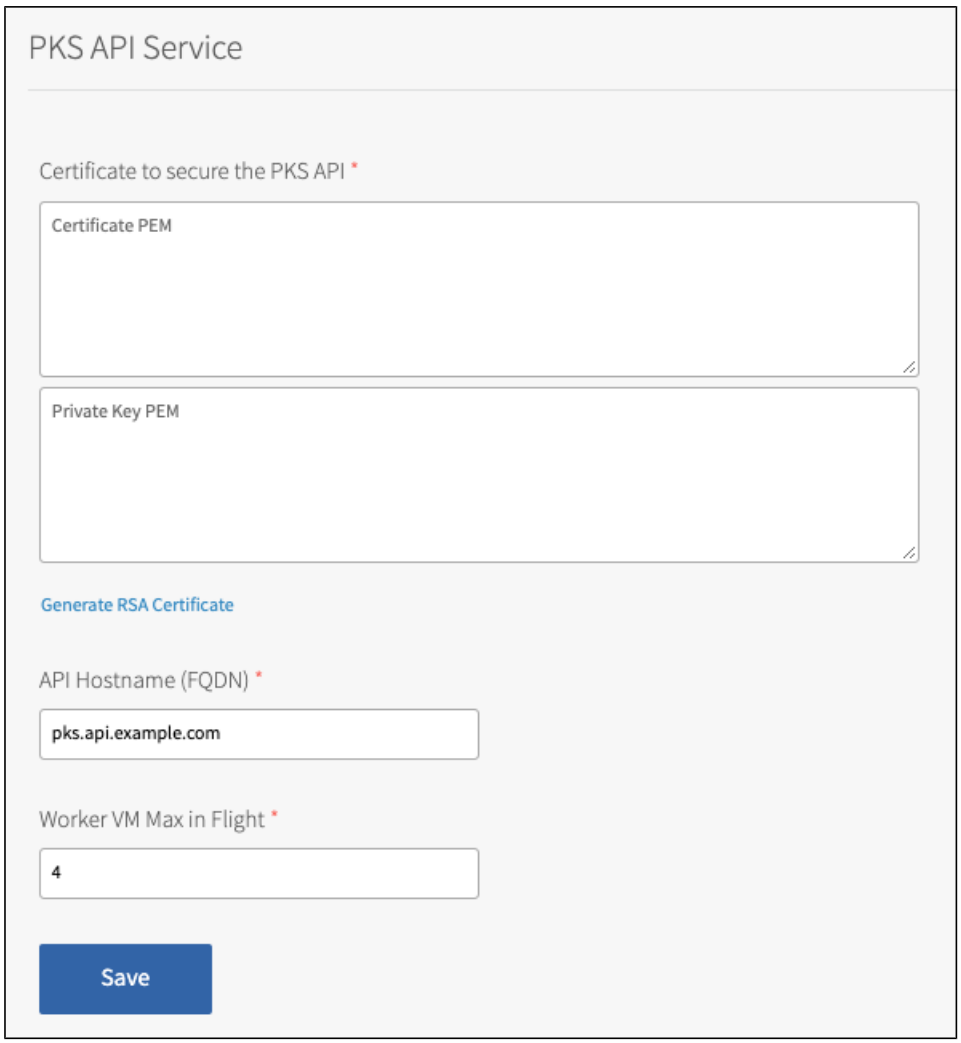
6. Click **Save**.

PKS API

Perform the following steps:

1. Click **PKS API**.

2. Under **Certificate to secure the PKS API**, provide a certificate and private key pair.

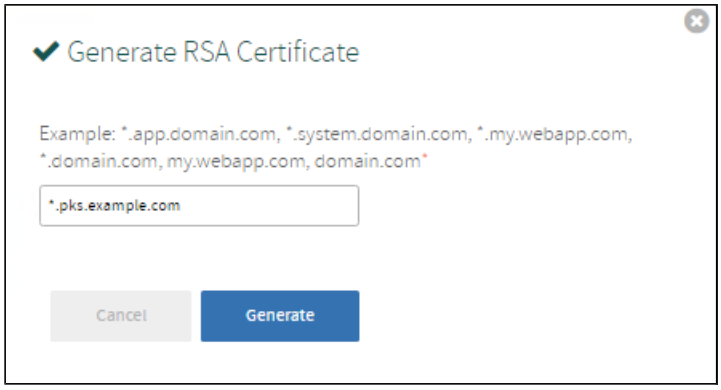


The certificate that you supply should cover the specific subdomain that routes to the PKS API VM with TLS termination on the ingress.

> **warning:** TLS certificates generated for wildcard DNS records only work for a single domain level. For example, a certificate generated for `*.pks.EXAMPLE.com` does not permit communication to `*.api.pks.EXAMPLE.com`. If the certificate does not contain the correct FQDN for the PKS API, calls to the API will fail.

You can enter your own certificate and private key pair, or have Ops Manager generate one for you.
To generate a certificate using Ops Manager:

a. If the **Generate RSA Certificate** option is not available, click **Change**.
b. Click **Generate RSA Certificate**.
c. Enter the domain for your API hostname. This can be a standard FQDN or a wildcard domain.
d. 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 value. When you create or resize a cluster, the `max_in_flight` value limits the number of component instances that can be created or started simultaneously. By default, the `max_in_flight` value is set to `4`, which means that up to four component instances are simultaneously created or started at a time.

5. Click **Save**.

Plans

A plan defines a set of resource types used for deploying a cluster.

Activate a Plan

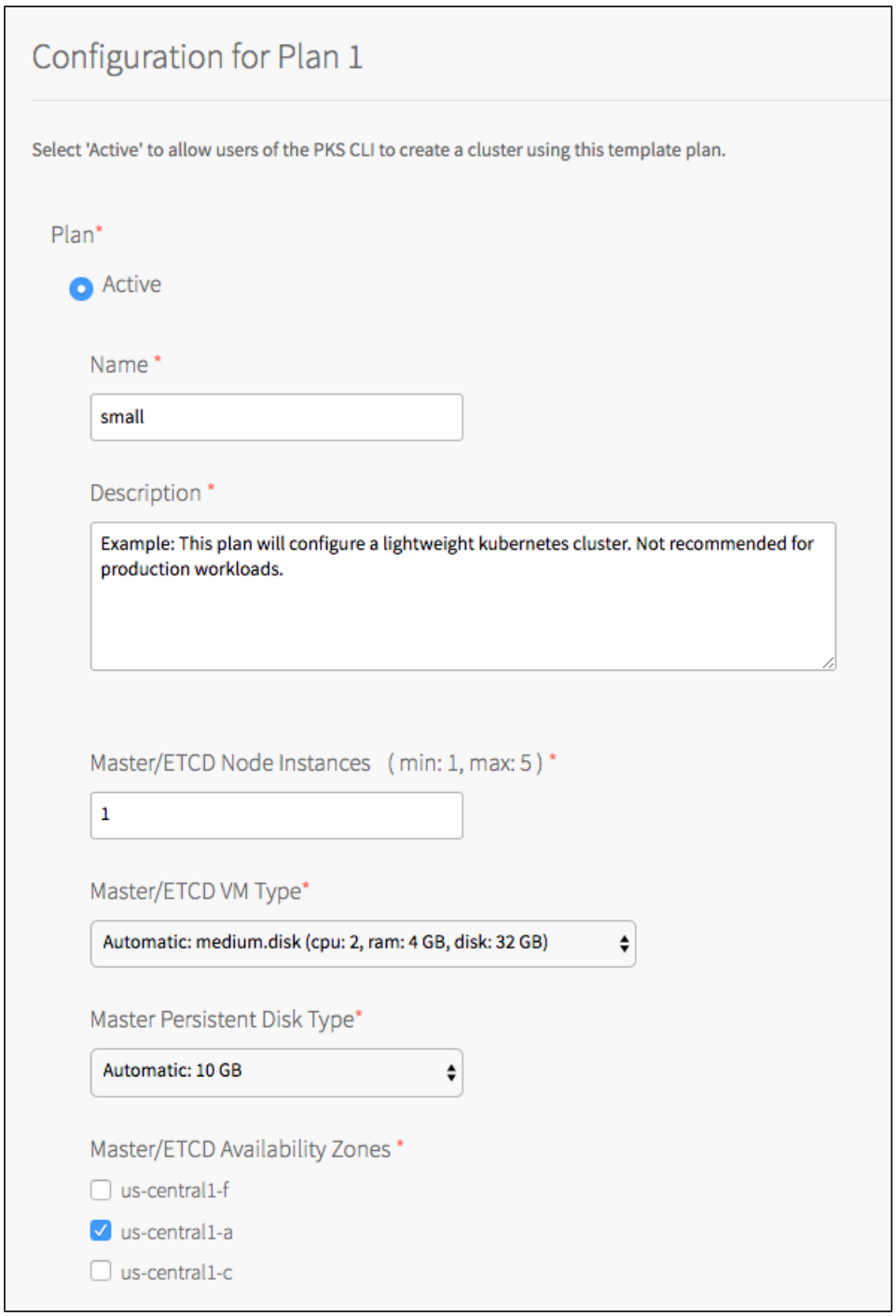
You must first activate and configure **Plan 1**, and afterwards you can optionally activate **Plan 2** through **Plan 10**.

To activate and configure a plan, perform the following steps:

1. Click the plan that you want to activate.

   > **Note**: Plans 11, 12 and 13 support only Windows worker-based Kubernetes clusters, on vSphere with Flannel.

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 the 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 `1`, `3`, or `5`.

> **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.

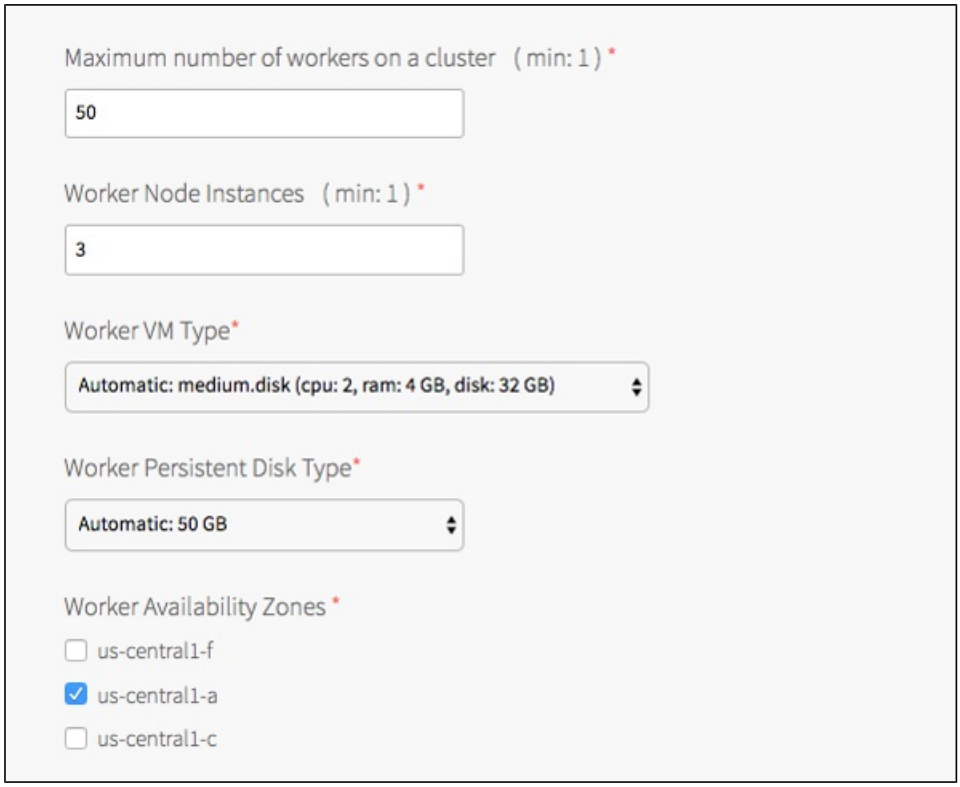
warning: To change the number of master/etcd nodes for a plan, you must ensure that no existing clusters use the plan. Enterprise PKS does not support changing the number of master/etcd nodes for plans with existing clusters.

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 Enterprise 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 Enterprise PKS. If you select more than one AZ, Enterprise PKS deploys the master VM in the first AZ and the worker VMs across the remaining AZs. If you are using multiple masters, Enterprise PKS deploys the master and worker VMs across the AZs in round-robin fashion.

9. Under **Maximum number of workers on a cluster**, set the maximum number of Kubernetes worker node VMs that Enterprise 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.

For more information about creating clusters, see Creating Clusters.

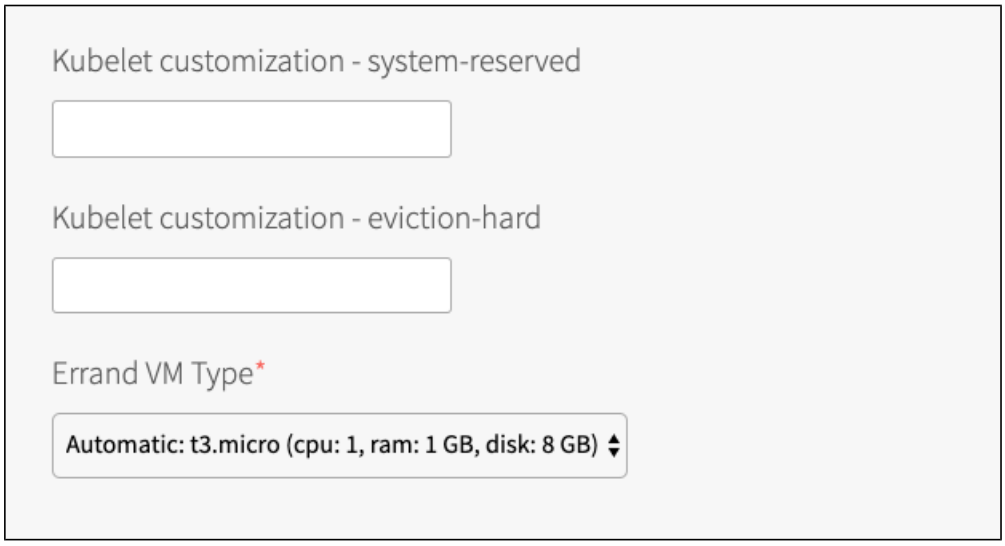
> **Note**: Changing a plan's **Worker Node Instances** setting does not alter the number of worker nodes on existing clusters. For information about scaling an existing cluster, see Scale Horizontally by Changing the Number of Worker Nodes Using the PKS CLI in *Scaling Existing Clusters*.

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 Enterprise PKS Clusters*.

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. Enterprise 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.

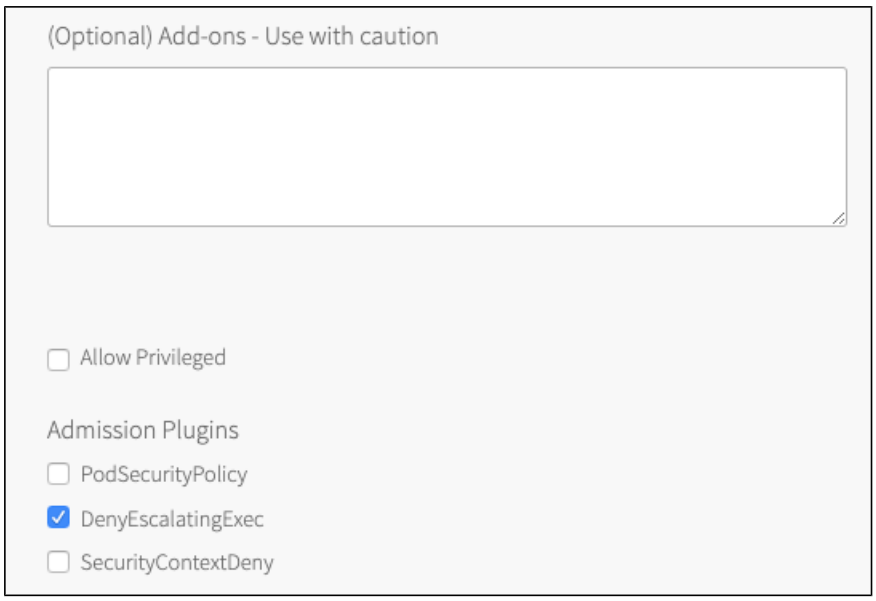


15. Under **Kubelet customization - eviction-hard**, enter threshold limits that Kubelet can use to evict pods when they exceed the limit. Enter limits in the format `EVICTION-SIGNAL=QUANTITY`. For example, `memory.available=100Mi, nodefs.available=10%, nodefs.inodesFree=5%`. For more information about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



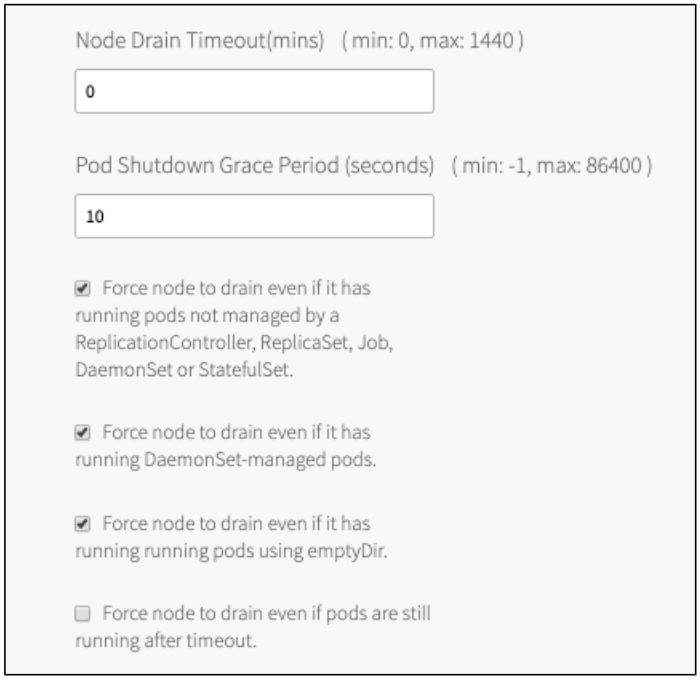
Linux Workloads.

18. (Optional) To allow users to create pods with privileged containers, select the **Allow Privileged** option. For more information, see Pods in the Kubernetes documentation.

> **Note:** Enabling the `Allow Privileged` option means that all containers in the cluster will run in privileged mode. Pod Security Policy provides a privileged parameter that can be used to enable or disable Pods running in privileged mode. As a best practice, if you enable `Allow Privileged`, define PSP to limit which Pods run in privileged mode. If you are implementing PSP for privileged pods, you must enable `Allow Privileged` mode.

19. (Optional) Enable or disable one or more admission controller plugins: **PodSecurityPolicy**, **DenyEscalatingExec**, and **SecurityContextDeny**. See Admission Plugins for more information.

20. (Optional) Under **Node Drain Timeout(mins)**, enter the timeout in minutes for the node to drain pods. If you set this value to



`0`, the node drain does not terminate.

21. (Optional) Under **Pod Shutdown Grace Period (seconds)**, enter a timeout in seconds for the node to wait before it forces the pod to terminate. If you set this value to `-1`, the default timeout is set to the one specified by the pod.

22. (Optional) To configure when the node drains, enable the following:
    - **Force node to drain even if it has running pods not managed by a ReplicationController, ReplicaSet, Job, DaemonSet or StatefulSet.**
    - **Force node to drain even if it has running DaemonSet-managed pods.**
    - **Force node to drain even if it has running running pods using emptyDir.**
    - **Force node to drain even if pods are still running after timeout.**

    > **warning:** If you select **Force node to drain even if pods are still running after timeout**, the node kills all running workloads on pods. Before enabling this configuration, set **Node Drain Timeout** to a value greater than `0`.

    For more information about configuring default node drain behavior, see Worker Node Hangs Indefinitely in *Troubleshooting*.

23. Click **Save**.

Deactivate a Plan

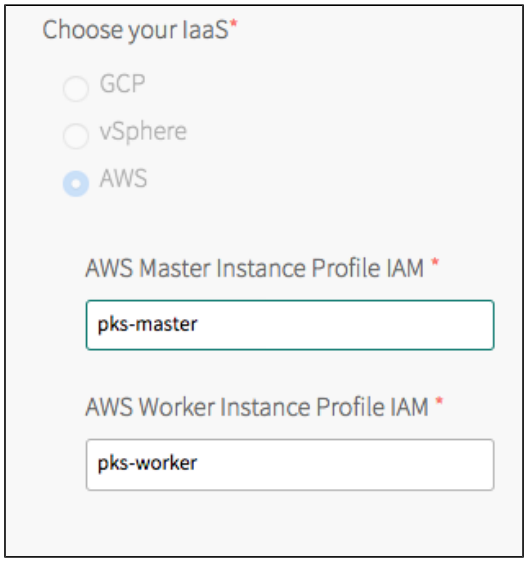
To deactivate a plan, perform the following steps:

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`.

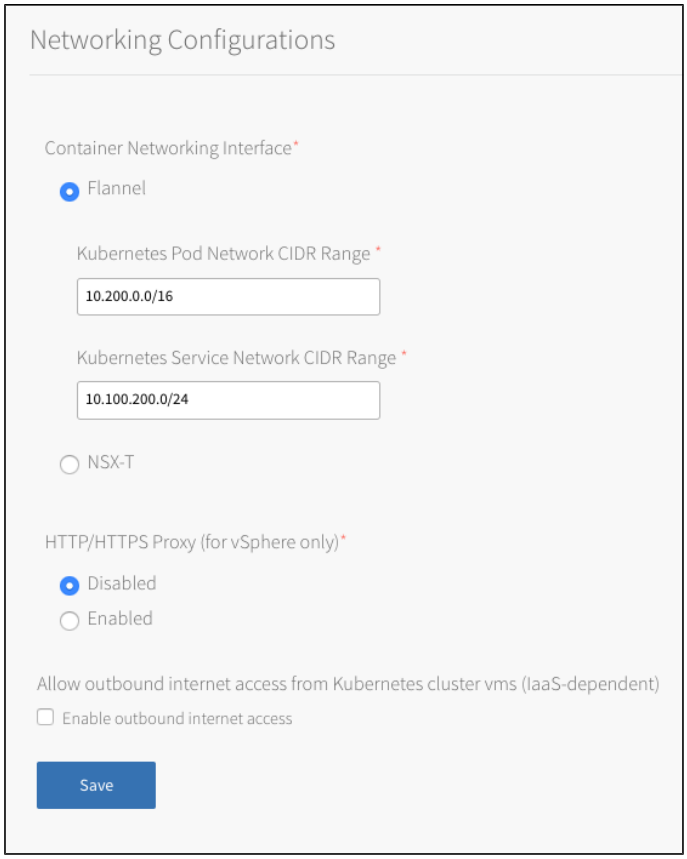
4. Enter your **AWS Worker Instance Profile IAM**. This is the instance profile name associated with the worker node. To retrieve the instance profile name, run `terraform output` and locate the value for the field `pks_worker_iam_instance_profile_name`.

5. Click **Save**.

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 and HTTPS traffic from your Kubernetes clusters and the PKS API server. See Using Proxies with Enterprise PKS on AWS for instructions on how to enable a proxy.

5. (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.

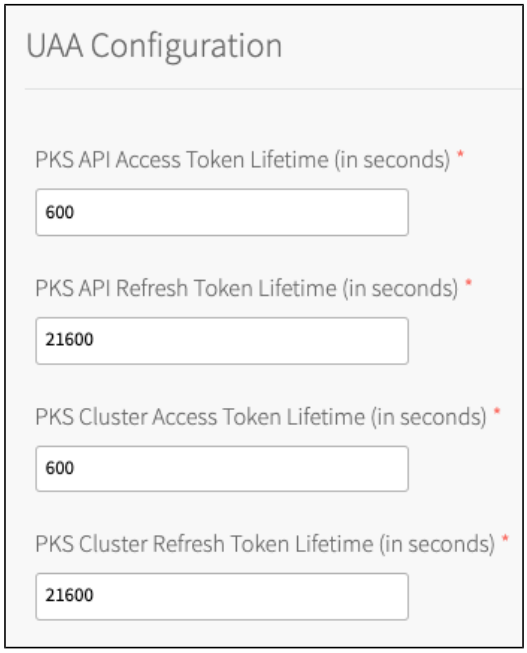
6. Click **Save**.

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. This field defaults to `600`.



3. Under **PKS API Refresh Token Lifetime**, enter a time in seconds for the PKS API refresh token lifetime. This field defaults to `21600`.

4. Under **PKS Cluster Access Token Lifetime**, enter a time in seconds for the cluster access token lifetime. This field defaults to `600`.

5. Under **PKS Cluster Refresh Token Lifetime**, enter a time in seconds for the cluster refresh token lifetime. This field defaults to `21600`.

> **Note**: Pivotal recommends using the default UAA token timeout values. By default, access tokens expire after ten minutes and refresh tokens expire after six hours. If you want to customize your token timeout values, see Token Management in *UAA Overview*.

6. Under **Configure created clusters to use UAA as the OIDC provider**, select **Enabled** or **Disabled**. If you click **Enabled**, Kubernetes verifies end-user identities based on authentication executed by UAA. For more information, see the table below.

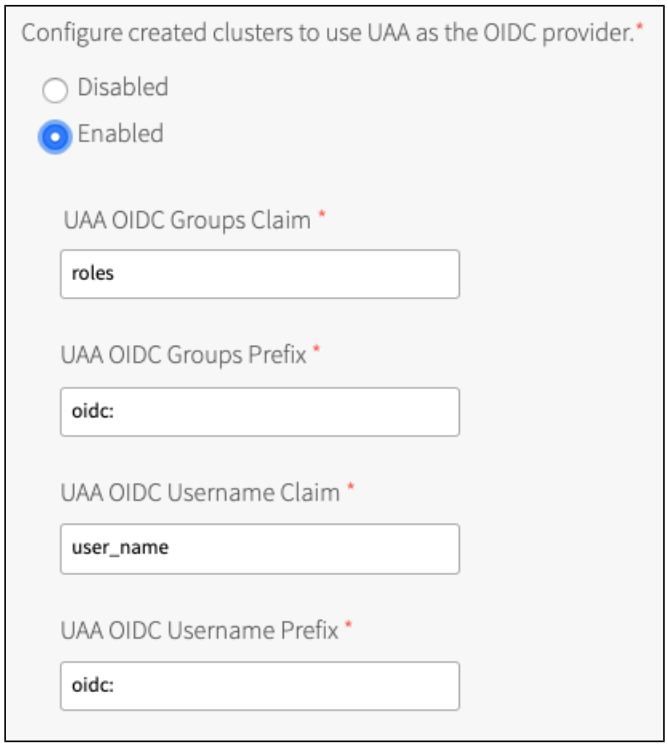
Option	Description
Disabled	If you do not enable UAA as the OpenID Connect (OIDC) provider, Kubernetes authenticates users against its internal user management system.

Enabled	If you enable UAA as the OIDC provider, Kubernetes authenticates users as follows: ◦ If you select **Internal UAA** in the next step, Kubernetes authenticates users against the internal UAA authentication mechanism. ◦ If you select **LDAP Server** in the next step, Kubernetes authenticates users against the LDAP server. ◦ If you select **SAML Identity Provider** in the next step, Kubernetes authenticates users against the SAML identity provider.

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

To configure Enterprise PKS to use UAA as the OIDC provider, do the following:

a. Under **Configure created clusters to use UAA as the OIDC provider**, select **Enabled**.



b. For **UAA OIDC Groups Claim**, enter the name of your groups claim. This is used to set a user's group in the JSON Web Token (JWT) claim. The default value is `roles`.
c. For **UAA OIDC Groups Prefix**, enter a prefix for your groups claim. This prevents conflicts with existing names. For example, if you enter the prefix `oidc:`, UAA creates a group name like `oidc:developers`. If you are configuring a new Enterprise PKS installation, the default is `oidc:`. If you are upgrading to Enterprise PKS v1.5, the default is `-`.
d. For **UAA OIDC Username Claim**, enter the name of your username claim. This is used to set a user's username in the JWT claim. The default value is `user_name`. Depending on your provider, admins can enter claims besides `user_name`, like `email` or `name`.
e. For **UAA OIDC Username Prefix**, enter a prefix for your username claim. This prevents conflicts with existing names. For example, if you enter the prefix `oidc:`, UAA creates a username like `oidc:admin`. If you are configuring a new Enterprise PKS installation, the default is `oidc:`. If you are upgrading to Enterprise PKS v1.5, the default is `-`.

Note: Pivotal recommends adding OIDC prefixes to prevent OIDC users and groups from gaining unintended cluster privileges. When you upgrade to Enterprise PKS v1.5, if you do not change the values for **UAA OIDC Groups Prefix** or **UAA OIDC Username Prefix**, Enterprise PKS does not add prefixes.

> **warning:** If you change the above values for a pre-existing Enterprise PKS installation, you must change any existing role bindings that bind to a username or group. If you do not change your role bindings, developers cannot access Kubernetes clusters. For instructions about creating role bindings, see Managing Cluster Access and Permissions.

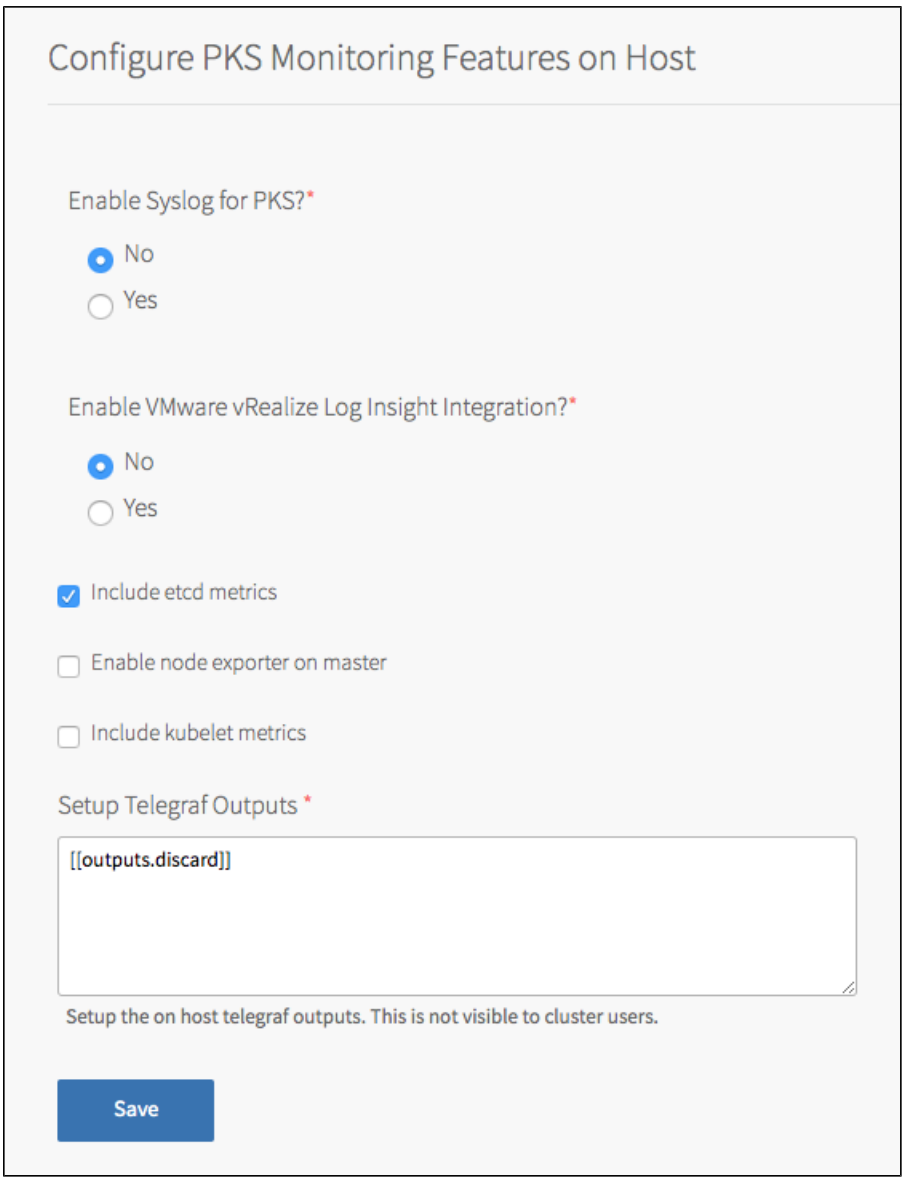
7. Select one of the following options:
   - To use an internal user account store for UAA, select **Internal UAA**. Click **Save** and continue to (Optional) Host Monitoring.
   - To use LDAP for UAA, select **LDAP Server** and continue to Connecting Enterprise PKS to an LDAP Server.
   - To use SAML for UAA, select **SAML Identity Provider** and continue to Connecting Enterprise PKS to a SAML Identity Provider.

(Optional) Host Monitoring

In **Host Monitoring**, you can configure one or more of the following:

- To configure Syslog for BOSH-deployed VMs, see Syslog. Syslog forwards log messages from all BOSH-deployed VMs to a syslog endpoint.
- To configure the Telegraf agent, see Telegraf. The Telegraf agent sends metrics from master node and worker node VMs to a monitoring service, such as Wavefront or Datadog.

For more information about these components, see Monitoring PKS and PKS-Provisioned Clusters.



Syslog

To configure Syslog for all BOSH-deployed VMs in Enterprise PKS:

1. Click **Host Monitoring**.
2. Under **Enable Syslog for PKS**, select **Yes**.
3. Under **Address**, enter the destination syslog endpoint.
4. Under **Port**, enter the destination syslog port.
5. Under **Transport Protocol**, select a transport protocol for log forwarding.
6. (Optional) To enable TLS encryption during log forwarding, complete the following steps:
   a. Ensure **Enable TLS** is selected.

> **Note**: Logs may contain sensitive information, such as cloud provider credentials. Pivotal strongly recommends that you enable TLS encryption for log forwarding.

b. Under **Permitted Peer**, provide the accepted fingerprint (SHA1) or name of remote peer. For example, `*.YOUR-LOGGING-SYSTEM.com`.
c. 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. (Optional) Under **Max Message Size**, enter a maximum message size for logs that are forwarded to a syslog endpoint. By default, the **Max Message Size** field is 10,000 characters.

8. Click **Save**.

Telegraf

To configure Enterprise PKS to use Telegraf for metric collection:

1. Create a configuration file for your monitoring service. For instructions, see Create a Configuration File.

2. Select one or more of the following checkboxes:

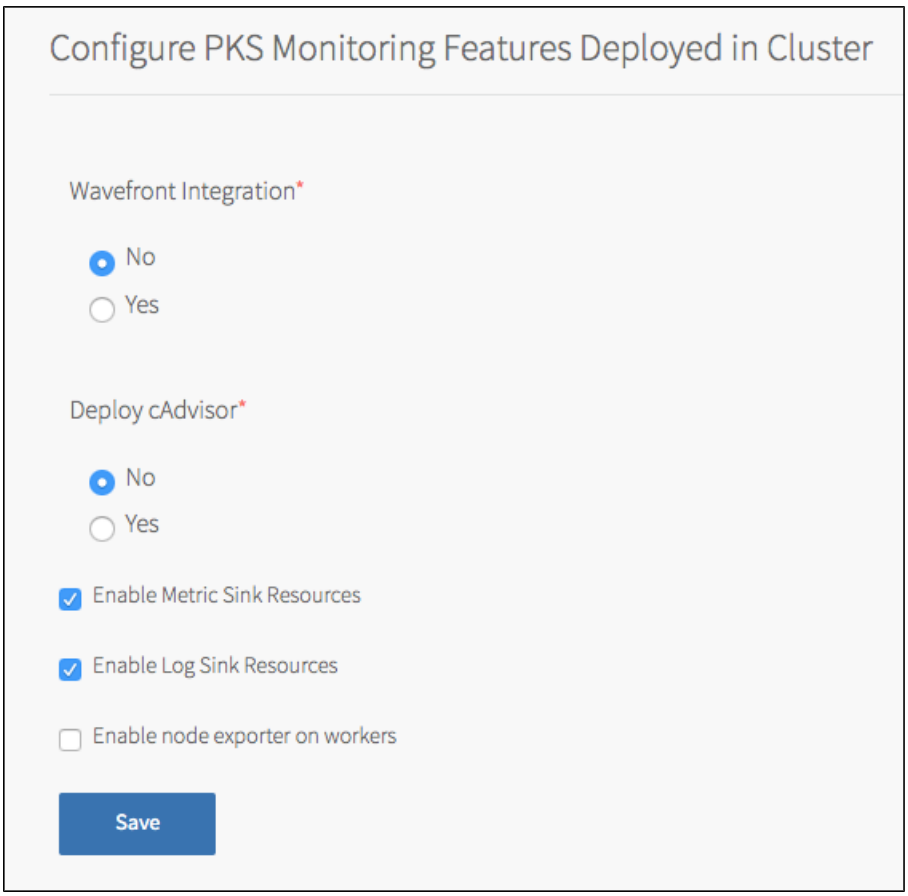
   - **Include etcd metrics**: This includes etcd server and debug metrics.
   - **Enable node exporter on master**: This enables Node Exporter on the *localhost* of each master node VM.
   - **Include kubelet metrics**: This includes all workload metrics across your Kubernetes clusters. Enabling **Include kubelet metrics** generates a high volume of metrics.

3. In **Setup Telegraf Outputs**, replace the default value `[[outputs.discard]]` with the contents of the configuration file that you created above.

4. Click **Save**.

(Optional) In-Cluster Monitoring

In **In-Cluster Monitoring**, you can configure one or more observability components and integrations that run in Kubernetes clusters and capture logs and metrics about your workloads. For more information, see Monitoring Workers and Workloads.



To configure in-cluster monitoring:

- To configure Wavefront, see Wavefront.
- To configure cAdvisor, see cAdvisor.
- To configure sink resources, see:
  - Metric Sink Resources
  - Log Sink Resources

  You can enable both log and metric sink resources or only one of them.

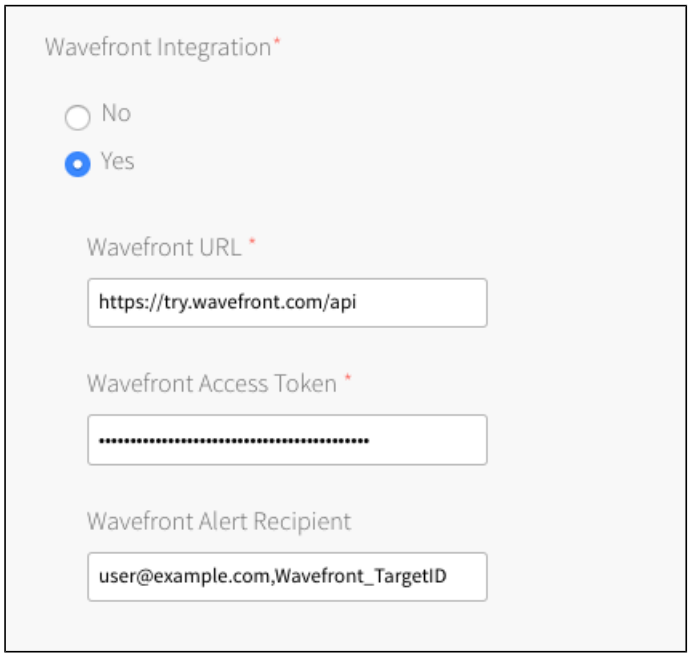
Wavefront

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 the Wavefront documentation.

To enable and configure Wavefront monitoring:

1. In the Enterprise PKS tile, select **In-Cluster Monitoring**.



2. 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**, using the following syntax:

```
USER-EMAIL,WAVEFRONT-TARGETID_001,WAVEFRONT-TARGETID_002
```

   Where:

   - `USER-EMAIL` is the alert recipient's email address.
   - `WAVEFRONT-TARGETID_001` and `WAVEFRONT-TARGETID_002` are your comma-delimited Wavefront Target IDs.

   For example:

```
randomuser@example.com,51n6psdj933ozdjf
```

6. Click **Save**.

To create alerts, you must enable errands in Enterprise PKS.

1. In the Enterprise PKS tile, select **Errands**.
2. On the **Errands** pane, enable **Create pre-defined Wavefront alerts errand**.
3. Enable **Delete pre-defined Wavefront alerts errand**.
4. Click **Save**. Your settings apply to any clusters created after you have saved these configuration settings and clicked **Apply Changes**.

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

cAdvisor

cAdvisor is an open source tool for monitoring, analyzing, and exposing Kubernetes container resource usage and performance statistics.

To deploy a cAdvisor container:

1. Select **In-Cluster Monitoring**.
2. Under **Deploy cAdvisor**, select **Yes**.
3. Click **Save**.

> **Note:** For information about configuring cAdvisor to monitor your running Kubernetes containers, see cAdvisor in the cAdvisor GitHub repository. For general information about Kubernetes cluster monitoring, see Tools for Monitoring Resources in the Kubernetes documentation.

Metric Sink Resources

You can configure PKS-provisioned clusters to send Kubernetes node metrics and pod metrics to metric sinks. For more information about metric sink resources and what to do after you enable them in the tile, see Sink Resources in *Monitoring Workers and Workloads*.

To enable clusters to send Kubernetes node metrics and pod metrics to metric sinks:

1. In **In-Cluster Monitoring**, select **Enable Metric Sink Resources**. If you enable this checkbox, Enterprise PKS deploys Telegraf as a `DaemonSet`, a pod that runs on each worker node in all your Kubernetes clusters.
2. (Optional) To enable Node Exporter to send worker node metrics to metric sinks of kind `ClusterMetricSink`, select **Enable node exporter on workers**. If you enable this checkbox, Enterprise PKS deploys Node Exporter as a `DaemonSet`, a pod that runs on each worker node in all your Kubernetes clusters.
   For instructions on how to create a metric sink of kind `ClusterMetricSink` for Node Exporter metrics, see Create a ClusterMetricSink Resource for Node Exporter Metrics in *Creating and Managing Sink Resources*.
3. Click **Save**.

Log Sink Resources

You can configure PKS-provisioned clusters to send Kubernetes API events and pod logs to log sinks. For more information about log sink resources and what to do after you enable them in the tile, see Sink Resources in *Monitoring Workers and Workloads*.

To enable clusters to send Kubernetes API events and pod logs to log sinks:

1. Select **Enable Log Sink Resources**. If you enable this checkbox, Enterprise PKS deploys Fluent Bit as a `DaemonSet`, a pod that runs on each worker node in all your Kubernetes clusters.
2. Click **Save**.

Tanzu Mission Control (Experimental)

Participants in the VMware Tanzu Mission Control beta program can use the **Tanzu Mission Control (Experimental)** pane of the Enterprise PKS tile to integrate their Enterprise PKS deployment with Tanzu Mission Control.

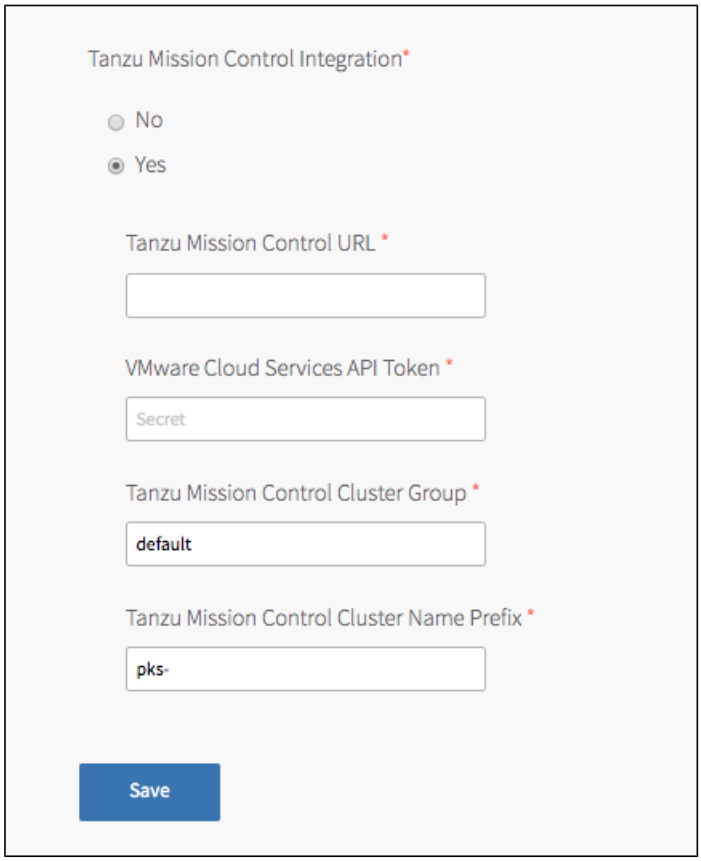
Tanzu Mission Control integration lets you monitor and manage Enterprise PKS clusters from the Tanzu Mission Control console, which makes the Tanzu Mission Control console a single point of control for all Kubernetes clusters.

> **warning:** VMware Tanzu Mission Control is currently experimental beta software and is intended for evaluation and test purposes only. For more information about Tanzu Mission Control, see the VMware Tanzu Mission Control home page.

To integrate Enterprise PKS with Tanzu Mission Control:

1. Confirm that the PKS Control Plane VM has internet access and can connect to `*.tmc.cloud.vmware.com` and the other outbound URLs listed in the What Happens When You Attach a Cluster section of the Tanzu Mission Control documentation.

2. Navigate to the **Enterprise PKS** tile > the **Tanzu Mission Control (Experimental)** pane and select **Yes** under **Tanzu Mission Control Integration**.



3. Configure the fields below:

   - **Tanzu Mission Control URL**: Enter the Org URL of your Tanzu Mission Control subscription, without a trailing slash `/`). For example, `YOUR-ORG.tmc.cloud.vmware.com`.
   - **VMware Cloud Services API token**: Enter your API token to authenticate with VMware Cloud Services APIs. You can retrieve this token by logging in to VMware Cloud Services and viewing your account information.
   - **Tanzu Mission Control Cluster Group**: Enter the name of a Tanzu Mission Control cluster group.

     The name can be `default` or another value, depending on your role and access policy:

- `Org Member` users in VMware cloud services have a `service.admin` role in Tanzu Mission Control. These users:
  - By default, can create and attach clusters only in the `default` cluster group.
  - Can create and attach clusters to other cluster groups after an `organization.admin` user grants them the `clustergroup.admin` or `clustergroup.edit` role for those groups.
- `Org Owner` users in VMware cloud services have `organization.admin` permissions in Tanzu Mission Control. These users:
  - Can create cluster groups.
  - Can grant `clustergroup` roles to `service.admin` users through the Tanzu Mission Control Access Policy view.

For more information about role and access policy, see Access Control in the *VMware Tanzu Mission Control Product Documentation*.

- **Tanzu Mission Control Cluster Name Prefix**: Enter a name prefix for identifying the Enterprise PKS clusters in Tanzu Mission Control.

4. Click **Save**.

warning: After the Enterprise PKS tile is deployed with a configured cluster group, the cluster group cannot be updated.

Note: When you upgrade your Kubernetes clusters and have Tanzu Mission Control integration enabled, existing clusters will be attached to Tanzu Mission Control.

CEIP and Telemetry

To configure VMware's Customer Experience Improvement Program (CEIP) and the Pivotal Telemetry Program (Telemetry), do the following:

1. Click **CEIP and Telemetry**.

2. Review the information about the CEIP and Telemetry.

CEIP and Telemetry

About the CEIP and Telemetry Program

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.

Customers who participate (at the enhanced tier) are eligible for several benefits, including Proactive Support.

Additional information regarding the data collected through CEIP or Telemetry, and the purposes for which it is used by VMware is set forth in the Trust & Assurance Center and for Pivotal on the Pivotal Telemetry pages. If you prefer not to participate in CEIP and Telemetry for PKS, you should not join below. You may join or leave CEIP and Telemetry for PKS at any time.

Levels of Participation

No personally identifiable information (PII) is collected at either level of participation. Please refer to the data dictionary for more information on the data we collect.

Standard: This participation tier is anonymous. Your data will be used to improve PKS, but is not identifiable to your organization

Enhanced: This participation tier allows us to identify your organization so we may provide proactive support and other benefits. See sample reports for more details.

Please Note

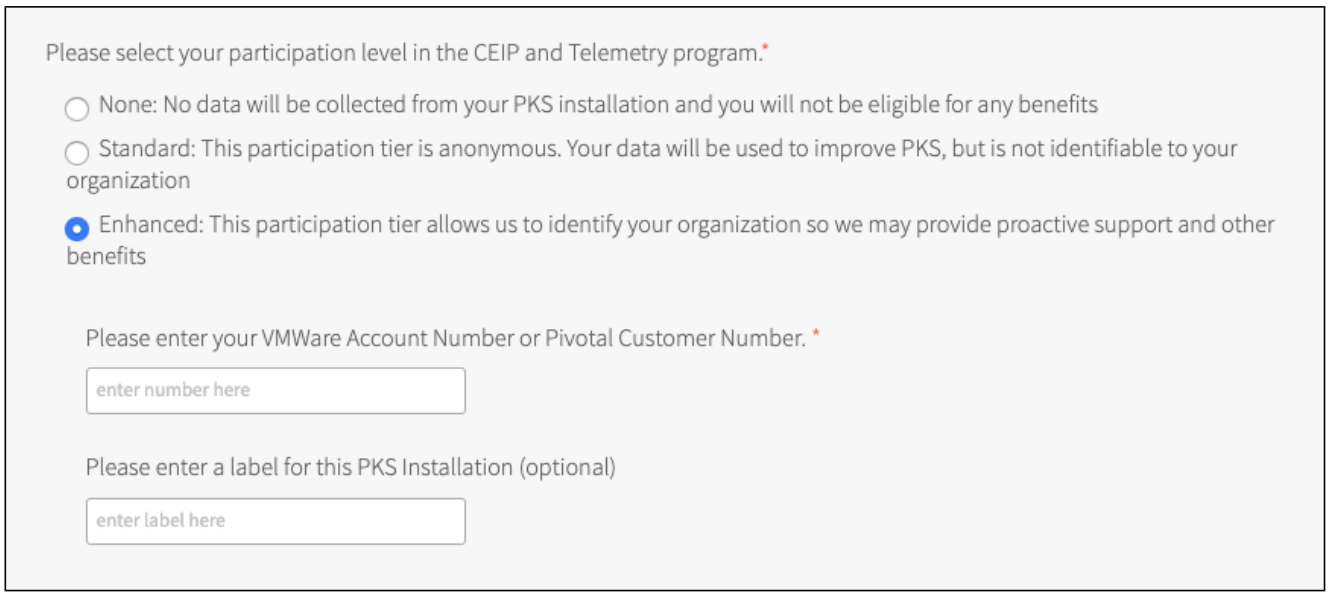
- If you are opting in on behalf of an organization (and not for you as an individual), you represent and warrant that you have legal authority to bind that organization, and you hereby join CEIP/Telemetry on behalf of your organization.
- If you are opting in to the enhanced tier, in the event a term or condition of CEIP or the Telemetry program conflicts with a term or condition of a previously executed license procurement agreement between you and Licensor (Pivotal or VMware), the CEIP or Telemetry program terms supersede solely for purposes of CEIP and Telemetry
- If you are running PKS on a private network, you will need to enable outgoing internet access by opening your firewall to allow traffic to `https://vcsa.vmware.com/ph` on port `443`

Resources

- Data Dictionary
- Participation Benefits
- Sample Reports
- Trust and Assurance Center
- Pivotal Telemetry

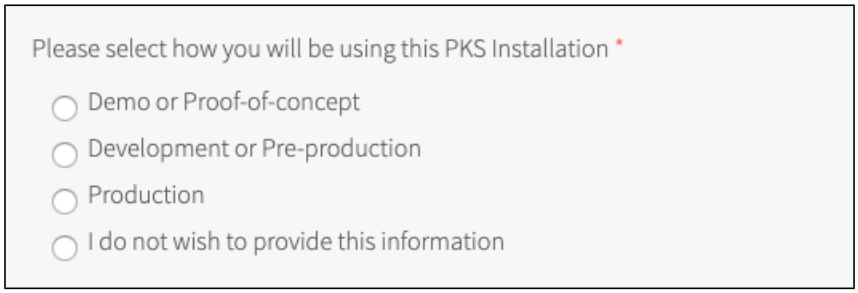
View a larger version of this image.

3. To specify your level of participation in the CEIP and Telemetry program, select one of the **Participation Level** options:
   - **None:** If you select this option, data is not collected from your Enterprise PKS installation.
   - (Default) **Standard:** If you select this option, data is collected from your Enterprise PKS installation to improve Enterprise PKS. This participation level is anonymous and does not permit the CEIP and Telemetry to identify your organization.
   - **Enhanced:** If you select this option, data is collected from your Enterprise PKS installation to provide you proactive support and other benefits. This participation level permits the CEIP and Telemetry to identify your organization.



For more information about the CEIP and Telemetry participation levels, see Participation Levels in *Telemetry*.

4. If you selected the **Enhanced** participation level, complete the following:
   - Enter your VMware account number or Pivotal customer number in the **VMware Account Number or Pivotal Customer Number** field. If you are a VMware customer, you can find your VMware Account Number in your **Account Summary** on my.vmware.com. If you are a Pivotal customer, you can find your Pivotal Customer Number in your Pivotal Order Confirmation email.
   - (Optional) Enter a descriptive name for your PKS installation in the **PKS Installation Label** field. The label you assign to this installation will be used in telemetry reports to identify the environment.
5. To provide information about the purpose for this installation, select an option in the **PKS Installation Type** list.



6. Click **Save**.

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

Note: Even if you select **None**, Enterprise PKS-provisioned clusters send usage data to the PKS control plane. However, this data is not sent to VMware or Pivotal and remains on your Enterprise PKS installation.

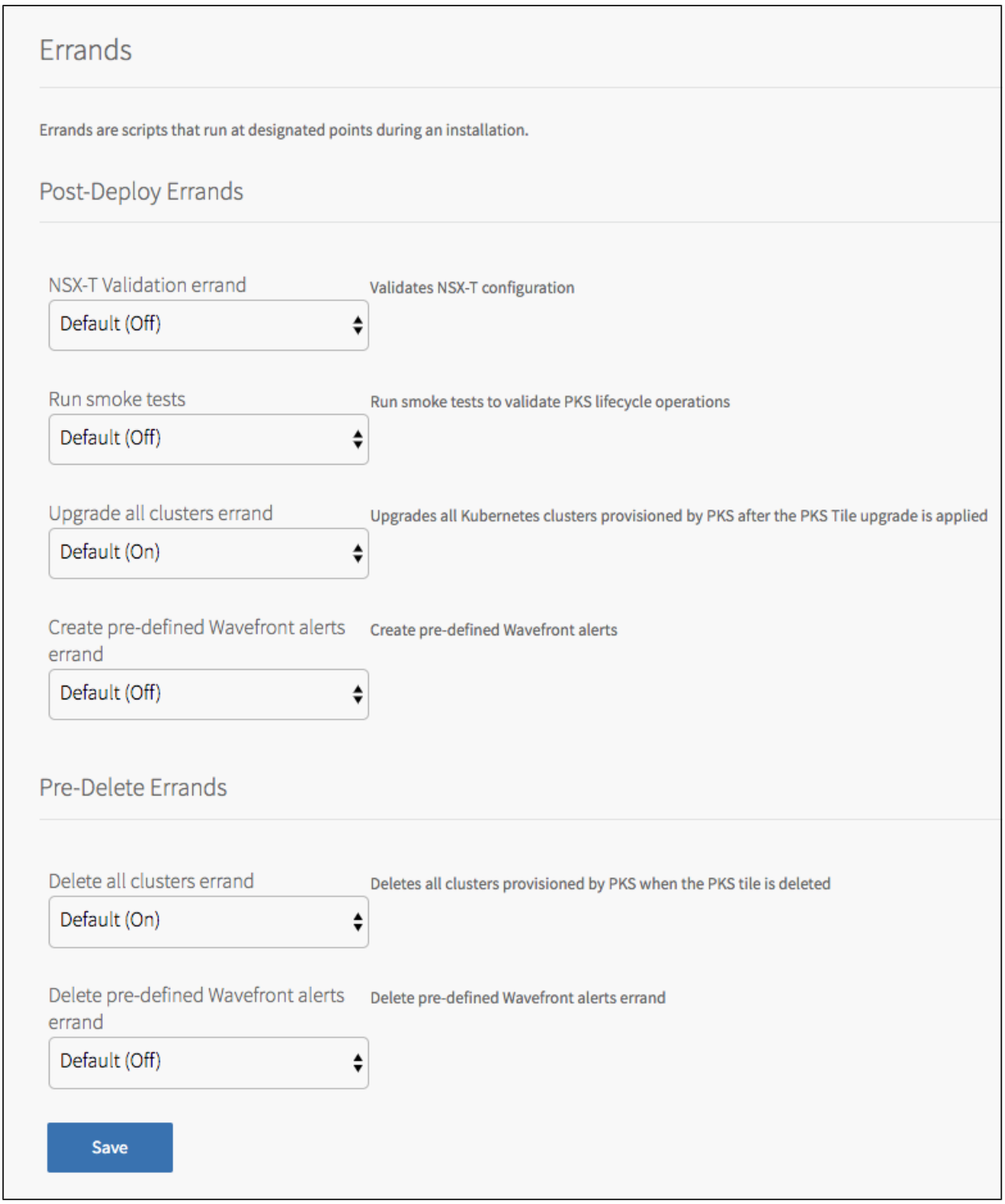
Errands

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

To configure when post-deploy and pre-delete errands for Enterprise 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 CLI to create a Kubernetes cluster and then delete it. If the creation or deletion fails, the errand fails and the installation of the Enterprise 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.

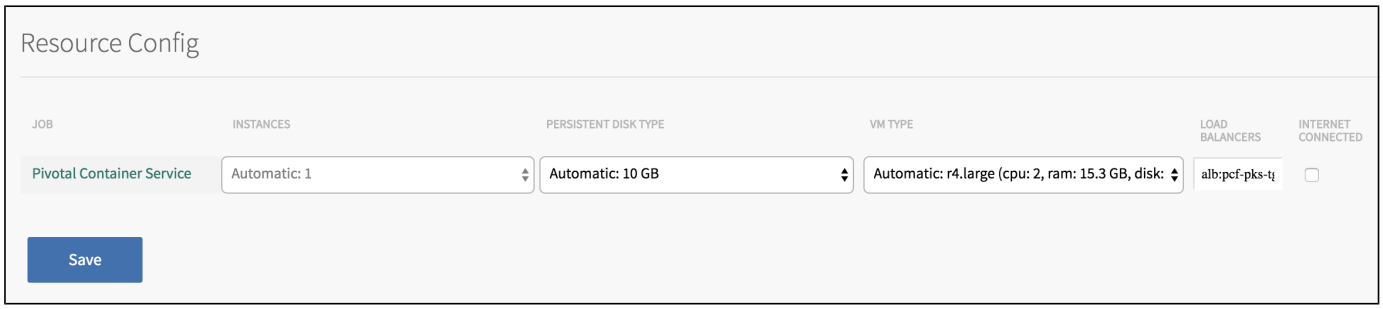
warning: If **Upgrade all clusters errand** is enabled, updating the Enterprise PKS tile with a new Linux stemcell triggers the

rolling of every Linux VM in each Kubernetes cluster. Similarly, updating the Enterprise PKS tile with a new Windows stemcell triggers the rolling of every Windows VM in your Kubernetes clusters. This automatic rolling ensures that all your VMs are patched. To avoid workload downtime, use the resource configuration recommended in What Happens During Enterprise PKS Upgrades and Maintaining Workload Uptime.

Resource Config

To modify the resource usage of Enterprise 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.

3. (Optional) Edit other resources used by the **Pivotal Container Service** job. The **Pivotal Container Service** job requires a VM with the following minimum resources:

CPU	Memory	Disk
2	8 GB	29 GB

> **Note**: The automatic **VM Type** value matches the minimum recommended size for the **Pivotal Container Service** job. If you experience timeouts or slowness when interacting with the PKS API, select a **VM Type** with greater CPU and memory resources.

4. (Optional) If you do not use a NAT instance, select **Internet Connected** to allow component instances direct access to the internet.

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 need to retrieve the PKS API endpoint to allow your organization to use the API to create, update, and delete Kubernetes clusters.

To retrieve the PKS API endpoint, do the following:

1. Navigate to the Ops Manager **Installation Dashboard**.
2. Click the **Enterprise PKS** 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 CLI and the Kubernetes CLI help you interact with your Enterprise 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 Authentication for Enterprise PKS

Follow the procedures in Setting Up Enterprise PKS Admin Users on AWS.

Next Steps

After installing Enterprise PKS on AWS, you might want to do one or more of the following:

- Create a load balancer for your Enterprise PKS clusters. For more information, see Creating and Configuring an AWS Load Balancer for Enterprise PKS Clusters.
- Create your first Enterprise PKS cluster. For more information, see Creating Clusters.

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

Using Proxies with Enterprise PKS on AWS

In this topic

Page last updated:

This topic describes how to use proxies with VMware Enterprise PKS on AWS.

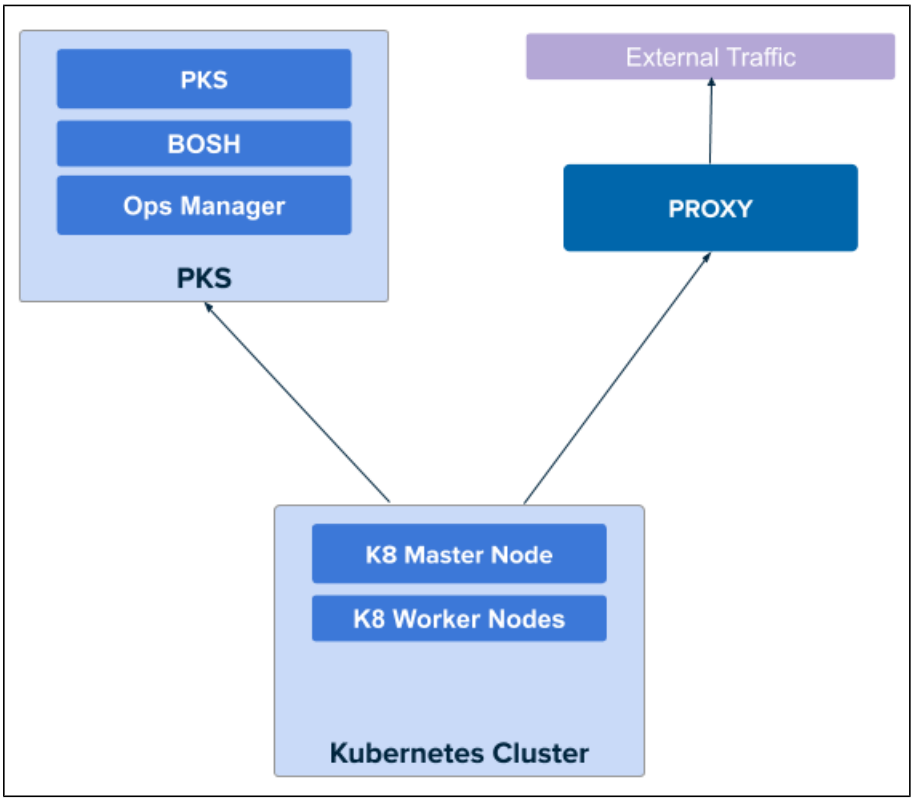
Overview

If your environment includes HTTP proxies, you can configure Enterprise PKS on AWS to use these proxies so that Enterprise PKS-deployed Kubernetes master and worker nodes access public Internet services and other internal services through a proxy.

In addition, Enterprise PKS proxy settings apply to the PKS API instance. When an Enterprise PKS operator creates a Kubernetes cluster, the PKS API instance VM behind a proxy is able to manage AWS components on the standard network.

You can also proxy outgoing HTTP/HTTPS traffic from Ops Manager and the BOSH Director so that all Enterprise PKS components use the same proxy service.

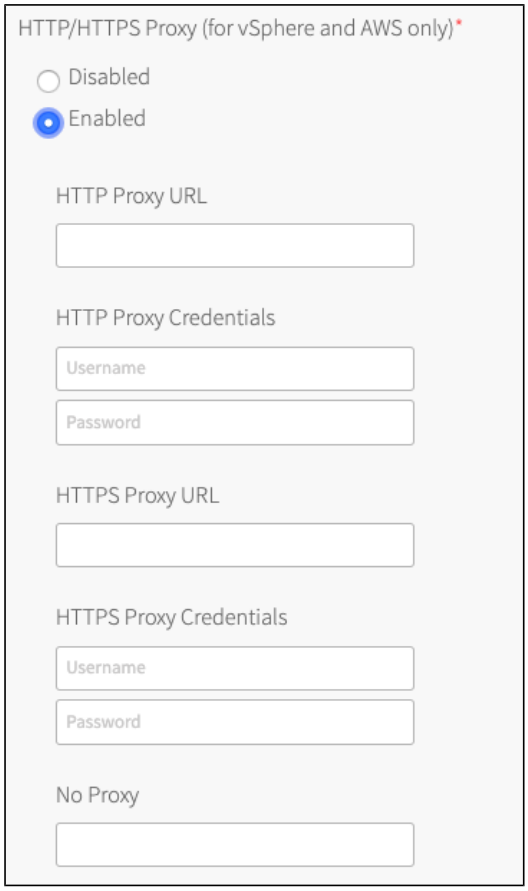
The following diagram illustrates the network architecture:



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 **Enterprise PKS** tile.
3. Click **Networking**.
4. Under **HTTP/HTTPS Proxy**, select **Enabled** to configure an Enterprise PKS global proxy for all outgoing HTTP and HTTPS traffic from your Kubernetes clusters.



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

Configuring Enterprise PKS to use these proxies allows Enterprise PKS-deployed Kubernetes nodes to access public Internet services and other internal services.

> **Note**: This setting will not set the proxy for running Kubernetes workloads or pods.

To complete your global proxy configuration for all outgoing HTTP/HTTPS traffic from your Kubernetes clusters, perform the following steps:

a. To proxy outgoing HTTP traffic, enter the URL of your HTTP proxy endpoint under **HTTP Proxy URL**. For example, `http://myproxy.com:1234`.
b. (Optional) If your outgoing HTTP proxy uses basic authentication, enter the username and password in the **HTTP Proxy Credentials** fields.

c. To proxy outgoing HTTPS traffic, enter the URL of your HTTP proxy endpoint under **HTTPS Proxy URL**. For example, `http://myproxy.com:1234`.

> **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.

d. (Optional) If your HTTPS proxy uses basic authentication, enter the username and password in the **HTTPS Proxy Credentials** fields.

e. Under **No Proxy**, enter the comma-separated list of IP addresses that must bypass the proxy to allow for internal Enterprise PKS communication.

   The list should include `127.0.0.1` and `localhost`.

   Also include:

   - Your Enterprise PKS environment's CIDRs, such as the service network CIDR where your Enterprise PKS cluster is deployed, the deployment network CIDR, the node network IP block CIDR, and the pod network IP block CIDR.
   - If a component is communicating with Enterprise PKS or a registry using a hostname, such as the Harbor API FQDN, instead of an IP address you will need to add the corresponding FQDN to the **No Proxy** list.
   - Any additional IP addresses or domain names that should bypass the proxy.

     The **No Proxy** property for AWS accepts wildcard domains denoted by a prefixed `*.` or `.`.

     For example:

```
127.0.0.1,localhost,
*.example1.com,
.example2.com,
example3.com,
198.51.100.0/24,
203.0.113.0/24,
192.0.2.0/24
```

> **Note**: By default the `169.254.169.254`, `10.100.0.0/8` and `10.200.0.0/8` IP address ranges, `.internal`, `.svc`, `.svc.cluster.local`, `.svc.cluster`, and your Enterprise PKS FQDN are not proxied. This allows internal Enterprise 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. To save your changes to the PKS tile, click **Save**.

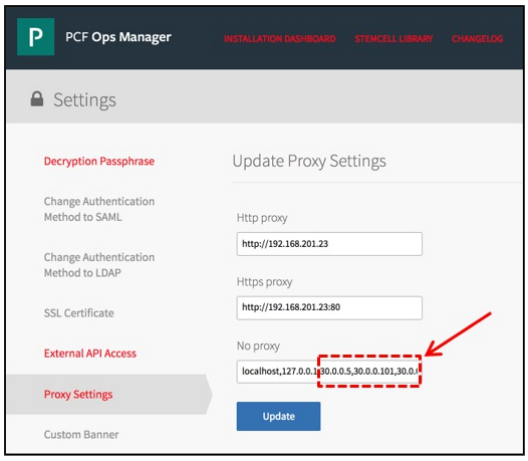
6. Proceed with any remaining Enterprise PKS tile configurations and deploy Enterprise PKS. See Installing Enterprise PKS on

AWS.

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.

Setting Up Enterprise PKS Admin Users on AWS

In this topic

Page last updated:

This topic describes how to create admin users in VMware Enterprise PKS with User Account and Authentication (UAA). Creating at least one admin user is a necessary step during the initial set up of Enterprise PKS.

Overview

UAA is the identity management service for Enterprise PKS. Enterprise PKS includes a UAA server, which is hosted on the PKS control plane VM.

To interact with the UAA server, you can use the UAA Command Line Interface (UAAC). You can either run UAAC commands from the Ops Manager VM or install UAAC on your local workstation.

Prerequisites

Before setting up admin users for Enterprise PKS, you must have one of the following:

- SSH access to the Ops Manager VM
- A machine that can connect to your PKS control plane VM

Step 1: Connect to the PKS Control Plane VM

You can connect to the PKS control plane VM from the Ops Manager VM or from a different machine such as your local workstation.

Option 1: Connect through the Ops Manager VM

You can connect to the PKS control plane VM by logging in to the Ops Manager VM through SSH. To SSH into the Ops Manager VM on AWS, do the following:

1. Retrieve the key pair you used when you created the Ops Manager VM. To see the name of the key pair:
    a. In the AWS console, click the Ops Manager VM
    b. Locate the `key pair name` in the properties.
2. On the AWS **EC2 instances** page, locate the Ops Manager FQDN.
3. Change the permissions on the `.pem` file to be more restrictive by running the `chmod 600` command. For example:

```
$ chmod 600 ops_mgr.pem
```

4. SSH into the Ops Manager VM by running the following command:

```
ssh -i ops_mgr.pem ubuntu@OPS-MANAGER-FQDN
```

Where `OPS-MANAGER-FQDN` is the FQDN of Ops Manager. For example:

```
$ ssh -i ops_mgr.pem ubuntu@my-opsmanager-fqdn.example.com
```

5. Proceed to the Log In as a UAA Admin section to manage users with UAAC.

Option 2: Connect through a Non-Ops Manager Machine

To connect to the PKS control plane VM and run UAA commands, do the following:

1. Install UAAC on your machine. For example:

```
gem install cf-uaac
```

2. Download a copy of your Ops Manager root CA certificate to the machine. To download the certificate, do the following:

   a. In a web browser, navigate to the FQDN of Ops Manager and log in.
   b. In Ops Manager, navigate to **Settings** in the drop-down menu under your username.
   c. Click **Advanced Options**.
   d. On the **Advanced Options** configuration page, click **Download Root CA Cert**
   e. Move the certificate to a secure location on your machine and record the path.

3. Proceed to the Log In as a UAA Admin section to create admin users with UAAC.

Step 2: Log In as a UAA Admin

Before creating PKS users, you must log in to the UAA server as a UAA admin. To log in to the UAA server, do the following:

1. Retrieve the UAA management admin client secret:

   a. In a web browser, navigate to the Ops Manager **Installation Dashboard** and click the **Enterprise PKS** tile.
   b. Click the **Credentials** tab.
   c. Click **Link to Credential** next to **Pks Uaa Management Admin Client** and copy the value of `secret`.

2. Target your UAA server by running the following command:

```
uaac target https://PKS-API:8443 --ca-cert CERTIFICATE-PATH
```

Where:

- `PKS-API` is the domain name of your PKS API server. You entered this domain name in the **Enterprise PKS** tile > **PKS API** > **API Hostname (FQDN)**.
- `CERTIFICATE-PATH` is the path to your Ops Manager root CA certificate. Provide this certificate to validate the PKS API

certificate with SSL.

- If you are logged in to the Ops Manager VM, specify `/var/tempest/workspaces/default/root_ca_certificate` as the path. This is the default location of the root certificate on the Ops Manager VM.
- If you downloaded the Ops Manager root CA certificate to your machine, specify the path where you stored the 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 ignore it because it has no impact.

3. Authenticate with UAA by running the following command:

```
uaac token client get admin -s ADMIN-CLIENT-SECRET
```

Where `ADMIN-CLIENT-SECRET` is your UAA management admin client secret that you retrieved in a previous step. The client username is `admin`.

Step 3: Assign Enterprise PKS Cluster Scopes

The `pks.clusters.manage` and `pks.clusters.admin` UAA scopes grant users the ability to create and manage Kubernetes clusters in Enterprise PKS. For information about UAA scopes in Enterprise PKS, see UAA Scopes for Enterprise PKS Users.

To create Enterprise PKS users with the `pks.clusters.manage` or `pks.clusters.admin` UAA scope, perform one or more of the following procedures based on the needs of your deployment:

- To assign PKS cluster scopes to an individual user, see Grant Enterprise PKS Access to an Individual User. Follow this procedure if you selected **Internal UAA** when you configured **UAA** in the Enterprise PKS tile. For more information, see Installing Enterprise PKS on AWS.
- To assign PKS cluster scopes to an LDAP group, see Grant Enterprise PKS Access to an External LDAP Group. Follow this procedure if you selected **LDAP Server** when you configured **UAA** in the Enterprise PKS tile. For more information, see Installing Enterprise PKS PKS on AWS.
- To assign PKS cluster scopes to a SAML group, see Grant Enterprise PKS Access to an External SAML Group. Follow this procedure if you selected **SAML Identity Provider** when you configured **UAA** in the Enterprise PKS tile. For more information, see Installing Enterprise PKS PKS on AWS.
- To assign PKS cluster scopes to a client, see Grant Enterprise PKS Access to a Client.

Next Step

After you create admin users in Enterprise PKS, the admin users can create and manage Kubernetes clusters in Enterprise PKS. For more information, see Managing Kubernetes Clusters and Workloads.

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

Azure

Page last updated:

This topic lists the procedures to follow to install VMware Enterprise PKS on Microsoft Azure.

Install Enterprise PKS on Azure

To install Enterprise PKS on Azure, follow the instructions below:

- Azure Prerequisites and Resource Requirements
- Installing and Configuring Ops Manager on Azure
- Creating Managed Identities in Azure for Enterprise PKS
- Installing Enterprise PKS on Azure
- Configuring an Azure Load Balancer for the PKS API
- Setting Up Enterprise PKS Admin Users on Azure

Install the PKS and Kubernetes CLIs

The PKS CLI and Kubernetes CLI help you interact with your Enterprise 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.

Azure Prerequisites and Resource Requirements

In this topic

Page last updated:

This topic describes the prerequisites and resource requirements for installing VMware Enterprise PKS on Microsoft Azure.

Prerequisites

Before installing Enterprise PKS:

1. Review the sections below.
2. Install and configure Ops Manager. To install Ops Manager, follow the instructions in Installing and Configuring Ops Manager on Azure.

Subscription Requirements

For Enterprise PKS and Kubernetes services to run correctly, you must have at least a `standard` subscription tier.

Resource Requirements

Installing Ops Manager and Enterprise 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 Pivotal Container Service VM as follows:

Number of Pods	Storage (Persistent Disk) Requirements
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 Enterprise 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 and Configuring Ops Manager on Azure

In this topic

Page last updated:

This topic describes how to install and configure Ops Manager before deploying VMware Enterprise PKS on Azure.

Prerequisites

You use Ops Manager to install and configure Enterprise PKS. Before you install Ops Manager, review Azure Prerequisites and Resource Requirements.

Install and Configure Ops Manager

Multiple versions of Ops Manager are compatible with each version of Enterprise PKS. To confirm that your Ops Manager version is compatible with the version of Enterprise PKS that you want to install, see the Enterprise PKS Release Notes.

To install and configure Ops Manager, follow the instructions in the table below:

Version	Instructions
Ops Manager v2.6	1. Preparing to Deploy Ops Manager on Azure Using Terraform 2. Deploying Ops Manager on Azure Using Terraform 3. Configuring BOSH Director on Azure Using Terraform
Ops Manager v2.7	1. Preparing to Deploy Ops Manager on Azure Using Terraform 2. Deploying Ops Manager on Azure Using Terraform 3. Configuring BOSH Director on Azure Using Terraform

Note: The topics above 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.

Next Installation Step

To create managed identities for Enterprise PKS, follow the instructions in Creating Managed Identities in Azure for Enterprise PKS.

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

Creating Managed Identities in Azure for Enterprise PKS

In this topic

Page last updated:

This topic describes how to create managed identities for VMware Enterprise 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 Enterprise PKS resource group.

You entered your subscription ID into the `terraform.tfvars` file in Step 1: Download and Edit the Terraform Variables File in *Deploying Ops Manager on Azure Using Terraform*.

The name of your Enterprise PKS resource group is exported from Terraform as the output `pcf_resource_group_name`.

To retrieve your subscription ID and the name of your Enterprise PKS resource group, you must have access to the output from when you ran `terraform apply` to create resources for the Enterprise PKS deployment in Create Azure Resources with Terraform in *Deploying Ops Manager to Azure Using Terraform* in the Pivotal documentation. You can view this output 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 Enterprise 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 Enterprise 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 Enterprise PKS resource group by performing the following steps:

   a. Navigate to the Azure Portal and log in.
   b. Open the Enterprise 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 Enterprise PKS resource group:

```
{
  "Name": "PKS worker",
  "IsCustom": true,
  "Description": "Permissions for PKS worker",
  "Actions": [
    "Microsoft.Compute/virtualMachines/read",
    "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 Enterprise PKS resource group.

5. Assign managed identity access to the Enterprise PKS resource group by performing the following steps:

   a. Navigate to the Azure Portal and log in.
   b. Open the Enterprise 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.

Next Installation Step

To install and configure Enterprise PKS, follow the instructions in Installing Enterprise PKS on Azure.

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

Installing Enterprise PKS on Azure

In this topic

Page last updated:

This topic describes how to install and configure VMware Enterprise 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 Enterprise 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 Enterprise PKS

To install Enterprise 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 **Enterprise PKS** in the left column, click the plus sign to add this product to your staging area.

Step 2: Configure Enterprise PKS

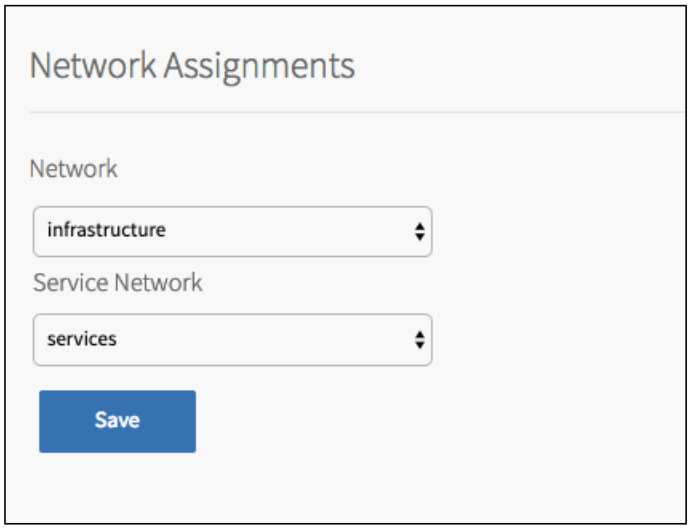
Click the orange **Enterprise PKS** tile to start the configuration process.

warning: When you configure the Enterprise 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 Enterprise PKS fails.

Assign Networks

To configure the networks used by the Enterprise PKS control plane:

1. Click **Assign Networks**.

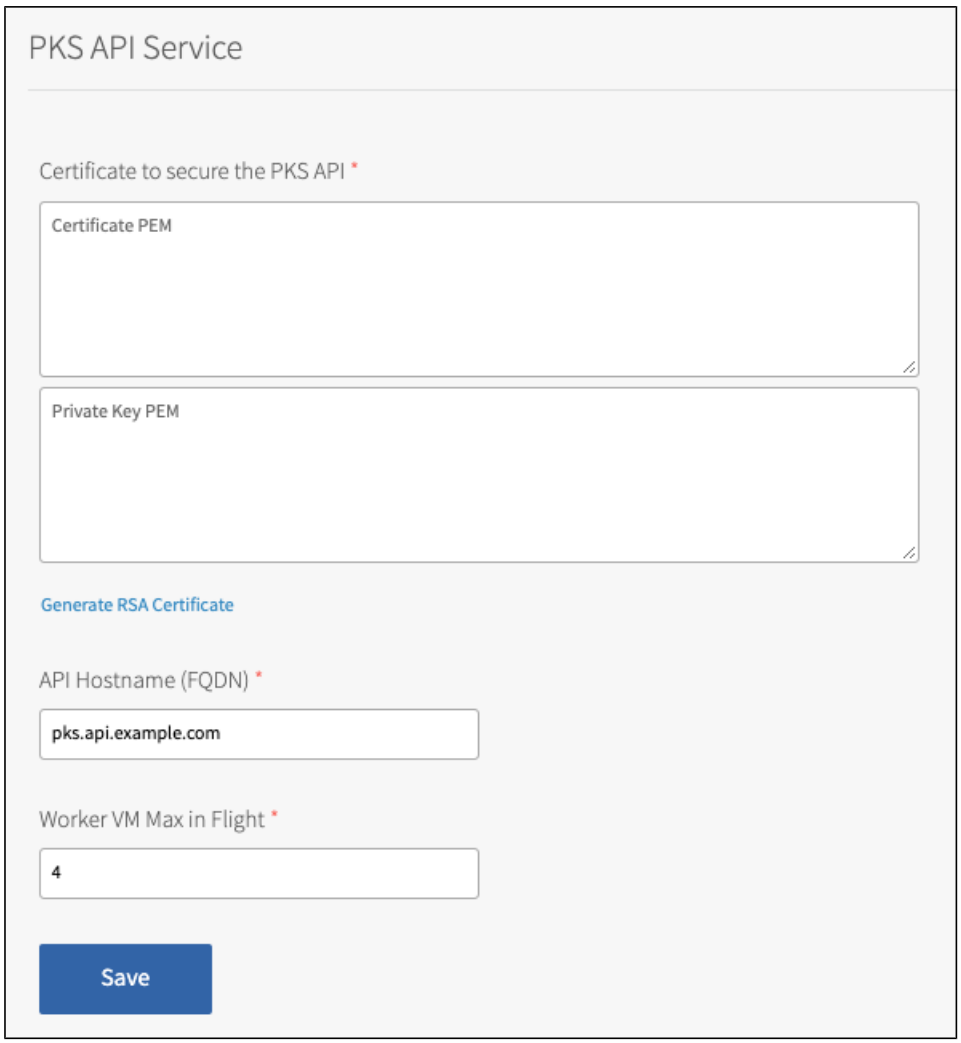


2. Under **Network**, select the infrastructure subnet that you created for the PKS API VM. For example, `infrastructure`.
3. Under **Service Network**, select the services subnet that you created for Kubernetes cluster VMs. For example, `services`.
4. Click **Save**.

PKS API

Perform the following steps:

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

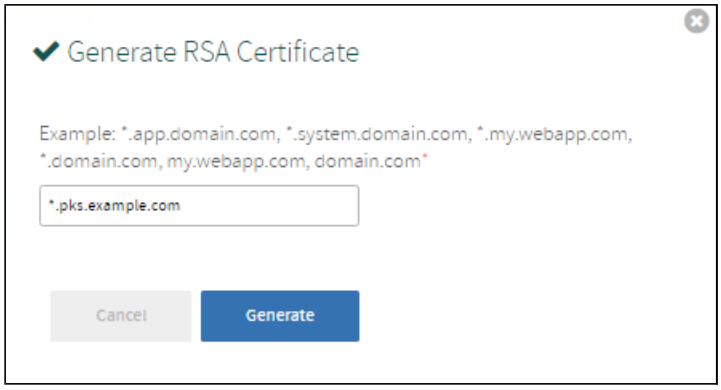


The certificate that you supply should cover the specific subdomain that routes to the PKS API VM with TLS termination on the ingress.

> **warning:** TLS certificates generated for wildcard DNS records only work for a single domain level. For example, a certificate generated for `*.pks.EXAMPLE.com` does not permit communication to `*.api.pks.EXAMPLE.com`. If the certificate does not contain the correct FQDN for the PKS API, calls to the API will fail.

You can enter your own certificate and private key pair, or have Ops Manager generate one for you.
To generate a certificate using Ops Manager:

a. If the **Generate RSA Certificate** option is not available, click **Change**.
b. Click **Generate RSA Certificate**.
c. Enter the domain for your API hostname. This can be a standard FQDN or a wildcard domain.
d. 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 value. When you create or resize a cluster, the `max_in_flight` value limits the number of component instances that can be created or started simultaneously. By default, the `max_in_flight` value is set to `4`, which means that up to four component instances are simultaneously created or started at a time.

5. Click **Save**.

Plans

A plan defines a set of resource types used for deploying a cluster.

Activate a Plan

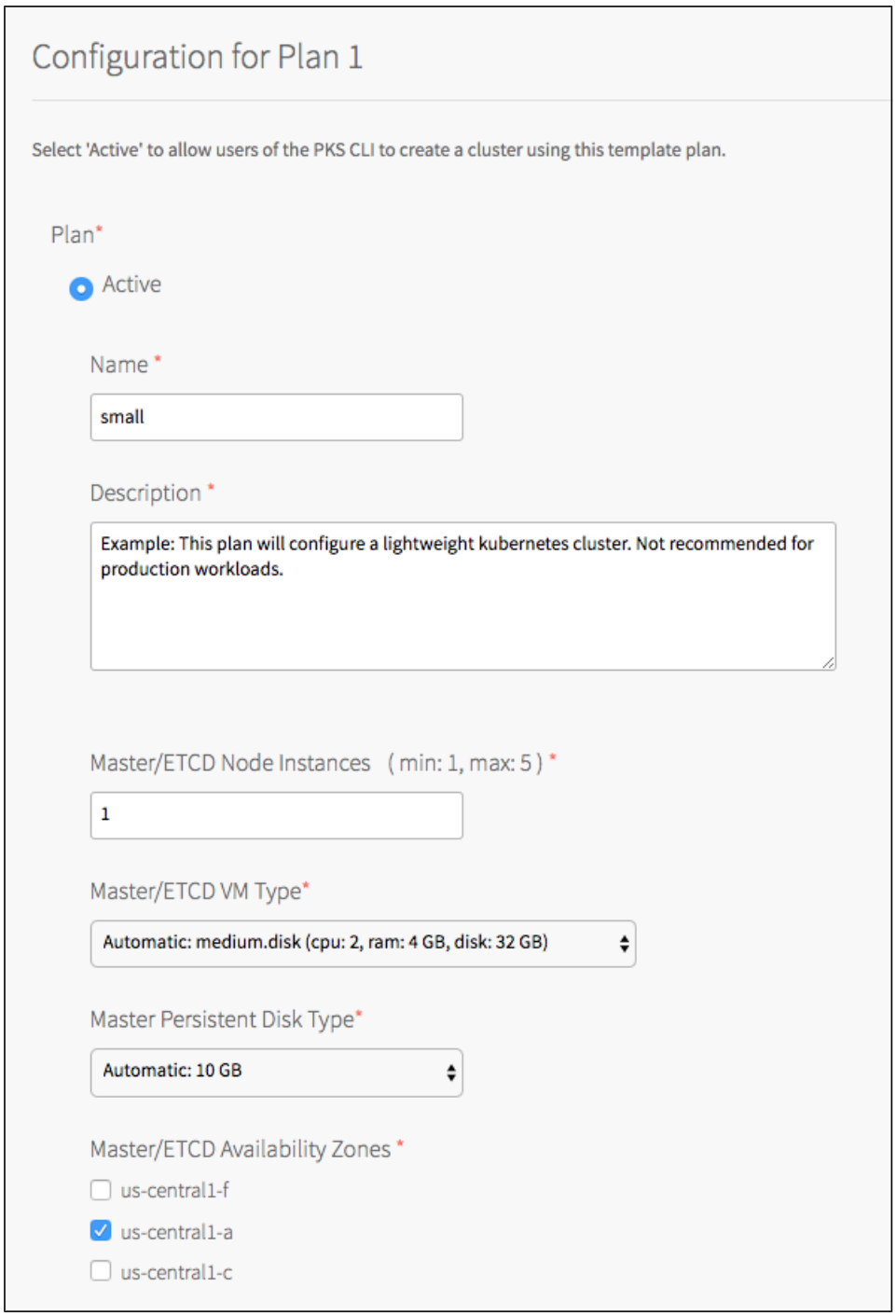
You must first activate and configure **Plan 1**, and afterwards you can optionally activate **Plan 2** through **Plan 10**.

To activate and configure a plan, perform the following steps:

1. Click the plan that you want to activate.

   > **Note**: Plans 11, 12 and 13 support only Windows worker-based Kubernetes clusters, on vSphere with Flannel.

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 the 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 `1`, `3`, or `5`.

> **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.

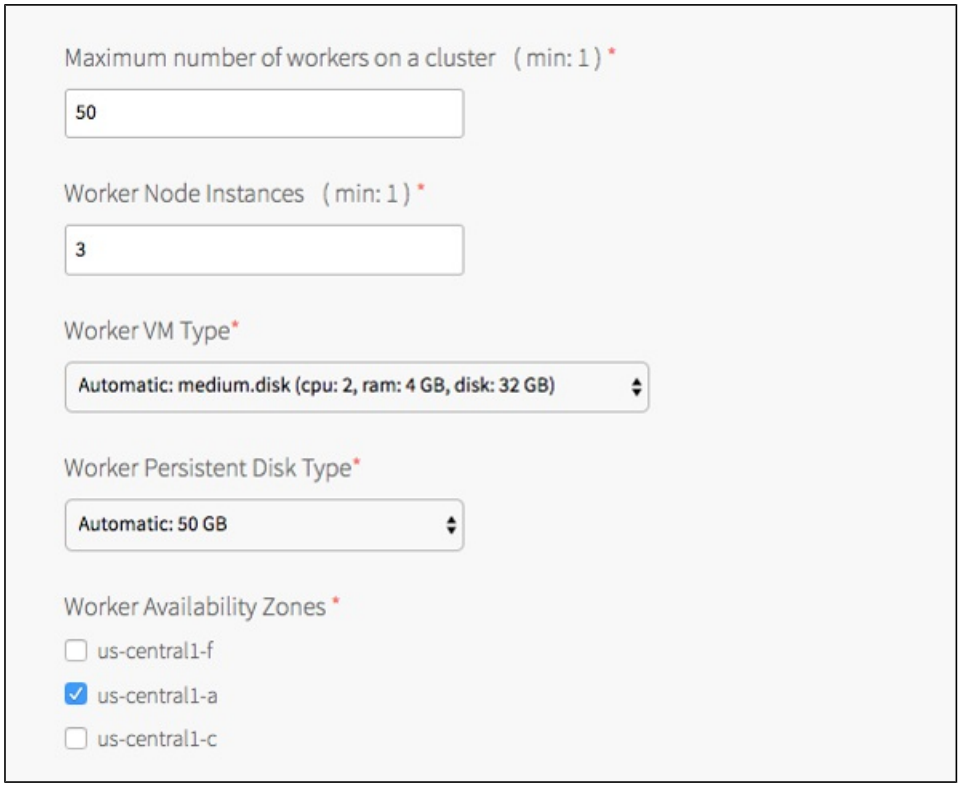
warning: To change the number of master/etcd nodes for a plan, you must ensure that no existing clusters use the plan. Enterprise PKS does not support changing the number of master/etcd nodes for plans with existing clusters.

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 Enterprise 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 Enterprise PKS. If you select more than one AZ, Enterprise PKS deploys the master VM in the first AZ and the worker VMs across the remaining AZs. If you are using multiple masters, Enterprise PKS deploys the master and worker VMs across the AZs in round-robin fashion.

9. Under **Maximum number of workers on a cluster**, set the maximum number of Kubernetes worker node VMs that Enterprise 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.

For more information about creating clusters, see Creating Clusters.

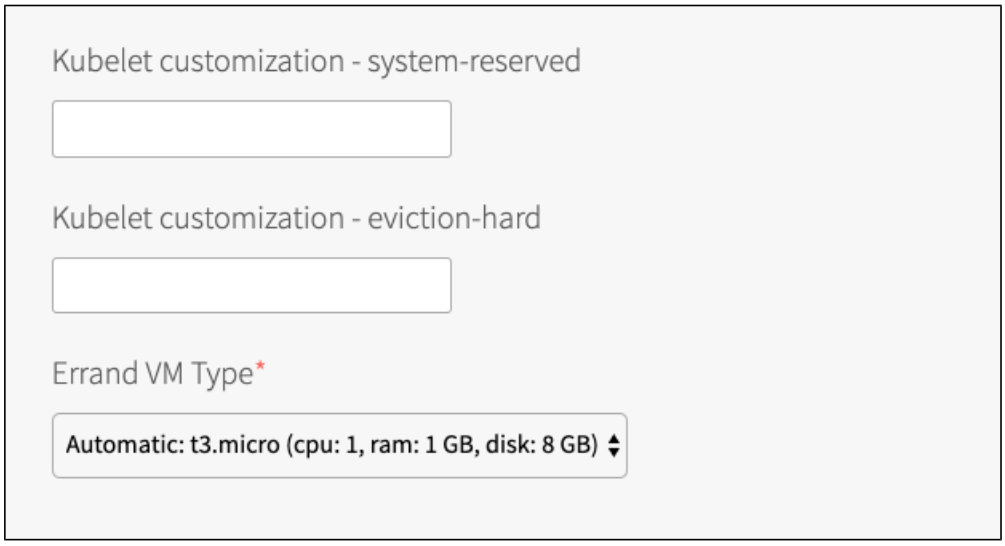
> **Note**: Changing a plan's **Worker Node Instances** setting does not alter the number of worker nodes on existing clusters. For information about scaling an existing cluster, see Scale Horizontally by Changing the Number of Worker Nodes Using the PKS CLI in *Scaling Existing Clusters*.

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 Enterprise PKS Clusters*.

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. Enterprise 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.

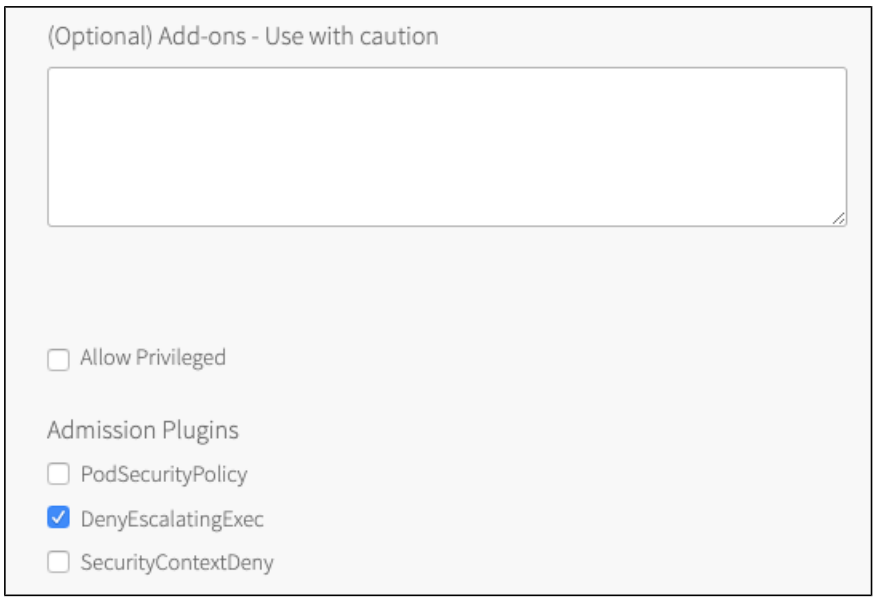


15. Under **Kubelet customization - eviction-hard**, enter threshold limits that Kubelet can use to evict pods when they exceed the limit. Enter limits in the format `EVICTION-SIGNAL=QUANTITY`. For example, `memory.available=100Mi, nodefs.available=10%, nodefs.inodesFree=5%`. For more information about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



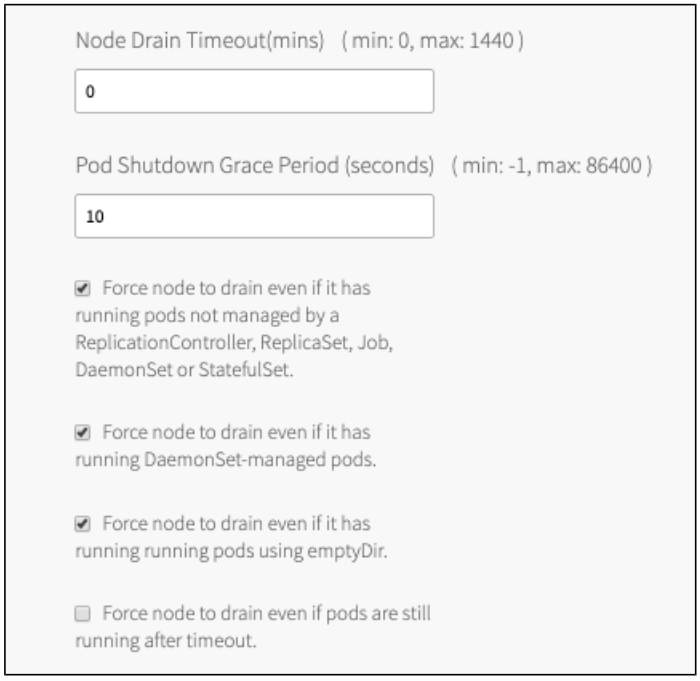
Linux Workloads.

18. (Optional) To allow users to create pods with privileged containers, select the **Allow Privileged** option. For more information, see Pods in the Kubernetes documentation.

> **Note:** Enabling the `Allow Privileged` option means that all containers in the cluster will run in privileged mode. Pod Security Policy provides a privileged parameter that can be used to enable or disable Pods running in privileged mode. As a best practice, if you enable `Allow Privileged`, define PSP to limit which Pods run in privileged mode. If you are implementing PSP for privileged pods, you must enable `Allow Privileged` mode.

19. (Optional) Enable or disable one or more admission controller plugins: **PodSecurityPolicy**, **DenyEscalatingExec**, and **SecurityContextDeny**. See Admission Plugins for more information.

20. (Optional) Under **Node Drain Timeout(mins)**, enter the timeout in minutes for the node to drain pods. If you set this value to



`0`, the node drain does not terminate.

21. (Optional) Under **Pod Shutdown Grace Period (seconds)**, enter a timeout in seconds for the node to wait before it forces the pod to terminate. If you set this value to `-1`, the default timeout is set to the one specified by the pod.

22. (Optional) To configure when the node drains, enable the following:
    - **Force node to drain even if it has running pods not managed by a ReplicationController, ReplicaSet, Job, DaemonSet or StatefulSet.**
    - **Force node to drain even if it has running DaemonSet-managed pods.**
    - **Force node to drain even if it has running running pods using emptyDir.**
    - **Force node to drain even if pods are still running after timeout.**

    > **warning:** If you select **Force node to drain even if pods are still running after timeout**, the node kills all running workloads on pods. Before enabling this configuration, set **Node Drain Timeout** to a value greater than `0`.

    For more information about configuring default node drain behavior, see Worker Node Hangs Indefinitely in *Troubleshooting*.

23. Click **Save**.

Deactivate a Plan

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**.

Choose your IaaS*

- GCP
- vSphere
- AWS
- Azure

Azure Cloud Name*

Azure Public Cloud

Subscription ID *

Tenant ID *

Location *

Resource Group *

Virtual Network *

Virtual Network Resource Group *

Default Security Group *

Primary Availability Set *

Master Managed Identity *

Worker Managed Identity *

Save

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.

    > **Note:** Enterprise PKS automatically assigns the default security group to each VM when you create a Kubernetes cluster. However, on Azure this automatic assignment may not occur. For more information, see Azure Default Security Group Is Not Automatically Assigned to Cluster VMs in *Enterprise PKS Release Notes*.

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. For more information, see Download Templates and Edit Variables File in *Deploying Ops Manager to Azure Using Terraform* in the Pivotal documentation. 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 Enterprise PKS*.

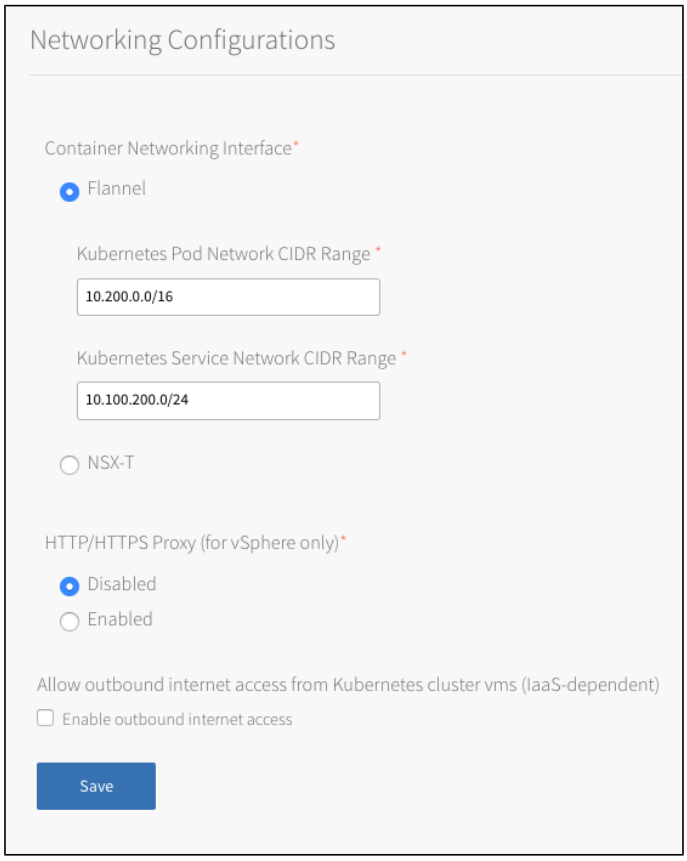
13. For **Worker Managed Identity**, enter `pks-worker`. You created the managed identity for the worker nodes in Create the Worker Nodes Managed Identity in *Creating Managed Identities in Azure for Enterprise PKS*.

14. Click **Save**.

Networking

To configure networking, do the following:

1. Click **Networking**.

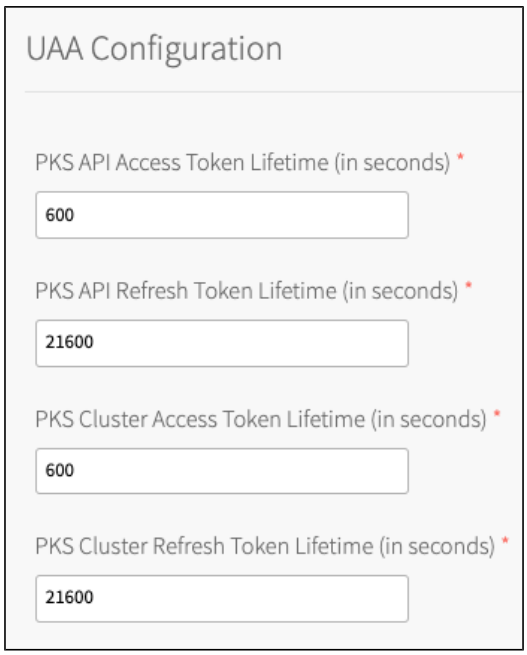


2. Under **Container Networking Interface**, select **Flannel**.
3. (Optional) Enter values for **Kubernetes Pod Network CIDR Range** and **Kubernetes Service Network CIDR Range**.
   - Ensure that the CIDR ranges do not overlap and have sufficient space for your deployed services.
   - Ensure that the 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. This field defaults to `600`.



3. Under **PKS API Refresh Token Lifetime**, enter a time in seconds for the PKS API refresh token lifetime. This field defaults to `21600`.

4. Under **PKS Cluster Access Token Lifetime**, enter a time in seconds for the cluster access token lifetime. This field defaults to `600`.

5. Under **PKS Cluster Refresh Token Lifetime**, enter a time in seconds for the cluster refresh token lifetime. This field defaults to `21600`.

> **Note**: Pivotal recommends using the default UAA token timeout values. By default, access tokens expire after ten minutes and refresh tokens expire after six hours. If you want to customize your token timeout values, see Token Management in *UAA Overview*.

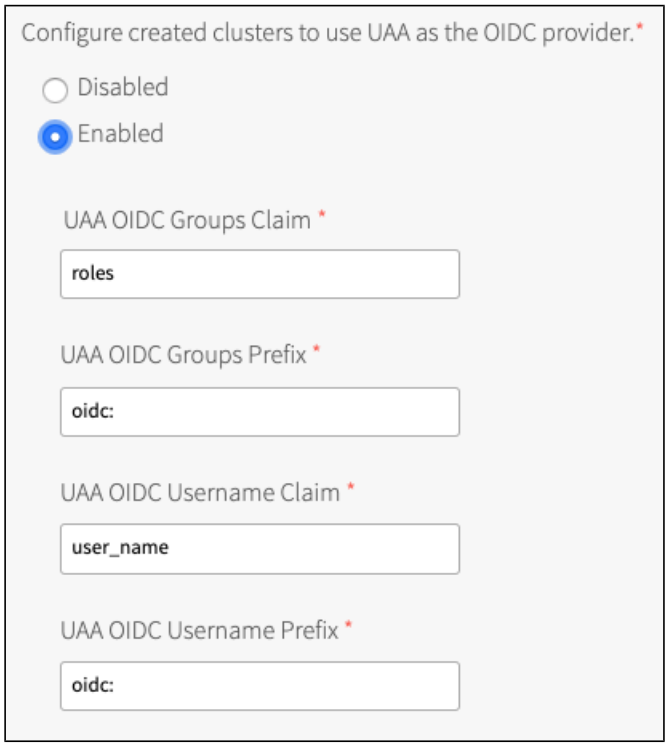
6. Under **Configure created clusters to use UAA as the OIDC provider**, select **Enabled** or **Disabled**. If you click **Enabled**, Kubernetes verifies end-user identities based on authentication executed by UAA. For more information, see the table below.

Option	Description
Disabled	If you do not enable UAA as the OpenID Connect (OIDC) provider, Kubernetes authenticates users against its internal user management system.
Enabled	If you enable UAA as the OIDC provider, Kubernetes authenticates users as follows: ○ If you select **Internal UAA** in the next step, Kubernetes authenticates users against the internal UAA authentication mechanism. ○ If you select **LDAP Server** in the next step, Kubernetes authenticates users against the LDAP server. ○ If you select **SAML Identity Provider** in the next step, Kubernetes authenticates users against the SAML identity provider.

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

To configure Enterprise PKS to use UAA as the OIDC provider, do the following:

a. Under **Configure created clusters to use UAA as the OIDC provider**, select **Enabled**.



b. For **UAA OIDC Groups Claim**, enter the name of your groups claim. This is used to set a user's group in the JSON Web Token (JWT) claim. The default value is `roles`.
c. For **UAA OIDC Groups Prefix**, enter a prefix for your groups claim. This prevents conflicts with existing names. For example, if you enter the prefix `oidc:`, UAA creates a group name like `oidc:developers`. If you are configuring a new Enterprise PKS installation, the default is `oidc:`. If you are upgrading to Enterprise PKS v1.5, the default is `-`.
d. For **UAA OIDC Username Claim**, enter the name of your username claim. This is used to set a user's username in the JWT claim. The default value is `user_name`. Depending on your provider, admins can enter claims besides `user_name`, like `email` or `name`.
e. For **UAA OIDC Username Prefix**, enter a prefix for your username claim. This prevents conflicts with existing names. For example, if you enter the prefix `oidc:`, UAA creates a username like `oidc:admin`. If you are configuring a new Enterprise PKS installation, the default is `oidc:`. If you are upgrading to Enterprise PKS v1.5, the default is `-`.

> **Note:** Pivotal recommends adding OIDC prefixes to prevent OIDC users and groups from gaining unintended cluster privileges. When you upgrade to Enterprise PKS v1.5, if you do not change the values for **UAA OIDC Groups Prefix** or **UAA OIDC Username Prefix**, Enterprise PKS does not add prefixes.

> **warning:** If you change the above values for a pre-existing Enterprise PKS installation, you must change any existing role bindings that bind to a username or group. If you do not change your role bindings, developers cannot access Kubernetes clusters. For instructions about creating role bindings, see Managing Cluster Access and Permissions.

7. Select one of the following options:

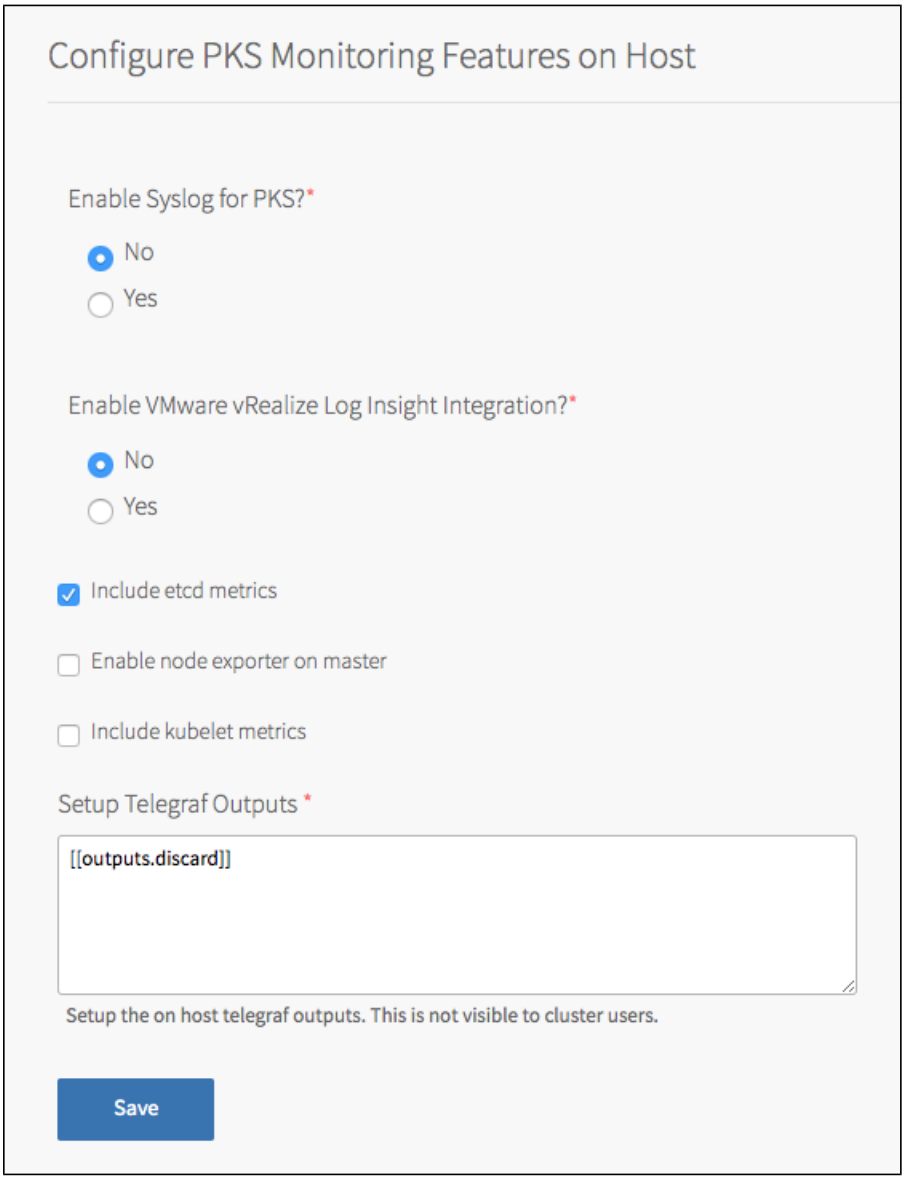
- To use an internal user account store for UAA, select **Internal UAA**. Click **Save** and continue to (Optional) Host Monitoring.
- To use LDAP for UAA, select **LDAP Server** and continue to Connecting Enterprise PKS to an LDAP Server.
- To use SAML for UAA, select **SAML Identity Provider** and continue to Connecting Enterprise PKS to a SAML Identity Provider.

(Optional) Host Monitoring

In **Host Monitoring**, you can configure one or more of the following:

- To configure Syslog for BOSH-deployed VMs, see Syslog. Syslog forwards log messages from all BOSH-deployed VMs to a syslog endpoint.
- To configure the Telegraf agent, see Telegraf. The Telegraf agent sends metrics from master node and worker node VMs to a monitoring service, such as Wavefront or Datadog.

For more information about these components, see Monitoring PKS and PKS-Provisioned Clusters.



Syslog

To configure Syslog for all BOSH-deployed VMs in Enterprise PKS:

1. Click **Host Monitoring**.
2. Under **Enable Syslog for PKS**, select **Yes**.
3. Under **Address**, enter the destination syslog endpoint.

4. Under **Port**, enter the destination syslog port.

5. Under **Transport Protocol**, select a transport protocol for log forwarding.

6. (Optional) To enable TLS encryption during log forwarding, complete the following steps:

   a. Ensure **Enable TLS** is selected.

   > **Note**: Logs may contain sensitive information, such as cloud provider credentials. Pivotal strongly recommends that you enable TLS encryption for log forwarding.

   b. Under **Permitted Peer**, provide the accepted fingerprint (SHA1) or name of remote peer. For example, `*.YOUR-LOGGING-SYSTEM.com`.

   c. 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. (Optional) Under **Max Message Size**, enter a maximum message size for logs that are forwarded to a syslog endpoint. By default, the **Max Message Size** field is 10,000 characters.

8. Click **Save**.

Telegraf

To configure Enterprise PKS to use Telegraf for metric collection:

1. Create a configuration file for your monitoring service. For instructions, see Create a Configuration File.

2. Select one or more of the following checkboxes:

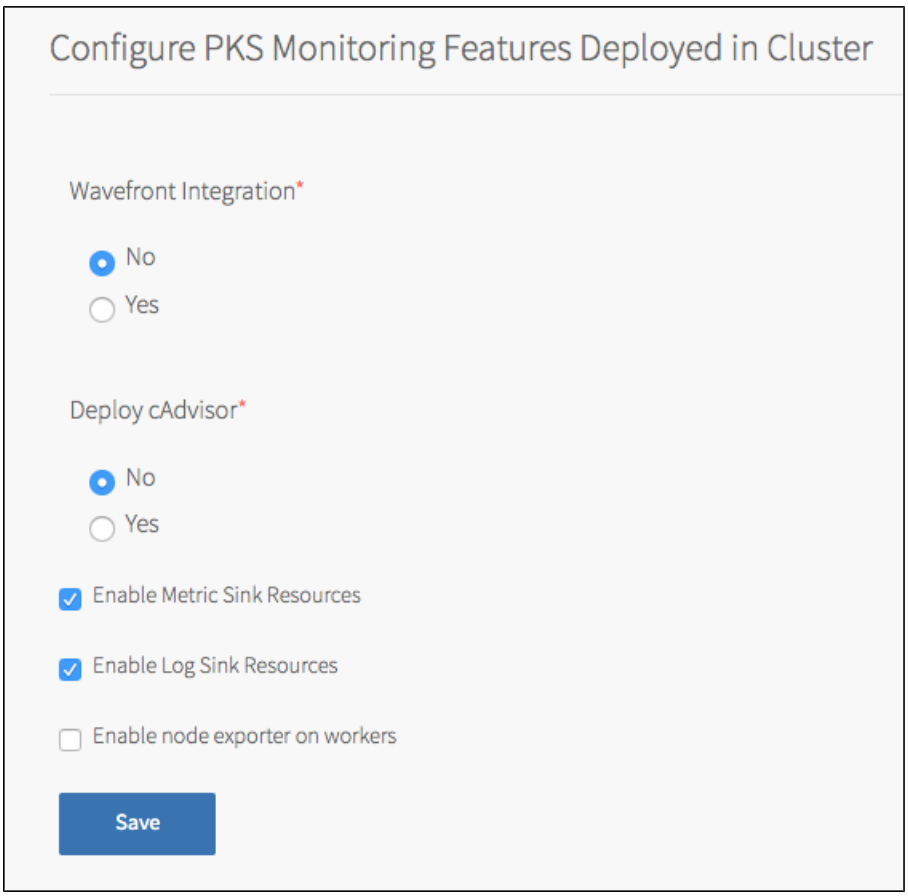
   - **Include etcd metrics**: This includes etcd server and debug metrics.
   - **Enable node exporter on master**: This enables Node Exporter on the *localhost* of each master node VM.
   - **Include kubelet metrics**: This includes all workload metrics across your Kubernetes clusters. Enabling **Include kubelet metrics** generates a high volume of metrics.

3. In **Setup Telegraf Outputs**, replace the default value `[[outputs.discard]]` with the contents of the configuration file that you created above.

4. Click **Save**.

(Optional) In-Cluster Monitoring

In **In-Cluster Monitoring**, you can configure one or more observability components and integrations that run in Kubernetes clusters and capture logs and metrics about your workloads. For more information, see Monitoring Workers and Workloads.



To configure in-cluster monitoring:

- To configure Wavefront, see Wavefront.
- To configure cAdvisor, see cAdvisor.
- To configure sink resources, see:
  - Metric Sink Resources
  - Log Sink Resources

  You can enable both log and metric sink resources or only one of them.

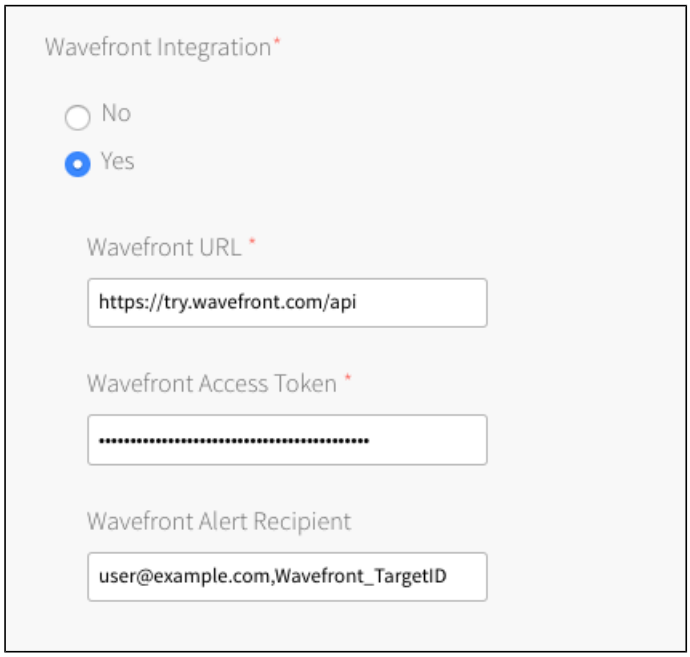
Wavefront

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 the Wavefront documentation.

To enable and configure Wavefront monitoring:

1. In the Enterprise PKS tile, select **In-Cluster Monitoring**.



2. 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**, using the following syntax:

```
USER-EMAIL,WAVEFRONT-TARGETID_001,WAVEFRONT-TARGETID_002
```

   Where:

   - `USER-EMAIL` is the alert recipient's email address.
   - `WAVEFRONT-TARGETID_001` and `WAVEFRONT-TARGETID_002` are your comma-delimited Wavefront Target IDs.

   For example:

```
randomuser@example.com,51n6psdj933ozdjf
```

6. Click **Save**.

To create alerts, you must enable errands in Enterprise PKS.

1. In the Enterprise PKS tile, select **Errands**.
2. On the **Errands** pane, enable **Create pre-defined Wavefront alerts errand**.
3. Enable **Delete pre-defined Wavefront alerts errand**.
4. Click **Save**. Your settings apply to any clusters created after you have saved these configuration settings and clicked **Apply Changes**.

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

cAdvisor

cAdvisor is an open source tool for monitoring, analyzing, and exposing Kubernetes container resource usage and performance statistics.

To deploy a cAdvisor container:

1. Select **In-Cluster Monitoring**.
2. Under **Deploy cAdvisor**, select **Yes**.
3. Click **Save**.

> **Note:** For information about configuring cAdvisor to monitor your running Kubernetes containers, see cAdvisor in the cAdvisor GitHub repository. For general information about Kubernetes cluster monitoring, see Tools for Monitoring Resources in the Kubernetes documentation.

Metric Sink Resources

You can configure PKS-provisioned clusters to send Kubernetes node metrics and pod metrics to metric sinks. For more information about metric sink resources and what to do after you enable them in the tile, see Sink Resources in *Monitoring Workers and Workloads*.

To enable clusters to send Kubernetes node metrics and pod metrics to metric sinks:

1. In **In-Cluster Monitoring**, select **Enable Metric Sink Resources**. If you enable this checkbox, Enterprise PKS deploys Telegraf as a `DaemonSet`, a pod that runs on each worker node in all your Kubernetes clusters.
2. (Optional) To enable Node Exporter to send worker node metrics to metric sinks of kind `ClusterMetricSink`, select **Enable node exporter on workers**. If you enable this checkbox, Enterprise PKS deploys Node Exporter as a `DaemonSet`, a pod that runs on each worker node in all your Kubernetes clusters.
   For instructions on how to create a metric sink of kind `ClusterMetricSink` for Node Exporter metrics, see Create a ClusterMetricSink Resource for Node Exporter Metrics in *Creating and Managing Sink Resources*.
3. Click **Save**.

Log Sink Resources

You can configure PKS-provisioned clusters to send Kubernetes API events and pod logs to log sinks. For more information about log sink resources and what to do after you enable them in the tile, see Sink Resources in *Monitoring Workers and Workloads*.

To enable clusters to send Kubernetes API events and pod logs to log sinks:

1. Select **Enable Log Sink Resources**. If you enable this checkbox, Enterprise PKS deploys Fluent Bit as a `DaemonSet`, a pod that runs on each worker node in all your Kubernetes clusters.
2. Click **Save**.

Tanzu Mission Control (Experimental)

Participants in the VMware Tanzu Mission Control beta program can use the **Tanzu Mission Control (Experimental)** pane of the Enterprise PKS tile to integrate their Enterprise PKS deployment with Tanzu Mission Control.

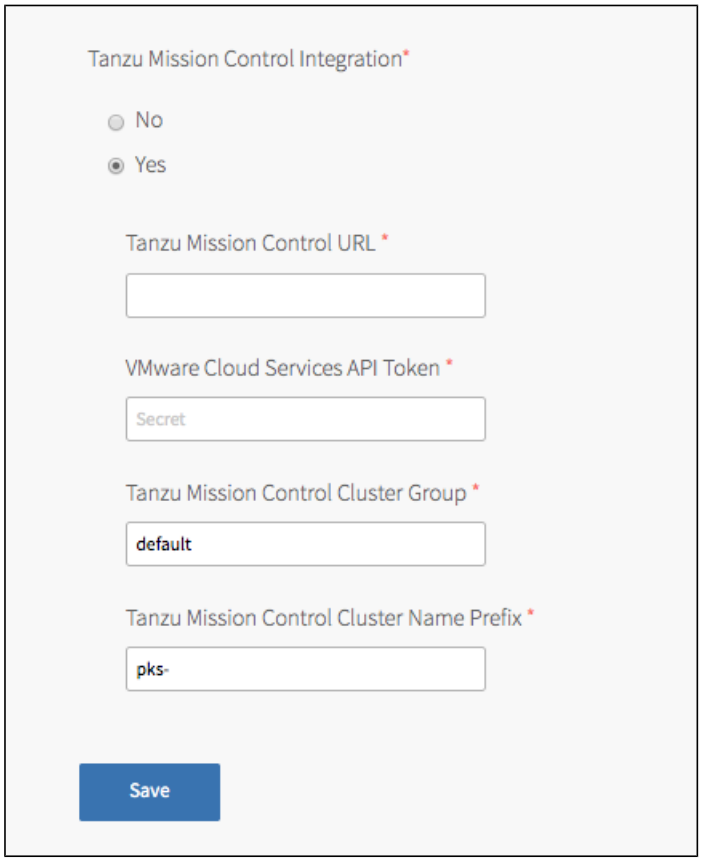
Tanzu Mission Control integration lets you monitor and manage Enterprise PKS clusters from the Tanzu Mission Control console, which makes the Tanzu Mission Control console a single point of control for all Kubernetes clusters.

> **warning:** VMware Tanzu Mission Control is currently experimental beta software and is intended for evaluation and test purposes only. For more information about Tanzu Mission Control, see the VMware Tanzu Mission Control home page.

To integrate Enterprise PKS with Tanzu Mission Control:

1. Confirm that the PKS Control Plane VM has internet access and can connect to `*.tmc.cloud.vmware.com` and the other outbound URLs listed in the What Happens When You Attach a Cluster section of the Tanzu Mission Control documentation.

2. Navigate to the **Enterprise PKS** tile > the **Tanzu Mission Control (Experimental)** pane and select **Yes** under **Tanzu Mission Control Integration**.



3. Configure the fields below:

   - **Tanzu Mission Control URL**: Enter the Org URL of your Tanzu Mission Control subscription, without a trailing slash `/`). For example, `YOUR-ORG.tmc.cloud.vmware.com`.
   - **VMware Cloud Services API token**: Enter your API token to authenticate with VMware Cloud Services APIs. You can retrieve this token by logging in to VMware Cloud Services and viewing your account information.
   - **Tanzu Mission Control Cluster Group**: Enter the name of a Tanzu Mission Control cluster group.

     The name can be `default` or another value, depending on your role and access policy:

- `Org Member` users in VMware cloud services have a `service.admin` role in Tanzu Mission Control. These users:
  - By default, can create and attach clusters only in the `default` cluster group.
  - Can create and attach clusters to other cluster groups after an `organization.admin` user grants them the `clustergroup.admin` or `clustergroup.edit` role for those groups.
- `Org Owner` users in VMware cloud services have `organization.admin` permissions in Tanzu Mission Control. These users:
  - Can create cluster groups.
  - Can grant `clustergroup` roles to `service.admin` users through the Tanzu Mission Control Access Policy view.

For more information about role and access policy, see Access Control in the *VMware Tanzu Mission Control Product Documentation*.

- **Tanzu Mission Control Cluster Name Prefix**: Enter a name prefix for identifying the Enterprise PKS clusters in Tanzu Mission Control.

4. Click **Save**.

warning: After the Enterprise PKS tile is deployed with a configured cluster group, the cluster group cannot be updated.

Note: When you upgrade your Kubernetes clusters and have Tanzu Mission Control integration enabled, existing clusters will be attached to Tanzu Mission Control.

CEIP and Telemetry

To configure VMware's Customer Experience Improvement Program (CEIP) and the Pivotal Telemetry Program (Telemetry), do the following:

1. Click **CEIP and Telemetry**.

2. Review the information about the CEIP and Telemetry.

CEIP and Telemetry

About the CEIP and Telemetry Program

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.

Customers who participate (at the enhanced tier) are eligible for several benefits, including Proactive Support.

Additional information regarding the data collected through CEIP or Telemetry, and the purposes for which it is used by VMware is set forth in the Trust & Assurance Center and for Pivotal on the Pivotal Telemetry pages. If you prefer not to participate in CEIP and Telemetry for PKS, you should not join below. You may join or leave CEIP and Telemetry for PKS at any time.

Levels of Participation

No personally identifiable information (PII) is collected at either level of participation. Please refer to the data dictionary for more information on the data we collect.

Standard: This participation tier is anonymous. Your data will be used to improve PKS, but is not identifiable to your organization

Enhanced: This participation tier allows us to identify your organization so we may provide proactive support and other benefits. See sample reports for more details.

Please Note

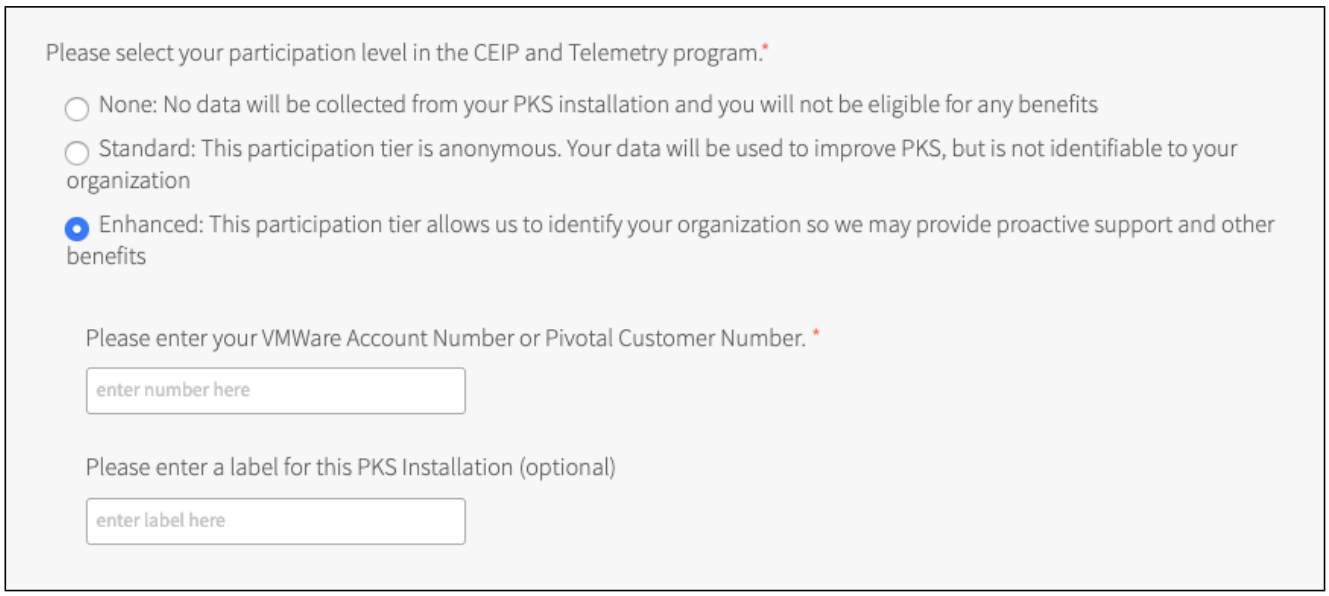
- If you are opting in on behalf of an organization (and not for you as an individual), you represent and warrant that you have legal authority to bind that organization, and you hereby join CEIP/Telemetry on behalf of your organization.
- If you are opting in to the enhanced tier, in the event a term or condition of CEIP or the Telemetry program conflicts with a term or condition of a previously executed license procurement agreement between you and Licensor (Pivotal or VMware), the CEIP or Telemetry program terms supersede solely for purposes of CEIP and Telemetry
- If you are running PKS on a private network, you will need to enable outgoing internet access by opening your firewall to allow traffic to `https://vcsa.vmware.com/ph` on port `443`

Resources

- Data Dictionary
- Participation Benefits
- Sample Reports
- Trust and Assurance Center
- Pivotal Telemetry

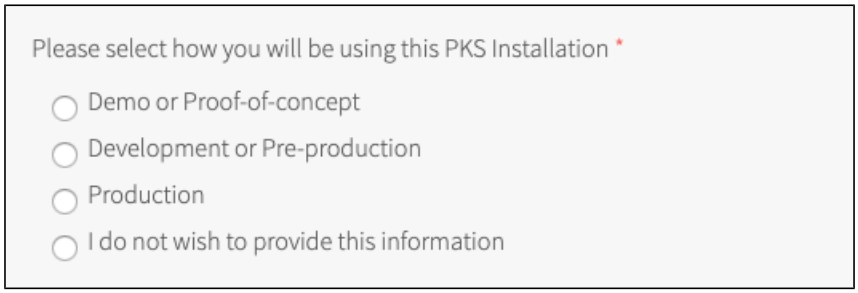
View a larger version of this image.

3. To specify your level of participation in the CEIP and Telemetry program, select one of the **Participation Level** options:
   - **None:** If you select this option, data is not collected from your Enterprise PKS installation.
   - (Default) **Standard:** If you select this option, data is collected from your Enterprise PKS installation to improve Enterprise PKS. This participation level is anonymous and does not permit the CEIP and Telemetry to identify your organization.
   - **Enhanced:** If you select this option, data is collected from your Enterprise PKS installation to provide you proactive support and other benefits. This participation level permits the CEIP and Telemetry to identify your organization.



For more information about the CEIP and Telemetry participation levels, see Participation Levels in *Telemetry*.

4. If you selected the **Enhanced** participation level, complete the following:
   - Enter your VMware account number or Pivotal customer number in the **VMware Account Number or Pivotal Customer Number** field. If you are a VMware customer, you can find your VMware Account Number in your **Account Summary** on my.vmware.com. If you are a Pivotal customer, you can find your Pivotal Customer Number in your Pivotal Order Confirmation email.
   - (Optional) Enter a descriptive name for your PKS installation in the **PKS Installation Label** field. The label you assign to this installation will be used in telemetry reports to identify the environment.
5. To provide information about the purpose for this installation, select an option in the **PKS Installation Type** list.



6. Click **Save**.

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

Note: Even if you select **None**, Enterprise PKS-provisioned clusters send usage data to the PKS control plane. However, this data is not sent to VMware or Pivotal and remains on your Enterprise PKS installation.

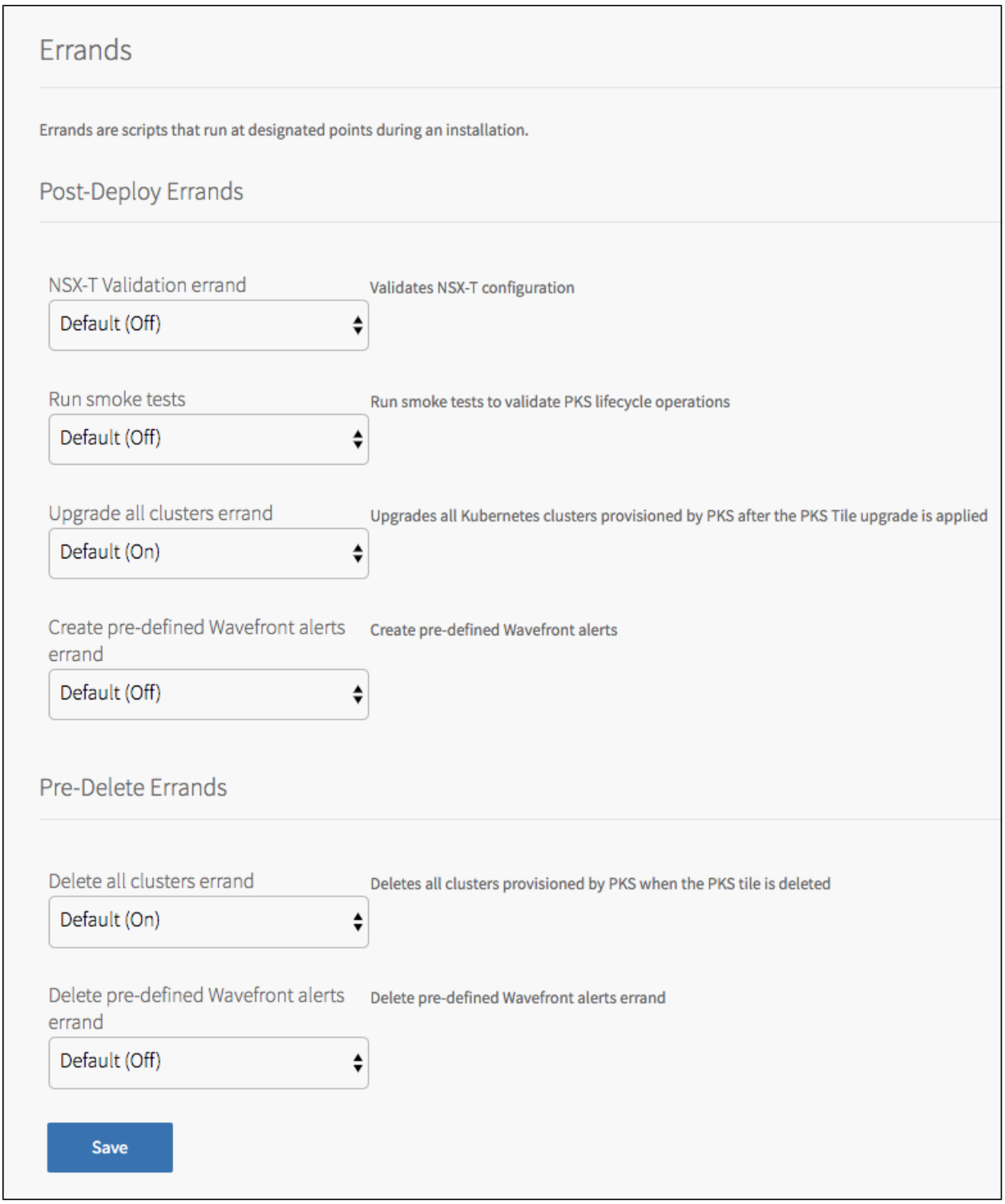
Errands

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

To configure when post-deploy and pre-delete errands for Enterprise 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 CLI to create a Kubernetes cluster and then delete it. If the creation or deletion fails, the errand fails and the installation of the Enterprise 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.

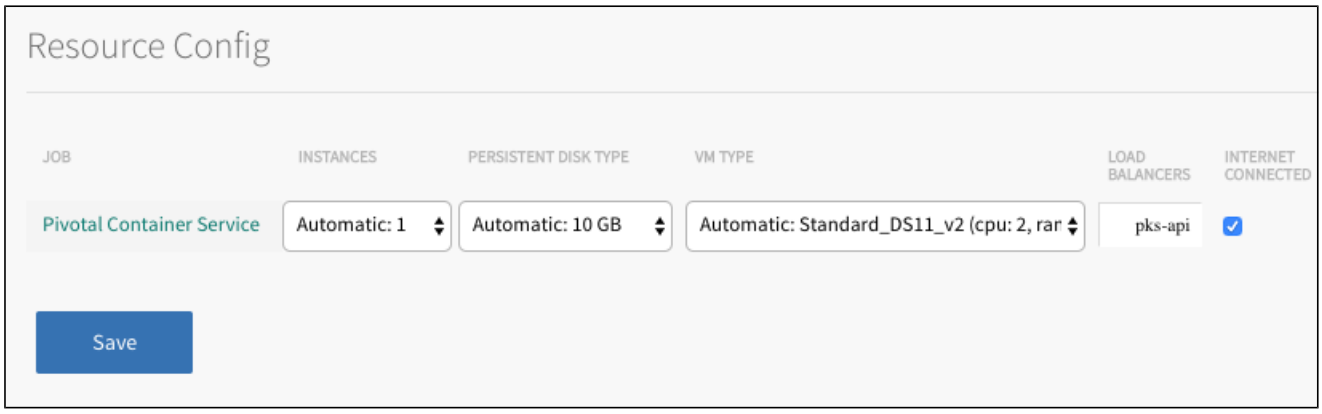
warning: If **Upgrade all clusters errand** is enabled, updating the Enterprise PKS tile with a new Linux stemcell triggers the

rolling of every Linux VM in each Kubernetes cluster. Similarly, updating the Enterprise PKS tile with a new Windows stemcell triggers the rolling of every Windows VM in your Kubernetes clusters. This automatic rolling ensures that all your VMs are patched. To avoid workload downtime, use the resource configuration recommended in What Happens During Enterprise PKS Upgrades and Maintaining Workload Uptime.

Resource Config

To modify the resource usage of Enterprise 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 that you configured in your `terraform.tfstate` file during Step 1: Download Templates and Edit Variables File in *Deploying Ops Manager on Azure Using Terraform*. 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.

3. (Optional) Edit other resources used by the **Pivotal Container Service** job. The **Pivotal Container Service** job requires a VM with the following minimum resources:

CPU	Memory	Disk
2	8 GB	29 GB

   Note: The automatic **VM Type** value matches the minimum recommended size for the **Pivotal Container Service** job. If you experience timeouts or slowness when interacting with the PKS API, select a **VM Type** with greater CPU and memory resources.

4. (Optional) If you do not use a NAT instance, select **Internet Connected** to allow component instances direct access to the internet.

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 need to retrieve the PKS API endpoint to allow your organization to use the API to create, update, and delete Kubernetes clusters.

To retrieve the PKS API endpoint, do the following:

1. Navigate to the Ops Manager **Installation Dashboard**.

2. Click the **Enterprise PKS** 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 CLI and the Kubernetes CLI help you interact with your Enterprise 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 Authentication for Enterprise PKS

Follow the procedures in Setting Up Enterprise PKS Admin Users on Azure.

Next Steps

After installing Enterprise PKS on Azure, you may want to do one or more of the following:

- Create a load balancer for your Enterprise PKS clusters. For more information, see Creating and Configuring an Azure Load Balancer for Enterprise PKS Clusters.
- Create your first Enterprise 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

In this topic

Page last updated:

This topic describes how to create a load balancer for the VMware Enterprise 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 Enterprise PKS using Terraform, 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 **Enterprise 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 Enterprise PKS installation by returning to the Install the PKS and Kubernetes CLIs step of *Installing Enterprise PKS on Azure*.

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

Setting Up Enterprise PKS Admin Users on Azure

In this topic

Page last updated:

This topic describes how to create admin users in VMware Enterprise PKS with User Account and Authentication (UAA). Creating at least one admin user is a necessary step during the initial set up of Enterprise PKS.

Overview

UAA is the identity management service for Enterprise PKS. Enterprise PKS includes a UAA server, which is hosted on the PKS control plane VM.

To interact with the UAA server, you can use the UAA Command Line Interface (UAAC). You can either run UAAC commands from the Ops Manager VM or install UAAC on your local workstation.

Prerequisites

Before setting up admin users for Enterprise PKS, you must have one of the following:

- SSH access to the Ops Manager VM
- A machine that can connect to your PKS control plane VM

Step 1: Connect to the PKS Control Plane VM

You can connect to the PKS control plane VM from the Ops Manager VM or from a different machine such as your local workstation.

Option 1: Connect through the Ops Manager VM

You can connect to the PKS control plane VM by logging in to the Ops Manager VM through SSH.

To log in to the Ops Manager VM using SSH on 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 SSH into the Ops Manager VM on 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.

Option 2: Connect through a Non-Ops Manager Machine

To connect to the PKS control plane VM and run UAA commands, do the following:

1. Install UAAC on your machine. For example:

```
gem install cf-uaac
```

2. Download a copy of your Ops Manager root CA certificate to the machine. To download the certificate, do the following:

   a. In a web browser, navigate to the FQDN of Ops Manager and log in.
   b. In Ops Manager, navigate to **Settings** in the drop-down menu under your username.
   c. Click **Advanced Options**.
   d. On the **Advanced Options** configuration page, click **Download Root CA Cert**.
   e. Move the certificate to a secure location on your machine and record the path.

3. Proceed to the Log In as a UAA Admin section to create admin users with UAAC.

Step 2: Log In as a UAA Admin

Before creating PKS users, you must log in to the UAA server as a UAA admin. To log in to the UAA server, do the following:

1. Retrieve the UAA management admin client secret:

   a. In a web browser, navigate to the Ops Manager **Installation Dashboard** and click the **Enterprise PKS** tile.
   b. Click the **Credentials** tab.
   c. Click **Link to Credential** next to **Pks Uaa Management Admin Client** and copy the value of `secret`.

2. Target your UAA server by running the following command:

```
uaac target https://PKS-API:8443 --ca-cert CERTIFICATE-PATH
```

Where:

- `PKS-API` is the domain name of your PKS API server. You entered this domain name in the **Enterprise PKS** tile > **PKS API** > **API Hostname (FQDN)**.
- `CERTIFICATE-PATH` is the path to your Ops Manager root CA certificate. Provide this certificate to validate the PKS API certificate with SSL.
  - If you are logged in to the Ops Manager VM, specify `/var/tempest/workspaces/default/root_ca_certificate` as the path. This is the default location of the root certificate on the Ops Manager VM.
  - If you downloaded the Ops Manager root CA certificate to your machine, specify the path where you stored the 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 ignore it because it has no impact.

3. Authenticate with UAA by running the following command:

```
uaac token client get admin -s ADMIN-CLIENT-SECRET
```

Where `ADMIN-CLIENT-SECRET` is your UAA management admin client secret that you retrieved in a previous step. The client username is `admin`.

Step 3: Assign Enterprise PKS Cluster Scopes

The `pks.clusters.manage` and `pks.clusters.admin` UAA scopes grant users the ability to create and manage Kubernetes clusters in Enterprise PKS. For information about UAA scopes in Enterprise PKS, see UAA Scopes for Enterprise PKS Users.

To create Enterprise PKS users with the `pks.clusters.manage` or `pks.clusters.admin` UAA scope, perform one or more of the following procedures based on the needs of your deployment:

- To assign PKS cluster scopes to an individual user, see Grant Enterprise PKS Access to an Individual User. Follow this procedure if you selected **Internal UAA** when you configured **UAA** in the Enterprise PKS tile. For more information, see Installing Enterprise PKS on Azure.
- To assign PKS cluster scopes to an LDAP group, see Grant Enterprise PKS Access to an External LDAP Group. Follow this procedure if you selected **LDAP Server** when you configured **UAA** in the Enterprise PKS tile. For more information, see Installing Enterprise PKS PKS on Azure.
- To assign PKS cluster scopes to a SAML group, see Grant Enterprise PKS Access to an External SAML Group. Follow this procedure if you selected **SAML Identity Provider** when you configured **UAA** in the Enterprise PKS tile. For more information, see Installing Enterprise PKS PKS on Azure.
- To assign PKS cluster scopes to a client, see Grant Enterprise PKS Access to a Client.

Next Step

After you create admin users in Enterprise PKS, the admin users can create and manage Kubernetes clusters in Enterprise PKS. For more information, see Managing Kubernetes Clusters and Workloads.

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

Installing the PKS CLI

In this topic

Page last updated:

This topic describes how to install the VMware Enterprise PKS 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 **Enterprise 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 CLI binary executable:

```
$ chmod +x pks
```

8. Move the binary file into your `PATH`.
9. Run `pks --version` to verify the version of your PKS CLI installed locally.

Linux

1. Navigate to Pivotal Network and log in.
2. Click **Enterprise 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 CLI binary executable:

```
$ chmod +x pks
```

8. Move the binary file into your `PATH`.
9. Run `pks --version` to verify the version of your PKS CLI installed locally.

Windows

1. Navigate to Pivotal Network and log in.
2. Click **Enterprise 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`.
8. Run `pks --version` to verify the version of your PKS CLI installed locally.

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

Installing the Kubernetes CLI

In this topic

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 **VMware Enterprise 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 **VMware Enterprise 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 **VMware Enterprise 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.

Configuring Windows Worker-Based Kubernetes Clusters (Beta)

In this topic

Page last updated:

This topic describes configuring Windows worker-based Kubernetes clusters in VMware Enterprise PKS.

Overview

In Enterprise PKS you can provision a Windows worker-based Kubernetes cluster on vSphere with Flannel.

To provision a Windows worker-based Kubernetes cluster:

1. Verify your environment meets the Windows worker-based Kubernetes cluster Prerequisites.
2. Configure a Windows Worker-Based Kubernetes Cluster.
3. Upload the Windows Server Stemcell.
4. Create a Windows Worker-Based Cluster.

> **warning:** Support for Windows-based Kubernetes clusters is in beta and supports only vSphere with Flannel.
>
> Do not enable this feature if you are using Enterprise PKS with vSphere with NSX-T, Google Cloud Platform (GCP), Azure, or Amazon Web Services (AWS).

We are actively looking for feedback on this beta feature. To submit feedback, send an email to pcf-windows@pivotal.io.

Prerequisites

The following are required for creating a Windows worker-based Kubernetes cluster in Enterprise PKS:

- Your vSphere environment meets the vSphere Prerequisites and Resource Requirements.
- Enterprise PKS must be installed in a vSphere with Flannel environment.

> **Note:** NSX-T does not support networking Windows containers. If this is a key requirement for you, submit feedback by sending an email to pcf-windows@pivotal.io.

- Enterprise PKS has been configured as described in Installing Enterprise PKS on vSphere.
- You must have a vSphere stemcell 2019.7 or later for Windows Server version 2019. The latest vSphere stemcell for Windows Server version 2019 is recommended.

> **Note:** Windows stemcells for vSphere are not available on Pivotal Network. These stemcells must be created using your own Windows Server disk image (ISO file). To create a Windows stemcell for vSphere, complete the procedures in Creating a Windows Stemcell for vSphere Using stembuild.

- If your Enterprise PKS installation is in an air-gapped environment, you must prepare a Windows pause image in a private registry. For information about setting up a Windows pause image see Prepare a Windows Pause Image for an Air-Gapped Environment.

Configure a Windows Worker-Based Kubernetes Cluster

1. Configure a Windows worker plan as described in Plans, below.
2. Configure Windows worker networking as described in Networking, below.
3. Upload the Windows Server stemcell as described in Upload the Windows Server Stemcell, below.
4. Click **Apply Changes** to complete the configuration changes.

Plans

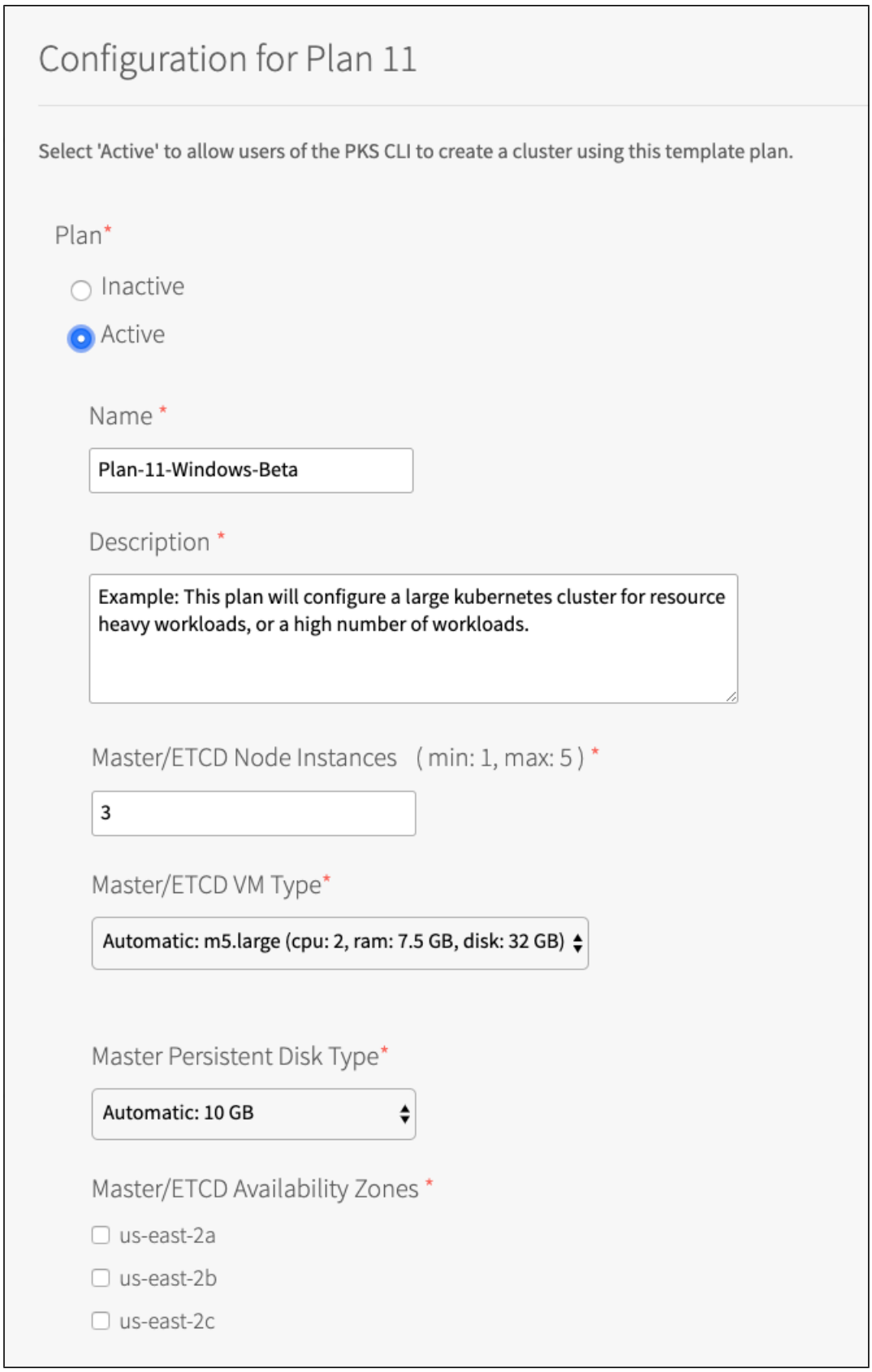
A plan defines a set of resource types used for deploying a cluster.

Activate a Plan

> **Note**: Before configuring your Windows worker plan, you must first activate and configure **Plan 1**. See Plans in *Installing Enterprise PKS on vSphere* for more information.

To activate and configure a plan, perform the following steps:

1. Click the plan that you want to activate. You must activate and configure either **Plan 11**, **Plan 12**, or **Plan 13** to deploy a Windows worker-based cluster.
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 the 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 1, 3, or 5.

> **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.

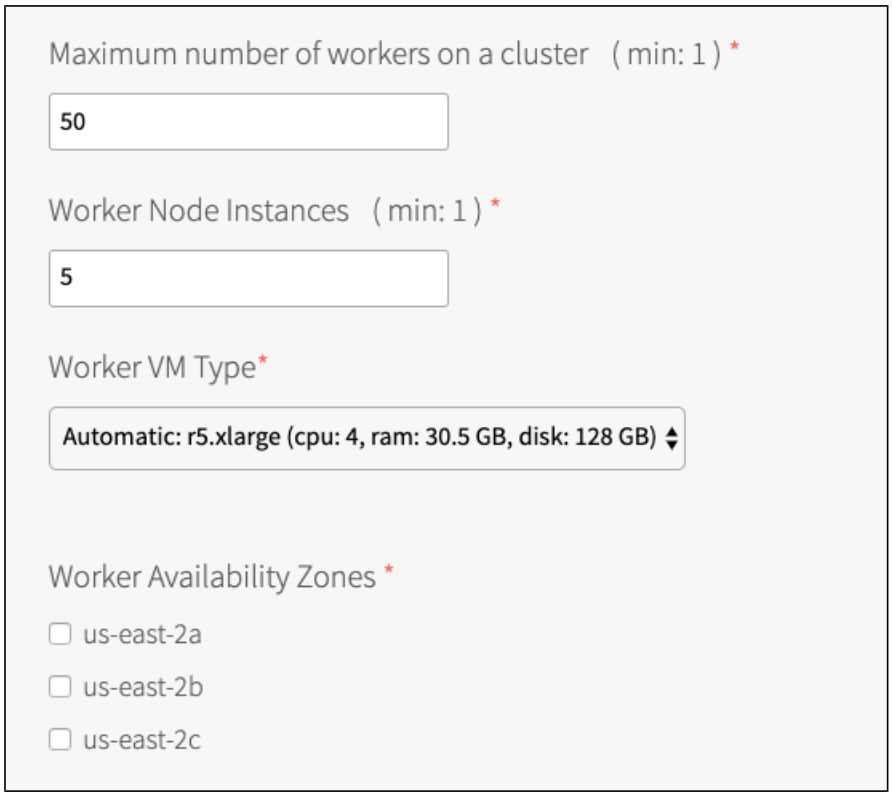
> **warning**: To change the number of master/etcd nodes for a plan, you must ensure that no existing clusters use the plan. Enterprise PKS does not support changing the number of master/etcd nodes for plans with existing clusters.

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 Enterprise 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 Enterprise PKS. If you select more than one AZ, Enterprise PKS deploys the master VM in the first AZ and the worker VMs across the remaining AZs. If you are using multiple masters, Enterprise PKS deploys the master and worker VMs across the AZs in round-robin fashion.

9. Under **Maximum number of workers on a cluster**, set the maximum number of Kubernetes worker node VMs that Enterprise 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.

For more information about creating clusters, see Creating Clusters.

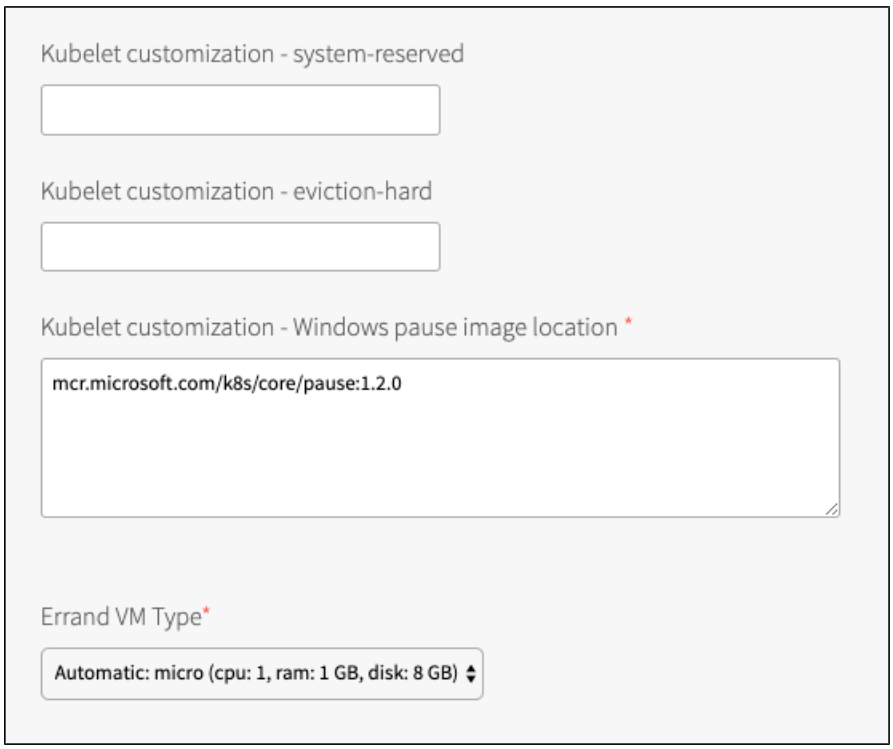
> **Note**: Changing a plan's **Worker Node Instances** setting does not alter the number of worker nodes on existing clusters. For information about scaling an existing cluster, see Scale Horizontally by Changing the Number of Worker Nodes Using the PKS CLI in *Scaling Existing Clusters*.

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 Enterprise PKS Clusters*.

> **Note:** BOSH does not support persistent disks for Windows VMs. If specifying **Worker Persistent Disk Type** on a Windows worker is a requirement for you, submit feedback by sending an email to pcf-windows@pivotal.io.

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

13. 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.



14. 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.

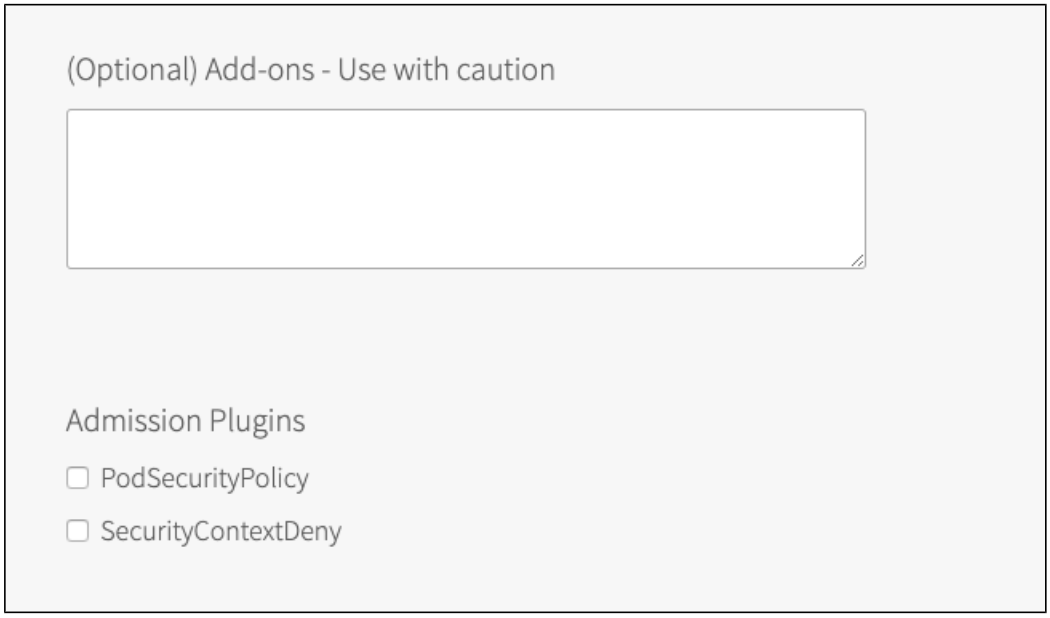
> **warning**: Use the Kubelet customization fields with caution. If you enter values that are invalid or that exceed the limits the system supports, Kubelet might fail to start. If Kubelet fails to start, you cannot create clusters.

15. Under **Kubelet customization - Windows pause image location**, enter the location of your Windows pause image. The **Kubelet customization - Windows pause image location** default value of `mcr.microsoft.com/k8s/core/pause:1.2.0` configures Enterprise PKS to pull the Windows pause image from the Microsoft Docker registry.

    The Microsoft Docker registry cannot be accessed from within air-gapped environments. If you want to deploy Windows pods in an air-gapped environment you must upload a Windows pause image to an accessible private registry, and configure the **Kubelet customization - Windows pause image location** field with the URI to this accessible Windows pause image. For more information about uploading a Windows pause image to a private registry, see Prepare a Windows Pause Image for an Internetless Environment, below.

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 Linux Workloads.



> **Note:** Windows in Kubernetes does not support privileged containers. See Feature Restrictions in the Kubernetes documentation for additional information.

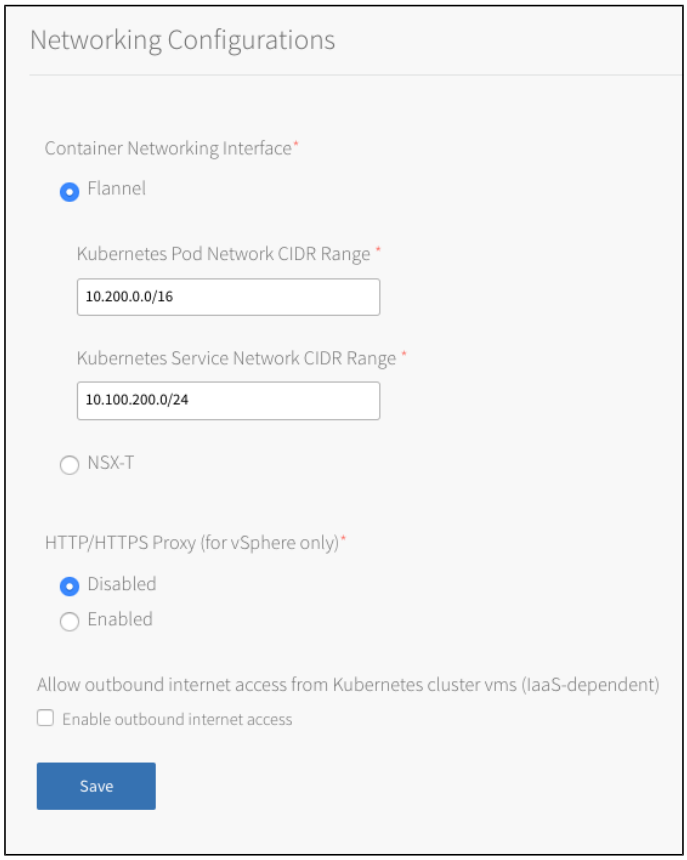
18. (Optional) Enable or disable one or more admission controller plugins: **PodSecurityPolicy**, and **SecurityContextDeny**. See Admission Plugins for more information. Windows in Kubernetes does not support the **DenyEscalatingExec** Admission Plugin feature. See API in the Kubernetes documentation for additional information.

19. Click **Save**.

Networking

To configure networking, do the following:

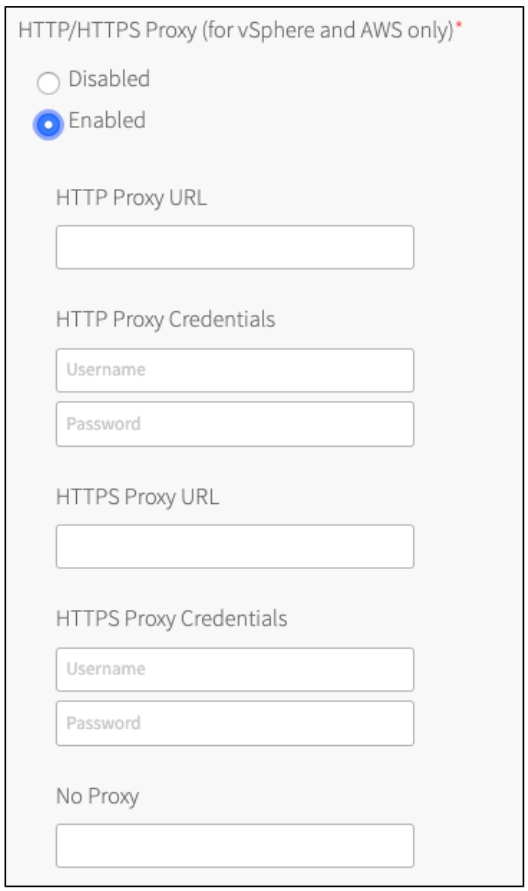
1. Click **Networking**.



2. Under **Container Networking Interface**, select **Flannel**.

3. (Optional) Enter values for **Kubernetes Pod Network CIDR Range** and **Kubernetes Service Network CIDR Range**.

   - For Windows worker-based clusters the **Kubernetes Service Network CIDR Range** setting must remain `10.220.0.0/16`. vSphere on Flannel does not support networking Windows containers. If customizing the Service Network CIDR range is a key requirement for you, submit feedback by sending an email to pcf-windows@pivotal.io.



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

Configuring Enterprise PKS to use these proxies allows Enterprise PKS-deployed Kubernetes nodes to access public Internet services and other internal services.

> **Note**: This setting will not set the proxy for running Kubernetes workloads or pods.

To complete your global proxy configuration for all outgoing HTTP/HTTPS traffic from your Kubernetes clusters, perform the following steps:

- To proxy outgoing HTTP traffic, enter the URL of your HTTP proxy endpoint under **HTTP Proxy URL**. For example, `http://myproxy.com:1234`.
- (Optional) If your outgoing HTTP proxy uses basic authentication, enter the username and password in the **HTTP Proxy Credentials** fields.
- To proxy outgoing HTTPS traffic, enter the URL of your HTTP proxy endpoint under **HTTPS Proxy URL**. For example, `http://myproxy.com:1234`.

> **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.

- (Optional) If your HTTPS proxy uses basic authentication, enter the username and password in the **HTTPS Proxy Credentials** fields.
- Under **No Proxy**, enter the comma-separated list of IP addresses that must bypass the proxy to allow for internal

Enterprise PKS communication.

The list should include `127.0.0.1` and `localhost`.

Also include:

- Your Enterprise PKS environment's CIDRs, such as the service network CIDR where your Enterprise PKS cluster is deployed, the deployment network CIDR, the node network IP block CIDR, and the pod network IP block CIDR.
- If a component is communicating with Enterprise PKS or a registry using a hostname, such as the Harbor API FQDN, instead of an IP address you will need to add the corresponding FQDN to the **No Proxy** list.
- Any additional IP addresses or domain names that should bypass the proxy.

  The **No Proxy** property for AWS accepts wildcard domains denoted by a prefixed `*.` or `.`.

  For example:

  ```
  127.0.0.1,localhost,
  *.example1.com,
  .example2.com,
  example3.com,
  198.51.100.0/24,
  203.0.113.0/24,
  192.0.2.0/24
  ```

  Note: By default the `10.100.0.0/8` and `10.200.0.0/8` IP address ranges, `.internal`, `.svc`, `.svc.cluster.local`, `.svc.cluster`, and your Enterprise PKS FQDN are not proxied. This allows internal Enterprise 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.

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

Upload the Windows Server Stemcell

1. When prompted by Ops Manager to upload a stemcell, follow the instructions and provide your previously created vSphere stemcell 2019.7 or later for Windows Server version 2019.

Create a Windows Worker-Based Cluster

1. To create a Windows worker-based cluster follow the steps in Creating Clusters.

Prepare a Windows Pause Image for an Air-Gapped Environment

To deploy a Windows pod, Kubelet deploys a Windows container image fetched from a Docker registry. Microsoft restricts distribution of Windows container base images and the fetched Windows container image is typically pulled from the Microsoft Docker registry.

To deploy Windows pods in an air-gapped environment you must have a Windows container image in a Docker registry accessible from your Enterprise PKS environment.

To prepare a Windows pause image for an air-gapped environment, perform the following:

1. Create an accessible Windows Server 2019 machine in your environment.
2. Install Docker on this Windows Server 2019 machine.
3. Configure the machine's Docker daemon to allow non-redistributable artifacts to be pushed to your private registry. For information about configuring your Docker daemon, see Allow push of nondistributable artifacts in the Docker documentation.
4. Open a command line on the Windows machine.
5. To download a Windows container image from the Microsoft Docker registry, run the following command:

   ```
   docker pull mcr.microsoft.com/k8s/core/pause:1.2.0
   ```

6. To tag the Windows container image, run the following command:

   ```
   docker tag mcr.microsoft.com/k8s/core/pause:1.2.0  REGISTRY-ROOT/windows/pause:1.2.0
   ```

   Where `REGISTRY-ROOT` is your private registry's URI.

7. To upload the Windows container image to your accessible private registry, run the following command:

   ```
   docker push  PAUSE-IMAGE-URI
   ```

   Where `PAUSE-IMAGE-URI` is the URI to the Windows pause image in your private registry. Your pause image URI should follow the pattern: `my.private.registry/windows/pause:1.2.0`.

To configure Enterprise PKS to fetch your accessible Windows container image when deploying Windows pods, perform the following:

1. Open the Enterprise PKS tile.
2. Click the Windows worker Plan that you want to configure to use your accessible private registry.
3. Modify the **Kubelet customization - Windows pause image location** property to be your pause image URI.

   For example:

   ```
   my.private.registry/windows/pause:1.2.0
   ```

4. Click **Save**.

Upgrading Enterprise PKS

Page last updated:

This section describes how to upgrade the VMware Enterprise PKS tile. See the following topics:

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

What Happens During Enterprise PKS Upgrades

In this topic

Page last updated:

This topic explains what happens to Kubernetes clusters during VMware Enterprise PKS upgrades.

Introduction

Enterprise PKS allows you to upgrade the Enterprise PKS tile and all Enterprise PKS-provisioned Kubernetes clusters simultaneously. Alternatively, you can upgrade the Enterprise PKS tile only and then upgrade your Enterprise PKS-provisioned Kubernetes clusters separately.

- Enterprise PKS Tile and Kubernetes Cluster Upgrades
- Upgrades of the Enterprise PKS Tile Only

During an upgrade of the Enterprise PKS tile, your configuration settings are automatically migrated to the new tile version. For upgrading instructions, see Upgrading Enterprise PKS.

Canary Instances

The Enterprise PKS tile is a BOSH 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 Enterprise PKS tile uses one canary instance when deploying or upgrading Enterprise PKS.

Enterprise PKS Tile and Kubernetes Cluster Upgrades

During an upgrade of the Enterprise PKS tile and Enterprise 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 Enterprise PKS is set to upgrade both the Enterprise PKS tile and Enterprise PKS-provisioned clusters, updating the Enterprise PKS tile with a new Linux or Windows stemcell rolls every Linux or Windows VM in each Kubernetes cluster. This automatic rolling ensures that all your VMs are patched. To avoid workload downtime, use the resource configuration recommended in the Master Nodes and Worker Nodes sections below and Maintaining Workload Uptime.

These control plane and cluster recreations cause the following outages:

PKS API Server and CLI Outage

When the PKS API server is recreated, you cannot interact with the PKS control plane or manage Kubernetes clusters. These restrictions prevent the following:

- Logging in to the PKS CLI and using all `pks` commands
- 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 *Enterprise PKS Cluster Management*.

Master Nodes Outage

When Enterprise 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 Outage

When Enterprise 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, Enterprise PKS upgrades Kubernetes to the version shipped with the Enterprise PKS tile. See the 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 Enterprise PKS Tile Only

During an upgrade of the Enterprise PKS tile only, the PKS API server is recreated.

To upgrade the Enterprise PKS tile only, you must set the **Upgrade all clusters errand** to **Off** before you begin the upgrade. For more information, see Upgrade the PKS Tile in *Upgrading Enterprise PKS*.

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 *Enterprise PKS Cluster Management*.

What Happens After Upgrading the Enterprise PKS Tile

All changes applied to the Enterprise PKS tile, including both upgrades and configuration changes, must be propagated to your Kubernetes clusters before the next change. Therefore, if you upgrade the Enterprise PKS only, you must upgrade all your Kubernetes clusters individually before the next Enterprise PKS tile upgrade. For instructions on upgrading individual Kubernetes clusters, see Upgrading Clusters.

> **Note**: When Enterprise PKS is set to upgrade only the Enterprise PKS tile and not the clusters, the Kubernetes cluster version falls behind the Enterprise PKS tile version. If the clusters fall more than one version behind the tile, Enterprise PKS cannot upgrade the clusters. The clusters must be upgraded to match the Enterprise PKS tile version before the next tile upgrade.

Upgrade Preparation Checklist for Enterprise PKS v1.6

In this topic

Page last updated:

This topic serves as a checklist for preparing to upgrade VMware Enterprise PKS v1.5 to VMware Enterprise PKS v1.6.

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 Enterprise PKS. If you are upgrading Enterprise PKS for environments using vSphere with NSX-T, continue to Upgrading Enterprise PKS with NSX-T.

Back Up Your Enterprise PKS Deployment

We recommend backing up your Enterprise PKS deployment before upgrading. To back up Enterprise PKS, see Backing Up and Restoring Enterprise PKS.

> **Note:** If you choose not to back up Enterprise PKS, NSX-T, or vCenter, we recommend backing up the NSX-T and NSX-T Container Plugin (NCP) logs.

Review What Happens During Enterprise PKS Upgrades

If you have not already done so, review What Happens During Enterprise PKS Upgrades.

Plan your upgrade based on your workload capacity and uptime requirements.

Review Changes in Enterprise PKS v1.6

Review the Release Notes for Enterprise PKS v1.6.

Set User Expectations and Restrict Cluster Access

Coordinate the Enterprise PKS upgrade with cluster admins and users. During the upgrade:

- Their workloads will remain active and accessible.
- They will be unable to perform cluster management functions, including creating, resizing, updating, and deleting clusters.
- They will be unable to log in to PKS or use the PKS CLI and other PKS control plane services.

Note: Cluster admins should not start any cluster management tasks right before an upgrade. Wait for cluster operations to complete before upgrading.

Upgrade All Clusters to Enterprise PKS v1.5.x

Before you upgrade to Enterprise PKS v1.6, you must upgrade all clusters. Doing this aligns the PKS version that the clusters run internally with the current patch version of the PKS control plane.

The PKS control plane supports clusters running the current version and previously-installed version of Enterprise PKS. It does not support clusters running PKS versions older than the last installed version. For example, you can upgrade to PKS v1.6.0 and continue running PKS v1.5.3 clusters.

To check the version of existing clusters and availability of upgrade, run:

```
pks clusters
```

To upgrade one or more clusters, see Upgrading Clusters.

Verify Your Clusters Support Upgrading

It is critical that you confirm that a cluster's resource usage is within the recommended maximum limits before upgrading the cluster.

VMware Enterprise PKS upgrades a cluster by upgrading master and worker nodes individually. The upgrade processes a master node by redistributing the node's workload, stopping the node, upgrading it and restoring its workload. This redistribution of a node's workloads increases the resource usage on the remaining nodes during the upgrade process.

If a Kubernetes cluster master VM is operating too close to capacity, the upgrade can fail.

warning: Downtime is required to repair a cluster failure resulting from upgrading an overloaded Kubernetes cluster master VM.

To prevent workload downtime during a cluster upgrade, complete the following before upgrading a cluster:

1. Ensure none of the master VMs being upgraded will become overloaded during the cluster upgrade. See Master Node VM Size for more information.

2. Review the cluster's workload resource usage in Dashboard. For more information, see Accessing Dashboard.

3. Scale up the cluster if it is near capacity on its existing infrastructure. Scale up your cluster by running `pks resize` or create a cluster using a larger plan. For more information, see Changing Cluster Configurations.

4. Run the cluster's workloads on at least three worker VMs using multiple replicas of your workloads spread across those VMs. For more information, see Maintaining Workload Uptime.

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 Enterprise PKS for environments using vSphere with NSX-T, perform the following steps:

1. Verify that the vSphere datastores have enough space.

2. Verify that the vSphere hosts have enough memory.

3. Verify that there are no alarms in vSphere.

4. Verify that the vSphere hosts are in a good state.

5. 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 Enterprise PKS*.

Clean up or Fix Failed Kubernetes Clusters

Clean up or fix any previous failed attempts to create PKS clusters with the PKS Command Line Interface (PKS CLI) by performing 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. To troubleshoot and fix failed clusters, perform the procedure in Cluster Creation Fails.

3. To clean up failed BOSH deployments related to failed clusters, perform the procedure in Cannot Re-Create a Cluster that Failed to Deploy.

4. After fixing and cleaning up any failed clusters, view your deployed clusters again by running `pks clusters`.

For more information about troubleshooting and fixing failed clusters, see the Pivotal Support Knowledge Base.

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 Enterprise 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 Enterprise PKS Users 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 by performing 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

Enterprise PKS upgrades can run without ever completing if any Kubernetes app has a `PodDisruptionBudget` with `maxUnavailable` set to `0`. To ensure that no apps have a `PodDisruptionBudget` 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
   ```

2. Examine the output. Verify that no app displays `0` in the `MAX UNAVAILABLE` column.

Configure Node Drain Behavior

During the Enterprise PKS tile upgrade process, worker nodes are cordoned and drained. Workloads can prevent worker nodes from draining and cause the upgrade to fail or hang.

To prevent hanging cluster upgrades, you can use the PKS CLI to configure the default node drain behavior. The new default behavior takes effect during the next upgrade, not immediately after configuring the behavior.

Configure with the Enterprise PKS Tile

To configure node drain behavior in the Enterprise PKS tile, see Worker Node Hangs Indefinitely in *Troubleshooting*.

Configure with the PKS CLI

To configure default node drain behavior, do the following:

1. View the current node drain behavior by running the following command:

   ```
   pks cluster CLUSTER-NAME --details
   ```

   Where `CLUSTER-NAME` is the name of your cluster.

   For example:

   ```
   $ pks cluster my-cluster --details

     Name:                     my-cluster
     Plan Name:                small
     UUID:                     f55ed6c4-c0a7-451d-b735-56c89fdb2ad7
     Last Action:              CREATE
     Last Action State:        succeeded
     Last Action Description:  Instance provisioning completed
     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:
     Kubernetes Settings Details:
       Set by Cluster:
       Kubelet Node Drain timeout (mins)          (kubelet-drain-timeout):              10
       Kubelet Node Drain grace-period (mins)     (kubelet-drain-grace-period):         10
       Kubelet Node Drain force                   (kubelet-drain-force):                true
       Set by Plan:
       Kubelet Node Drain force-node              (kubelet-drain-force-node):           true
       Kubelet Node Drain ignore-daemonsets       (kubelet-drain-ignore-daemonsets):    true
       Kubelet Node Drain delete-local-data       (kubelet-drain-delete-local-data):    true
   ```

2. Configure the default node drain behavior by running the following command:

   ```
   pks update-cluster CLUSTER-NAME FLAG
   ```

   Where:

   - `CLUSTER-NAME` is the name of your cluster.
   - `FLAG` is an action flag for updating the node drain behavior.

For example:

```
$ pks update-cluster my-cluster --kubelet-drain-timeout 1 --kubelet-drain-grace-period 5

  Update summary for cluster my-cluster:
  Kubelet Drain Timeout: 1
  Kubelet Drain Grace Period: 5
  Are you sure you want to continue? (y/n): y
  Use 'pks cluster my-cluster' to monitor the state of your cluster
```

For a list of the available action flags for setting node drain behavior, see pks update-cluster in *PKS CLI*.

Update Your Role for the Worker Node Managed Identity on Azure

If you are running Enterprise PKS on Azure, you must add the `"Microsoft.Compute/virtualMachines/read"` action to the worker node managed identity.

> **Note**: You do not need to modify the worker node managed identity role if you are running Enterprise PKS on AWS, GCP, vSphere, or vSphere with NSX-T. Modifying the role for Azure is a requirement as of Kubernetes v1.14.5.

To add the `"Microsoft.Compute/virtualMachines/read"` action, do the following:

1. List your roles using the Azure CLI. For example:

   ```
   $ az role definition list --custom-role-only true -o json
   ```

2. Retrieve the definition of the `"PKS worker"` role using the `roleName` key. For example:

   ```
   $ az role definition list --custom-role-only true -o json | jq -r '.[] | select(.roleName=="PKS worker")'
   ```

3. Copy the JSON to a file and add `"Microsoft.Compute/virtualMachines/read"` under `"Actions"`.

4. Save your template as `pks_worker_role.json`.

5. Run the following command to update the role:

   ```
   az role definition update --role-definition pks_worker_role.json
   ```

For more information about creating managed identities for Enterprise PKS, see Creating Managed Identities in Azure for Enterprise PKS.

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

Upgrading Enterprise PKS

In this topic

Page last updated:

This topic explains how to upgrade VMware Enterprise PKS from v1.5 to v1.6 on vSphere without NSX-T integration, Google Cloud Platform (GCP), Amazon Web Services (AWS), and Azure.

For instructions on upgrading Enterprise PKS on vSphere with NSX-T integration, see Upgrading Enterprise PKS with NSX-T.

Overview

The supported upgrade paths to Enterprise PKS v1.6 are from Enterprise PKS v1.5.x. Enterprise PKS v1.6 is compatible with Ops Manager v2.6.16 and later and Ops Manager v2.7.6 and later.

warning: Do not manually upgrade your Kubernetes version. Enterprise PKS includes the compatible Kubernetes version.

Before you begin the upgrade, follow the procedures in the Prerequisites section to plan and prepare your upgrade.

After you complete the prerequisite steps, continue to the procedures in the During the Upgrade section. These steps guide you through the process of upgrading Ops Manager, upgrading the Enterprise PKS tile, importing an updated stemcell, and applying the changes to your deployment.

After you complete the upgrade, follow the procedures in the After the Upgrade section to verify that your upgraded Enterprise PKS deployment is running properly.

Prerequisites

Before you upgrade Enterprise PKS, you must do the following:

1. Determine Your Upgrade Path
2. Prepare to Upgrade

Determine Your Upgrade Path

Use the following table to determine your upgrade path to Enterprise PKS v1.6.

If you want to upgrade Enterprise PKS on Ops Manager …	Then use the following upgrade path:
v2.5	1. Upgrade to Ops Manager v2.6.16 and later. 2. Upgrade to Enterprise PKS v1.6. 3. (Optional) Upgrade to Ops Manager v2.7.6 and later.
v2.6	1. (Optional) Upgrade to Ops Manager v2.7.6 and later. 2. Upgrade to Enterprise PKS v1.6.

Prepare to Upgrade

If you have not already, complete all of the steps in the Upgrade Preparation Checklist for Enterprise PKS v1.6.

During the Upgrade

This section describes the steps required to upgrade to Enterprise PKS v1.6:

1. Upgrade to Ops Manager v2.7.6+ or v2.6.16+
2. Upgrade to Enterprise PKS v1.6
3. Download and Import the Stemcell
4. Verify Errand Configuration
5. Apply Changes to the Enterprise PKS Tile

Upgrade Ops Manager to v2.6.16+ or v2.7.6+

Before you upgrade to Enterprise PKS v1.6, you must upgrade to Ops Manager v2.6.16 and later or v2.7.6 and later.

To upgrade Ops Manager, do the following:

1. Complete the procedures in Import Installation to Ops Manager v2.6 VM or Import Installation to Ops Manager v2.7 VM.
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, see About Cluster Upgrades in *Maintaining Workload Uptime* and Creating Clusters.
b. Monitor the PKS control plane VM by clicking the **Enterprise PKS** tile, selecting the **Status** tab, and reviewing the **Pivotal Container Service** VM's data points. If any data points are at capacity, scale your deployment accordingly.

Upgrade Enterprise PKS to v1.6

When you upgrade Enterprise PKS, your configuration settings typically migrate to the new version automatically. To perform an upgrade:

1. Review the Release Notes for the version you are upgrading to.
2. Download the desired version of the product from Pivotal Network.
3. Navigate to the Ops Manager Installation Dashboard and click **Import a Product** to upload the product file.
4. Under the **Import a Product** button, click **+** next to **Enterprise PKS**. This adds the tile to your staging area.

Download and Import the Stemcell

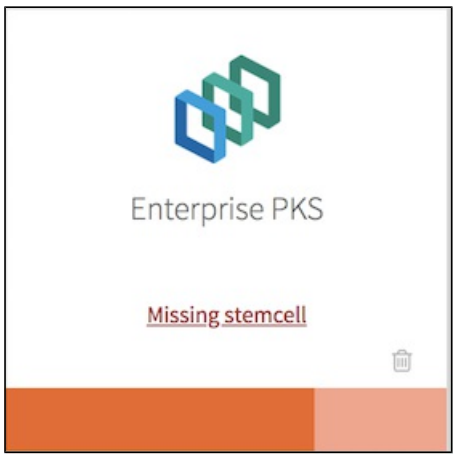
Enterprise PKS requires a Xenial stemcell. A stemcell for Windows 2019 is also required if you intend to create Windows worker-based clusters. For information about Windows stemcells, see Configuring Windows Worker-Based Clusters (Beta).

If Ops Manager does not have the Xenial stemcell required for Enterprise PKS, the Enterprise 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 Enterprise PKS tile, click the **Missing stemcell** link.



2. In the **Stemcell Library**, locate the **Enterprise PKS** tile and note the required stemcell version.
3. Navigate to the Pivotal Stemcells (Ubuntu Xenial) page on Pivotal Network and download the required stemcell version for your IaaS.

4. Return to the **Installation Dashboard** in Ops Manager and click **Stemcell Library**.
5. On the **Stemcell Library** page, click **Import Stemcell** and select the stemcell file you downloaded from Pivotal Network.
6. Select the Enterprise PKS tile 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 Enterprise PKS.
8. Select the **Installation Dashboard** link to return to the Installation Dashboard.

(Optional) Configure Host Monitoring

In Enterprise PKS v1.6, the **Logging** screen is renamed to **Host Monitoring** and adds options to configure the Telegraf agent to collect metrics from Node Exporter, etcd, and the Kubelet agent and send the metrics to a third-party monitoring service.

For information about configuring **Host Monitoring**, see the *(Optional) Host Monitoring* section of the installation topic for your IaaS:

- Installing Enterprise PKS on vSphere
- Installing Enterprise PKS on AWS
- Installing Enterprise PKS on Azure
- Installing Enterprise PKS on GCP

(Optional) Configure In-Cluster Monitoring

In Enterprise PKS v1.6, the **Monitoring** screen is renamed to **In-Cluster Monitoring**.

For information about configuring **In-Cluster Monitoring**, see the *(Optional) In-Cluster Monitoring* section of the installation topic for your IaaS:

- Installing Enterprise PKS on vSphere
- Installing Enterprise PKS on AWS
- Installing Enterprise PKS on Azure
- Installing Enterprise PKS on GCP

(Optional) Configure Tanzu Mission Control Integration

If you are a participant in the VMware Tanzu Mission Control beta program, you can choose to integrate your Enterprise PKS deployment with Tanzu Mission Control.

For information about configuring **Tanzu Mission Control (Experimental)**, see the *Tanzu Mission Control (Experimental)* section of the installation topic for your IaaS:

- Installing Enterprise PKS on vSphere
- Installing Enterprise PKS on AWS
- Installing Enterprise PKS on Azure
- Installing Enterprise PKS on GCP

Verify Errand Configuration

To verify that errands are configured correctly in the Enterprise PKS tile, do the following:

1. In the **Enterprise PKS** tile, click **Errands**.

2. Under **Post-Deploy Errands**, configure **Upgrade all clusters errand**:

   - If you want to upgrade the Enterprise PKS tile and all your existing Kubernetes clusters simultaneously, verify that **Upgrade all clusters errand** is set to **Default (On)**. The errand upgrades all clusters, one Kubernetes cluster at a time. Upgrading Enterprise PKS-deployed Kubernetes clusters can temporarily interrupt the service as described in Service Interruptions.
   - If you want to upgrade the Enterprise PKS tile only and then upgrade your existing Kubernetes clusters separately, disable **Upgrade all clusters errand**. For more information, see Upgrading Clusters.

   > **warning:** If you disable **Upgrade all clusters errand** when upgrading the Enterprise PKS tile or updating your tile configuration, you must upgrade all your Kubernetes clusters before the next Enterprise PKS upgrade or tile configuration update.

3. Under **Post-Deploy Errands**, set the **Run smoke tests** errand to **On**. The errand uses the Enterprise 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 Enterprise PKS tile is aborted.

4. Click **Save**.

5. Review the other configuration panes. Click **Save** on any panes where you make changes.

   > **warning**: Do not change the number of master/etcd nodes for any plan that was used to create currently-running clusters. Enterprise PKS does not support changing the number of master/etcd nodes for plans with existing clusters.

   > **Note:** When you upgrade Enterprise PKS, you must place singleton jobs in the AZ you selected when you first installed the Enterprise PKS tile. You cannot move singleton jobs to another AZ.

Apply Changes to the Enterprise PKS Tile

To complete the upgrade of the Enterprise 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 Enterprise PKS deployment. For more information, see Using BOSH Diagnostic Commands in Enterprise PKS.
   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 Enterprise PKS v1.6, complete the following verifications and upgrades:

- Update PKS and Kubernetes CLIs
- Verify the Upgrade
- Upgrade vSphere

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 Enterprise PKS.

To update your CLIs, download and re-install the PKS and Kubernetes CLI distributions that are provided with Enterprise 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 Enterprise 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, see About Cluster Upgrades in *Maintaining Workload Uptime* and Creating Clusters.
   b. Monitor the PKS control plane VM by clicking the **Enterprise PKS** tile, selecting the **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 desired, upgrade vSphere to a version that is supported by Enterprise PKS.

For Enterprise PKS on vSphere version requirements, refer to the VMware Product Interoperability Matrices.

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

Upgrading Enterprise PKS with NSX-T

In this topic

Page last updated:

This topic explains how to upgrade VMware Enterprise PKS for environments using vSphere with NSX-T.

> **warning:** Do not manually upgrade your Kubernetes version. Enterprise PKS includes the compatible Kubernetes version.

Overview

The supported upgrade paths to Enterprise PKS v1.6.0 are from Enterprise PKS v1.5.0 and later. Enterprise PKS v1.6.0 is compatible with Ops Manager v2.6.16 and later or Ops Manager v2.7.6 and later.

Before you begin the upgrade, follow the procedures in the Prerequisites section to plan and prepare your upgrade.

After you complete the prerequisite steps, continue to the procedures in the During the Upgrade section. These steps guide you through the process of upgrading the Enterprise PKS tile, upgrading Ops Manager, importing an updated stemcell, and applying the changes to your deployment.

After you complete the upgrade, follow the procedures in the After the Upgrade section to verify that your upgraded Enterprise PKS deployment is running properly.

Prerequisites

Before you upgrade Enterprise PKS, you must do the following:

1. Determine Your Upgrade Path

2. Prepare to Upgrade

Determine Your Upgrade Path

Use the following table to determine your upgrade path to Enterprise PKS v1.6.

Target Ops Manager version	Upgrade Path for NSX-T v2.4	Upgrade Path for NSX-T v2.5
v2.6	1. Upgrade to NSX-T Data Center v2.4.3 2. Upgrade to Ops Manager v2.6.16 and later. 3. Upgrade to Enterprise PKS v1.6. 4. Upgrade to NSX-T Data Center v2.5.	1. Upgrade to Ops Manager v2.6.16 and later. 2. Upgrade to Enterprise PKS v1.6.
v2.7	1. Upgrade to NSX-T Data Center v2.4.3 2. Upgrade to Ops Manager v2.7.6 and later. 3. Upgrade to Enterprise PKS v1.6. 4. Upgrade to NSX-T Data Center v2.5.	1. Upgrade to Ops Manager v2.7.6 and later. 2. Upgrade to Enterprise PKS v1.6.

Prepare to Upgrade

If you have not already, complete all of the steps in the Upgrade Preparation Checklist for Enterprise PKS v1.6.

Review new features that are available in Enterprise PKS v1.6 by reading the Release Notes. In particular, review the new networking-related features that are available for NSX-T deployments.

During the Upgrade

This section describes the steps required to upgrade to Enterprise PKS v1.6:

1. Review the Release Notes for the supported software versions.
2. Upgrade to Ops Manager v2.6.16+ or v2.7.6+
3. Upgrade to Enterprise PKS v1.6
4. Download and Import the Stemcell
5. Verify Errand Configuration
6. Apply Changes to the Enterprise PKS Tile
7. Upgrade your Kubernetes clusters by selecting the **Upgrade all clusters errand** in Ops Manager, or by using the PKS CLI as described in Upgrading Clusters.

> **Note:** This operation updates the Kubernetes clusters to the version of Kubernetes that is included with with Enterprise PKS v1.6, and updates your Kubernetes clusters to the version of NCP that is included with Enterprise PKS v1.6.

8. (Recommended) If you are running NSX-T v2.4.3, upgrade to v2.5.x.

Upgrade Ops Manager to v2.6.16+ or v2.7.6+

Before you upgrade to Enterprise PKS v1.6, you must upgrade to Ops Manager v2.6.16 and later or v2.7.6 and later.

To upgrade Ops Manager, do the following:

1. Complete the procedures in Upgrade Ops Manager and Installed Products to v2.6 or Upgrade Ops Manager and Installed Products to v2.7.

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, see About Cluster Upgrades in *Maintaining Workload Uptime* and Creating Clusters.
   b. Monitor the PKS control plane VM by clicking the **Enterprise PKS** tile, selecting the **Status** tab, and reviewing the **Pivotal Container Service** VM's data points. If any data points are at capacity, scale your deployment accordingly.

Upgrade Enterprise PKS to v1.6

To upgrade to Enterprise PKS v1.6, 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 **Enterprise PKS**. This adds the tile to your staging area.

Download and Import the Stemcell

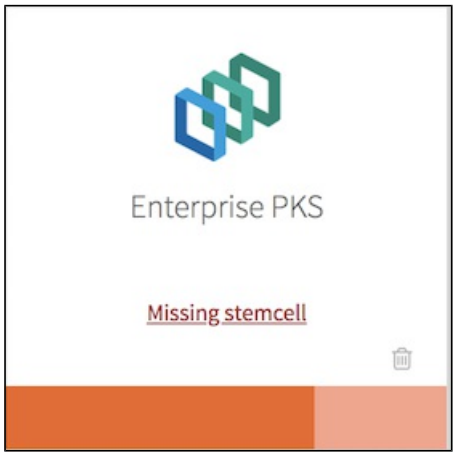
Enterprise PKS v1.6 requires a Xenial stemcell.

If Ops Manager does not have the Xenial stemcell required for Enterprise PKS, the Enterprise 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 Enterprise PKS tile, click the **Missing stemcell** link.



2. In the **Stemcell Library**, locate the **Enterprise PKS** tile and note the required stemcell version.
3. Navigate to the Pivotal Stemcells (Ubuntu Xenial) page on Pivotal Network and download the required stemcell version for your IaaS.
4. Return to the **Installation Dashboard** in Ops Manager and click **Stemcell Library**.
5. On the **Stemcell Library** page, click **Import Stemcell** and select the stemcell file you downloaded from Pivotal Network.
6. Select the Enterprise PKS tile 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 Enterprise PKS.
8. Select the **Installation Dashboard** link to return to the Installation Dashboard.

(Optional) Configure Host Monitoring

In Enterprise PKS v1.6, the **Logging** screen is renamed to **Host Monitoring** and adds options to configure the Telegraf agent to collect metrics from Node Exporter, etcd, and the Kubelet agent and send the metrics to a third-party monitoring service.

For information about configuring **Host Monitoring**, see the *(Optional) Host Monitoring* section of Installing Enterprise PKS on vSphere with NSX-T.

(Optional) Configure In-Cluster Monitoring

In Enterprise PKS v1.6, the **Monitoring** screen is renamed to **In-Cluster Monitoring**.

For information about configuring **In-Cluster Monitoring**, see the *(Optional) In-Cluster Monitoring* section of Installing Enterprise PKS on vSphere with NSX-T.

(Optional) Configure Tanzu Mission Control Integration

If you are a participant in the VMware Tanzu Mission Control beta program, you can choose to integrate your Enterprise PKS deployment with Tanzu Mission Control.

For information about configuring **Tanzu Mission Control (Experimental)**, see the *Tanzu Mission Control (Experimental)* section of

Installing Enterprise PKS on vSphere with NSX-T.

Verify Errand Configuration

To verify that errands are configured correctly in the Enterprise PKS tile, do the following:

1. In the **Enterprise PKS** tile, click **Errands**.
2. Under **Post-Deploy Errands**, configure **Upgrade all clusters errand**:
   - If you want to upgrade the Enterprise PKS tile and all your existing Kubernetes clusters simultaneously, verify that **Upgrade all clusters errand** is set to **Default (On)**. The errand upgrades all clusters, one Kubernetes cluster at a time. Upgrading Enterprise PKS-deployed Kubernetes clusters can temporarily interrupt the service as described in Service Interruptions.

     > **Note:** If you are upgrading Enterprise PKS on NSX-T v2.4.3, you must select the **Upgrade all clusters errand** or plan to upgrade all clusters individually prior to upgrading to NSX-T v2.5.

   - If you want to upgrade the Enterprise PKS tile only and then upgrade your existing Kubernetes clusters separately, disable **Upgrade all clusters errand**. For more information, see Upgrading Clusters.

     > **warning:** If you disable **Upgrade all clusters errand** when upgrading the Enterprise PKS tile or updating your tile configuration, you must upgrade all your Kubernetes clusters before the next Enterprise PKS upgrade or tile configuration update.

3. Under **Post-Deploy Errands**, set the **Run smoke tests** errand to **On**. The errand uses the Enterprise 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 Enterprise PKS tile is aborted.
4. Click **Save**.
5. Review the other configuration panes. Click **Save** on any panes where you make changes.

   > **warning**: Do not change the number of master/etcd nodes for any plan that was used to create currently-running clusters. Enterprise PKS does not support changing the number of master/etcd nodes for plans with existing clusters.

   > **Note:** When you upgrade Enterprise PKS, you must place singleton jobs in the AZ you selected when you first installed the Enterprise PKS tile. You cannot move singleton jobs to another AZ.

Apply Changes to the Enterprise PKS Tile

Perform the following steps to complete the upgrade of the Enterprise 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 Enterprise PKS deployment. For more information, see Using BOSH Diagnostic Commands in Enterprise PKS.
    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 Enterprise PKS v1.6, complete the following verifications and upgrades:

- Update PKS and Kubernetes CLIs
- Verify the Upgrade

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 Enterprise PKS.

To update your CLIs, download and re-install the PKS and Kubernetes CLI distributions that are provided with Enterprise 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 Enterprise 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, see About Cluster Upgrades in *Maintaining Workload Uptime* and Creating Clusters.
    b. Monitor the PKS control plane VM by clicking the **Enterprise PKS** tile, selecting the **Status** tab, and reviewing the **Pivotal Container Service** VM's data points. If any data points are at capacity, scale your deployment accordingly.

Upgrade NSX-T Data Center v2.4.3 to v2.5

If you are using NSX-T v2.4.3, Pivotal recommends that you upgrade to NSX-T v2.5.

1. Confirm that you have upgraded all PKS-provisioned Kubernetes clusters to PKS v1.6 using **Upgrade all clusters errand** in Ops Manager or using the PKS CLI.

2. Confirm that your vSphere v6.5 or v6.7 installation is on the supported version and patch for NSX-T v2.5.

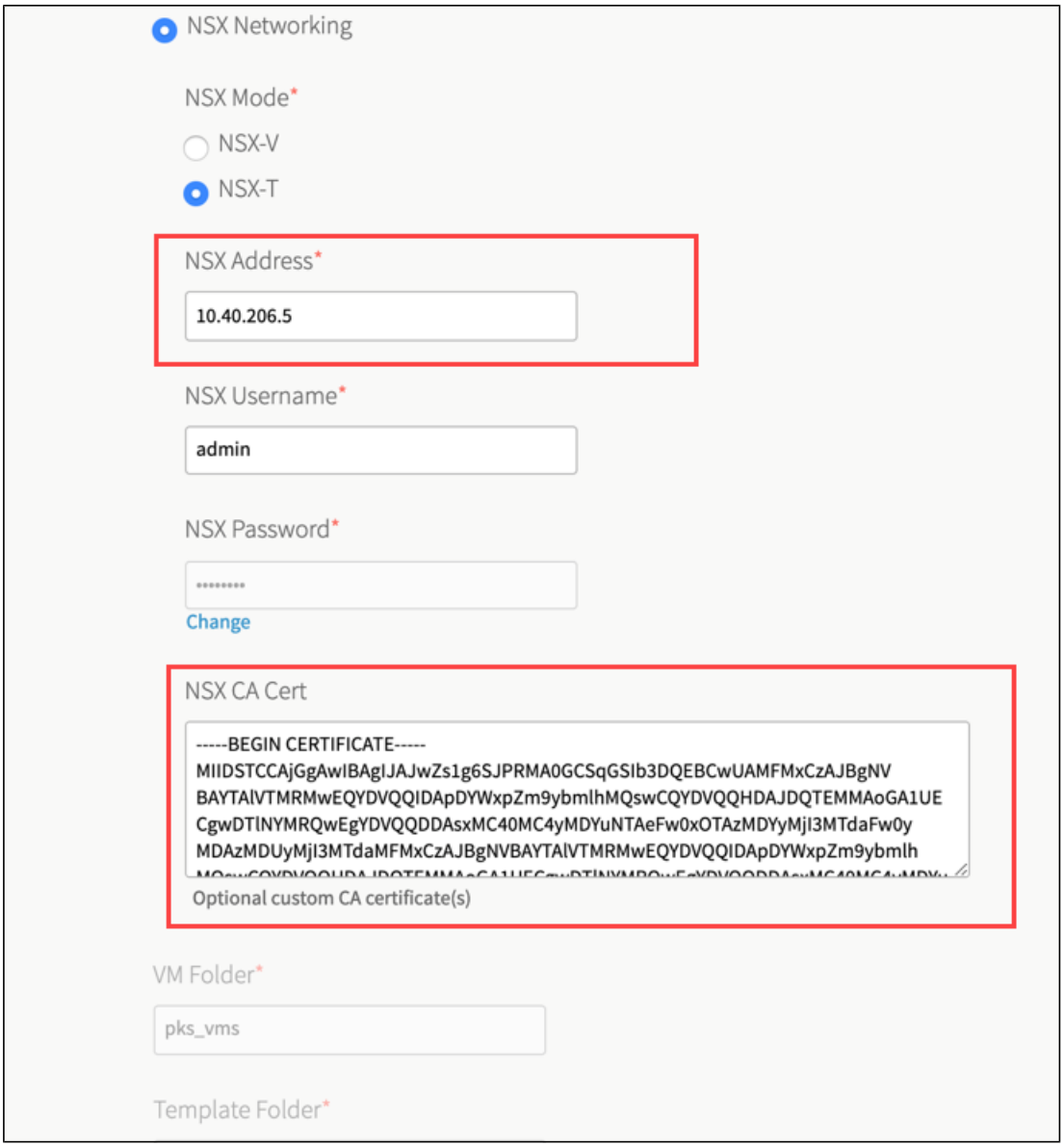
- Refer to the VMware Product Interoperability Matrices.
- If necessary, update to the required vSphere version or patch before proceeding with the upgrade of NSX-T.

3. To upgrade NSX-T, refer to the NSX-T Data Center Upgrade Guide in the VMware documentation.

4. If you made any architectural changes to your NSX-T environment that affect Enterprise PKS, such as adding a VIP address or load balancer for the NSX-T Management Cluster, update the PKS and BOSH Director tile with the new information, specifically:

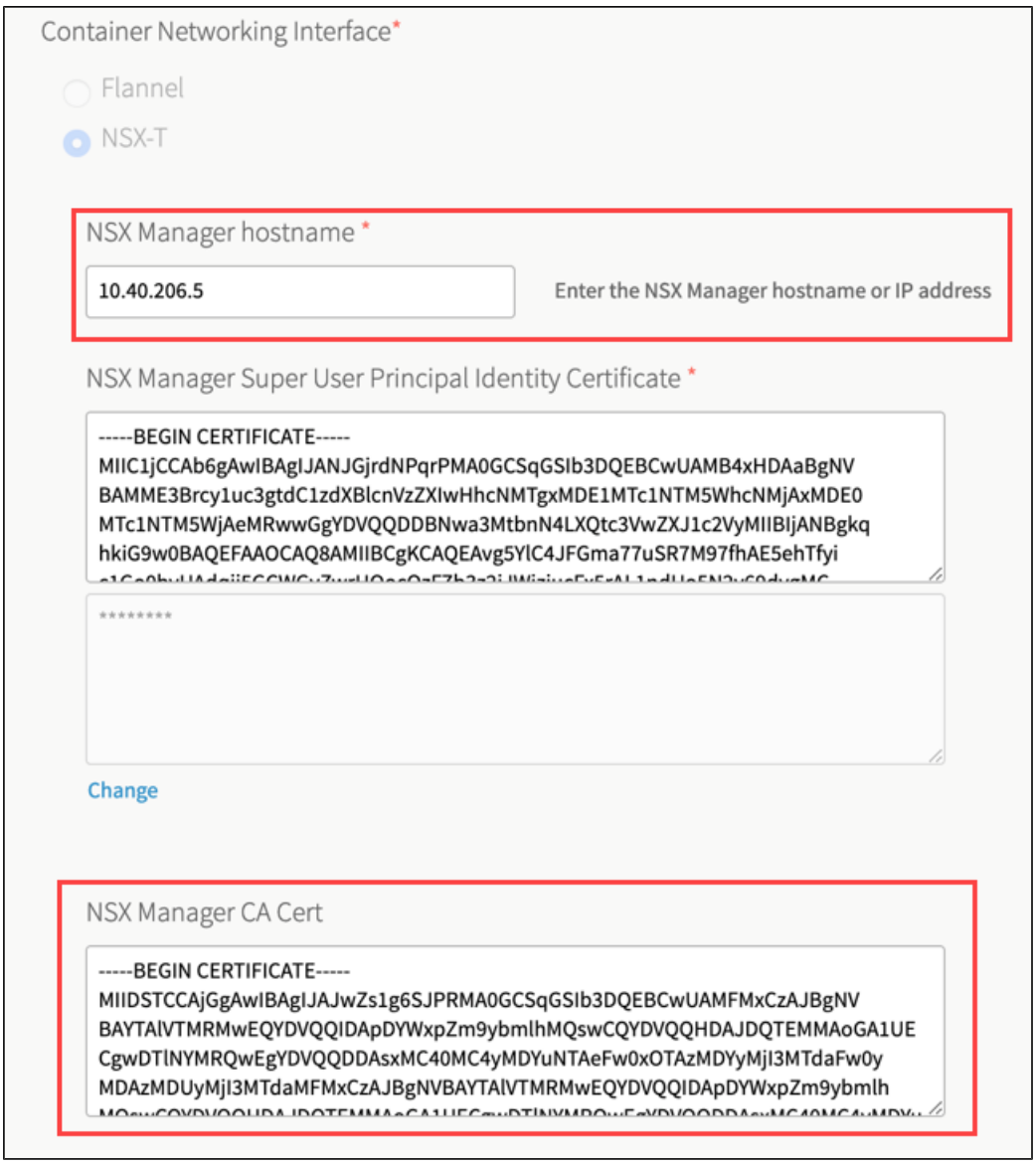
   - In the BOSH Director tile, select the **vCenter Configuration** tab and update the following:

     - The **NSX Address** field
     - The **NSX CA Cert** field
     - Save the BOSH tile changes.

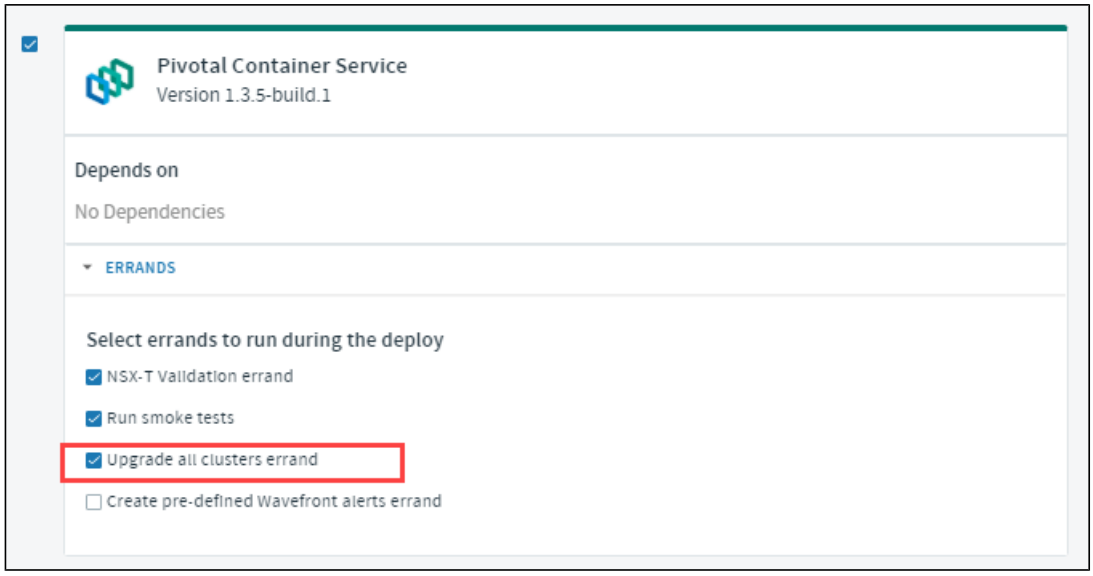


   - In the PKS tile, select the **Networking** tab and update the following:

     - The **NSX Manager hostname** field
     - The **NSX Manager CA Cert** field
     - Save the PKS tile changes



5. If you made changes to the BOSH Director or PKS tiles:
   - Go to the **Installation Dashboard** in Ops Manager.
   - Click **Review Pending Changes.**
   - Expand the **Errands** list for Enterprise PKS.
   - Ensure that the **Upgrade all clusters errand** is selected.
   - Click **Apply Changes**.



6. Verify the upgrade of NSX-T Data Center.

After you apply changes to the Enterprise PKS tile and the upgrade is complete, verify that your Kubernetes environment is healthy. To verify the health of your Kubernetes environment, see Verifying Deployment Health.

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

Maintaining Workload Uptime

In this topic

Page last updated:

This topic describes how you can maintain workload uptime for Kubernetes clusters deployed with VMware Enterprise 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 Cluster Upgrades

The Enterprise PKS tile contains an errand that upgrades all Kubernetes clusters. Upgrades run on a single VM at a time:

- While a master VM is upgraded, the VM's workloads are distributed to the cluster's remaining master VMs.
- While a worker VM is upgraded, the workload on that VM goes down. The cluster's 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.

Upgrading a cluster with only a single master or worker VM results in a workload outage.

To prevent workload downtime during a cluster upgrade, VMware recommends the following:

- Ensure none of the master VMs being upgraded will become overloaded during the cluster upgrade. See Master Node VM Size for more information.
- Run 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 Enterprise PKS, see Configuring and Using PersistentVolumes.

For information about the supported storage topologies for Enterprise PKS on vSphere, see PersistentVolume Storage Options on vSphere.

Configuring the Upgrade Pipeline

In this topic

Page last updated:

This topic describes how to set up a Concourse pipeline to perform automatic upgrades of a VMware Enterprise PKS installation.

When you configure the upgrade pipeline, the pipeline upgrades your installation when a new Enterprise 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 Enterprise PKS

Page last updated:

This section describes how to manage VMware Enterprise PKS.

See the following topics:

- Managing Enterprise PKS Users
- Managing Kubernetes Cluster Resources
- Adding Infrastructure Password Changes to the Enterprise PKS Tile
- Shutting Down and Restarting Enterprise PKS
- Deleting Enterprise PKS

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

Managing Enterprise PKS Users

Page last updated:

This section describes how to connect UAA to external user stores when configuring VMware Enterprise PKS and how to manage users with UAA.

See the following topics:

- Connecting Enterprise PKS to an LDAP Server
- Configuring Okta as a SAML Identity Provider
- Configuring Azure Active Directory as a SAML Identity Provider
- Connecting Enterprise PKS to a SAML Identity Provider
- Managing Enterprise PKS Users with UAA

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

Connecting Enterprise PKS to an LDAP Server

In this topic

Page last updated:

This topic describes how to connect VMware Enterprise PKS to an external LDAP server.

Overview

User Account and Authentication (UAA), the identity management service for Enterprise PKS, can authenticate users either through its internal user account store or external authentication mechanisms such as an LDAP server or a SAML identity provider.

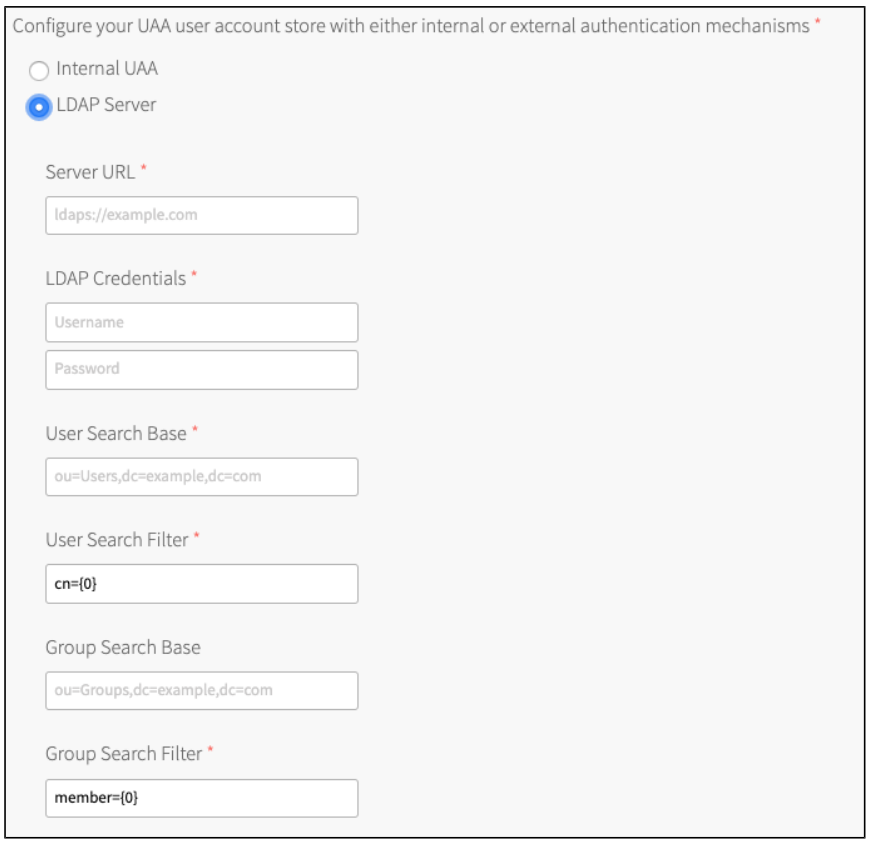
To enable an internal user account store for UAA, you select **Internal UAA** in the **Enterprise PKS** tile > **UAA**.

If you want to connect Enterprise PKS to an external LDAP server, you must integrate the UAA server with your LDAP server by following the instructions in Integrate UAA with an LDAP Server below. This enables UAA to delegate authentication to your LDAP user store.

Integrate UAA with an LDAP Server

To integrate UAA with one or more LDAP servers:

1. In **Enterprise PKS** > **UAA**, under **Configure your UAA user account store with either internal or external authentication mechanisms**, select **LDAP Server**.



2. Under **Server URL**, enter the URLs that point to your LDAP server. For example, `ldaps://example.com`. If you have multiple LDAP servers, separate their URLs with spaces. Each URL must include one of the following protocols:
   - `ldap://`: Enter this protocol if your LDAP server uses an unencrypted connection.
   - `ldaps://`: Enter 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. Under **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. Under **User Search Base**, enter the location in the LDAP directory tree where LDAP user search begins. For example, a domain named `cloud.example.com` may use `ou=Users,dc=example,dc=com` as its LDAP user search base.

5. Under **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 Enterprise 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 for Active Directory, `sAMAccountName`.

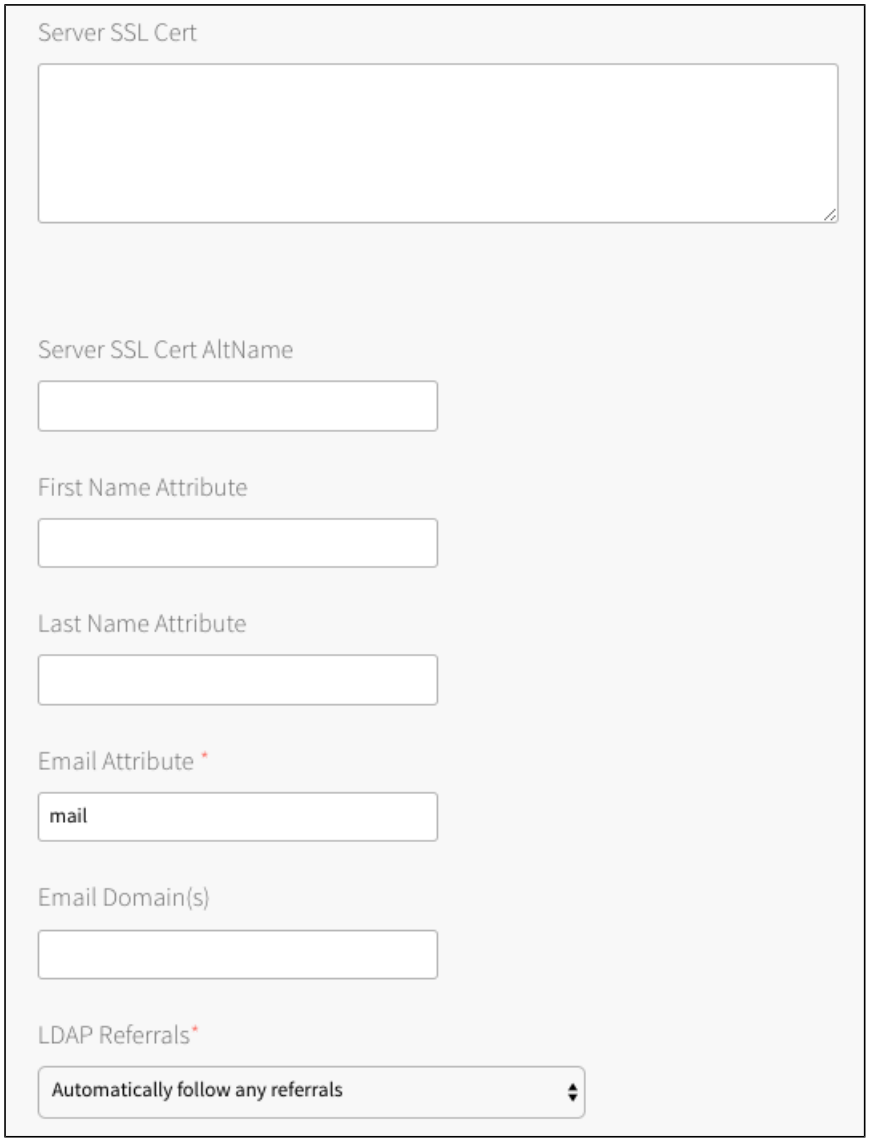
   > **Note:** For information about testing and troubleshooting your LDAP search filters, see Configuring LDAP Integration with Pivotal Cloud Foundry.

6. Under **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. You must configure **Group Search Base** if you want to map an external LDAP group to a role in Enterprise PKS or a Kubernetes group.

> **Note:** To map the groups under this search base to roles in Enterprise PKS, follow the instructions in Grant Enterprise PKS Access to an External LDAP Group.

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

8. Under **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. Under **First Name Attribute**, enter the attribute name in your LDAP directory that contains user first names. For example, `cn`.

11. Under **Last Name Attribute**, enter the attribute name in your LDAP directory that contains user last names. For example, `sn`.

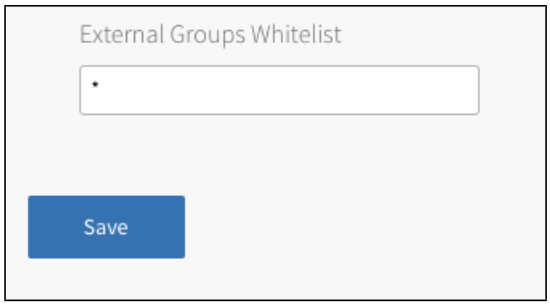
12. Under **Email Attribute**, enter the attribute name in your LDAP directory that contains user email addresses. For example, `mail`.

13. Under **Email Domain(s)**, enter a comma-separated list of the email domains for external users who can receive invitations to Apps Manager.

14. Under **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. Under **External Groups Whitelist**, enter a comma-separated list of group patterns that need to be populated in the user's `id_token`. For more information about accepted patterns, see the description of `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**.

Complete Your Tile Configuration

- If you do not need to configure any other settings in the Enterprise PKS tile, return to the Ops Manager Installation Dashboard and click **Review Pending Changes** > **Apply Changes**.
- If you need to configure any other settings in the Enterprise PKS tile, return to the *Installing Enterprise PKS* topic for your IaaS and follow the instructions for the pane you want to configure:
    - Installing Enterprise PKS on vSphere
    - Installing Enterprise PKS on vSphere with NSX-T
    - Installing Enterprise PKS on GCP
    - Installing Enterprise PKS on AWS
    - Installing Enterprise PKS on Azure

Next Steps

For information about creating Enterprise PKS roles and managing Kubernetes cluster access, see:

- Setting Up Enterprise PKS Admin Users for your IaaS
- Managing Cluster Access and Permissions

Configuring Okta as a SAML Identity Provider

In this topic

Page last updated:

This topic explains how to configure single sign-on (SSO) between Okta and VMware Enterprise PKS.

Prerequisites

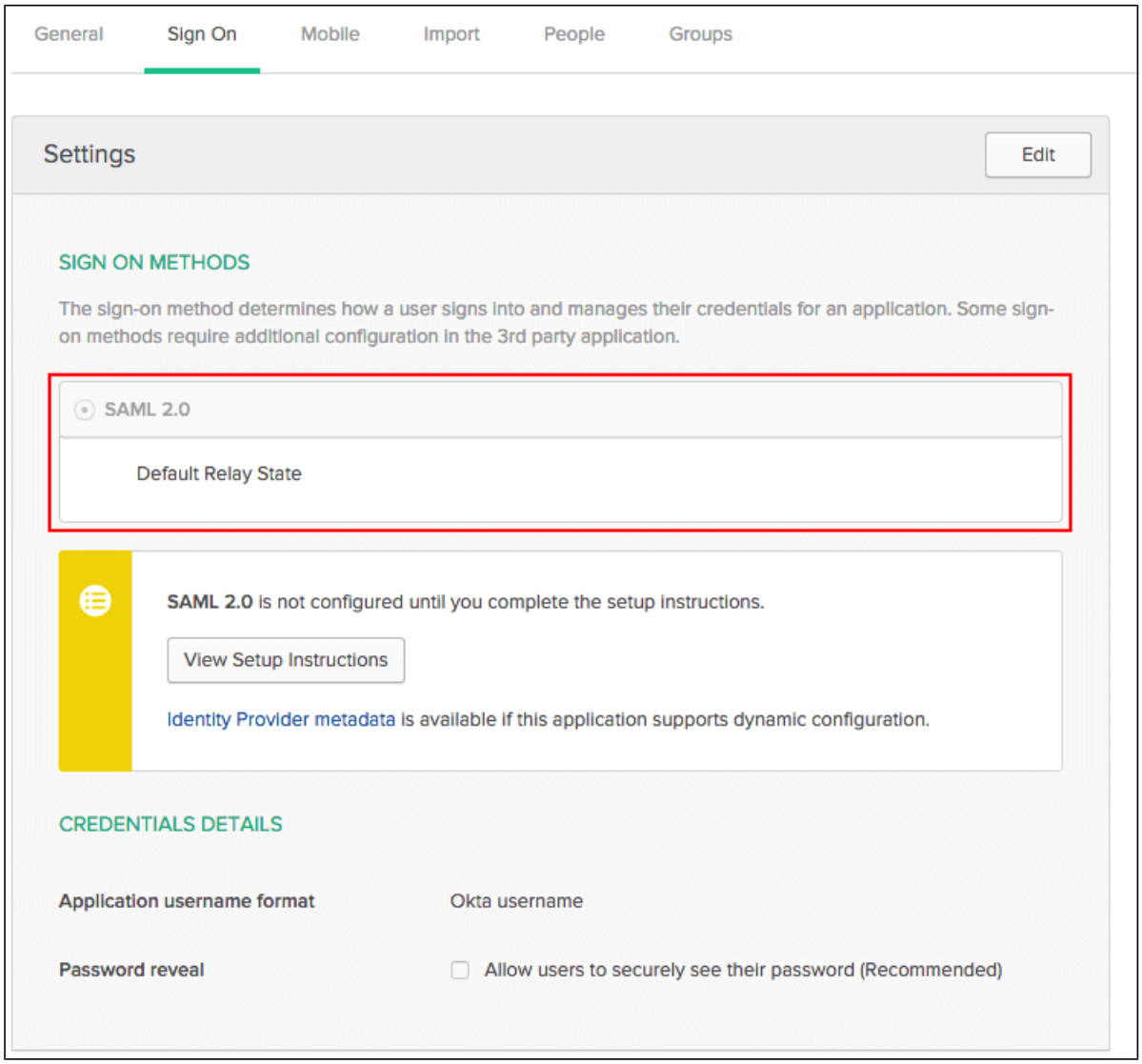
To configure Okta to designate Enterprise PKS as a service provider, you must have the following:

- An Okta Single-Sign On admin account
- An app with SAML 2.0 enabled in Okta

Configure SAML in Okta

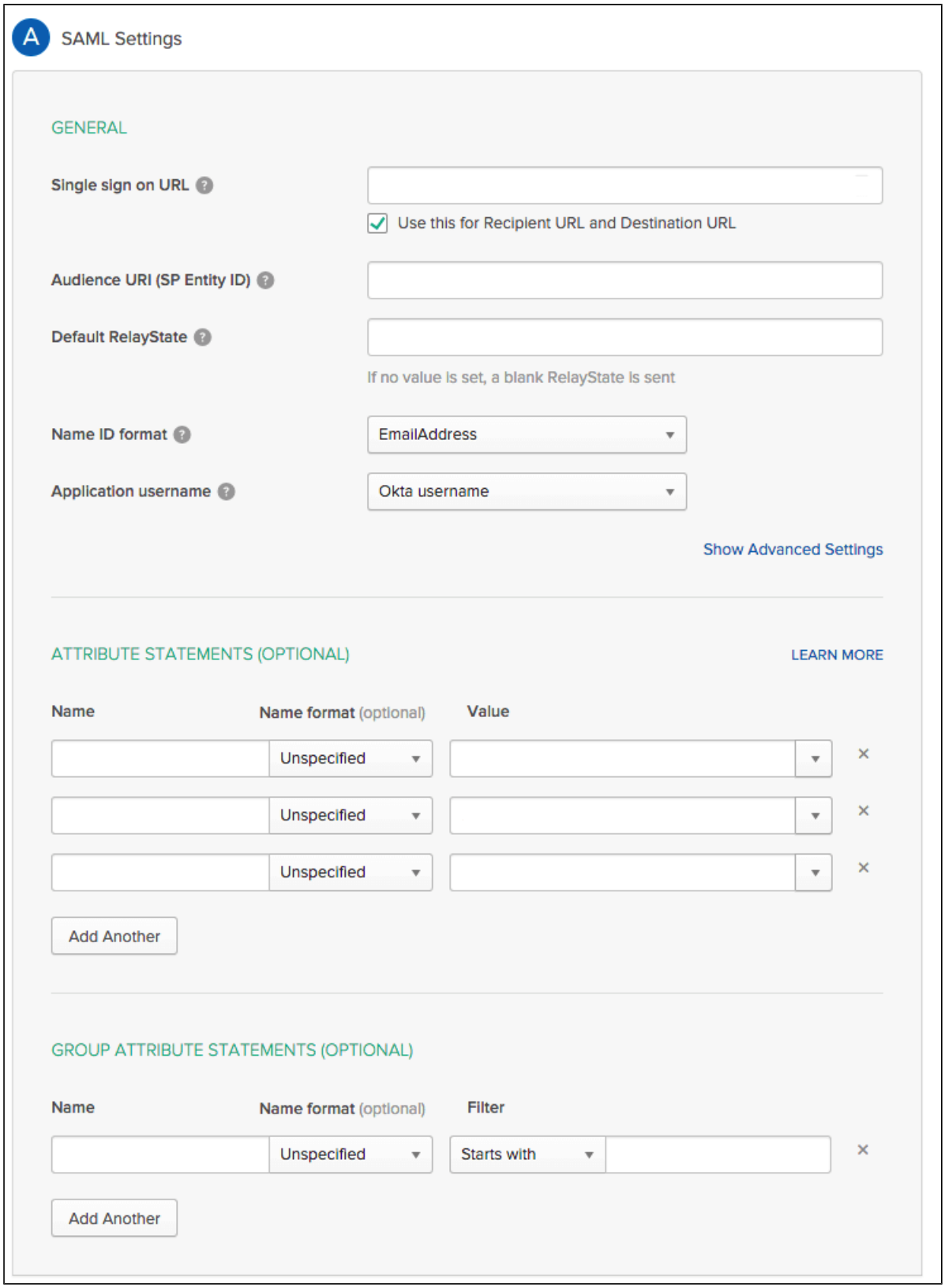
To configure Okta as a SAML identity provider for Enterprise PKS, do the following:

1. Log in to Okta as an admin.
2. Navigate to your app and click **Sign On**.
3. Under **Settings**, click **Edit**, and select **SAML 2.0**.



4. Click the **General** tab.

5. Under **SAML Settings**, click the **Edit** button followed by the **Next** button.



6. Configure the fields as follows:

Field	Instructions
Single sign on URL	Enter `https://PKS-API:8443/saml/SSO/alias/PKS-API:8443`. For example: `https://api.pks.example.com:8443/saml/SSO/alias/api.pks.example.com:8443`

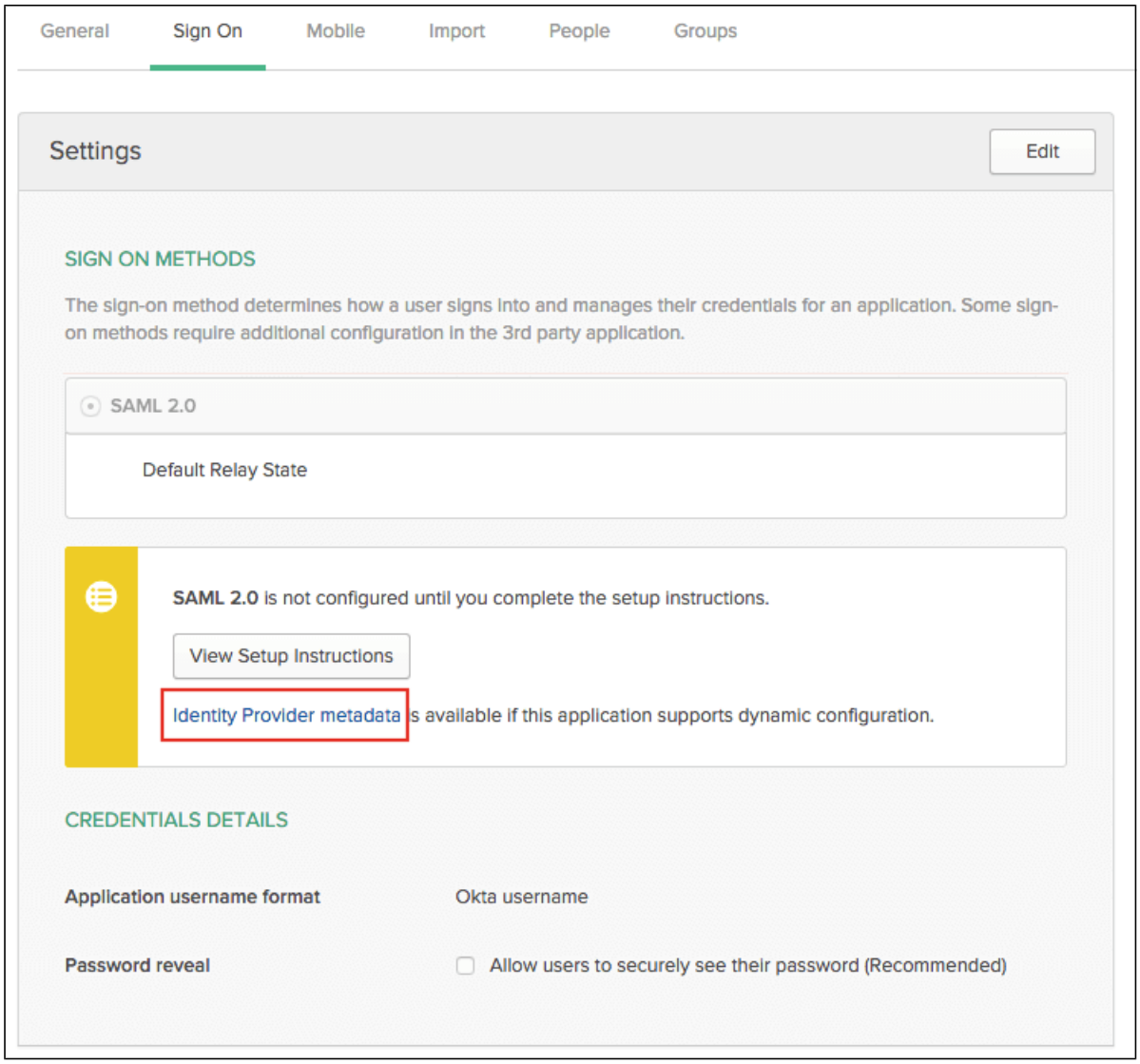
Use this for Recipient URL and Destination URL	Ensure this checkbox is enabled.
Audience URI (SP Entity ID)	Enter `PKS-API:8443`. For example: `api.pks.example.com:8443`
Name ID format	Select a name identifier format. By default, Enterprise PKS uses `EmailAddress`.
Attribute Statements	Enter any attribute statements that you want to map to users in the ID token. In Enterprise PKS you can define first name, last name, and email attributes.
Group Attribute Statements	Enter any group attribute statements that you want to map to users in the ID token. In Okta, these are groups that users belong to. You can use filters to define which groups are passed to Enterprise PKS.

Note: Pivotal recommends using the default settings for the fields that are not referenced in the above table.

7. Click the **Next** button followed by the **Finish** button.

8. (Optional) If you want to enable multi-factor authentication (MFA), you can add a SSO policy rule to your app. To enable MFA, do the procedure in Add Sign On policies for applications in the Okta documentation.

9. Click **Identity Provider metadata** to download the metadata, or copy and save the link address of the **Identity Provider metadata**.



10. Use the Okta metadata you retrieved in the above step to configure SAML in the Enterprise PKS tile. See Connecting Enterprise PKS to a SAML Identity Provider.

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

Configuring Azure Active Directory as a SAML Identity Provider

In this topic

Page last updated:

This topic explains how to configure single sign-on (SSO) between Azure Active Directory (Azure AD) and VMware Enterprise PKS.

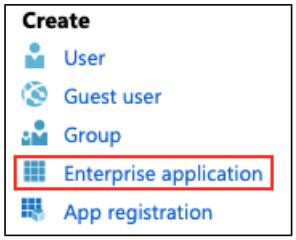
Prerequisites

To configure Azure AD to designate Enterprise PKS as a service provider, you must have an Azure AD Global Administrator account.

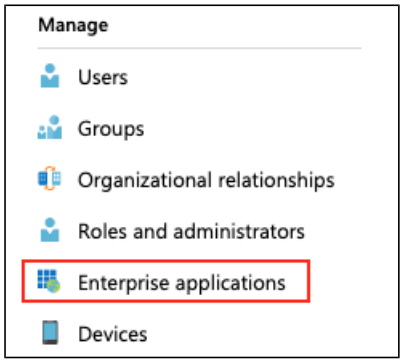
Configure SAML in Azure AD

To configure Azure AD as a SAML identity provider for Enterprise PKS, do the following:

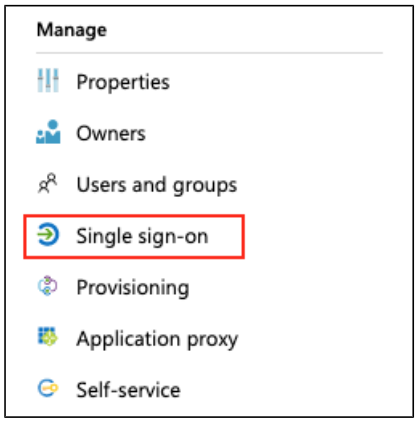
1. Log in to Azure AD as a Global Administrator.
2. Navigate to **Azure Active Directory**.
3. Under **Create**, click **Enterprise application**.



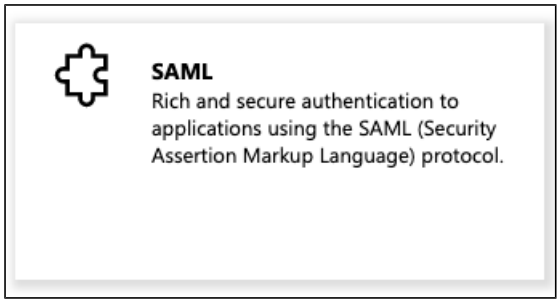
4. Under **Add your own app**, select **Non-gallery application**. Enter a **Name** and click **Add**.
5. Navigate to **Azure Active Directory** > **Enterprise applications**.



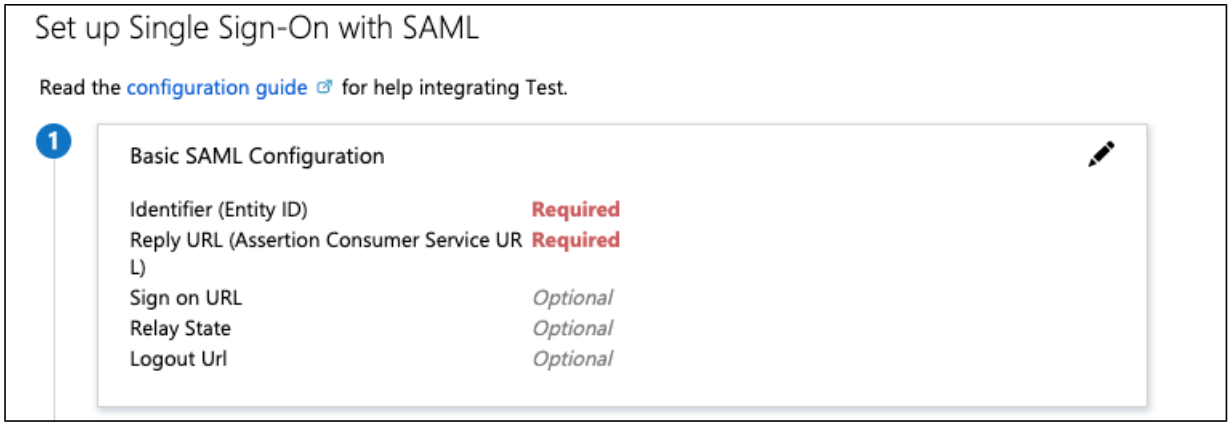
6. Click your app and then click **Single sign-on**.



7. Under **Select a single sign-on method**, select **SAML**.



8. Under **Set up Single Sign-On with SAML**, click the pencil icon for **Basic SAML Configuration**.



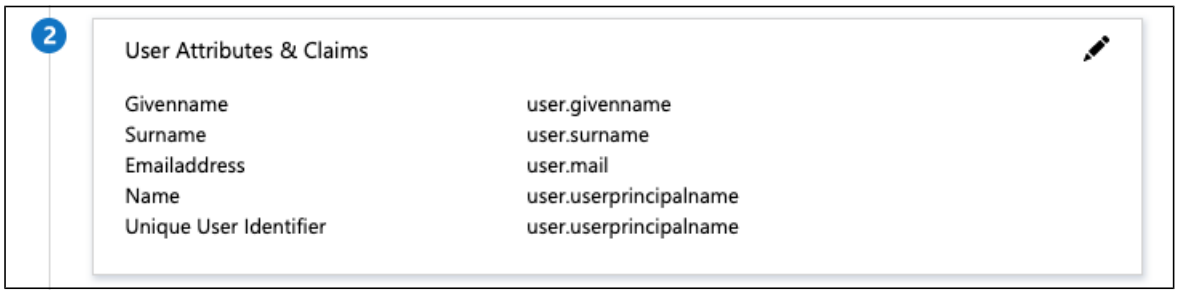
9. Configure the following fields:

Field	Instructions
Identifier (Entity ID)	Enter `PKS-API:8443`. For example: `api.pks.example.com:8443`
Reply URL	Enter `https://PKS-API:8443/saml/SSO/alias/PKS-API:8443`. For example: `https://api.pks.example.com:8443/saml/SSO/alias/api.pks.example.com:8443`

Sign on URL	Enter `https://PKS-API:8443/saml/SSO/alias/PKS-API:8443`. For example: https://api.pks.example.com:8443/saml/SSO/alias/api.pks.example.com:8443

Note: Pivotal recommends that you use the default settings for the fields that are not referenced in the above table.

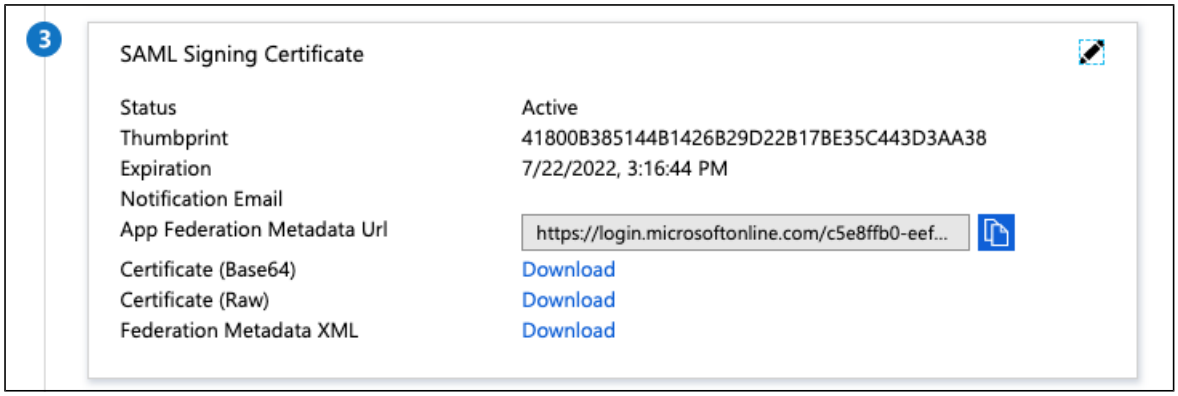
10. Click the pencil icon for **User Attributes & Claims**.



11. Configure your user attributes and claims by doing the procedures in How to: Customize claims issued in the SAML token for enterprise applications in the Microsoft Azure documentation. By default, Enterprise PKS uses the `EmailAddress` name identifier format.

12. Configure your group attributes and claims by doing the procedures in the Configure group claims for SAML applications using SSO configuration section of *Configure group claims for applications with Azure Active Directory (Public Preview)* in the Microsoft Azure documentation.

13. Under **SAML Signing Certificate**, copy and save the link address for **App Federation Metadata Url** or download **Federation Metadata XML**. You use the Azure AD metadata to configure SAML in the Enterprise PKS tile. For more information, see Connecting Enterprise PKS to a SAML Identity Provider.



Connecting Enterprise PKS to a SAML Identity Provider

In this topic

Page last updated:

This topic describes how to connect VMware Enterprise PKS to a SAML identity provider (IdP).

Overview

User Account and Authentication (UAA), the identity management service for Enterprise PKS, can authenticate users either through its internal user account store or external authentication mechanisms such as an LDAP server or a SAML IdP.

To enable an internal user account store for UAA, you select **Internal UAA** in the **Enterprise PKS** tile > **UAA**.

If you want to connect Enterprise PKS to a SAML IdP, you must integrate the UAA server with your SAML IdP by following the instructions in Integrate UAA with a SAML IdP below. This enables UAA to delegate authentication to your SAML IdP.

Prerequisites

Before you configure a SAML IdP in the Enterprise PKS tile, you must configure your IdP to designate Enterprise PKS as a service provider (SP).

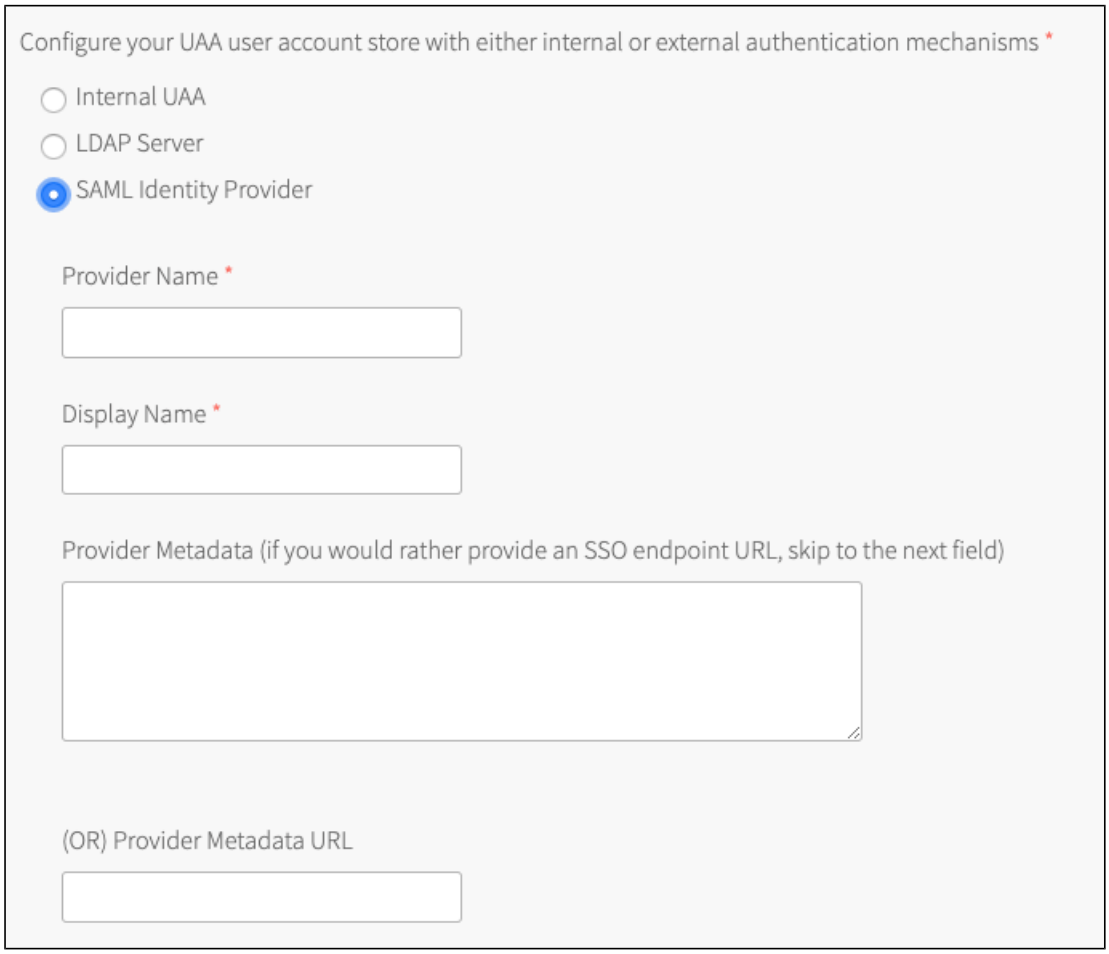
See the table below for information about industry-standard SAML IdPs and how to integrate them with Enterprise PKS:

Solution Name	Integration Guide
Okta Single Sign-On	Configuring Okta as a SAML Identity Provider
Azure Active Directory	Configuring Azure Active Directory as a SAML Identity Provider

Integrate UAA with a SAML IdP

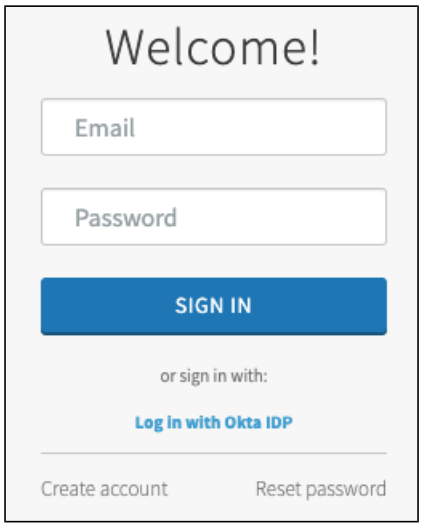
To integrate UAA with a SAML IdP:

1. In **Enterprise PKS** > **UAA**, under **Configure your UAA user account store with either internal or external authentication mechanisms**, select **SAML Identity Provider**.



2. For **Provider Name**, enter a unique name you create for the IdP. This name can include only alphanumeric characters, +, _, and -. You must not change this name after deployment because all external users use it to link to the provider.

3. For **Display Name**, enter a display name for your provider. This display name appears as a link on your Pivotal login page, which you can access at `https://PKS-API:8443/login`.



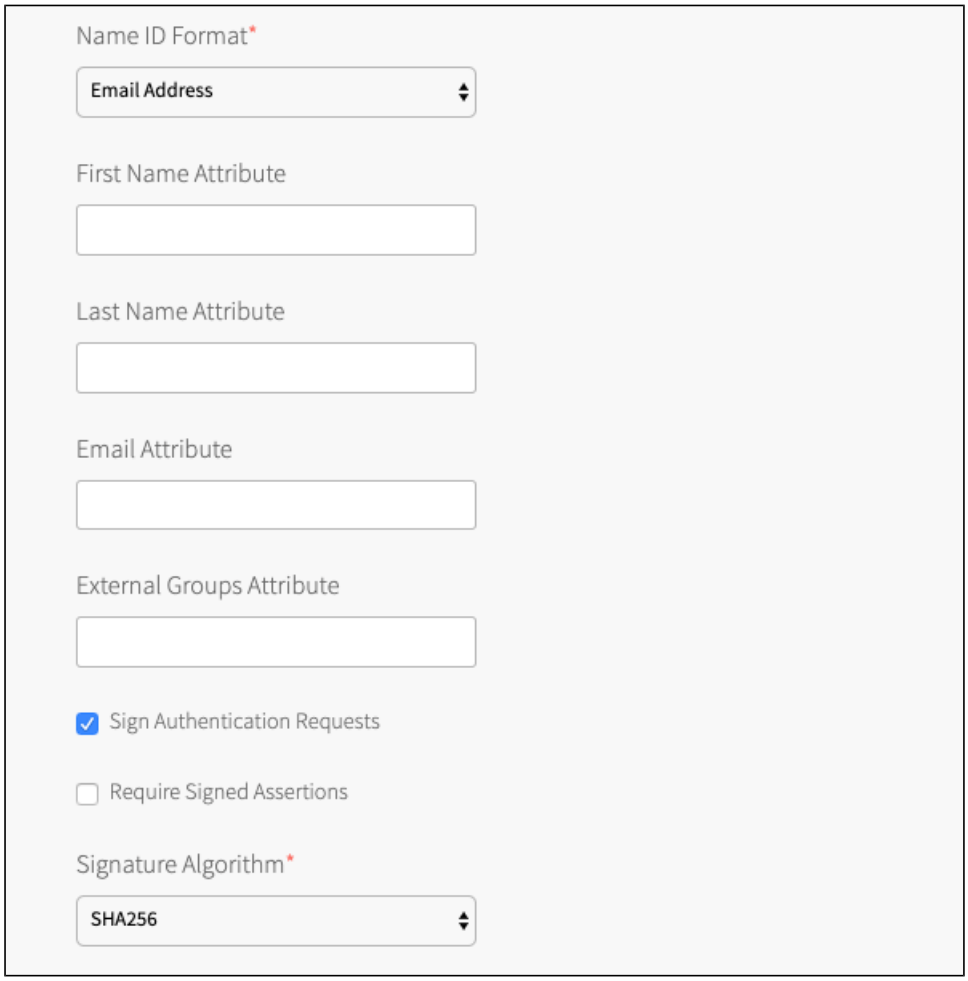
4. Retrieve the metadata from your IdP. You recorded your IdP metadata when you configured your IdP to designate Enterprise PKS as a SP. See Prerequisites above.

5. Enter your IdP metadata into either the **Provider Metadata** or the **Provider Metadata URL** fields:

- If your IdP exposes a metadata URL, enter it in **Provider Metadata URL**.
- If your IdP does not expose a metadata URL, paste the XML you retrieved into **Provider Metadata**.

> **Note:** Pivotal recommends that you use the Provider Metadata URL rather than Provider Metadata because the metadata can change. You need to select only one of the above configurations. If you configure both, your IdP defaults to the **(OR) Provider Metadata URL**.

6. For **Name ID Format**, select the name identifier format for your SAML IdP. This translates to `username` in Enterprise PKS. The default is `Email Address`.



7. For **First Name Attribute** and **Last Name Attribute**, enter the attribute names in your SAML database that correspond to the first and last names in each user record. This field is case sensitive.

8. For **Email Attribute**, enter the attribute name in your SAML assertion that corresponds to the email address in each user record, for example, `EmailID`. This field is case sensitive.

9. For **External Groups Attribute**, enter the attribute name in your SAML database for your user groups. This field is case sensitive. To map the groups from the SAML assertion to admin roles in PKS, see Grant Enterprise PKS Access to an External SAML Group in *Managing Enterprise PKS Users with UAA*.

10. By default, all SAML authentication requests from Enterprise PKS are signed. To change this, disable **Sign Authentication Requests** and configure your IdP to verify SAML authentication requests.

11. To validate the signature for the incoming SAML assertions, enable **Required Signed Assertions** and configure your IdP to

send signed SAML assertions.

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

13. Click **Save**.

Complete Your Tile Configuration

- If you do not need to configure any other settings in the Enterprise PKS tile, return to the Ops Manager Installation Dashboard and click **Review Pending Changes** > **Apply Changes**.
- If you need to configure any other settings in the Enterprise PKS tile, return to the *Installing Enterprise PKS* topic for your IaaS and follow the instructions for the pane you want to configure:
  - Installing Enterprise PKS on vSphere
  - Installing Enterprise PKS on vSphere with NSX-T
  - Installing Enterprise PKS on GCP
  - Installing Enterprise PKS on AWS
  - Installing Enterprise PKS on Azure

Next Steps

For information about creating Enterprise PKS roles and managing Kubernetes cluster access, see:

- Setting Up Enterprise PKS Admin Users for your IaaS
- Managing Cluster Access and Permissions

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

Managing Enterprise PKS Users with UAA

In this topic

Page last updated:

This topic describes how to manage users in VMware Enterprise PKS with User Account and Authentication (UAA).

Overview

UAA is the identity management service for Enterprise PKS. Enterprise PKS includes a UAA server, which is hosted on the PKS control plane VM.

To interact with the UAA server, you can use the UAA Command Line Interface (UAAC). You can either run UAAC commands from the Ops Manager VM or install UAAC on your local workstation.

UAA Scopes for Enterprise PKS Users

By assigning UAA scopes, you grant users the ability to create, manage, and audit Kubernetes clusters in Enterprise PKS.

A UAA admin user can assign the following UAA scopes to Enterprise PKS users:

- `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.
- `pks.clusters.admin.read`: Accounts with this scope can access any information about all clusters except for cluster credentials.

You can assign these scopes to individual users, external identity provider groups, or clients for automation purposes.

For more information about UAA scopes in Enterprise PKS, see UAA Scopes.

Prerequisites

Before managing users for Enterprise PKS, you must connect to the PKS control plane VM. To connect to the PKS control plane VM, you need one of the following:

- SSH access to the Ops Manager VM
- A machine that can connect to your PKS control plane VM

For instructions on how to connect to the PKS control plane, see Connect to the PKS Control Plane VM for your IaaS.

Log In as a UAA Admin

Before creating PKS users, you must log in to the UAA server as a UAA admin. To log in to the UAA server, do the following:

1. Retrieve the UAA management admin client secret:

   a. In a web browser, navigate to the Ops Manager **Installation Dashboard** and click the **Enterprise PKS** tile.
   b. Click the **Credentials** tab.
   c. Click **Link to Credential** next to **Pks Uaa Management Admin Client** and copy the value of `secret`.

2. Target your UAA server by running the following command:

   ```
   uaac target https://PKS-API:8443 --ca-cert CERTIFICATE-PATH
   ```

   Where:

   - `PKS-API` is the domain name of your PKS API server. You entered this domain name in the **Enterprise PKS** tile > **PKS API** > **API Hostname (FQDN)**.
   - `CERTIFICATE-PATH` is the path to your Ops Manager root CA certificate. Provide this certificate to validate the PKS API certificate with SSL.
     - If you are logged in to the Ops Manager VM, specify `/var/tempest/workspaces/default/root_ca_certificate` as the path. This is the default location of the root certificate on the Ops Manager VM.
     - If you downloaded the Ops Manager root CA certificate to your machine, specify the path where you stored the 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 ignore it because it has no impact.

3. Authenticate with UAA by running the following command:

   ```
   uaac token client get admin -s ADMIN-CLIENT-SECRET
   ```

   Where `ADMIN-CLIENT-SECRET` is your UAA management admin client secret that you retrieved in a previous step. The client username is `admin`.

Grant Enterprise PKS Access to an Individual User

To create a new UAA user with Enterprise PKS access, do the following:

1. If you are not logged in as the UAA admin, perform the steps in Log In as a UAA Admin.

2. Create a new user by running the following command:

   ```
   uaac user add USERNAME --emails USER-EMAIL -p USER-PASSWORD
   ```

For example:

```
$ uaac user add cody --emails cody@example.com -p password
```

Note: If your operator has configured Enterprise PKS to use a SAML identity provider, you must add `--origin SAML-ORIGIN` to the above command. `SAML-ORIGIN` is the domain name for your SAML identity provider. To find `SAML-ORIGIN`, click on the PKS tile, select **Settings** > **UAA** > **SAML**, and locate the `Provider Name`. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider.

3. Assign a PKS cluster scope to the new user by running the following command:

```
uaac member add UAA-SCOPE USERNAME
```

Where:

- `UAA-SCOPE` is one of the UAA scopes described in UAA Scopes for Enterprise PKS Users.
- `USERNAME` is the user that you created in the previous step.

For example:

```
$ uaac member add pks.clusters.admin cody
```

After you assign this scope, the user can create and manage Kubernetes clusters. For more information, see Managing Kubernetes Clusters and Workloads.

Grant Enterprise PKS Access to an External Group

Connecting Enterprise PKS to an external LDAP or SAML user store enables the UAA server to delegate authentication to existing enterprise user stores.

Note: When integrating UAA with an external identity provider, authentication within UAA becomes chained. UAA first attempts to authenticate with user credentials against the UAA user store before the external identity provider. For more information about integrating LDAP, 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 Enterprise PKS access to an external identity provider group, do one the following procedures:

- Grant Enterprise PKS Access to an External LDAP Group
- Grant Enterprise PKS Access to an External SAML Group

Grant Enterprise PKS Access to an External LDAP Group

To grant Enterprise PKS access to an external LDAP group, do the following:

1. If you are not logged in as the UAA admin, do the steps in Log In as a UAA Admin.

2. Assign a PKS cluster scope to all users in an LDAP group by running the following command:

```
uaac group map --name UAA-SCOPE GROUP-DISTINGUISHED-NAME
```

Where:

- `UAA-SCOPE` is one of the UAA scopes described in UAA Scopes for Enterprise PKS Users.
- `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.

Grant Enterprise PKS Access to an External SAML Group

To grant Enterprise PKS access to an external SAML group, do the following:

1. If you are not logged in as the UAA admin, do the steps in Log In as a UAA Admin.

2. Assign a PKS cluster scope to all users in a SAML group by running the following command:

```
uaac group map --name UAA-SCOPE SAML-GROUP --origin SAML-ORIGIN
```

Where:

- `UAA-SCOPE` is one of the UAA scopes described in UAA Scopes for Enterprise PKS Users.
- `SAML-GROUP` is name of your SAML identity provider group.
- `SAML-ORIGIN` is the domain name for your SAML identity provider. To find `SAML-ORIGIN`, click on the PKS tile, select **Settings** > **UAA** > **SAML**, and locate the `Provider Name`.

For example:

```
$ uaac group map --name pks.clusters.manage pks-devs --origin my-sso.example.com
```

Grant Enterprise PKS Access to a Client

To grant Enterprise PKS access to a client for a script or service automation, do the following:

1. If you are not logged in as the UAA admin, perform the steps in Log In as a UAA Admin.

2. Create a client with the desired scopes by running the following command:

```
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 one or more of the UAA scopes described in UAA Scopes for Enterprise PKS Users, 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
```

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

Managing Kubernetes Cluster Resources

Page last updated:

This section describes how to manage resources for Kubernetes clusters provisioned by VMware Enterprise PKS.

See the following topics:

- Load Balancing
- CPU and Memory
- Network Profiles (NSX-T Only)
- Compute Profiles and Host Groups (vSphere Only)

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

Load Balancing

Page last updated:

This section describes how to create and configure load balancers for VMware Enterprise PKS clusters. See the following topics:

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

---

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

Creating and Configuring a GCP Load Balancer for Enterprise PKS Clusters

In this topic

Page last updated:

This topic describes how to configure a Google Cloud Platform (GCP) load balancer for a Kubernetes cluster deployed by VMware Enterprise 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 Enterprise PKS tile that 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 Enterprise 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 Enterprise 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 Enterprise 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.

        > **Note:** If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

     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 that 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 Enterprise 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`.

   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.
   ```

   The `pks get-credentials` command creates a local `kubeconfig` that enables you to manage the cluster. For more information about the `pks get-credentials` command, see Retrieving Cluster Credentials and Configuration.

   > **Note:** If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

2. Run `kubectl cluster-info` to confirm you can access your cluster using the Kubernetes CLI.

See Managing Enterprise 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**.

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

Creating and Configuring an AWS Load Balancer for Enterprise PKS Clusters

In this topic

Page last updated:

This topic describes how to configure a Amazon Web Services (AWS) load balancer for your VMware Enterprise 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 Enterprise PKS deployment see Load Balancers in PKS.

You can use an AWS Enterprise PKS cluster load balancer to secure and facilitate access to a Enterprise PKS cluster from outside the network. You can also reconfigure your AWS Enterprise PKS cluster load balancers.

Using an AWS Enterprise 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 Enterprise PKS tile that you are installing.

> **Note**: This procedure uses example commands which you should modify to represent the details of your Enterprise 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 Enterprise PKS Clusters

In this topic

Page last updated:

This topic describes how to create and configure an Azure load balancer for your VMware Enterprise 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 Enterprise 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 Enterprise 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 Enterprise PKS deployed.
    g. **Resource group**: Select the resource group which has Enterprise PKS deployed.
    h. **Location**: Select the location group which has Enterprise 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 Enterprise PKS using Terraform, 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**.

CPU and Memory

Page last updated:

This section describes how to define and use resource quotas for Kubernetes clusters in VMware Enterprise PKS.

See the following topics:

- Managing Resource Usage with Quotas
- Viewing Usage Quotas

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

Managing Resource Usage with Quotas

In this topic

Page last updated:

> **warning**: This feature is a beta component and is intended for evaluation and test purposes only. Do not use this feature in a production environment. Product support and future availability are not guaranteed for beta components.

This topic describes how to restrict and review the usage of VMware Enterprise PKS resources by Enterprise PKS users.

Overview

As an Enterprise PKS administrator, you can set a limit on each user's total resource allocation within Enterprise PKS.

You manage resources in Enterprise PKS by defining quotas for individual users with the PKS API.

The `quotas` API endpoint allows you to restrict the total amount of memory and number of CPUs that a user can allocate in total across their deployed clusters.

In addition, you can limit the total number of clusters a user can provision within Enterprise PKS.

To review overall resource usage and for individual users, you access the PKS API `usages` endpoint.

> **Note**: Quota settings affect only non-admin user accounts. A quota applied to an admin user account is ignored.

Set up Your API Access Token

The curl commands in this topic use an access token environment variable to authenticate into the PKS API.

1. To export your access token into an environment variable, run the following command:

```
pks login -a PKS-API -u USER-ID -p 'PASSWORD' -k; \
export YOUR-ACCESS-TOKEN=$(bosh int ~/.pks/creds.yml --path /access_token)
```

Where:

- `PKS-API` is the FQDN of your PKS API endpoint. For example, `api.pks.example.com`.
- `USER-ID` is your Enterprise PKS user ID.
- `PASSWORD` is your Enterprise PKS password.
- `YOUR-ACCESS-TOKEN` is the name of your access token environment variable.

For example:

```
$ pks login -a pks.my.lab -u alana -p 'psswrdabc123...!' -k; \
export my_token=$(bosh int ~/.pks/creds.yml --path /access_token)
```

Note: If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

Manage Quotas

This section describes how to add, modify and delete user quotas.

Add a Quota

To enforce a quota on a specific user, run the following command:

```
curl -k -X POST \
-H "Authorization: Bearer $YOUR-ACCESS-TOKEN" \
-H "Content-Type: application/json" \
-d \
'{
   "owner": "USER-ID",
   "limit": {
    "cpu": MAX-CPU,
    "memory": MAX-MEM,
    "cluster": MAX-CLUSTER
    }
   }' \
https://PKS-API:9021/v1/quotas
```

Where:

- `YOUR-ACCESS-TOKEN` is your access token environment variable.
- `USER-ID` is the user account ID to enforce the quota restriction on.
- `MAX-CPU` is the maximum total amount of CPU resources that the user can allocate to containers and pods. If set to `0`, the user cannot create clusters.
- `MAX-MEM` is the maximum total amount of memory, in gigabytes, that the user can allocate to containers and pods. If set to `0`, the user cannot create clusters.
- `MAX-CLUSTER` is the maximum number of clusters that the user can provision. This value must greater than or equal to `1`.
- `PKS-API` is the FQDN of your PKS API server.

For example:

```
$ user=exampleuser
$ pks login -a pks.my.lab -u $user -p 'psswrdabc123...!' -k; export TOKEN=$(bosh int ~/.pks/creds.yml --path /access_token)
$ curl -k -X POST \
-H "Authorization: Bearer $TOKEN" \
-H "Content-Type: application/json" \
-d \
'{
   "owner": "cody",
   "limit": {
      "cpu": 4,
      "memory": 5,
      "cluster": 10
   }
  }' \
https://example.com:9021/v1/quotas
```

Modify an Existing Quota

To modify a specific user's existing quota, run the following command:

```
curl -k -X PATCH  \
-H "Authorization: Bearer $YOUR-ACCESS-TOKEN" \
-H "Content-Type: application/json" \
-d \
'{
   "owner": "USER-ID",
   "limit": {
     "cpu": MAX-CPU,
     "memory": MAX-MEM,
     "cluster": MAX-CLUSTER
     }
   }' \
https://PKS-API:9021/v1/quotas/USER-ID
```

Where:

- `YOUR-ACCESS-TOKEN` is your access token environment variable.
- `USER-ID` is the user account ID to enforce the quota restriction on.
- `MAX-CPU` is the maximum total amount of CPU resources that the user can allocate to containers and pods. If set to `0`, the user cannot create clusters.
- `MAX-MEM` is the maximum total amount of memory, in gigabytes, that the user can allocate to containers and pods. If set to `0`, the user cannot create clusters.
- `MAX-CLUSTER` is the maximum number of clusters that the user can provision. This value must greater than or equal to `1`.
- `PKS-API` is the FQDN of your PKS API server. For example, `api.pks.example.com`.

For example:

```
$ user=exampleuser
$ pks login -a pks.my.lab -u $user -p 'psswrdabc123...!' -k; export TOKEN=$(bosh int ~/.pks/creds.yml --path /access_token)
$ curl -k -X PATCH \
-H "Authorization: Bearer $TOKEN" \
-H "Content-Type: application/json" \
-d \
'{
   "owner": "cody",
   "limit": {

      "cpu": 2,
      "memory": 3,
      "cluster": 6
   }
 }' \
https://example.com:9021/v1/quotas/$user
```

Delete a Quota

To delete a specific user's existing quota, run the following command:

```
curl -k -X DELETE -H "Authorization: Bearer $YOUR-ACCESS-TOKEN" \
https://PKS-API:9021/v1/quotas/USER-ID
```

Where:

- `YOUR-ACCESS-TOKEN` is your access token environment variable.
- `PKS-API` is the FQDN of your PKS API server.
- `USER-ID` is the user account ID to enforce the quota restriction on.

For example:

```
$ user=exampleuser
$ pks login -a pks.my.lab -u $user -p 'psswrdabc123...!' -k; export TOKEN=$(bosh int ~/.pks/creds.yml --path /access_token)
$ curl -k -X DELETE -H "Authorization: Bearer $TOKEN" \
https://example.com:9021/v1/quotas/$user
{
  "body":"The quota owner named: \"exampleuser\" not found."
}
```

View Quotas

The PKS API `quotas` endpoint reports on resource usage quotas in the JSON format.

View Quotas for a Single User

To list the resource quota restrictions currently applied to a single user, run the following command:

```
curl -k -H "Authorization: Bearer $YOUR-ACCESS-TOKEN" \
https://PKS-API:9021/v1/quotas/USER-ID
```

Where:

- `YOUR-ACCESS-TOKEN` is your access token environment variable.
- `PKS-API` is the FQDN of your PKS API server.
- `USER-ID` is the user account ID to report on.

For example:

```
$ user=exampleuser
$ pks login -a pks.my.lab -u $user -p 'psswrdabc123...!' -k; export TOKEN=$(bosh int ~/.pks/creds.yml --path /access_token)
$ curl -k -H "Authorization: Bearer $TOKEN" \
https://example.com:9021/v1/quotas/$user
{
  "owner":"cody",
  "limit":{
      "cpu":2,
      "memory":1.0,
      "cluster": 6
  }
}
```

View All Quotas

To list all current resource and cluster quota restrictions, run the following command:

```
curl -k -H "Authorization: Bearer $YOUR-ACCESS-TOKEN" \
https://PKS-API:9021/v1/quotas
```

Where:

- `YOUR-ACCESS-TOKEN` is your access token environment variable.
- `PKS-API` is the FQDN of your PKS API server.

For example:

```
$ user=exampleuser
$ pks login -a pks.my.lab -u $user -p 'psswrdabc123...!' -k; export TOKEN=$(bosh int ~/.pks/creds.yml --path /access_token)
$ curl -k -H "Authorization: Bearer $TOKEN" \
https://example.com:9021/v1/quotas
[
  {
    "owner":"cody",
    "limit":{
        "cpu":2,
        "memory":1.0,
        "cluster": 6
    }
  }
]
```

Error Message When User Exceeds Cluster Quota

If a user has exceeded their set cluster creation quota, then the following error message appears when the user attempts to create a cluster.

Error: You do not have enough privileges to perform this action.
Please contact the PKS administrator.

View Usage

The PKS API `usages` endpoint returns resource usage per user in the JSON format.

View Resource Usage by User

To list the current resource usage of a single user, run the following command:

```
curl -k -H "Authorization: Bearer $YOUR-ACCESS-TOKEN" https://PKS-API:9021/v1/usages/USER-ID
```

Where:

- `YOUR-ACCESS-TOKEN` is your access token environment variable.
- `PKS-API` is the FQDN of your PKS API server.
- `USER-ID` is the user account ID whose resource utilization you want to view.

View All Resource Usage

To list the current resource utilization for all users and clusters, run the following command:

```
curl -k -H "Authorization: Bearer $YOUR-ACCESS-TOKEN" \
https://PKS-API:9021/v1/usages
```

Where:

- `YOUR-ACCESS-TOKEN` is your access token environment variable.
- `PKS-API` is the FQDN of your PKS API server.

For example:

```
$ user=exampleuser
$ pks login -a pks.my.lab -u $user -p 'psswrdabc123...!' -k; export TOKEN=$(bosh int ~/.pks/creds.yml --path /access_token)
$ curl -k -H "Authorization: Bearer $TOKEN" \
https://example.com:9021/v1/usages
[
  {
    "owner": "cody",
    "totals": {
      "cpu": 20,
      "memory": 52,
      "cluster": 2
    },
    "clusters": [
      {
        "name": "vsp1",
        "cpu": 12,
        "memory": 36
      }
    ]
  }
]
```

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

Viewing Usage Quotas

In this topic

Page last updated:

warning: This feature is a beta component and is intended for evaluation and test purposes only. Do not use this feature in a production environment. Product support and future availability are not guaranteed for beta components.

This topic describes how to review your resource usage and quotas in VMware Enterprise PKS using the Enterprise PKS API.

Overview

Your Enterprise PKS administrator might set a limit on the number of clusters you can provision and the resources, such as amount of memory and number of CPUs, that are allocated in total to any clusters you create and workloads you deploy.

The resource quota limitations are based on the total allocated size of the VM instances you create, not their actual utilization.

By using the PKS API, you can check the resource and cluster limitations that the administrator has assigned to you as well as review your current usage.

Set up Your API Access Token

The curl commands in this topic use an access token environment variable to authenticate to the PKS API endpoints.

1. To export your access token into an environment variable, run the following command:

   ```
   pks login -a PKS-API -u USER-ID -p 'PASSWORD' -k; \
   export YOUR-ACCESS-TOKEN=$(bosh int ~/.pks/creds.yml --path /access_token)
   ```

   Where:

   - `PKS-API` is the FQDN of your PKS API endpoint. For example, `api.pks.example.com`.
   - `USER-ID` is your Enterprise PKS user ID.
   - `PASSWORD` is your Enterprise PKS password.
   - `YOUR-ACCESS-TOKEN` is the name of your access token environment variable.

   For example:

   ```
   $ pks login -a pks.my.lab -u alana -p 'psswrdabc123...!' -k; \
   export my_token=$(bosh int ~/.pks/creds.yml --path /access_token)
   ```

> **Note:** If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

View Quotas

The PKS API `quotas` endpoint returns your resource usage and cluster quota in the JSON format.

To view your resource and cluster quota, run the following command:

```
curl -k -H "Authorization: Bearer $YOUR-ACCESS-TOKEN" \
https://PKS-API:9021/v1/quotas/USER-ID
```

Where:

- `YOUR-ACCESS-TOKEN` is your access token environment variable.
- `PKS-API` is the FQDN of your PKS API endpoint. For example, `api.pks.example.com`.
- `USER-ID` is your Enterprise PKS user ID.

For example:

```
$ user=exampleuser
$ pks login -a pks.my.lab -u $user -p 'psswrdabc123...!' -k; export TOKEN=$(bosh int ~/.pks/creds.yml --path /access_token)
$ curl -k -H "Authorization: Bearer $TOKEN" \
https://example.com:9021/v1/quotas/$user
{
  "owner":"cody",
  "limit":{
      "cpu":2,
      "memory":1.0,
      "cluster": 6
  }
}
```

View Usage

The PKS API `usages` endpoint reports your actual resource usage in the JSON format.

To view your current allocated resource usage, run the following command:

```
curl -k -H "Authorization: Bearer $YOUR-ACCESS-TOKEN" \
https://PKS-API:9021/v1/usages/USER-ID
```

Where:

- `YOUR-ACCESS-TOKEN` is your access token environment variable.
- `PKS-API` is the FQDN of your PKS API endpoint. For example, `api.pks.example.com`.
- `USER-ID` is your Enterprise PKS user ID.

For example:

```
$ user=exampleuser
$ pks login -a pks.my.lab -u $user -p 'psswrdabc123...!' -k; export TOKEN=$(bosh int ~/.pks/creds.yml --path /access_token)
$ curl -k -H "Authorization: Bearer $TOKEN" \
https://example.com:9021/v1/usages
[
  {
    "owner": "cody",
    "totals": {
      "cpu": 2,
      "memory": 1,
      "cluster": 2
    },
  }
]
```

Error Message When You Exceed Cluster Quota

If you exceed your set cluster creation quota, then the following error message appears when you attempt to create a cluster.

```
Error: 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.

Network Profiles (NSX-T Only)

Page last updated:

This section describes how to define and use network profiles for VMware Enterprise PKS clusters deployed on NSX-T with vSphere.

See the following topics:

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

Defining Network Profiles

In this topic

Page last updated:

This topic describes how to define network profiles for Kubernetes clusters provisioned with VMware Enterprise 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 service that is created when a Kubernetes cluster is provisioned.
Custom Node Networks	Customize Kubernetes Node Networks, including the IP addresses, subnet size, and routability.
Custom Pod Networks	Customize Kubernetes Pod Networks, including IP addresses, subnet size, and routability.
Custom Floating IP Pool	Specify a custom floating IP pool.
Security Groups Namespace	Specify an NSX-T Namespace Group where Kubernetes master nodes will be added to during cluster creation.
Edge Router Selection	Specify the NSX-T Tier-0 router where Kubernetes node and Pod networks will be connected to.
Nodes DNS Configuration	Specify one or more DNS servers for Kubernetes clusters.
DNS Lookup of Pre-Provisioned IPs	Configure DNS lookup of the Kubernetes API load balancer or ingress controller.
Load Balancer Configuration	Configure layer 4 TCP load balancer settings; use third-party load balancer.
Ingress Configuration	Configure layer 7 HTTP/S ingress controller settings; use third-party ingress controller.
FW Section Marking	Configure top or bottom section markers for explicit DFW rule placement.
Shared Tier-1 Router	Use a single Shared Tier-1 Router for each Kubernetes cluster.
NCP Logging	Configure NCP logging.

Network Profile Format

Network profiles are defined using JSON. Here is an example network profile that includes all available parameters.

Note: This example network profile is for illustration purposes only. It is not intended to be used as a template for network

profile definition.

```
{
  "name": "example-network-profile",
  "description": "Example Network Profile with All Available Parameters -- FOR ILLUSTRATION PURPOSES ONLY",
  "parameters": {
    "lb_size": "large",
    "pod_ip_block_ids": [
      "ebe78a74-a5d5-4dde-ba76-9cf4067eee55",
      "ebe78a74-a5d5-4dde-ba76-9cf4067eee56" ],
    "pod_subnet_prefix": 27,
    "pod_routable": true,
    "fip_pool_ids": [
      "e50e8f6e-1a7a-45dc-ad49-3a607baa7fa0",
      "ebe78a74-a5d5-4dde-ba76-9cf4067eee55" ],
    "t0_router_id": "5a7a82b2-37e2-4d73-9cb1-97a8329e1a90",
    "master_vms_nsgroup_id": "9b8d535a-d3b6-4735-9fd0-56305c4a5293",
    "node_ip_block_ids": [
      "2250dc43-63c8-4bb8-b8cf-c6e12ccfb7de", "3d577e5c-dcaf-4921-9458-d12b0e1318e6" ],
    "node_routable": true,
    "node_subnet_prefix": 20,
    "nodes_dns": [
      "8.8.8.8", "192.168.115.1", "192.168.116.1" ],
    "dns_lookup_mode": "API_INGRESS",
    "ingress_prefix": "ingress",
    "single_tier_topology": true,
    "infrastructure_networks": [
      "30.0.0.0/24",
      "192.168.111.0/24",
      "192.168.115.1" ],
    "cni_configurations": {
      "type": "nsxt",
      "parameters": {
        "nsx_lb": false,
        "nsx_ingress_controller": true,
        "x_forwarded_for": "insert",
        "ingress_ip": "192.168.160.212",
        "log_settings": {
          "log_level": "DEBUG",
        "ingress_persistence_settings": {
          "persistence_type": "cookie",
          "persistence_timeout": 1 },
        "max_l4_lb_service": 10,
        "l4_persistence_type": "source_ip",
        "l4_lb_algorithm": "weighted_round_robin",
        "top_firewall_section_marker":"section-id",
        "bottom_firewall_section_marker":"section-id"
      }
    }
  }
}
```

WARNING: None of the parameters defined under `cni_configurations` can be updated.

Network Profile Parameters

The network profile JSON accepts the following top-level parameters:

Parameter	Type	Description
`name`	String	Name of the network profile.

Parameter	Type	Description
`description`	String	Description of the network profile.
`parameters`	Map	One or more name-value pairs.
`lb_size`	String	Size of the NSX-T load balancer service: `small` (default), `medium`, and `large`.
`pod_ip_block_ids`	String	Array of Pod IP Block UUIDs.
`pod_subnet_prefix`	Integer	Size of the Pods IP Block subnet.
`pod_routable`	Boolean	Make routable the Pods subnet. Default is `false`.
`fip_pool_ids`	String	Array of floating IP pool UUIDs defined in NSX-T.
`t0_router_id`	String	Tenant Tier-0 Router UUID defined in NSX-T.
`master_vms_nsgroup_id`	String	Namespace Group UUID as defined in NSX-T.
`nodes_dns`	String	Array (up to 3) of DNS server IP addresses for lookup of Kubernetes nodes and pods.
`dns_lookup_mode`	String	DNS lookup mode for Kubernetes API load balancer (`API`) and ingress controller (`API_INGRESS`).
`ingress_prefix`	String	Ingress controller hostname prefix for DNS lookup.
`single_tier_topology`	Boolean	Use a single Tier-1 Router per cluster (shared). Default is `false`.
`infrastructure_networks`	String	Array of IP addresses and subnets for Node Networks for use with a Shared Tier-1 topology in a Multi-Tier-0 environment.
`cni_configurations`	Map	Map containing key-value pairs for configuring NCP (see table below).

The network profile JSON accepts the following parameters for `cni_configurations`:

Parameter	Type	Description
`type`	String	Only the constant `nsxt` is accepted.
`parameters`	Map	One or more name-value pairs for NCP settings.
`x_forwarded_for`	String	Use the same source IP for calling clients. Accepts `"insert"` and `"replace"`.
`nsx_lb`	Boolean	Use NSX-T layer 4 virtual server for each Kubernetes service of type LoadBalancer. Default is `true`.
`nsx_ingress_controller`	Boolean	Use NSX-T layer 7 virtual server as the ingress controller for the Kubernetes cluster. Default is `true`.
`ingress_ip`	String	IP address to use for the ingress controller.
`log_settings`	Map	Parameters for configuring NCP logging.
`log_level`	String	Accepted values: “INFO”, “WARNING”, “DEBUG”, “ERROR”, and “CRITICAL”.
`log_dropped_traffic`	Boolean	Log dropped firewall traffic. Default is `false`.
`ingress_persistence_settings`	String	Parameters for customizing Layer 7 persistence.
`persistence_type`	String	Specify the ingress persistence type: `none`, `cookie`, or `source_ip`.
`persistence_timeout`	Integer	Persistence timeout interval in seconds.
`max_l4_lb_service`	Integer	Limit the maximum number of layer 4 virtual servers per cluster. Minimum is `1`.

Parameter	Type	Description
`l4_persistence_type`	String	Connection stickiness based on `source_ip`.
`l4_lb_algorithm`	String	Layer 4 load balancer behavior: `round_robin` (default), `least_connection`, `ip_hash`, `weighted_round_robin`.
`top_firewall_section_marker`	String	UUID of the top `section-id` for the distributed firewall (DFW) section as defined in NSX-T.
`bottom_firewall_section_marker`	String	UUID of the bottom `section-id` for the distributed firewall (DFW) section as defined in NSX-T.

Network Profile Creation

After the network profile is defined in a JSON file, an Enterprise PKS administrator can create the network profile using the PKS CLI. The Kubernetes administrator can use the network profile when creating a cluster.

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.
```

For more information, see the Create and Use Network Profiles section of *Using Network Profiles (NSX-T Only)*.

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

Using Network Profiles (NSX-T Only)

In this topic

Page last updated:

This topic describes how to use network profiles for Kubernetes clusters provisioned with VMware Enterprise 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 Enterprise 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 an Enterprise 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 Enterprise 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.
```

Change the Network Profile for a Cluster

PKS supports changing the network profile for an already created cluster.

To change the network profile, you create a new network profile and then apply it to the existing cluster.

Currently you only can change the `pod_ip_block_ids` field as follows:

- Reorder the IP Block IDs in the array
- Add more IP Block IDs in the array

For more information on the `pod_ip_block_ids` field, see Network Profile Parameters. For more information on the supported use cases for the `pod_ip_block_ids` field, see Network Profile Use Cases.

To update the `pod_ip_block_ids` network profile field for a cluster, do the following:

1. Define a new network profile. In the `pod_ip_block_ids` field, reorder the IP Block IDs or add additional IP Block IDs. For more information on defining the network profile JSON file, see Defining Network Profiles.

   Note: If possible, start with the exact contents of the original network profile and update the `pod_ip_block_ids` field, as well as the `name` field. If it is not possible to obtain the original network profile, create a new network profile with a unique `name` and the original values in the `pod_ip_block_ids` field. Then reorder or add IPs as desired.

2. Create a new network profile as described in the Create a Network Profile section.

   Note: The name that you specify for the new network profile must be unique and differ from the previously assigned network profile. Otherwise, you receive an error when you try to create the network profile.

3. Run the following command to update the cluster with the new network profile:

   ```
   pks update-cluster CLUSTER-NAME --network-profile NEW-NETWORK-PROFILE-NAME
   ```

   Where:

   - `CLUSTER-NAME` is the name of the existing Kubernetes cluster
   - `NEW-NETWORK-PROFILE-NAME` is the name of the new network profile you want to apply to the cluster.

Network Profile Update Validation

There are strict validation rules for the `pks update-cluster --network-profile` command:

- If a field in the original network profile is empty, the system ignores the empty field even if the field is included in the new network profile.
- If the existing `pod_ip_block_ids` field contains the same entries as the new network profile, the `update-cluster --network-profile` operation passes validation.

- If a field in the existing network profile conflicts with a field in the new network profile, the system reports the conflict and fails the validation.
- If the field is empty in the new network profile, then the system ignores the field even if the field is not empty in the original network profile.

View NSX-T Network Details

You can use the following PKS CLI command to view NSX-T network details:

```
pks cluster CLUSTER-NAME --details
```

Where CLUSER-NAME is the name of a Kubernetes cluster provisioned by Enterprise PKS.

For example:

```
Name:                     some-cluster-name
Plan Name:                some-plan-name
UUID:                     7a161098-986a-4a3c-8b02-d21e2a523463
Last Action:              CREATE
Last Action State:        in progress
Last Action Description:  Instance update in progress
Kubernetes Master Host:   85da06e7-bd08-43b8-82bf-6fa946339356.internal
Kubernetes Master Port:   8443
Worker Nodes:             2
Kubernetes Master IP(s):  10.11.12.13, 10.20.30.40
Network Profile Name:

NSXT Network Details:
  Load Balancer Size                     (lb_size):                "small"
  Nodes DNS Setting                      (nodes_dns):              ["192.168.115.1"]
  POD IP addresses are routable [no-nat] (pod_routable):           false
  POD subnet prefix                      (pod_subnet_prefix):      24
  NS Group ID of master VMs              (master_vms_nsgroup_id):  ""
  Tier 0 Router identifier               (t0_router_id):           "1b9014c1-fdaa-49b2-b2c2-e5a7f6215200"
  Floating IP Pool identifiers           (fip_pool_ids):           ["21a65cd7-2162-4dc1-ad51-020e617f3595", "d33ec8ca-04ed-4e8e-9a05-af4a2b48d906"]
  Node IP block identifiers              (node_ip_block_ids):      ["59bc3d66-d5ca-4c02-b509-8b09c950f52c", "59bc3d66-d5ca-4c02-b509-8b09c950f52c"]
  POD IP block identifiers               (pod_ip_block_ids):       ["d33ec8ca-04ed-4e8e-9a05-af4a2b48d906"]
[ubuntu-lc(46041) Outgoing]$
```

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

Defining a Network Profile for Load Balancer Sizing

In this topic

Page last updated:

This topic describes how to size a load balancer using a network profile.

Load Balancer Sizing

When you deploy a Kubernetes cluster using Enterprise 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 this load balancer and use a `medium` or `large` load balancer for Kubernetes clusters.

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 Enterprise PKS:

- 1 global virtual server for the Kubernetes API which runs on the master nodes
- 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 controller resources

The number of virtual servers that you can run depends on the size of the load balancer, which in turn depends on the size of the NSX-T Edge Node hosting the load balancer service. See Scaling Load Balancer Resources in the NSX-T documentation. Because of the number of virtual servers required by Enterprise PKS, you can only use the large NSX Edge Node VM or the bare metal NSX Edge Node with Enterprise PKS.

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"
   }
}
```

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

Defining Network Profiles for Pod Networks

In this topic

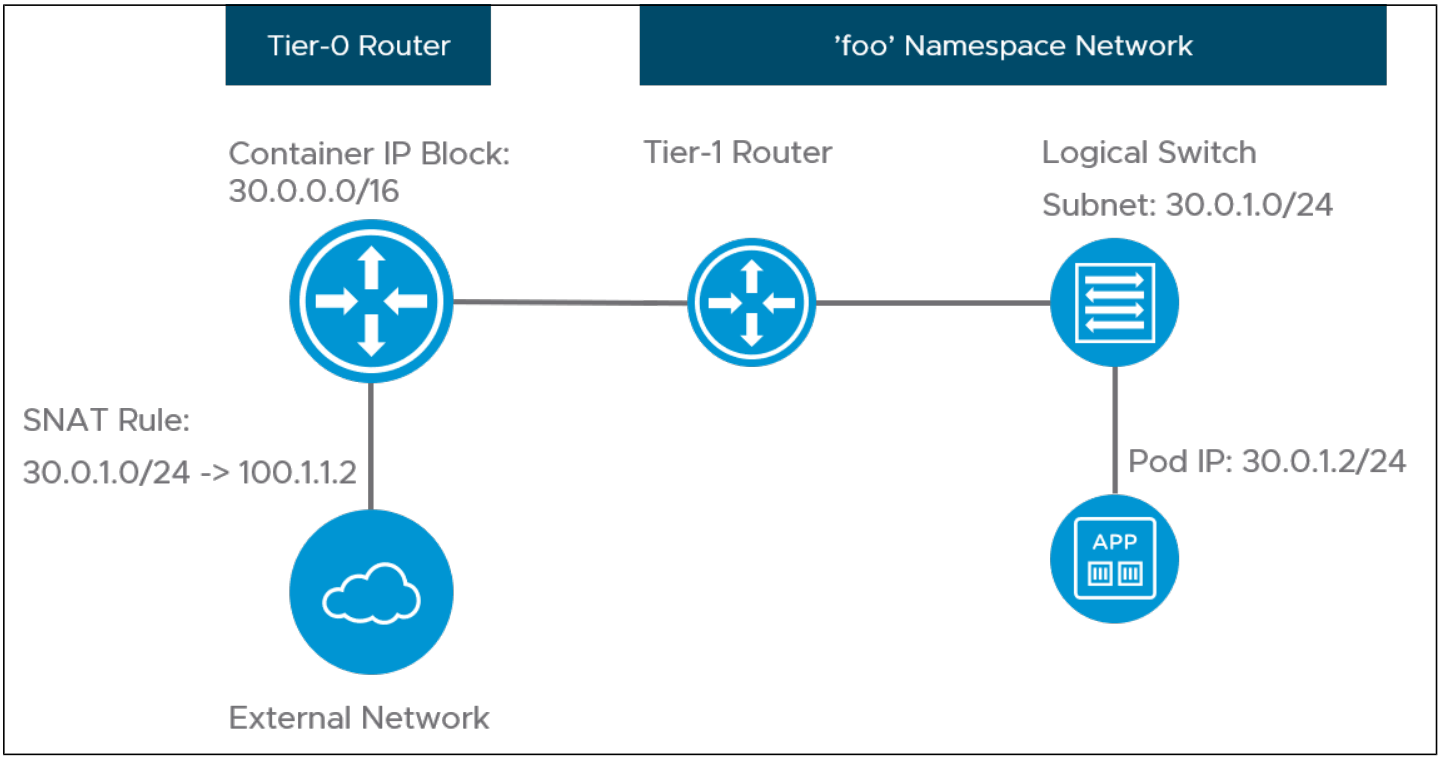
Page last updated:

This topic describes how to define network profiles for pod networks.

Custom Pod Networks

When you configure your NSX-T infrastructure for Enterprise 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 Enterprise 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 Enterprise PKS tile with a custom IP block. To use a custom pods network, do the following after you deploy Enterprise PKS:

1. Define a custom IP block in NSX-T. For more information, see Creating NSX-T Objects for Enterprise 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": "Example network profile with 2 non-routable pod networks",
  "name": "non-routable-pod",
  "parameters": {
    "pod_ip_block_ids": [
      "ebe78a74-a5d5-4dde-ba76-9cf4067eee55",
      "ebe78a74-a5d5-4dde-ba76-9cf4067eee56"
    ]
  }
}
```

Note: You cannot use the same Pod IP Block ID (UUID) that is specified in the PKS Tile. Create a new Pod IP Block ID (UUID) that is not referenced in the PKS Tile and use it to define a network profile.

Pod Subnet Prefix

Each time a Kubernetes namespace is created, a subnet from the pods IP block is allocated. The default 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 Enterprise 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 two custom Pod IP Block IDs:

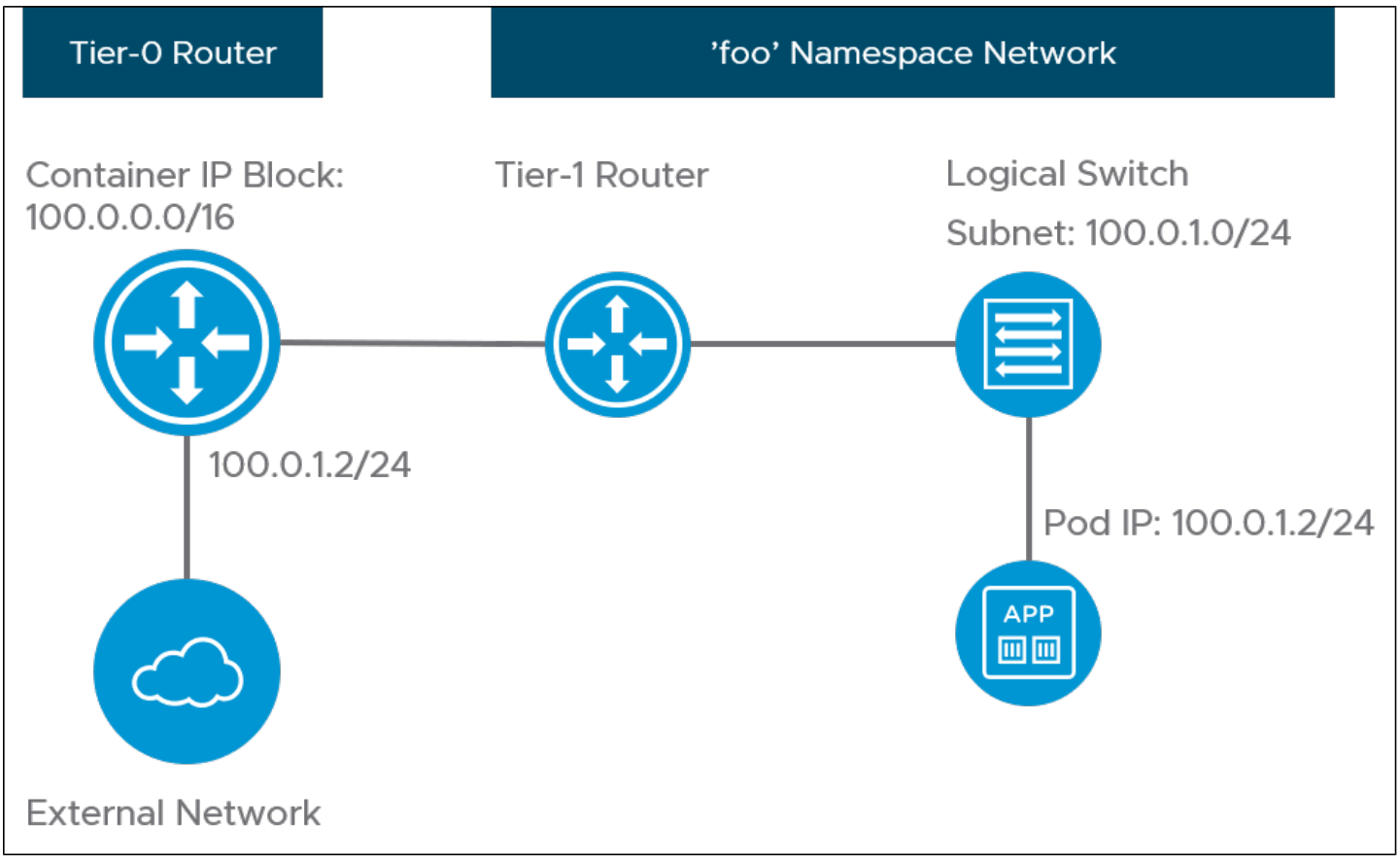
```
{
  "description": "Example network profile with 2 non-routable pod networks and custom prefix",
  "name": "non-routable-pod",
  "parameters": {
    "pod_subnet_prefix": 27,
    "pod_ip_block_ids": [
      "ebe78a74-a5d5-4dde-ba76-9cf4067eee55",
      "ebe78a74-a5d5-4dde-ba76-9cf4067eee56"
    ]
  }
}
```

Note: You cannot customize the size of the Pod IP Block ID (UUID) that is specified in the PKS Tile. To customize the size of the Pod subnet block you must create a new Pod IP Block ID (UUID) that is not referenced in PKS Tile and use it to define a network profile.

Note: The subnet size for a Pods IP Block must be consistent across all Network Profiles. Enterprise PKS does not support variable subnet sizes for a given IP Block.

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 Enterprise 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 Enterprise PKS:

1. Define a routable IP block in NSX-T. For more information, see Creating NSX-T Objects for Enterprise 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": "Example network profile with 2 routable pod networks and custom prefix",
  "name": "small-routable-pod",
  "parameters": {
    "pod_routable": true,
    "pod_subnet_prefix": 27,
    "pod_ip_block_ids": [
      "ebe78a74-a5d5-4dde-ba76-9cf4067eee55",
      "ebe78a74-a5d5-4dde-ba76-9cf4067eee56"
    ]
  }
}
```

Note: You cannot use the same Pod IP Block ID (UUID) that is specified in the PKS Tile. Create a new Pod IP Block ID (UUID) that is not referenced in PKS Tile and use it to define a network profile.

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

Customizing Node Networks

In this topic

Page last updated:

This topic describes how to define network profiles for customizing Kubernetes node networks provisioned with VMware Enterprise PKS on vSphere with NSX-T.

Configurable Node Network IP Blocks

The **Nodes IP Block** is used by Enterprise PKS to assign address space to Kubernetes nodes when new clusters are deployed or a cluster increases its scale. By default each Kubernetes cluster deployed by Enterprise PKS is allocated a /24 subnet, which allows up to 256 IP addresses to be assigned.

Using a network profile you can define one or more custom Node IP Block networks, specify the size of the nodes subnet, and specify if the network is routable.

Using the `node_ip_block_ids` parameter in a network profile, you can specify one or more **Nodes IP Blocks** for the Kubernetes node networks such that if one of IP block is exhausted, an alternative IP block can be used by Kubernetes clusters to create the Nodes subnet.

> **Note:** Specifying a new node subnet for an existing cluster is not supported. In other words, you cannot autoscale the node network for an existing cluster. For any new clusters created using a network profile with `node_ip_block_ids` configured, Enterprise PKS automatically creates a node subnet from one of the IP blocks that is available.

The `node_routable` boolean lets you specify if the Node network is routable or non-routable. This is the equivalent of enabling or disabling NAT mode in the PKS tile. If `"node_routable":false`, the Node network uses NAT mode. In this case you must make sure that Kubernetes nodes have access to BOSH and other PKS control plane components. See Creating the Enterprise PKS Management Plane for more information. If `"node_routable":true`, the IP address space must be an externally routable address block.

> **Note:** The default routable setting for the Node network is determined based on the selection made in the PKS tile. If **NAT mode** is selected, the Node network is non-routable. To override the default selection, provide the `node_routable` parameter in the network profile.

Depending on the size of the cluster (number of Kubernetes nodes), you can specify a subnet size using the `node_subnet_prefix` parameter that optimizes the use of network address space. This configuration is especially useful when the cluster nodes are using globally routable address space with the `node_routable` option set to “true”.

For example, if the Enterprise PKS administrator has configured the default in the PKS tile to be a Routable network for the*Nodes IP Block**, the Kubernetes cluster administrator can deploy Kubernetes cluster in the NAT'ed mode (non-routable) by specifying a network profile with an IP block that supports the NAT'ed address range.

> **Note:** The default size of the Node network is /24. If you want to use a different size, you must specify the `node_subnet_prefix` size.

```
nodes-network.json
{
  "description": "Configurable Nodes Network IP Block",
  "name": "network-profile_nodes-ip-block",
  "parameters": {
    "node_ip_block_ids": [
      "2250dc43-63c8-4bb8-b8cf-c6e12ccfb7de", "3d577e5c-dcaf-4921-9458-d12b0e1318e6"
    ],
    "node_routable":true,
    "node_subnet_prefix":20
  }
}
```

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

Defining Network Profiles for Custom Floating IP Pools

In this topic

Page last updated:

This topic describes how to define network profiles for custom floating IP pools.

Custom Floating IP Pool

To deploy Enterprise PKS to vSphere with NSX-T, you must define a Floating IP Pool in NSX Manager. IP addresses from the Floating IP Pool are used for SNAT IP addresses whenever a Namespace is created (NAT mode). In addition, IP addresses from the Floating IP Pool are assigned to load balancers automatically provisioned by NSX-T, including the load balancer fronting the PKS API server and load balancers for pod ingress. For more information, see the Plan Network CIDRs section of *Planning, Preparing, and Configuring NSX-T for Enterprise PKS*.

You can define a network profile that specifies a custom floating IP pool to use instead of the default pool specified in the Enterprise 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 Enterprise 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`.

Note: If you are using multiple Floating IP Pools within the same Tier-0 router, the Floating IP Pools cannot overlap. Overlapping Floating IP Pools are allowed across Tier-0 routers, but not within the same Tier-0 router.

Defining Network Profiles for Bootstrapping Security Groups

In this topic

Page last updated:

This topic describes how to define network profiles for Kubernetes clusters provisioned with VMware Enterprise PKS on vSphere with NSX-T.

Bootstrap Security Group

Most of the NSX-T virtual interface tags used by Enterprise 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 Enterprise PKS.

To address this bootstrap issue, Enterprise 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 Enterprise 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"
  }
}
```

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

Defining Network Profiles for Edge Router Selection

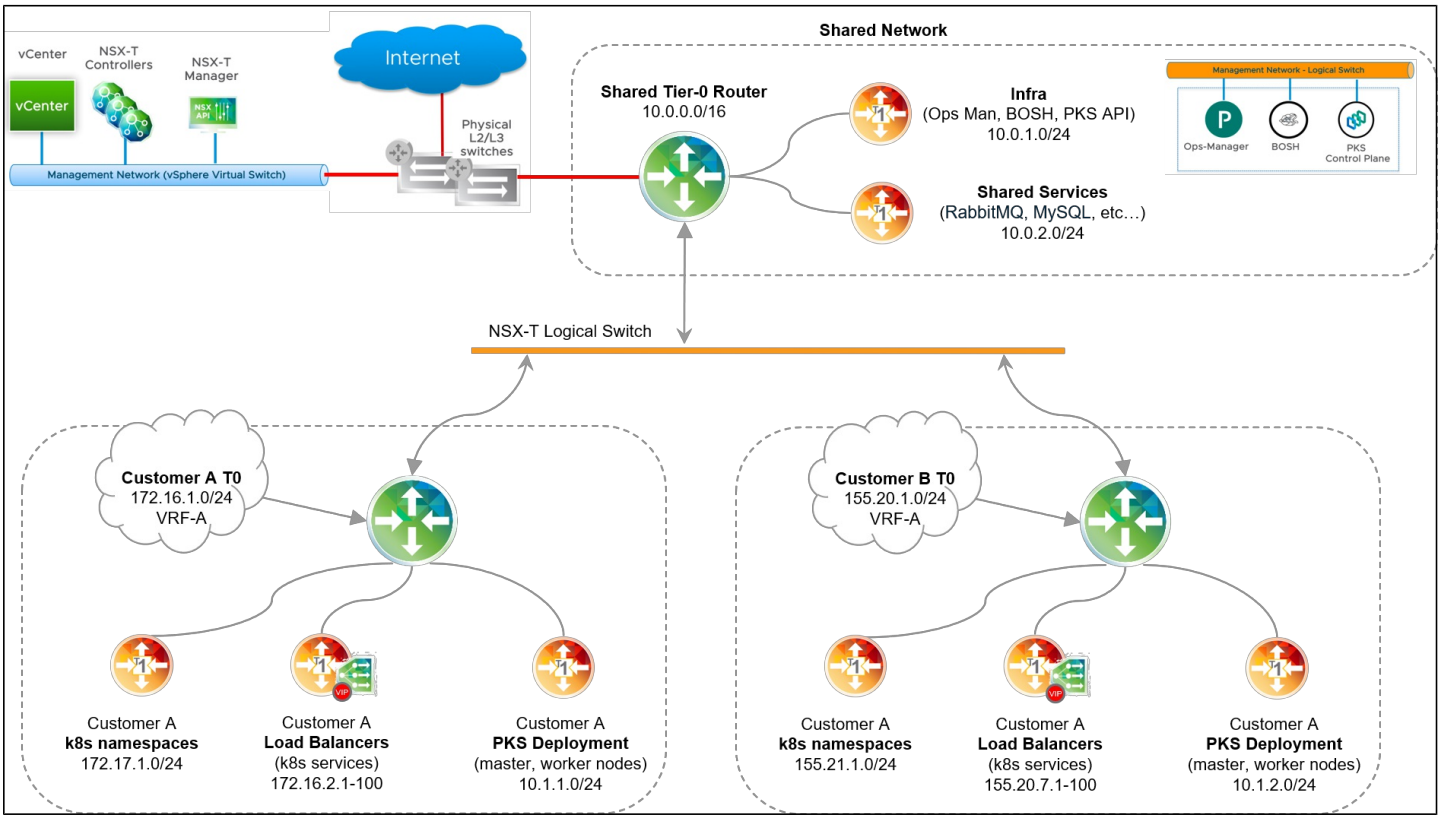
In this topic

Page last updated:

This topic describes how to define network profiles for Kubernetes clusters provisioned with VMware Enterprise PKS on vSphere with NSX-T.

Edge Router Selection

Using Enterprise 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.

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

Defining Network Profiles for Nodes DNS

In this topic

Page last updated:

This topic describes how to define network profiles for Kubernetes clusters provisioned with VMware Enterprise PKS on vSphere with NSX-T.

DNS Configuration for Kubernetes Clusters

You can specify multiple DNS entries in a Network Profile to override the **Nodes DNS** parameter configured in the PKS tile. In a multi-tenant environment, for example, each tenant can have a different set of DNS servers to do a DNS lookup.

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 Enterprise PKS tile. For more information, see Networking.

The `nodes_dns` field accepts an array with up to three elements. Each element must be a valid IP address of a DNS server. If you are deploying Enterprise PKS in a multi-tenant environment with multiple Tier-0 routers and a single PKS foundation (installation) shared across all the tenants, or if you have shared services that can be accessed by all Kubernetes clusters deployed across multiple Tier-0 routers, the first DNS server entered should be a shared DNS server. Subsequent DNS entries in the Network Profile can be 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.

Defining Network Profile for DNS Lookup of Pre-Provisioned IP Addresses

In this topic

Page last updated:

This topic describes how to define network profile for performing DNS lookup of the pre-provisioned IP addresses for the Kubernetes API load balancer and ingress controller.

About DNS Lookup of Pre-Provisioned IP Addresses

In an Enterprise PKS environment on NSX-T, when you provision a Kubernetes cluster using the command `pks create-cluster`, NSX-T creates a layer 4 load balancer that fronts the Kubernetes API server running on the master node(s). In addition, NCP creates two layer 7 virtual servers (HTTP and HTTPS) as front-end load balancers for the ingress resources in Kubernetes servers.

The IP addresses that are assigned to the API load balancer and ingress controller are derived from the floating IP pool in NSX-T. These IP addresses are not known in advance, and you have to wait for the IP addresses to be allocated to know what they are so you can update your DNS records.

If you want to pre-provision these IP addresses, you define a network profile to lookup the IP addresses for these components from your DNS server. In this way you can tell PKS what IP addresses to use for these resources when the cluster is created, and be able to have DNS records for them so FQDNs can be used.

DNS Lookup Parameters

Using the `dns_lookup_mode` parameter, you can define a network profile to specify the lookup mode: `API` or `API_INGRESS`. If the mode is `API`, PKS will perform a lookup of the pre-provisioned IP address for the Kubernetes API load balancer. If the mode is `API_INGRESS`, PKS will perform a lookup of the pre-provisioned IP addresses for the Kubernetes API load balancer and the ingress controller.

The IP addresses used must come from the floating IP pool. The floating IP pool, if not specified in the network profile, will come from the PKS tile configuration.

The DNS lookup, whether for the Kubernetes master(s) load balancer or the ingress controller, is performed in the Kubernetes master VM using the DNS server(s) configured in the PKS tile or the `nodes_dns` field in the network profile.

Example API Load Balancer Lookup

The following network profile, api.json, triggers a DNS lookup for the Kubernetes master node(s) IP address. In this example, a custom floating IP pool is specified, and DNS servers. If these parameters are not specified, the values in the PKS tile are used.

```
{
  "name": "example-network-profile",
  "description": "Network profile using API lookup mode",
  "parameters": {
    "nodes_dns": [
      "8.8.8.8", "192.168.115.1", "192.168.116.1"
      ],
    "fip_pool_ids": [
      "ENTER-FIP-POOL-ID1",
      "ENTER-FIP-POOL-ID2"
      ],
    "dns_lookup_mode": "API"
  }
}
```

Performing DNS Lookup of the Ingress Controller Using Network Profile

The following example network profile, api_ingress.json, triggers a DNS lookup for the Kubernetes master node(s) IP address and the ingress controller IP address.

```
{
  "name": "api_ingress",
  "description": "Network profile using API_INGRESS dns lookup mode",
  "parameters": {
     "fip_pool_ids": [
        "ENTER-FIP-POOL-ID1",
        "ENTER-FIP-POOL-ID2"
     ],
     "dns_lookup_mode": "API_INGRESS",
     "ingress_prefix": "ingress"
  }
}
```

Setting the Master Node IP Address on the Command Line

As an alternative to DNS lookup, you can specify a fixed IP address in the command line so that it will be used for the Kubernetes master node(s) load balancer.

Previously, to create a cluster, you were required to specify an external hostname for the cluster. For example:

```
$ pks create-cluster my-cluster --external-hostname example.hostname --plan small
```

Now you can specify the IP address for the load balancer that fronts the Kubernetes master node(s) using the `--external-hostname` or `-e` flag. For example:

```
$ pks create-cluster my-cluster -e 192.168.160.20 -p small
```

The IP address that you use must belong to a valid floating IP pool created in NSX-T.

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

Defining Network Profiles for the Layer 4 Load Balancer

In this topic

Page last updated:

This topic describes how to define network profile to configure the NSX-T Load Balancer for VMware Enterprise PKS.

About the NSX-T Load Balancer

The NSX-T Load Balancer is a logical load balancer that handles a number of functions using virtual servers and pools.

By default the following NSX-T virtual servers are deployed by Enterprise PKS:

- 1 per cluster TCP layer 4 virtual server for the Kubernetes API server
- 1 TCP layer 4 virtual server for **each** Kubernetes service of `type: LoadBalancer`
- 2 per cluster layer 7 virtual servers for the Kubernetes Ingress Controller (HTTP and HTTPS)

The TCP layer 4 virtual server that fronts the Kubernetes API server is always created, and it is not controlled by the parameters exposed in the network profile.

The `nsx_lb` parameter is used to control the TCP layer 4 virtual server that is provisioned for each Kubernetes service of type: LoadBalancer.

The `nsx_ingress_controller` parameter is used to control if NCP is used as the Ingress Controller for the Kubernetes cluster. By default when you define an ingress resource for a Kubernetes cluster, NCP instructs the NSX-T load balancer to provision 2 layer 7 virtual services (HTTP and HTTPS) as the Ingress Controller.

The `nsx_ingress_controller` parameter is subject to the `nsx_lb` parameter as described in the following table.

`nsx_lb setting`	**nsx_ingress_controller: true**	**nsx_ingress_controller: false**
nsx_lb: true	Use the NSX-T Layer 4 LoadBalancer and the NCP-provisioned Layer 7 Ingress Controller.	Use the NSX-T Layer 4 LoadBalancer and a third-party Ingress Controller, such as NGINX.
nsx_lb: false	Invalid configuration. You cannot disable the NSX-T Load Balancer and use NCP as the Ingress Controller. The network profile will fail validation.	Use a third-party load balancer and a third-party ingress controller.

NSX-T Load Balancer Parameters

The following table shows the parameters for configuring the NSX-T Load Balancer:

`cni_configurations`	Map	Map containing two key-value pairs: `type` and `parameters`.
`type`	Constant String	Only `nsxt` is accepted.
`parameters`	Map	Map containing one or more key-value pairs for NCP settings.
`nsx_lb`	Boolean	Flag to control if the NSX-T Load Balancer is used for the Kubernetes cluster.
`x_forwarded_for`	String	Sets the original client source IP in the request header. Default is none. Accepts `"insert"` and `"replace"`.
`max_l4_lb_service`	Integer	Limit the maximum number of layer 4 virtual servers per cluster. Minimum is `1`.
`l4_persistence_type`	String	Connection stickiness based on `source_ip` (only accepted value).
`l4_lb_algorithm`	String	Specify the layer 4 load balancer behavior. Accepts `"round_robin"` (default), `"least_connection"`, `"ip_hash"`, `"weighted_round_robin"`.

Example Network Profile for NSX-T Load Balancer Configuration

The following network profile is an example that illustrates the parameters exposed for configuring `ncp.ini`.

```
{
 "name": "network_profile",
 "description": "Example network profile for nsxt load balancer",
 "parameters" : {
   "cni_configurations": {
      "type": "nsxt",
      "parameters": {
        "nsx_lb": true,
        "x_forwarded_for": "replace",
        "max_l4_lb_service": 10,
        "l4_persistence_type": "source_ip",
        "l4_algorithm": "weighted_round_robin"
        }
      }
   }
}
```

NSX Load Balancer

The `nsx_lb` flag gives you the choice to of deploying the NSX-T Load Balancer or a third-party load balancer, such as Nginx. The `nsx_lb` parameter maps to the `use_native_loadbalancer` parameter in NCP.ini. The default setting is `true`. To use a third party load balancer, set this parameter to false.

For example:

```
{
 "name": "example_network_profile",
 "description": "nsx_lb and nsx_ingress_controller are enabled",
 "parameters": {
  "cni_configurations": {
   "type": "nsxt",
   "parameters": {
    "nsx_lb": false,
    "nsx_ingress_controller": false
   }
  }
 }
}
```

Since the `nsx_ingress_controller` is a component of the NSX-T Load Balancer, if you choose to use a third-party load balancer (`nsx_lb: false`), you must also explicitly disable the `nsx_ingress_controller`. See Defining Network Profile for the HTTP/S Ingress Controller.

Set Client Source IP Address

The X-Forwarded-For HTTP header field is used to identify the originating IP address of a client connecting to a web server through an HTTP proxy or load balancer.

In network profile, the `x_forwarded_for` parameter can be enabled to ensure that the client IP address will be set in the HTTP request header. The `x_forwarded_for` parameter is useful in situations where it is important to know the source IP address of the client request, such as for auditing purposes.

The `x_forwarded_for` parameter type is String that accepts `"insert"` and `"replace"`. Any other type will be rejected. Missing entry is accepted.

If set to “insert”, the client source IP will be appended (comma separated) to the existing set of client source IP addresses. If set to “replace”, any existing client source IP address will be replaced with the current client source IP address.

In the following example network profile, the source IP address of the client will be appended to the existing set of client source IP addresses.

```
{
 "name": "example-network-profile",
 "description": "x_forwarded_for insert",
 "parameters" : {
  "cni_configurations": {
    "type": "nsxt",
    "parameters": {
     "x_forwarded_for": "insert"
   }
  }
 }
}
```

In the following example network, the existing source IP address of the client will replace all other client source IP entries.

```
{
 "name": "example-network-profile",
 "description": "x_forwarded_for replace",
 "parameters" : {
   "cni_configurations": {
      "type": "nsxt",
      "parameters": {
        "x_forwarded_for": "replace"
     }
   }
  }
}
```

Max Number of Layer 4 Load Balancer Virtual Servers

The default NSX-T Load Balancer behavior is that auto-scaling is unlimited. This means that the number of layer 4 virtual servers that can be deployed is governed only by the capacity of the Edge Cluster where the load balancer service is deployed. As a result, in theory it is possible for a single Kubernetes cluster to use up all of the layer 4 virtual servers that the Edge Cluster can support.

The `max_l4_service` parameter sets the upper limit for the number of virtual servers that can be used by a Kubernetes cluster. You can use this parameter to limit the number of virtual servers that can be created per Kubernetes cluster.

The `max_l4_lb_service` data type is an integer. The value must be larger or equal to 1. Missing entry is accepted.

For example, the following network profile uses the `max_l4_lb_service` parameter to limit the number of layer 4 virtual servers to 100 per cluster:

```
{
 "name": "example_network_profile",
 "description": "max_l4_lb_service",
 "parameters" : {
   "cni_configurations": {
      "type": "nsxt",
      "parameters": {
        "max_l4_lb_service": 100
      }
   }
  }
}
```

Layer 4 Persistence Type

The `l4_persistence_type` is used to set connection stickiness based on `source_ip`.

The `l4_persistence_type` data type is string. The only accepted value is `source_ip`.

```
{
 "name": "example_network_profile",
 "description": "l4_persistence_type",
 "parameters": {
  "cni_configurations": {
   "type": "nsxt",
   "parameters": {
    "l4_persistence_type": "source_ip"
   }
  }
 }
}
```

Layer 4 Load Balancer Algorithm

The `l4_lb_algorithm` is used to set the algorithm type for the layer 4 NSX-T Load Balancer service.

The `l4_lb_algorithm` data type is string enumeration that accepts the following values: `"round_robin"` (default), `"least_connection"`, `"ip_hash"`, and `"weighted_round_robin"`.

For example, the following network profile specifies the `weighted_round_robin` as the load balancer algorithm.

```
{
 "name": "example_network_profile",
 "description": "l4_lb_algorithm",
 "parameters": {
  "cni_configurations": {
   "type": "nsxt",
   "parameters": {
    "l4_lb_algorithm": "weighted_round_robin"
   }
  }
 }
}
```

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

Defining Network Profile for the HTTP/S Ingress Controller

In this topic

Page last updated:

This topic describes how to define network profiles for Kubernetes clusters provisioned with VMware Enterprise PKS on vSphere with NSX-T.

About the NCP Ingress Controller

The NSX-T Load Balancer is a logical load balancer that handles a number of functions using virtual servers and pools.

By default the following NSX-T virtual servers are deployed by Enterprise PKS:

- 1 per cluster TCP layer 4 virtual server for the Kubernetes API server
- 1 TCP layer 4 virtual server for **each** Kubernetes service of `type: LoadBalancer`
- 2 per cluster layer 7 virtual servers for the Kubernetes Ingress Controller (HTTP and HTTPS)

The TCP layer 4 virtual server that fronts the Kubernetes API server is always created, and it is not controlled by the parameters exposed in the network profile.

The `nsx_lb` parameter is used to control the TCP layer 4 virtual server that is provisioned for each Kubernetes service of type: LoadBalancer.

The `nsx_ingress_controller` parameter is used to control if NCP is used as the Ingress Controller for the Kubernetes cluster. By default when you define an ingress resource for a Kubernetes cluster, NCP instructs the NSX-T load balancer to provision 2 layer 7 virtual services (HTTP and HTTPS) as the Ingress Controller.

The `nsx_ingress_controller` parameter is subject to the `nsx_lb` parameter as described in the following table.

`nsx_lb` setting	"nsx_ingress_controller": true (default)	"nsx_ingress_controller": false
"nsx_lb": true (default)	This means you want to use the NSX-T Layer 4 Load Balancer and the NCP-provisioned Layer 7 Ingress Controller.	This means you want to use the NSX-T Layer 4 Load Balancer and a third-party Ingress Controller, such as NGINX.
"nsx_lb": false	This is an invalid configuration because you cannot disable the NSX-T Load Balancer and use NCP as the Ingress Controller. The network profile will fail validation.	This means you want to use a third-party load balancer and a third-party ingress controller.

Ingress Parameters

The following table shows the ingress parameters for configuring NCP:

Parameter	Type	Description
`name`	String	User-defined name of the network profile.
`description`	String	User-defined description for the network profile.
`parameters`	Map	One or more name-value pairs.
`cni_configurations`	Map	Map containing two key-value pairs: `type` and `parameters`.
`type`	Constant String	Only `nsxt` is accepted.
`parameters`	Map	Map containing one or more key-value pairs for NCP settings.
`nsx_ingress_controller`	Boolean	Flag to control if NCP is used as the Ingress Controller for the Kubernetes cluster.
`ingress_ip`	String	IP address to use for ingress controller load balancer.
`ingress_persistence_settings`	Map	Holds parameters for customizing Layer 7 persistence.
persistence_type	String	Valid values are `cookie` or `source_ip`. An empty value is not accepted.
`persistence_timeout`	Integer	Value that is equal to `1` or larger. Empty value is not accepted.

Example Network Profile for Ingress

The following network profile is an example that illustrates the parameters exposed for configuring the NSX Ingress Controller.

```
{
 "name": "ncp_network_profile",
 "description": "Example network profile for ingress controller",
 "parameters" : {
   "cni_configurations": {
      "type": "nsxt",
      "parameters": {
        "nsx_ingress_controller": false,
        "ingress_ip": "192.168.160.212",
        "ingress_persistence_settings": {
          "persistence_type": "cookie",
          "persistence_timeout": 1
          }
        }
      }
   }
}
```

NSX Ingress Controller

NCP depends on the NSX-T Load Balancer to fulfill its role as an Ingress Controller. To use a third-party ingress controller, such as the NGINX Ingress Controller, set the `nsx_ingress_controller` to `false`.

For example:

```
{
 "name": "example_network_profile",
 "description": "Use the nsx_lb with a 3rd party ingress controller",
 "parameters": {
  "cni_configurations": {
   "type": "nsxt",
   "parameters": {
    "nsx_lb": true,
    "nsx_ingress_controller": false
   }
  }
 }
}
```

The following network profile uses a third party load balancer and a third-party ingress controller.

```
{
 "name": "example_network_profile",
 "description": "Use the nsx_lb with a 3rd party ingress controller",
 "parameters": {
  "cni_configurations": {
   "type": "nsxt",
   "parameters": {
    "nsx_lb": false,
    "nsx_ingress_controller": false
   }
  }
 }
}
```

The following network profile is invalid; you cannot disable the `nsx_lb` and use the NCP Ingress Controller.

```
{
 "name": "example_network_profile",
 "description": "Use the nsx_lb with a 3rd party ingress controller",
 "parameters": {
  "cni_configurations": {
   "type": "nsxt",
   "parameters": {
    "nsx_lb": false,
    "nsx_ingress_controller": true
   }
  }
 }
}
```

Ingress IP

The `ingress_ip` parameter instructs NCP to create an ingress virtual server with the given IP address.

The `ingress_ip` parameter type is a string that accepts any valid IP address. Missing entry is accepted.

Example network profile for `ingress_ip`:

```
{
 "name": "example-network-profile",
 "description": "ingress_ip",
 "parameters" : {
   "cni_configurations": {
      "type": "nsxt",
      "parameters": {
        "ingress_ip": "192.168.160.212"
      }
   }
 }
}
```

An invalid IP address is rejected with an invalid parameter value error. For example, the following network profile parameters cannot be parsed becase the 'ingress_ip': '192.168.460.212' address is an invalid IP address.

```
{
 "name": "example-network-profile",
 "description": "ingress_ip-ERROR",
 "parameters" : {
   "cni_configurations": {
      "type": "nsxt",
      "parameters": {
        "ingress_ip": "192.168.460.212"
      }
   }
 }
}
```

The following network profile cannot be parsed because the JOSN input is invalid (not a string).

```
{
 "name": "example-network-profile",
 "description": "ingress_ip-ERROR",
 "parameters" : {
   "cni_configurations": {
      "type": "nsxt",
      "parameters": {
        "ingress_ip": 192.168.160.212
      }
   }
 }
}
```

Ingress Persistence Settings

The `ingress_persistence` parameter lets you customize layer 7 persistence for Kubernetes services.

The `ingress_persistence_settings` parameter is a map that supports two keys: `persistence_type` and `persistence_timeout`. These two keys are correlated and must be set/unset at the same time. If `persistence_type` and `persistence_timeout` are not both specified, the network profile fails validation.

Parameter	Data type	Description
`persistence_type`	String	Valid values are `cookie` or `source_ip`. An empty value is not accepted.

Parameter	Data type	Description
`persistence_timeout`	Integer	Value that is equal to `1` or larger. Empty value is not accepted.

Example 1: network profile for `ingress_persistence_settings`:

```
{
 "name": "example_network_profile",
 "description": "ingress_persistence_settings",
 "parameters" : {
   "cni_configurations": {
      "type": "nsxt",
      "parameters": {
        "ingress_ip": "192.168.160.212"
        "ingress_persistence_settings": {
         "persistence_type": "cookie",
         "persistence_timeout": 1
         }
      }
   }
 }
}
```

Example 2: network profile for `ingress_persistence_settings`:

```
{
 "name": "example_network_profile",
 "description": "ingress_persistence_settings",
 "parameters" : {
   "cni_configurations": {
      "type": "nsxt",
      "parameters": {
        "ingress_ip": "192.168.160.212"
        "ingress_persistence_settings": {
         "persistence_type": "source_ip",
         "persistence_timeout": 100
         }
      }
   }
 }
}
```

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

Defining Network Profile for Distributed Firewall Section Marking

In this topic

Page last updated:

This topic describes how to define network profiles to create markers for NSX-T distributed firewall (DFW).

About DFW Section Markers

NSX-T applies distributed firewall (DFW) rules on ESXi transport nodes to control east-west traffic. Edge firewall (EFW) rules run on Edge transport nodes and control north-south traffic.

When a Kubernetes developer creates a network policy, NCP translates that policy into a DFW rule. “Allow” rules are placed at the top. “Deny” or “Drop” rules placed at the bottom, after the allow rules. All “Allow” rules are created above all “Deny” rules. This means the “Allow” rules take higher precedence than “Deny” rules.

NCP does not place the network policy DFW rule in any particular order in the stack of allow and deny rules. If multiple “Allow” rules are created, they are executed in chronological order: the previously created “Allow” rule takes precedence over the recently created one. This may not be the ordering that network administrators want, or it may disrupt the presumed ordering. Similarly, if a network policy results in an NCP-defined deny rule, it is placed at the top of the deny/drop stack. Again, this may not be the desired location.

With the top section marker, the operational rules can be created in the top section, NCP will create any rules when Kubernetes network policies are created always below this top section marker. Thus, the operational rules created will not be over-ridden by developers. Drop/deny rules operate in the bottom section. You can define a bottom section marker so that drop/deny rules created by NCP do not displace allow rules defined.

Top Firewall Section Marker

Using the `top_firewall_section_marker`, the operational rules can be created in the top section, NCP will create any rules when Kubernetes network policies are created always below this top section marker. Thus, the operational rules created will not be over-ridden by developers.

```
{
"name": "Example network profile",
"description": "Network profile to enable DFW section marking",
"parameters": {
  "cni_configurations":
  {
    "type":"nsxt",
    "parameters": {
      "top_firewall_section_marker":"section-id",
      "bottom_firewall_section_marker":"section-id"
    }
  }
}
}
```

Bottom Firewall Section Marker

Drop/deny rules operate in the bottom section. You can define a `bottom_firewall_section_marker` so that drop/deny rules created by NCP do not displace existing rules.

```
{
 "name": "Example network profile",
 "description": "Network profile to enable DFW section marking",
 "parameters": {
  "cni_configurations":
  {
    "type":"nsxt",
    "parameters": {
     "top_firewall_section_marker":"section-id",
     "bottom_firewall_section_marker":"section-id"
    }
  }
 }
}
```

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

Defining Network Profiles for NCP Logging

In this topic

Page last updated:

This topic describes how to define network profiles for logging NCP configurations.

About Logging for NCP Configurations

VMware Enterprise PKS provides network profile parameters for logging `ncp.ini` configurations.

Parameters for NCP Logging

The parameter `cni_configurations` is a map with two keys: `type` and `parameters`. The following table shows the parameters for configuring NCP:

Parameter	Type	Description
`name`	String	User-defined name of the network profile.
`description`	String	User-defined description for the network profile.
`parameters`	Map	One or more name-value pairs.
`cni_configurations`	Map	Map containing two key-value pairs: `type` and `parameters`.
`type`	Constant String	Only `nsxt` is accepted.
`parameters`	Map	Map containing one or more key-value pairs for NCP settings.
`log_settings`	Map	Holds parameters for configuring NCP logging.
`log_level`	String Enumeration	“INFO”, “WARNING”, “DEBUG”, “ERROR”, “CRITICAL”
`log_dropped_traffic`	Boolean	Default is `false`. Set to `true` to log dropped firewall traffic.

Example Network Profile for NCP Logging

The following network profile is an example that illustrates the parameters exposed for NCP logging.

```
{
 "name": "ncp_network_profile",
 "description": "Example network profile for NCP logging and error handling",
 "parameters" : {
   "cni_configurations": {
      "type": "nsxt",
      "parameters": {
        "log_settings": {
         "log_level": "WARNING",
         "log_dropped_traffic": true
        }
      }
      }
   }
}
```

Log Settings

The parameter `log_settings` is a map that supports two keys: `log_level` and `log_dropped_traffic`.

Log Level

The `log_level` parameter type is a string. The `log_level` value is an enumeration that supports the following values:

- “INFO”
- “WARNING”
- “DEBUG”
- “ERROR”
- “CRITICAL”

Any other value results in an error.

The value is set for three *ncp.ini* keys: `coe.nsxlib_loglevel`, `coe.loglevel`, and `k8s.loglevel`. The default log levels for these keys are as follows:

ncp.ini key	Default log level
`coe.nsxlib_loglevel`	INFO
`coe.loglevel`	NONE
`k8s.loglevel`	NONE

Log Dropped Traffic

The `log_dropped_traffic` type is a boolean: `true` or `false`. Any other type is rejected, such as “true”. Missing entry is accepted. Enabling this parameter is used in distributed firewall configurations to log the traffic for dropped rules.

Example network profile for logging:

```
{
 "name": "example-network-profile",
 "description": "log_settings",
 "parameters" : {
   "cni_configurations": {
      "type": "nsxt",
      "parameters": {
        "log_settings": {
          "log_level": "DEBUG",
          "log_dropped_traffic": true
        }
      }
   }
 }
}
```

enable_err_crd

The `enable_err_crd` parameter provides a mechanism of reporting NSX backend errors to Kubernetes cluster using a custom resource definition (CRD).

The Kubernetes resource “NSXError” is defined to hold error information. For each kubernetes resource object that has NSX backend failures, one “NSXError” object is generated with error information. There is a ‘common error object’ containing all cluster-wide errors.

The `enable_err_crd` data type is a boolean: `true` or `false`. Missing entry is accepted. If set to `true`, you define a CRD to handle the “NSXError” common error object.

Example: network profile for `enable_err_crd`:

```
{
 "name": "example_network_profile",
 "description": "enable_err_crd",
 "parameters" : {
   "cni_configurations": {
      "type": "nsxt",
      "parameters": {
        "enable_err_crd": true
      }
   }
 }
}
```


Defining Network Profiles for Shared Tier-1 Router

In this topic

Page last updated:

This topic describes how to define network profiles to enable Kubernetes clusters to use a shared Tier-1 Router.

About the Shared Tier-1 Topology

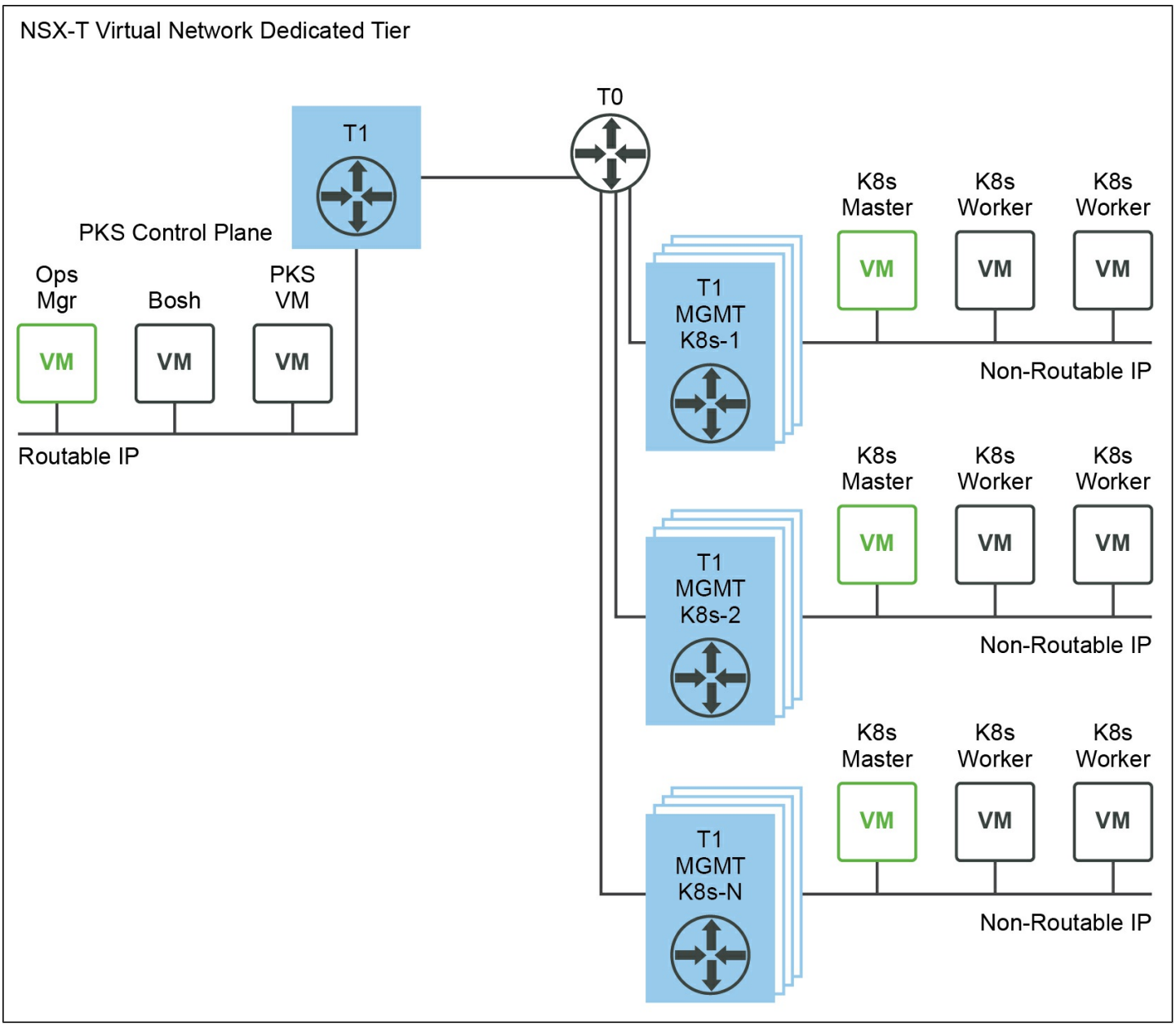
A Shared Tier-1 network topology provides an alternative approach for deploying Kubernetes clusters.

> **Note:** The Shared Tier-1 topology requires NSX-T Data Center v2.5.

By default when a Kubernetes cluster is provisioned by Enterprise PKS, the system creates following NSX-T objects:

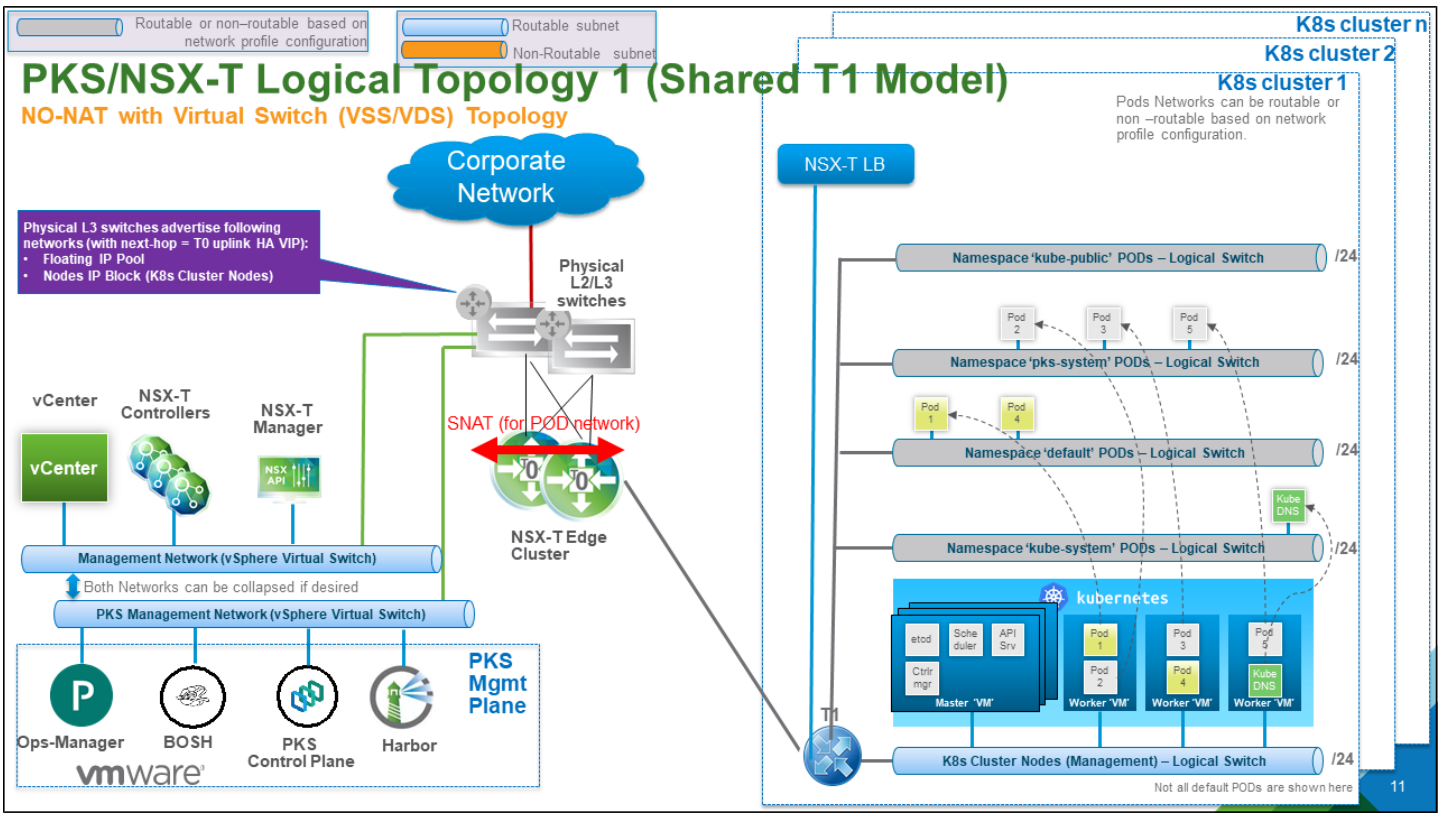
- 1 Logical Switch and Tier-1 Router for each Kubernetes Nodes subnet
- 1 Logical Switch and Tier-1 Router for each Kubernetes namespace
- 1 Logical Switch and Tier-1 Router each NSX-T Load Balancer that is allocated for the Kubernetes cluster

As depicted, the result is that a given Kubernetes cluster will run several Tier-1 switches and routers in its topology.



An alternative topology is to use a single, shared Tier-1 switch and router for each Kubernetes cluster. As depicted, the shared Tier-1 model only uses one Tier-1 router and multiple logical switches connected to the shared Tier-1 to connect all Kubernetes cluster components, including:

- Kubernetes Nodes Networks
- Kubernetes Namespaces
- NSX-T load balancer instances allocated for the Kubernetes cluster



The shared Tier-1 model configures any necessary NAT rules (if using NAT mode) on the single Tier-1 router directly. The Tier-0 router is no longer used for any NAT configuration. As a result, the Tier-0 router can operate in Active/Active mode if all Kubernetes clusters are deployed using the Shared Tier-1 model. The Shared Tier-1 model enables higher scale numbers for PKS as number of NSX-T objects allocated per Kubernetes cluster is drastically reduced. The advantage of the shared Tier-1 topology is that you can increase the number of NSX-T objects that can be supported in a given cluster.

Implementing a Single Tier Topology

To implement a shared Tier-1 topology, you define a network profile that enables the `single_tier_topology` key. Shown below is an example network profile to enable a Shared Tier-1 Router for Kubernetes clusters:

```
{
 "name": "example-network-profile-shared-t1",
 "description": "Shared-Tier-1 topology network profile",
 "parameters": {
  "single_tier_topology": true
 }
}
```

To create a Shared Tier-1 network profile, see Create Network Profile.

To create a cluster using a Shared Tier-1 network profile, see Create a Cluster with a Network Profile.

Implementing a Shared Tier-1 Topology in a Multi-Tier-0 Environment

In a Shared Tier-1 Router topology, all Kubernetes cluster traffic is automatically NATed in the single Tier-1 router that services that cluster. However, in a Multi-Tier-0 environment, traffic from Kubernetes Node Networks to the Shared Tier-0 Router cannot be NATed.

To implement a Shared Tier-1 topology in a Multi-Tier-0 environment, use the `infrastructure_networks` field in the network profile and include the subnets where your infrastructure is running. During Kubernetes cluster creation, Enterprise PKS will add a NO_SNAT rule from the Node Network to subnets specified in the `infrastructure_networks` field.

In the following example network profile, the `infrastructure-networks` field includes three subnets for which NO_SNAT rules will be created. These subnets map to the PKS Control Plane (`30.0.0.0/24`), vCenter and NSX-T VMs (`192.168.111.0/24`), and the Nodes DNS server (`192.168.115.1`).

```
{
 "name": "tenant-A-shared-T1",
 "description": "Example Network Profile for Tenant A Shared Tier-1 Router Topology",
 "parameters": {
  "t0_router_id": "a6addd27-24ce-469a-979e-cf742a19ef5c",
  "fip_pool_ids": [
   "a8b7f715-42f0-46bf-a4f2-1599c55058b6" ],
  "pod_ip_block_ids": [
   "edd59bf6-ff04-420c-88de-2c43d47f7130" ],
  "infrastructure_networks": [
   "30.0.0.0/24",
   "192.168.111.0/24",
   "192.168.115.1"
  ],
  "single_tier_topology": true
 }
}
```

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

Compute Profiles and Host Groups (vSphere Only)

Page last updated:

This section describes how to use compute profiles and host groups for VMware Enterprise PKS clusters on vSphere.

See the following topics:

- Using Compute Profiles
- Using vSphere Host Groups with Enterprise PKS

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

Using Compute Profiles (vSphere Only)

In this topic

Page last updated:

warning: This feature is a beta component and is intended for evaluation and test purposes only. Do not use this feature in a production environment. Product support and future availability are not guaranteed for beta components.

This topic describes how to use compute profiles for Kubernetes clusters provisioned with VMware Enterprise PKS on vSphere with NSX-T integration.

The procedures in this topic use the PKS API.

Overview

Compute profiles allow you to specify which vSphere resources are used when deploying Kubernetes clusters in a PKS deployment. Use this feature to control where your Kubernetes clusters are created within your virtual infrastructure.

With compute profiles, you can define Availability Zones (AZs) dynamically instead of adding new AZs in the BOSH Director tile. As a result, you do not need to make AZ changes to each plan and the overall impact to the PKS control plane is smaller.

Note: Compute profiles only override the AZ values specified in Enterprise PKS plans. The rest of the cluster's configuration is inherited from the plan.

You define a compute profile within a JSON file, and then create clusters that use the compute profile using the PKS API.

You can customize any number of compute profiles.

Prerequisites

Before you define a compute profile, you must have the following:

- A configured Enterprise PKS installation on vSphere with NSX-T. See Installing Enterprise PKS on vSphere with NSX-T.
- A PKS UAA admin access token to provide when calling the PKS API. For instructions that explain how to export your access token to an environment variable, see Export PKS API Token.

Define a Compute Profile

Define a compute profile in a JSON file.

1. In a new text file, enter the parameters for the compute profile in the following format:

```
{
  "name" : "PROFILE-NAME",
  "description" : "PROFILE-DESCRIPTION",
  "parameters" : {
    "azs": [{
      "name": "AZ-NAME",
      "cpi": "CPI-NAME",
      "cloud_properties": {
        "datacenters": [{
          "name": "DC-NAME",
          "clusters": [{
            "CLUSTER-NAME": {
              "resource_pool": "RESOURCE-POOL-NAME"
            }
          }]
        }]
      }
    }],
    "master_azs": ["AZ-NAME"],
    "worker_azs": ["AZ-NAME"]
  }
}
```

   Where:

   - `PROFILE-NAME` is the name of the compute profile that you want to define. Enter a string value up to 26 characters long. For example, `dev`.
   - (Optional) `PROFILE-DESCRIPTION` describes the compute profile.
   - `AZ-NAME` is a name for the availability zone (AZ) where you want to deploy cluster VMs. For example, `z1`.

> **Note**: If you define multiple AZs in the `azs` array, you can specify different AZs for worker node VMs and master node VMs.

- `CPI-NAME` is the BOSH CPI ID of your Enterprise PKS deployment. For instructions on how to obtain the `CPI-NAME`, see Retrieve the BOSH CPI ID. For example, `abc012abc345abc567de`.
- `DC-NAME` is the name of your data center as it appears in Ops Manager and your cloud provider console. For example, `dc-east`. For additional cloud properties related to AZs, see the vSphere CPI AZs section of the BOSH documentation.
- `CLUSTER-NAME` is the name that you want to give your cluster.
- `RESOURCE-POOL-NAME` is the name of the resource pool where you want to deploy your cluster.

For example, this compute profile shows one AZ for one cluster assigned to the `resource_pool` named `my-res-pool`:

```
{
  "name": "dev",
  "description": "For development clusters",
  "parameters": {
    "azs": [{
      "cpi": "abc012abc345abc567des",
      "name": "z1",
      "cloud_properties": {
        "datacenters": [{
          "name": "my-dc",
          "clusters": [{
            "my-vsphere-cluster": {
              "resource_pool": "my-res-pool1"
            }
          }]
        }]
      }
    }],
    "master_azs": ["z1"],
    "worker_azs": ["z1"]
  }
}
```

2. Save the file with a JSON extension. For example, `dev-clusters.json`.

For more example compute profiles, see Example Compute Profiles.

Retrieve the BOSH CPI ID

Use the following procedure to retrieve the BOSH CPI ID for your Enterprise PKS deployment.

1. Locate the credentials that were used to import the PCF .ova or .ovf file into your virtualization system. You configured 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.

2. 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.

3. When prompted, enter the password that you configured during the .ova deployment into vCenter. For example:

   ```
   $ ssh ubuntu@my-opsmanager-fqdn.example.com
   Password: **********
   ```

4. Run `bosh cpi-config` to locate the Cloud Provider Interface (CPI) name for your deployment. For example:

   ```
   $ bosh cpi-config
   Using environment 'BOSH-DIRECTOR-IP' as client 'ops_manager'
   cpis:
   - migrated_from:
     - name: ""
     name: YOUR-CPI-NAME
   ```

   For more information about running BOSH commands in your Enterprise PKS deployment, see Using BOSH Diagnostic Commands in Enterprise PKS.

Create a Compute Profile

Use the JSON file that contains your compute profile parameters to make a request to the PKS API. The PKS API applies the parameters to your PKS deployment as a compute profile.

To create the compute profile, run the following command:

```
curl -s -k -X POST "https://PKS-API:9021/v1/compute-profiles" \
-H "cache-control: no-cache" -H "authorization: Bearer $YOUR-ACCESS-TOKEN" \
-H "Content-Type: application/json" -d "@PATH-TO-YOUR-COMPUTE-PROFILE.json"
```

Where:

- `PKS-API` is the FQDN of your PKS API endpoint. For example, `api.pks.example.com`.
- `YOUR-ACCESS-TOKEN` is your PKS UAA admin access token specified as an environment variable. For information on how to export this token as an environment variable, see Export PKS API Token.
- `@PATH-TO-YOUR-COMPUTE-PROFILE.json` is the path and filename for the file that contains your compute profile parameters. For example, `@../compute-profiles/dev-clusters.json`.

List Compute Profiles

You can list compute profiles in your PKS deployment by making a request to the PKS API.

To list compute profiles, run the following command:

```
curl -s -k  "https://PKS-API:9021/v1/compute-profiles" \
-H "cache-control: no-cache" -H "authorization: Bearer $YOUR-ACCESS-TOKEN" \
-H "Content-Type: application/json"
```

Where:

- `PKS-API` is the FQDN of your PKS API endpoint. For example, `api.pks.example.com`.
- `YOUR-ACCESS-TOKEN` is your PKS UAA admin access token specified as an environment variable. For information on how to export this token as an environment variable, see Export PKS API Token.

For example:

```
$ curl -s -k  "https://api.pks.example.com:9021/v1/compute-profiles" \
-H "cache-control: no-cache" -H "authorization: Bearer $YOUR-ACCESS-TOKEN" \
-H "Content-Type: application/json"
[
  {
    "description": "For development clusters",
    "name": "dev",
    "parameters": {
      "azs": [
        {
          "cloud_properties": {
            "datacenters": [
              {
                "clusters": [
                  {
                    "my-vsphere-cluster": {
                      "drs_rules": null,
                      "host_group": null,
                      "resource_pool": "my-res-pool1"
                    }
                  }
                ],
                "name": "my-dc"
              }
            ]
          },
          "cpi": "abc012abc345abc567des",
          "name": "z1"
        }
      ],
      "master_azs": [
        "z1"
      ],
      "worker_azs": [
        "z1"
      ]
    },
    "uuid": "618abbcc-4dc5-4080-a24a-56c788b5ed44"
  }
]
```

Create a Cluster with a Compute Profile

Define cluster parameters in a JSON file. Include the name of the compute profile and plan in this file. Then make a request to the PKS API to create the cluster.

To create a cluster with a compute profiles, do the following:

1. Create a new JSON file that describes the cluster that you want to create. Include the compute profile names in the file. For example:

   ```
   {
     "name": "my-cluster",
     "plan_name": "small",
     "parameters": {
       "kubernetes_master_host": "master.host"
     },
     "compute_profile_name": "dev-clusters",
     "network_profile_name": "my-network-profile"
   }
   ```

   If you are using Enterprise PKS with NSX-T integration, you can optionally specify a network profile. For more information on network profiles, see Using Network Profiles.

2. To create the cluster using the compute profile, run the following command:

   ```
   curl -k -X POST "https://PKS-API:9021/v1/clusters/" \
   -H "cache-control: no-cache" -H "authorization: Bearer $YOUR-ACCESS=TOKEN" \
   -H "Content-Type: application/json" -d "@PATH-TO-CLUSTER.json"
   ```

   Where:

   - `PKS-API` is the FQDN of your PKS API endpoint. For example, `api.pks.example.com`.
   - `YOUR-ACCESS-TOKEN` is your PKS UAA admin access token specified as an environment variable. For information on how to export this token as an environment variable, see Export PKS API Token.
   - `@PATH-TO-CLUSTER.json` is the path and filename for the JSON file that you created in Define a Compute Profile.

   If successful, the cluster creation command outputs something similar to the following:

```
{
  "name": "k8s1",
  "plan_name": "small",
  "last_action": "CREATE",
  "last_action_state": "in progress",
  "last_action_description": "Creating cluster",
  "uuid": "abcdefg-a123-b456-c789-1011121314",
  "kubernetes_master_ips": ["In Progress"],
  "network_profile_name": "np-tenant-a",
  "compute_profile_name": "",
  "parameters": {
    "kubernetes_master_host": "k8s1.example.com",
    "kubernetes_master_port": 8443,
    "worker_haproxy_ip_addresses": null,
    "kubernetes_worker_instances": 2,
    "authorization_mode": null,
    "nsxt_network_profile": "{\"fip_pool_ids\":[\"abcdefg-b234-c567-d891-hijklmnop\"],\"lb_size\":\"small\",
    \"master_vms_nsgroup_id\":\"0e96f87d-891c-4b32-9757-54051a7f2b91\",\"pod_ip_block_ids\":
    [\"abcdefg-b234-c567-d891-hijklmnop\"],\"pod_routable\":true,\"pod_subnet_prefix\":24,\"t0_router_id\":
    \"zyxwvu-b234-c567-d891-hijklmnop\"}",
    "compute_profile": "{\"azs\":[{\"name\":\"az-1\",\"cpi\":\"abc012abc345abc567des\",\"cloud_properties\":{\"datacenters\":[{\"name\":\"my-dc\",\"clusters\":[{\"cluster1\":{\"resource_pool\":\"pks-respool-1\"}}]}}},{\"nam
    "k8s_customization_parameters": {
      "vsphere_cloud_provider_config_key": null,
      "unset_http_proxy": null,
      "insecure_registries": null
    }
  }
}
```


Resize a Cluster with a Compute Profile

Currently, you cannot use the PKS API to resize an existing cluster that uses a compute profile.

To resize an existing cluster, perform the following steps:

1. Delete the cluster by running the `pks delete cluster` command.
2. Recreate the cluster with a compute profile as described in Create a Cluster with a Compute Profile, and specify a larger value for the `kubernetes_worker_instances` parameter in your cluster JSON file.

List Clusters with Compute Profile

When you execute the `pks clusters` command, the output does not include compute profile information for clusters that use compute profiles.

To view compute profile information for all clusters, run the following command:

```
curl -s -k -H "authorization: Bearer $YOUR-ACCESS-TOKEN" "https://PKS-API:9021/v1/clusters"
```

Where:

- `PKS-API` is the FQDN of your PKS API endpoint. For example, `api.pks.example.com`.
- `$YOUR-ACCESS-TOKEN` is your PKS UAA admin access token specified as an environment variable. For information on how to export this token as an environment variable, see Export PKS API Token.

The output includes compute profile information.

For example:

```
$ curl -s -k  "https://api.pks.example.com:9021/v1/clusters" \
-H "cache-control: no-cache" -H "authorization: Bearer $YOUR-ACCESS-TOKEN" \
-H "Content-Type: application/json"
 [
   {
     "name": "cluster-1",
     "plan_name": "Plan-1",
     "last_action": "UPDATE",
     "last_action_state": "succeeded",
     "last_action_description": "Instance update completed",
     "uuid": "123a45bc-de6f-7891-gh23-45hi678912j3",
     "kubernetes_master_ips": [
       "10.0.8.6"
     ],
     "network_profile_name": "my-net-profile",
     "compute_profile_name": "my-compute-profile",
     "parameters": {
       "kubernetes_master_host": "cluster-1.virtmerlin.io",
       "kubernetes_master_port": 8443,
       "kubernetes_worker_instances": 1
     }
   },
   {
     "name": "cluster-2",
     "plan_name": "Plan-1",
     "last_action": "UPDATE",
     "last_action_state": "succeeded",
     "last_action_description": "Instance update completed",
     "uuid": "a9876b54-3c21-9def-87g6-efgh1543219i",
     "kubernetes_master_ips": [
       "10.0.8.4"
     ],
     "network_profile_name": "my-net-profile-2",
     "compute_profile_name": "my-compute-profile-2",
     "parameters": {
       "kubernetes_master_host": "cluster-2.virtmerlin.io",
       "kubernetes_master_port": 8443,
       "kubernetes_worker_instances": 2
     }
   }
 ]
```

Delete Compute Profiles

You can delete compute profiles in your PKS deployment by making a request to the PKS API.

You cannot delete a compute profile that is applied to an existing cluster. First delete any clusters that use the compute profile. To delete a cluster using a compute profile, use the `pks delete cluster` command.

To delete a compute profile, run the following command:

```
curl -k -X DELETE  "https://PKS-API:9021/v1/compute-profiles" \
-H "cache-control: no-cache" -H "authorization: Bearer $YOUR-ACCESS-TOKEN" \
-H "Content-Type: application/json"
```

Where:

- `PKS-API` is the FQDN of your PKS API endpoint. For example, `api.pks.example.com`.
- `YOUR-ACCESS-TOKEN` is your PKS UAA admin access token specified as an environment variable. For information on how to export this token as an environment variable, see Export PKS API Token.

Example Compute Profiles

This section includes example compute profiles that define different sets of vSphere resources.

dev-clusters.json

The following example compute profile for a single cluster uses one AZ and one resource pool:

```
{
  "name": "dev",
  "description": "For development clusters",
  "parameters": {
    "azs": [{
      "cpi": "abc012abc345abc567des",
      "name": "z1",
      "cloud_properties": {
        "datacenters": [{
          "name": "my-dc",
          "clusters": [{
            "my-vsphere-cluster": {
              "resource_pool": "my-res-pool1"
              }
          }]
        }]
      }
    }],
    "master_azs": ["z1"],
    "worker_azs": ["z1"]
  }
}
```

prod-clusters.json

The following example compute profile uses two AZs, with one cluster in the first AZ and three clusters in the second AZ:

```
{
  "name": "prod",
  "description": "For production clusters",
  "parameters": {
    "azs": [{
        "name": "z1",
        "cpi": "abc012abc345abc567des",
        "cloud_properties": {
          "datacenters": [{
            "name": "my-dc",
            "clusters": [{
              "cluster1": {
                "resource_pool": "rp3"
              }
            }]
          }]
        }
      },
      {
        "name": "z2",
        "cpi": "abc012abc345abc567des",
        "cloud_properties": {
          "datacenters": [{
            "name": "my-dc",
            "clusters": [{
              "cluster2": {
                "resource_pool": "rp1"
              }
            },
              {
              "cluster3": {
                "resource_pool": "rp1"
              }
            },
              {
              "cluster4": {
                "resource_pool": "rp2"
              }
            }]
          }]
        }
      }
   ],
   "master_azs": ["z1","z2"],
   "worker_azs": ["z1","z2"]
  }
}
```

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

Using vSphere Host Groups with Enterprise PKS

In this topic

Page last updated:

Topic provided by VMware

This topic describes how to use vSphere Host Groups with VMware Enterprise PKS.

About vSphere Host Groups

In vSphere, a cluster is a collection of EXSi servers that run virtual machines (VMs). A typical way to organize resources within a cluster is using resource pools. A resource pool is a collection of vSphere resources.

Another way to segment resources within a cluster is using host groups. This means that within a cluster object you can specify certain ESXi hosts to be part of a host group.

Enterprise PKS users can define host groups in vSphere, then in the PKS tile can specify the host group. Host groups align with the Availability Zone (AZ) construct in BOSH.

For more information on vSphere host groups, refer to the vSphere documentation.

Host Group Use Cases for Enterprise PKS

This subsection describes use cases for using host groups with Enterprise PKS.

Enabling Support for vSAN Fault Domains

The vSAN fault domains feature instructs vSAN to spread redundancy components across the servers in separate computing racks. In this way, you can protect the environment from a rack-level failure such as loss of power or connectivity. For more information, see Designing and Sizing vSAN Fault Domains in the VMware documentation.

Fault domains map to host groups. If you have set up fault domains in your vSAN architecture, you can now leverage host groups with PKS.

Using Host Group as a New AZ in BOSH

Previously, the two types of AZs available with PKS on vSphere were Datacenter and Datacenter plus Resource Pool. Host groups gives you a third option: Datacenter plus HostGroups.

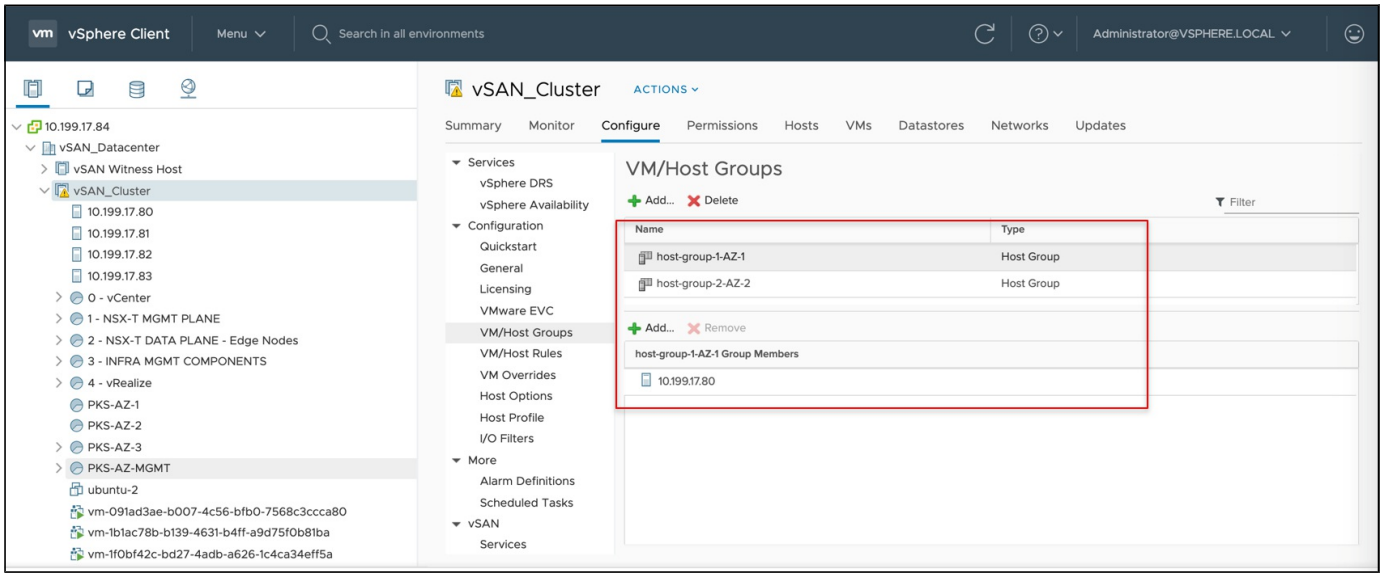
In the case of multi-master Kubernetes clusters, with the Datacenter and Datacenter plus Resource Pool AZs, there is no guarantee that master nodes will reside on separate ESXi hosts. With the Datacenter plus HostGroups AZ you can guarantee that Kubernetes

master nodes will reside on separate ESXi hosts.

Defining a Host Group in vSphere

To implement host groups with Enterprise PKS, the first step is to define a host group in vSphere.

1. Log in to vCenter.
2. Select the compute **Cluster**.
3. Select the **Configure** tab.
4. Under **Configuration**, select **VM/Host Groups**.
5. Click **Add** and configure the host group as follows:
   - Name: Enter a name for the host group.
   - Type: Select **Host Group** from the drop down.
   - Click **Add** and select the ESXi host(s) to include in the host group.
   - Click **OK**.
6. Once done, you should see that the host group is configured.



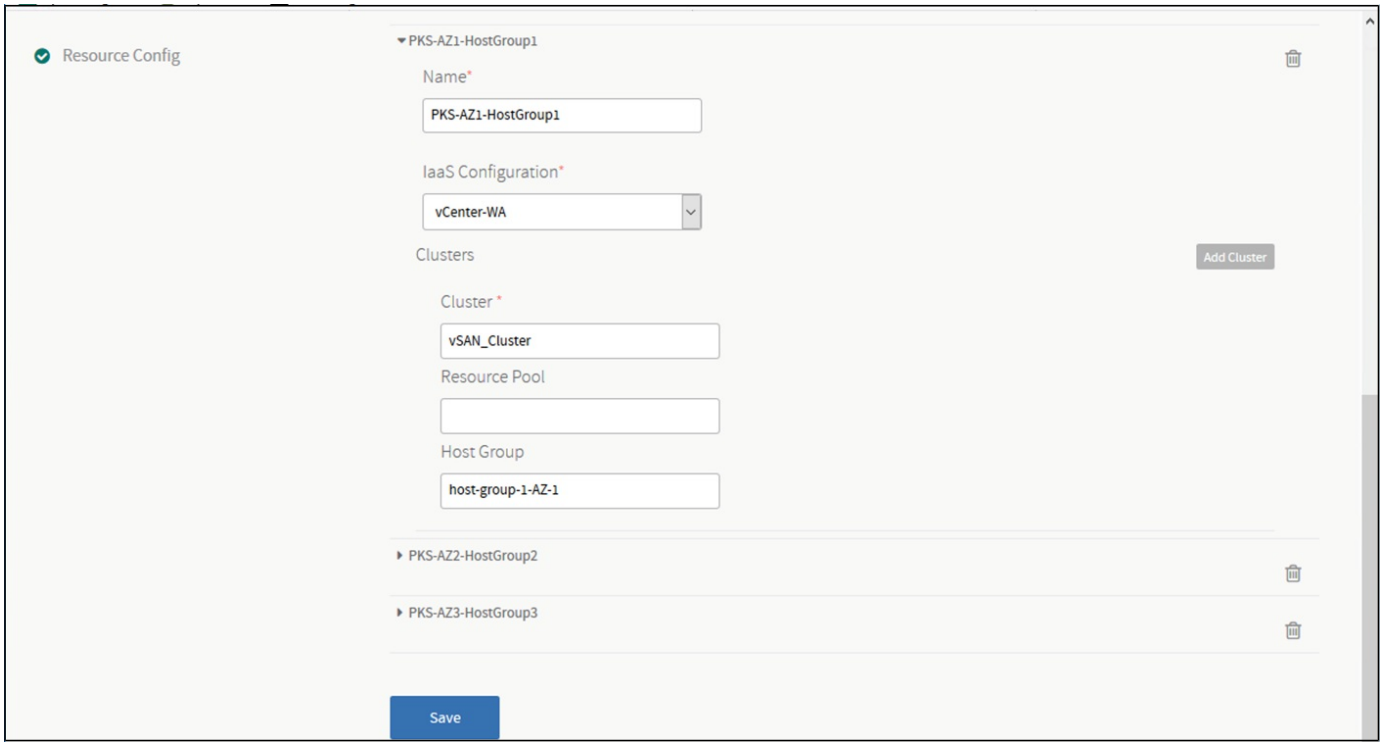
Using a Host Group with Enterprise PKS

Once the host group is defined in vSphere, the next step is to declare this host group when defining the BOSH Availability Zone (AZ) for use with Enterprise PKS.

1. Log in to Ops Manager.
2. Select the BOSH Director tile.
3. Select the **Create Availability Zones** tab.
4. Select the desired AZ, or create a new one.

5. In the **Clusters** section, enter the name of the **Host Group**.

6. Click **Save**.



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

Adding Infrastructure Password Changes to the Enterprise PKS Tile

In this topic

Page last updated:

This topic describes how to manage VMware Enterprise PKS after changing a BOSH Director or Enterprise PKS service account password.

Manage Your Service Account Passwords

When you installed Enterprise PKS you created two service accounts:

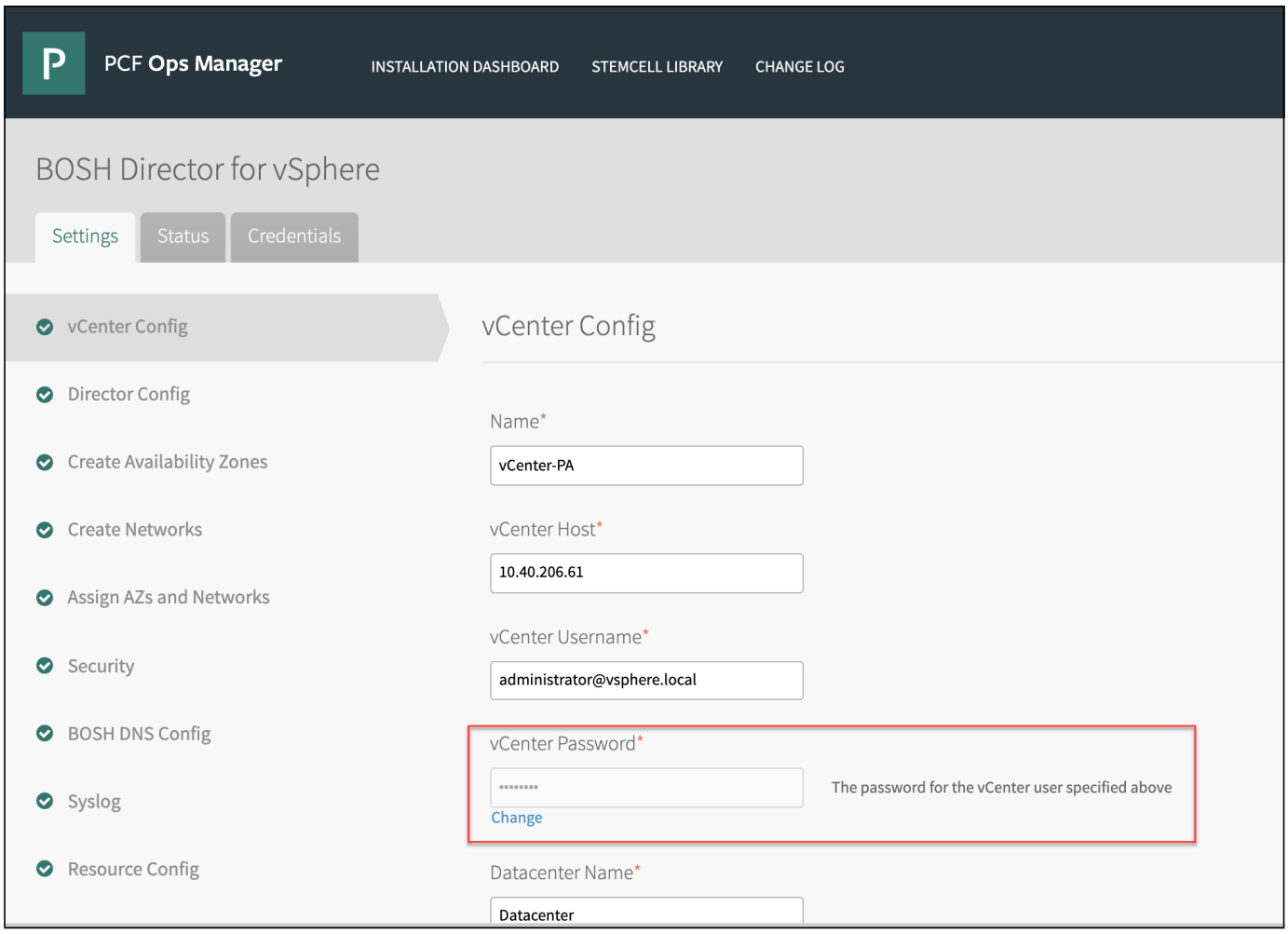
- **BOSH/Ops Manager Service Account**: This service account is configured in the BOSH Director tile.
- **Master Node Service Account**: This service account is configured in the Enterprise PKS tile.

You must update a tile's copy of a service account password after changing the password on your network.

Step 1: Update Your Service Account Passwords

To update BOSH Director with a new **BOSH/Ops Manager Service Account** password, perform the following steps:

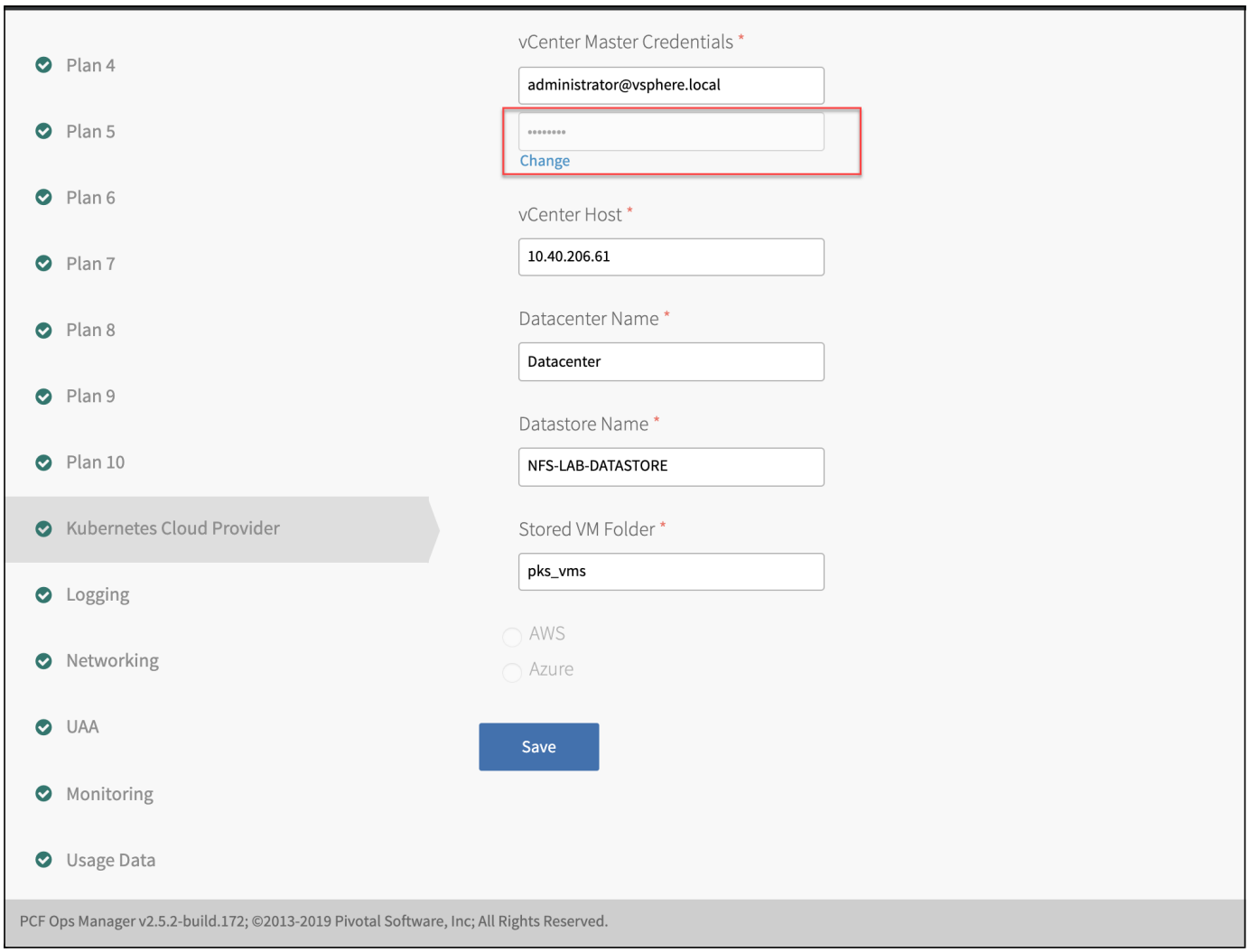
1. Access the **Installation Dashboard** in Ops Manager.
2. Select the BOSH Director tile.
3. Select your IaaS' **Config** tab.
4. Click **Change**, the link beneath the IaaS **Password** field, to modify the password.



5. Enter the new service account password.
6. Click **Save** to save the new password to the BOSH Director tile.

To update Enterprise PKS with a new **Master Node Service Account** password, perform the following steps:

1. Access the **Installation Dashboard** in Ops Manager.
2. Select the Enterprise PKS tile.
3. Select the **Kubernetes Cloud Provider** tab.
4. Click **Change**, the link beneath your IaaS' **Master Credentials** field, to modify the password.



5. Enter the new master node service account password.
6. Click **Save** to save the new password to the Enterprise PKS tile.

Step 2: Deploy Your New Service Account Passwords

After updating an Ops Manager tile's service account password you must also deploy the new password.

To deploy a new password to BOSH Director and Enterprise PKS, perform the following steps:

1. Access the **Installation Dashboard** in Ops Manager.
2. Click **Review Pending Changes**.
3. In the **Errands** section for Enterprise PKS, select **Update all clusters errand**.
4. Click **Apply Changes** to update the Enterprise PKS installation with the new password(s).

> **Note**: The **Update all clusters errand** must be enabled to update the Kubernetes cloud provider password stored in Kubernetes clusters.

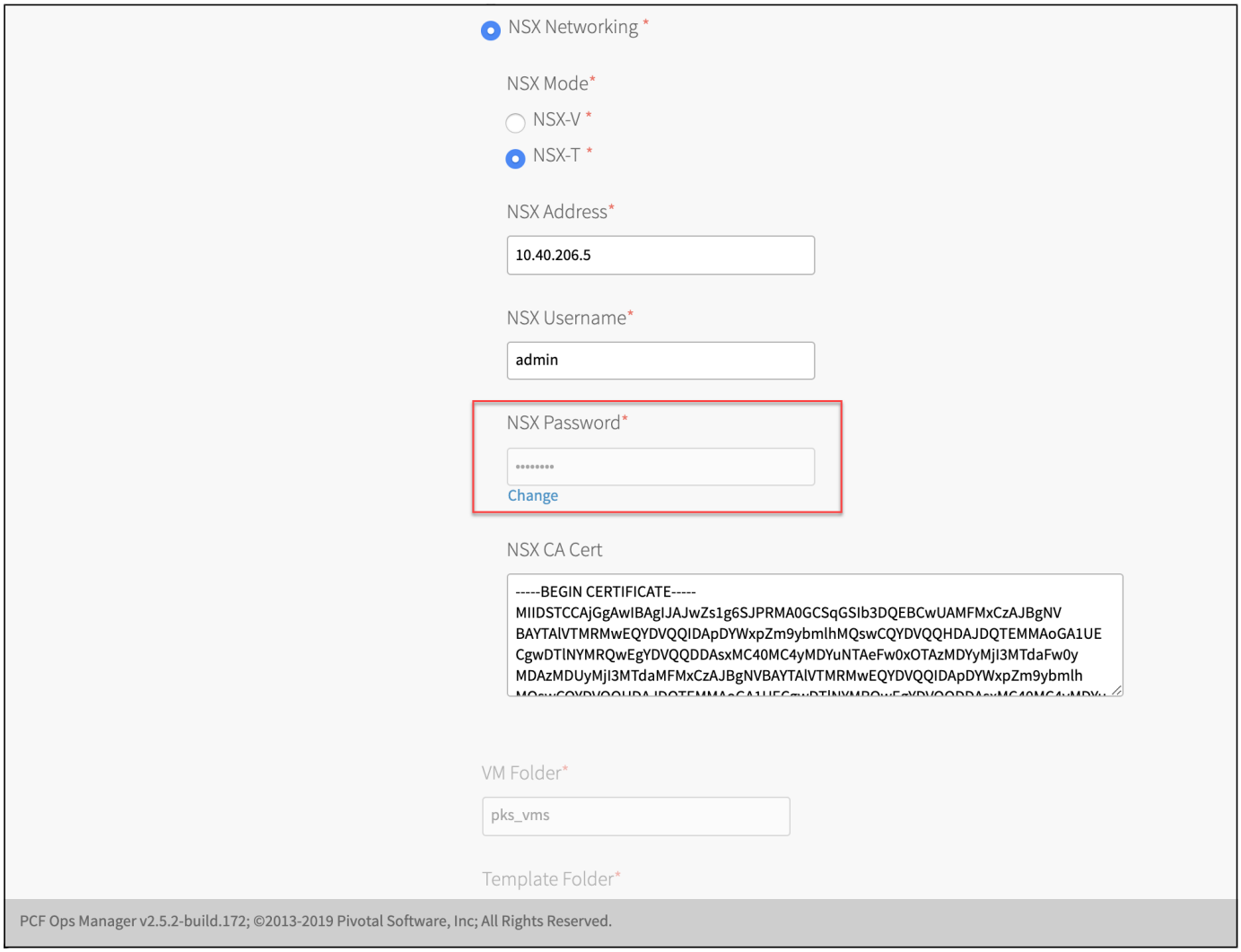
Manage Your NSX Manager Password (vSphere and vSphere with NSX-T only)

If you are on vSphere or vSphere with NSX-T only, you also configured the **NSX Manager Account** and password when you installed Enterprise PKS. This service account is configured in the BOSH Director tile.

After changing the password on your network, you must also update the BOSH Director tile's copy of the **NSX Manager Account** password.

To update the BOSH Director with the new NSX Manager password, perform the following steps:

1. Access the **Installation Dashboard** in Ops Manager.
2. Select the BOSH Director tile.
3. Select the **vCenter Config** tab.
4. Click **Change**, the link beneath the **NSX Username** field, to modify the password.



5. Enter the new password.
6. Click **Save** to save the changes to the BOSH Director tile.
7. On the Ops Manager **Installation Dashboard**, select **Review Pending Changes**.

8. Click **Apply Changes**.

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

Shutting Down and Restarting Enterprise PKS

In this topic

Page last updated:

This topic lists and describes the shutdown and startup sequence for VMware Enterprise PKS including Enterprise PKS-provisioned Kubernetes cluster nodes, PKS control plane components, and vSphere hosts.

Shutdown Sequence and Tasks

To perform a graceful shutdown of all Kubernetes, Enterprise PKS, and infrastructure components, complete the following tasks in sequence.

Step 1: Disable BOSH Resurrection

If you have the **Enable VM Resurrector Plugin** checkbox selected in the BOSH Director tile > **Director Config** pane, you must turn BOSH resurrection off before restarting PKS, to prevent BOSH from recreating VMs.

To do this, run the command `bosh update-resurrection off`.

Step 2: Shut Down Customer Apps

Shut down all customer apps running on Enterprise 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 3: Shut Down Kubernetes Clusters

Shut down all Enterprise 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 Enterprise 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 Enterprise 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:

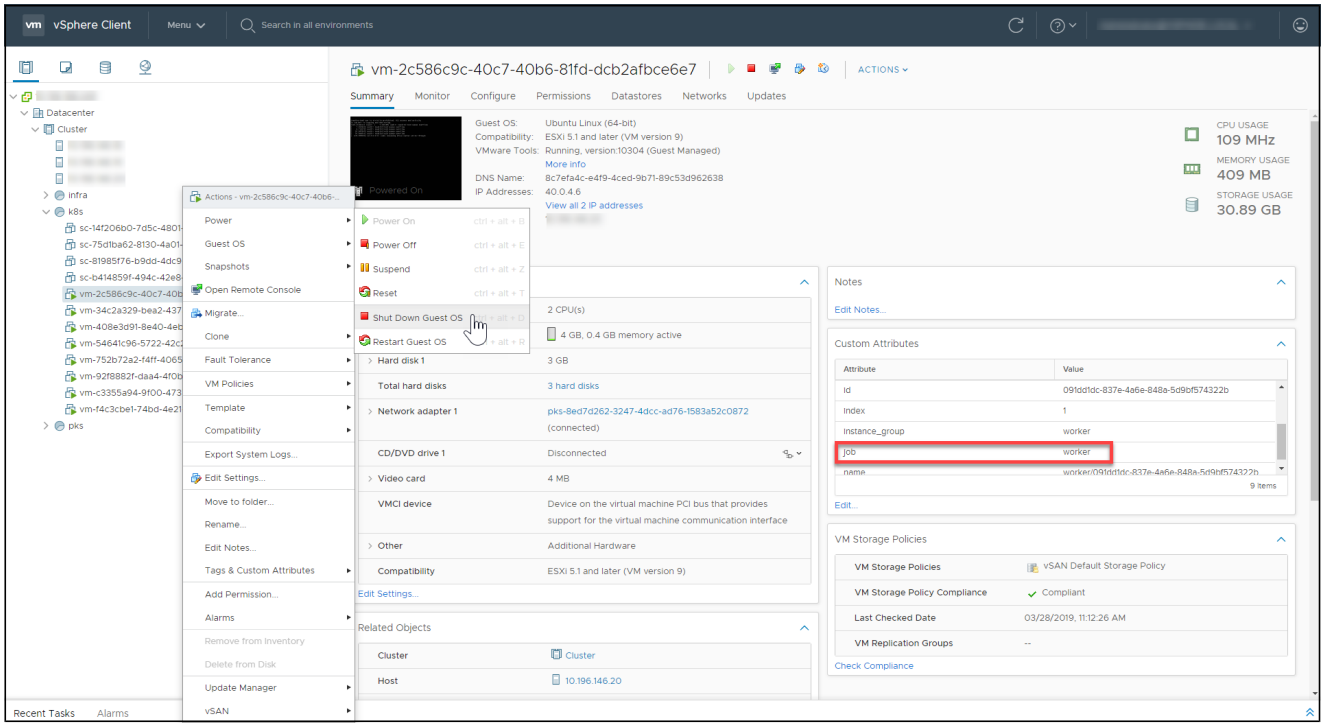
   ```
   bosh -d service-instance_CLUSTER-UUID stop master
   ```

   Where `CLUSTER-UUID` is the BOSH deployment name of your Enterprise 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 **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 4: Shut Down Enterprise 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 Enterprise PKS deployment by running the following command:

   ```
   bosh deployments
   ```

   Enterprise PKS deployment names begin with `pivotal-container-service` and include a unique BOSH-generated hash.

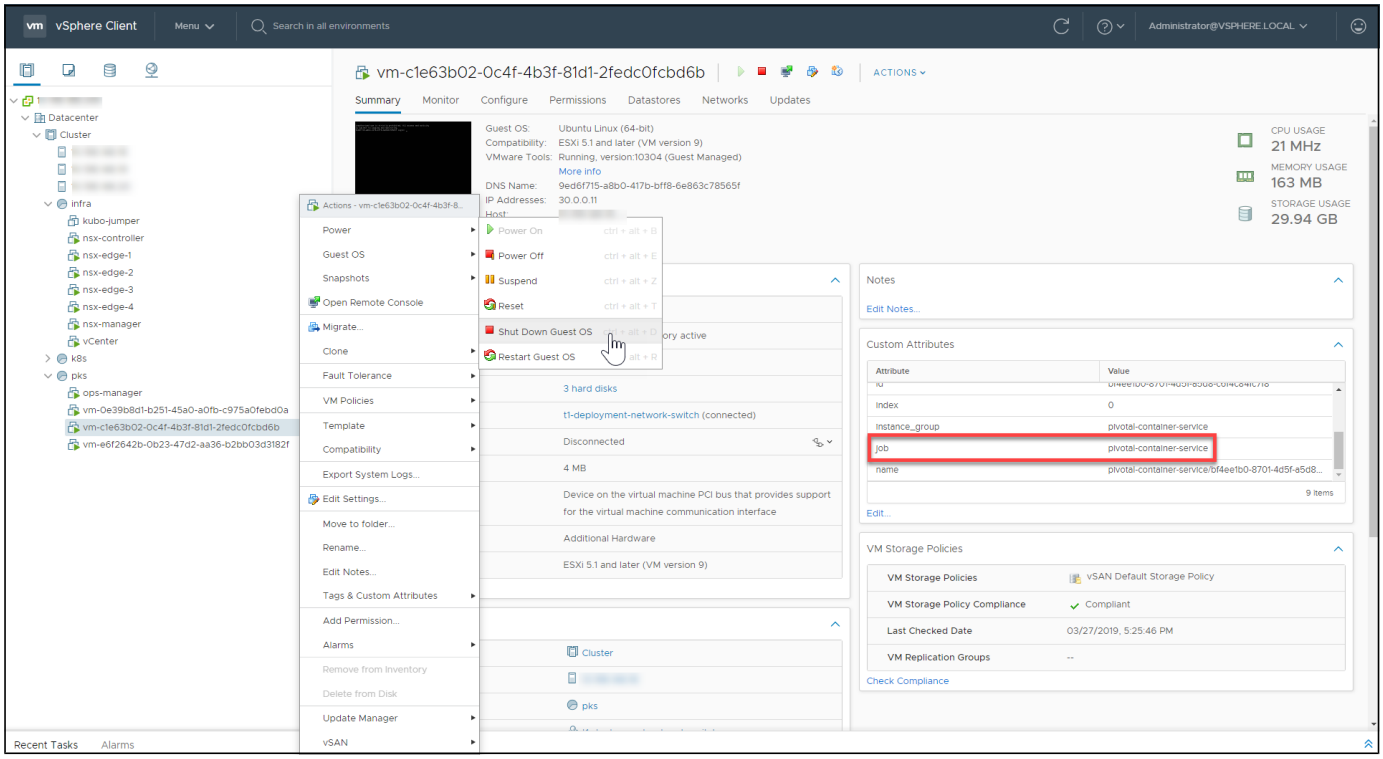
2. Using the BOSH CLI, stop the PKS control plane VM by running the following command:

   ```
   bosh -d pivotal-container-service-DEPLOYMENT-ID stop
   ```

   Where `DEPLOYMENT-ID` is the BOSH-generated ID of your Enterprise 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 5: 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.

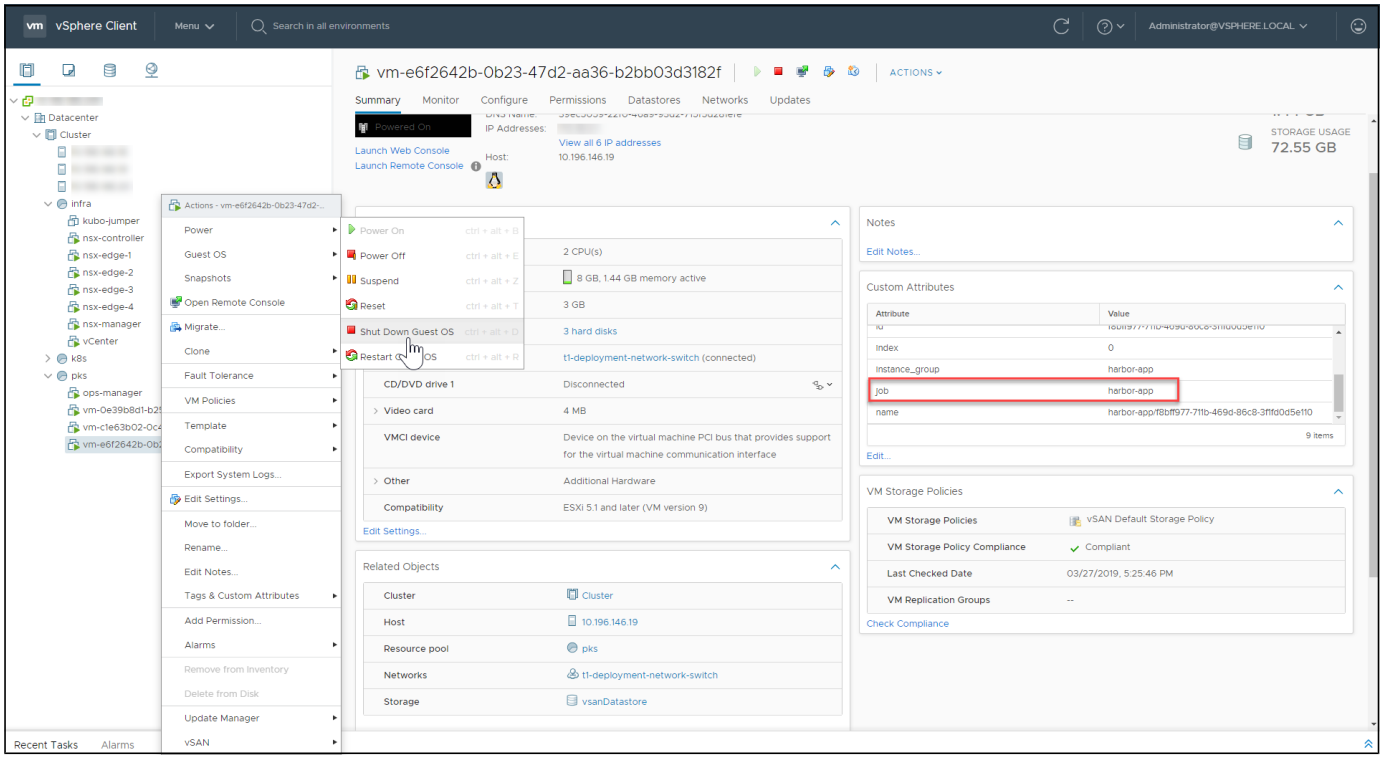
2. Using the BOSH CLI, stop the Harbor Registry VM by running the following command:

   ```
   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
   ```

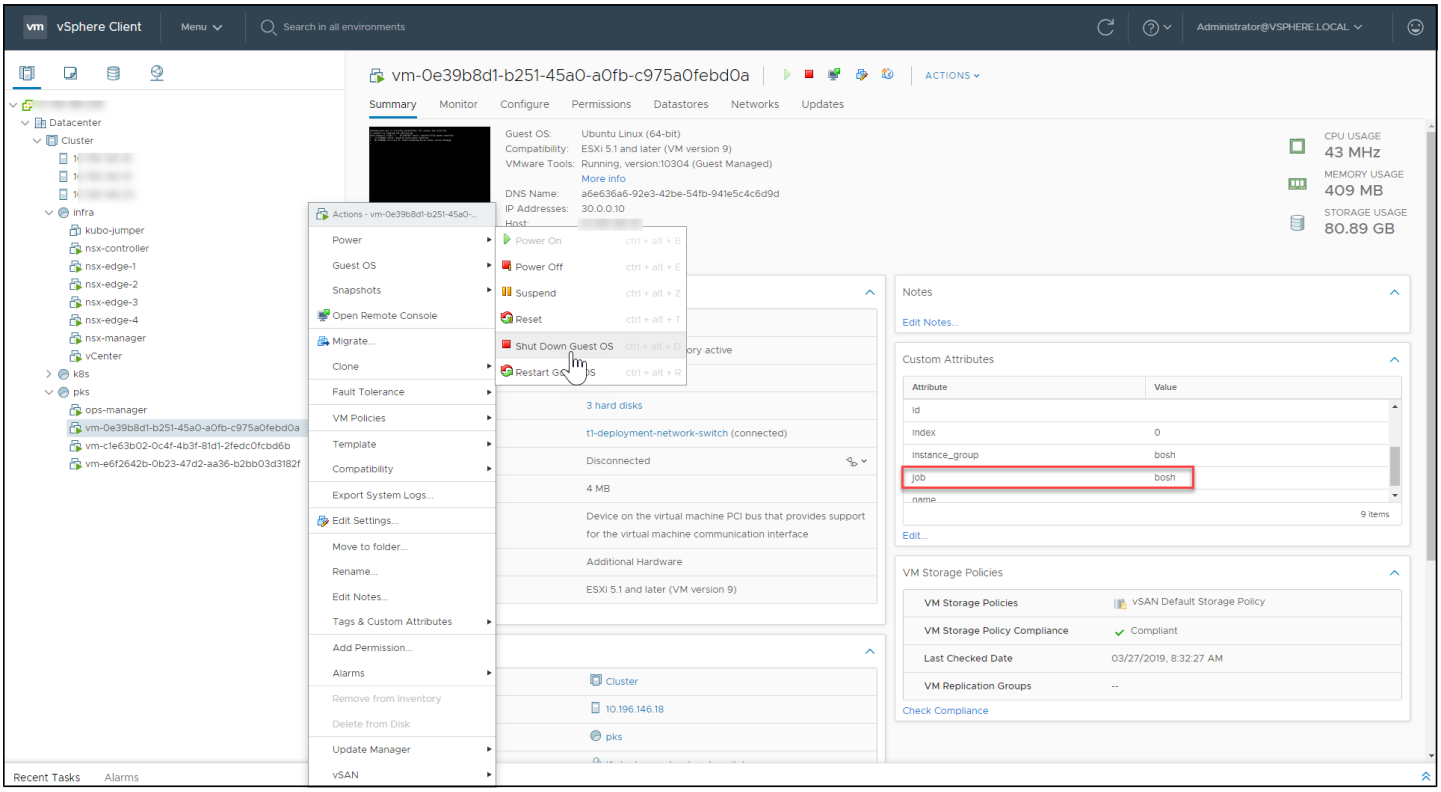
3. Using vCenter, locate and gracefully shut down the Harbor Registry VM.



View a larger version of this image.

Step 6: Shut Down BOSH Director

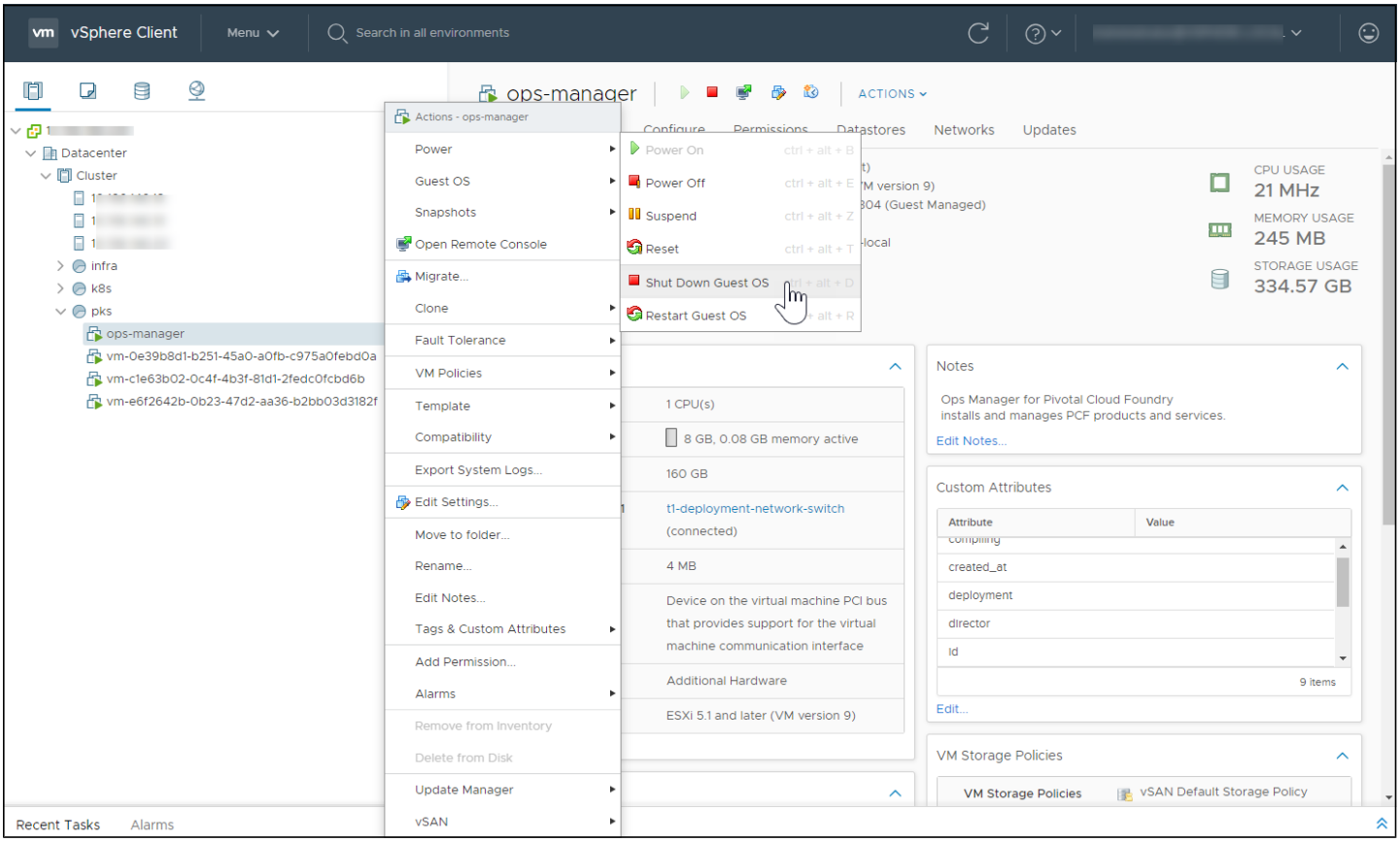
Using vCenter, locate and gracefully shut down the BOSH Director VM.



View a larger version of this image.

Step 7: Shut Down Ops Manager

Using vCenter, locate and gracefully shut down the Ops Manager VM.

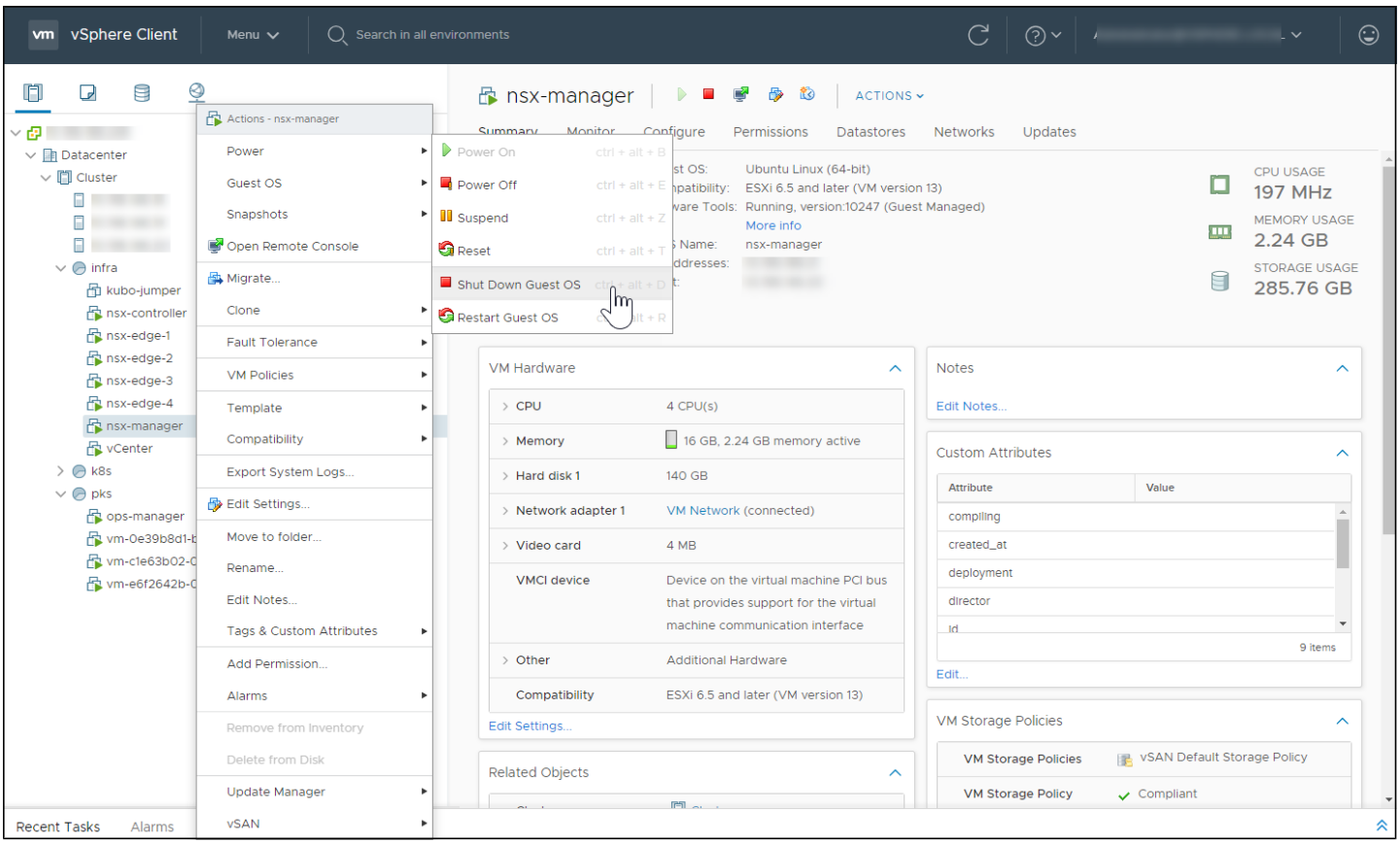


View a larger version of this image.

Step 8: 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 9: Shut Down vCenter Server

To shut down the vCenter Server VM, do the following:

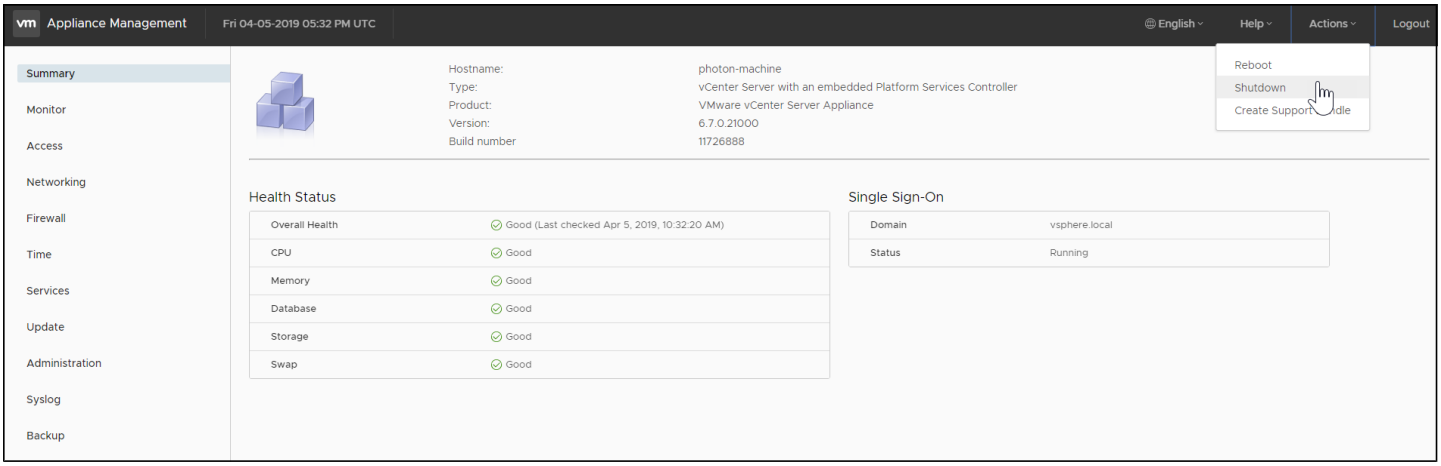
1. Navigate to the vCenter Appliance Management Interface at `https://YOUR-VCENTER-HOSTNAME-OR-IP-ADDRESS>:5480`, where `YOUR-VCENTER-HOSTNAME-OR-IP-ADDRESS` is the hostname or IP address that you use to connect to vCenter through the vSphere Web Client.

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.



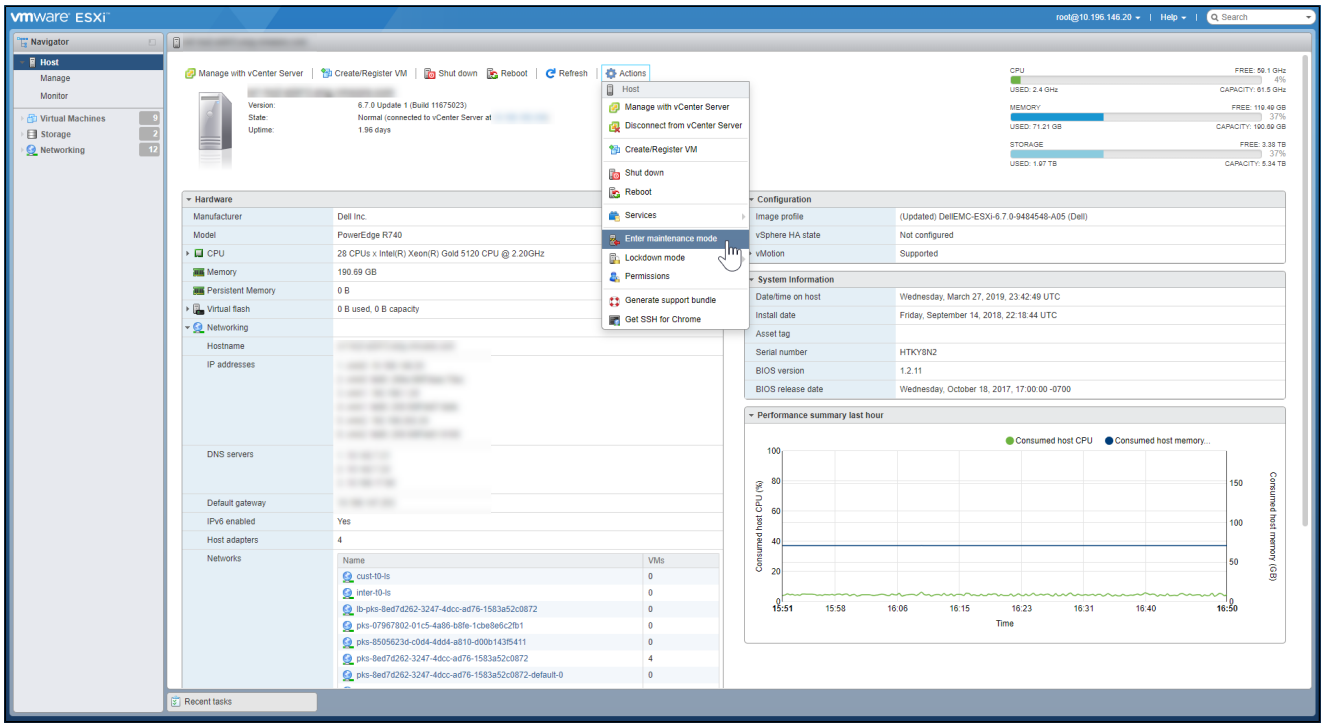
View a larger version of this image

Step 10: 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:

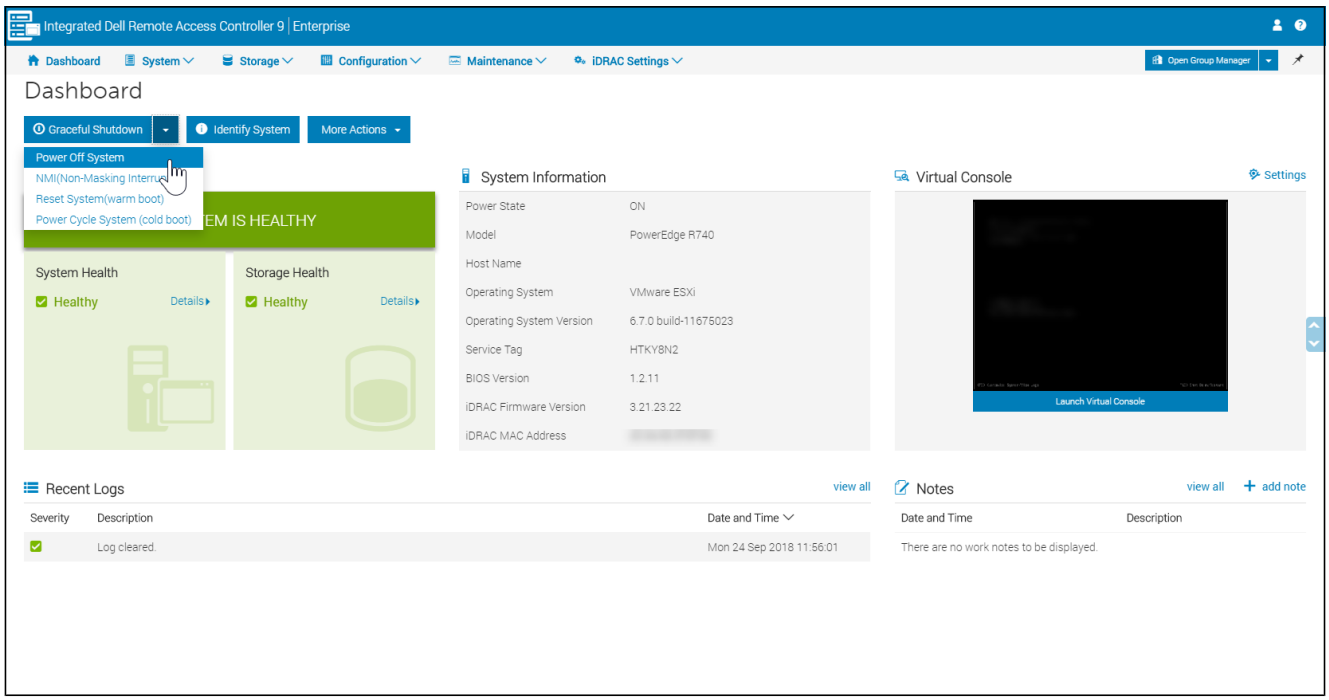
   a. Using a browser, navigate to the HTTPS IP address of the ESXi host, for example: https://10.196.146.20/.
   b. Log in using vSphere administrative credentials.
   c. Put the ESXi host in maintenance mode by selecting **Actions** > **Enter maintenance mode**.



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, Enterprise 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

1. Using vCenter, power on the Ops Manager VM.
2. Using a browser, go to the Ops Manager URL.
3. Enter the Ops Manager passphrase.
4. Log in to the Ops Manager UI.

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 Enterprise 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 Enterprise 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 Enterprise PKS deployment. For example:

```
$ bosh -d pivotal-container-service-1bf7b02738056cdc37e6 start
```

Note: Because you stopped the Enterprise 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 Enterprise PKS master node and start etcd.
2. Using the BOSH CLI, start the next Enterprise PKS master node.
3. Using the BOSH CLI, start all remaining Enterprise PKS master nodes including the master where you started etcd.
4. Using the BOSH CLI, start all Enterprise 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 Enterprise PKS-provisioned Kubernetes clusters.

Step 10: Re-enable BOSH Resurrection

Turn BOSH resurrection back on by running the command `bosh update-resurrection on`.

Deleting Enterprise PKS

In this topic

Page last updated:

This topic explains how to delete the Enterprise PKS tile.

Delete the Enterprise PKS Tile

To delete the Enterprise PKS tile, perform the following steps:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the trash can icon on the Enterprise PKS tile.
3. Click **Confirm**.
4. Click **Review Pending Changes**.
5. (Optional) By default, deleting the Enterprise PKS tile also deletes all the clusters created by Enterprise PKS. To preserve the clusters, click **Errands** and deselect the **Delete all clusters** errand.
6. Click **Apply Changes**.

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

Managing Kubernetes Clusters and Workloads

Page last updated:

This section describes how to manage VMware Enterprise PKS-provisioned Kubernetes clusters and workloads.

See the following topics:

- Managing Clusters in Enterprise PKS
- Supporting Clusters
- Deploying Workloads

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

Managing Clusters in Enterprise PKS

Page last updated:

This section describes how to manage Kubernetes clusters in VMware Enterprise PKS.

See the following topics:

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

Logging in to Enterprise PKS

In this topic

Page last updated:

This topic describes how to log in to VMware Enterprise PKS.

Overview

To manage Enterprise PKS-deployed clusters, you use the PKS Command Line Interface (PKS CLI). When you log in to Enterprise 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 Enterprise PKS, you must have the following:

- A running Enterprise PKS environment, including an external load balancer configured to forward traffic to the PKS API endpoint. See the *Installing Enterprise PKS* section for your cloud provider.
- A username and password that has access to the PKS API. See Managing Enterprise PKS Users with UAA.
- The PKS CLI installed on your local system. See Installing the PKS CLI.

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** > **Enterprise PKS** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
- `USERNAME` and `PASSWORD` belong to the account you created in the Grant Enterprise PKS Access to an Individual User

section of *Managing Enterprise PKS Users 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
  ```

  Note: If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

Log in to the PKS CLI as an Automated Client

To log in to the PKS CLI as an automated client for a script or service, run the following command:

```
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** > **Enterprise PKS** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
- `CLIENT-NAME` is an OAuth client ID for either:
  - A UAA admin client created with `--authorities "pks.clusters.admin"`
  - The default admin client **Pks Uaa Management Admin Client**
- `CLIENT-SECRET` the OAuth client secret for the `--client-name` value above.
- `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
  ```

For information on how to create a UAA client, see Grant Enterprise PKS Access to a Client in *Managing Enterprise PKS Users with UAA*.

Export PKS API Access Token

This procedure stores a PKS API access token as an environment variable that you can use when executing PKS API calls on the

command line.

1. To export your access token into an environment variable, run the following command:

```
pks login -a PKS-API -u USER-ID -p 'PASSWORD' -k; \
export YOUR-ACCESS-TOKEN=$(bosh int ~/.pks/creds.yml --path /access_token)
```

   Where:

   - `PKS-API` is the FQDN of your PKS API endpoint. For example, `api.pks.example.com`.
   - `USER-ID` is your Enterprise PKS user ID.
   - `PASSWORD` is your Enterprise PKS password.
   - `YOUR-ACCESS-TOKEN` is the name of your access token environment variable.

   For example:

```
$ pks login -a pks.my.lab -u alana -p 'psswrdabc123...!' -k; \
export my_token=$(bosh int ~/.pks/creds.yml --path /access_token)
```

   Note: If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

Creating Clusters

In this topic

Page last updated:

This topic describes how to create a Kubernetes cluster with VMware Enterprise PKS using the PKS Command Line Interface (PKS CLI).

Configure Cluster Access

Cluster access configuration differs by the type of Enterprise PKS deployment.

vSphere with NSX-T

Enterprise 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 Enterprise PKS Deployments with NSX-T.

> **Note**: For a complete list of the objects that Enterprise 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 Enterprise 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 Enterprise PKS cluster load balancer, see the following:

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

Create the Enterprise 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. For more information, see the Grant Enterprise PKS Access to an Individual User section of *Managing Enterprise PKS Users with UAA*.

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** > **Enterprise PKS** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
   - `USERNAME` is your user name.

     See Logging in to Enterprise PKS for more information about the `pks login` command.

     > **Note:** If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

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. If you are using NSX-T, you can pre-provision the IP address to use for the Kubernetes API server load balancer using an available IP address from the floating IP pool and define a network profile to perform DNS lookup, or specify the IP address to use for load balancer on the command line. See Defining Network Profile for DNS Lookup of Pre-Provisioned IP Addresses for details.

- `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 Enterprise PKS 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 Enterprise PKS* topic for your IaaS, such as Installing Enterprise 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 Enterprise 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 Enterprise 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 <cluster-name>
```

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.

> **Note:** If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

8. To confirm you can access your cluster using the Kubernetes CLI, run the following command:

```
kubectl cluster-info
```

See Managing Enterprise PKS for information about checking cluster health and viewing cluster logs.

Identify Kubernetes Cluster Master VMs

> **Note**: This section applies only to Enterprise PKS deployments on GCP or on vSphere without NSX-T. Skip this section if your Enterprise PKS deployment is on vSphere with NSX-T, AWS, or Azure. For more information, see Load Balancers in

Enterprise 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** > **Enterprise PKS** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
   - `USERNAME` is your user name.

     See Logging in to Enterprise PKS for more information about the `pks login` command.

     > **Note:** If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

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 Enterprise PKS deployment is on AWS, you must tag your subnets with your new cluster's unique identifier before adding the subnets to the Enterprise PKS workload load balancer. After you complete the Create a Kubernetes Cluster procedure, follow the instructions in AWS Prerequisites.

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

Viewing Cluster Lists

Page last updated:

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** > **Enterprise PKS** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
   - `USERNAME` is your user name.

   See Logging in to Enterprise PKS for more information about the `pks login` command.

   > **Note:** If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

2. Run the following command to view the list of deployed clusters, including cluster names and status:

   ```
   pks clusters
   ```

   For example:

   ```
   $ pks clusters

   Name          Plan Name     UUID                                  Status    Action
   cluster-one    small        881543kd-64fg-7826-hea6-3h7g1o04kh0e  succeeded  CREATE
   cluster-two    small        951547dl-67kg-9631-bju8-7h9s3o98br0q  succeeded  CREATE
   ```

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

Viewing Cluster Details

Page last updated:

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** > **Enterprise PKS** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
   - `USERNAME` is your user name.

   See Logging in to Enterprise PKS for more information about the `pks login` command.

   > **Note:** If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

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
   ```

3. Run the following command to view additional details of an individual cluster, including NSX-T network details and Kubernetes settings details:

   ```
   pks cluster CLUSTER-NAME --details
   ```

   Replace `CLUSTER-NAME` with the unique name for your cluster. For example:

   ```
   $ pks cluster my-cluster --details
   ```

The following shows the sample output for an example Kubernetes cluster named `my-cluster`:

```
$ pks cluster my-cluster --details
PKS Version:              1.5.0-build.30
Name:                     my-cluster
K8s Version:              1.14.5
Plan Name:                small
UUID:                     4b1a9b6d-3594-4cad-ad0f-22043fb26480
Last Action:              CREATE
Last Action State:        succeeded
Last Action Description:  Instance provisioning completed
Kubernetes Master Host:   example-hostname
Kubernetes Master Port:   8443
Worker Nodes:             3
Kubernetes Master IP(s):  10.197.100.130
Network Profile Name:

NSXT Network Details:
  Load Balancer Size                    (lb_size):                "small"
  Nodes DNS Setting                     (nodes_dns):              ["10.142.7.2"]
  Node IP addresses are routable [no-nat]  (node_routable):       false
  Nodes subnet prefix                   (node_subnet_prefix):     24
  POD IP addresses are routable [no-nat]   (pod_routable):        false
  PODs subnet prefix                    (pod_subnet_prefix):      24
  NS Group ID of master VMs             (master_vms_nsgroup_id):  ""
  Tier 0 Router identifier              (t0_router_id):           "1e8371ac-1718-4617-8734-3975c6cd373b"
  Floating IP Pool identifiers          (fip_pool_ids):           ["901341c7-2e14-49d0-a3c1-66748664a062"]
  Node IP block identifiers             (node_ip_block_ids):      ["c5f0eb13-9691-4170-a9cd-c988f336ebd2"]
  POD IP block identifiers              (pod_ip_block_ids):       ["fead2c9a-96e8-4c5f-98a3-e797f06bc8d4"]
Kubernetes Settings Details:
  Set by Plan:
  Kubelet Node Drain timeout (mins)         (kubelet-drain-timeout):           0
  Kubelet Node Drain grace-period (seconds)  (kubelet-drain-grace-period):      10
  Kubelet Node Drain force                  (kubelet-drain-force):             true
  Kubelet Node Drain force-node             (kubelet-drain-force-node):        false
  Kubelet Node Drain ignore-daemonsets      (kubelet-drain-ignore-daemonsets):  true
  Kubelet Node Drain delete-local-data      (kubelet-drain-delete-local-data):  true
```

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

Viewing Cluster Plans

Page last updated:

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** > **Enterprise PKS** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
   - `USERNAME` is your user name.

   See Logging in to Enterprise PKS for more information about the `pks login` command.

   > **Note:** If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

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

In this topic

Page last updated:

This topic explains how to scale an existing cluster horizontally by adding worker nodes and vertically by changing the size of the node VMs.

warning: Do not change the number of master/etcd nodes for any plan that was used to create currently-running clusters. Enterprise PKS does not support changing the number of master/etcd nodes for plans with existing clusters.

Note: To change the default number of worker nodes created in new clusters, change your plan's **Worker Node Instances** setting. For more information, see Plans in the *Installing PKS* topic for your IaaS.

Scale Horizontally by Changing the Number of Worker Nodes Using the PKS CLI

Follow the steps below to scale an existing cluster using the PKS CLI by increasing or decreasing the number of worker nodes.

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** > **Enterprise PKS** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
   - `USERNAME` is your user name.

   See Logging in to Enterprise PKS for more information about the `pks login` command.

   Note: If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

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 Enterprise PKS 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.

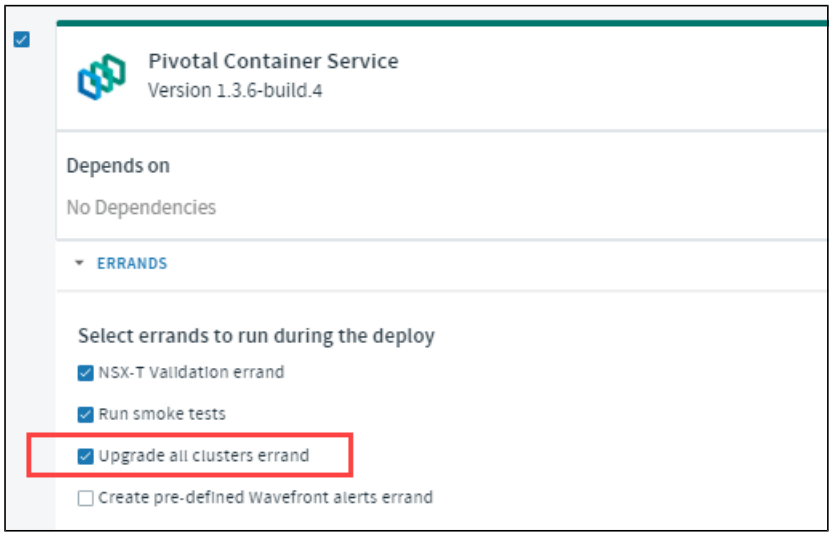
Scale Vertically by Changing Cluster Node VM Sizes in the PKS Tile

You can scale an existing cluster vertically by changing the size of the master or worker node VMs. When this is done, BOSH will recreate the VMs sequentially, one cluster at a time, and one node after another within the cluster. See VM Sizing for PKS Clusters for more information.

To change the size of a Kubernetes cluster node VM, complete the following steps:

1. Log in to Ops Manager.
2. Select the PKS tile.
3. Select the plan that is in use by the cluster(s) you want to resize.
4. To change the VM size:
   - For Master nodes, select the desired VM size from the **Master/ETCD VM Type** menu.
   - For Worker nodes, select the desired VM size from the **Worker VM Type** menu.

     Note: See Customize Master and Worker Node VM Size and Type for information on creating a custom VM size for use with a PKS cluster.
5. Click **Save** to preserve tile changes.
6. At the **Installation Dashboard**, click **Review Pending Changes**.
7. For the PKS tile, expand the **ERRANDS** list.
8. Select the **Update all clusters errand** if it is not already selected. You must ensure that **Update all clusters errand** is selected so that the cluster deployment manifest is regenerated after the plan is updated.



9. Click **Apply Changes**.

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

Upgrading Clusters

In this topic

Page last updated:

This topic describes how to upgrade Kubernetes clusters provisioned by VMware Enterprise PKS. Upgrading a cluster makes the cluster's internal PKS version match your PKS control plane's PKS version.

For information about updating clusters, which modifies a cluster's configuration, see pks update-cluster in the *PKS CLI* documentation.

Overview

When upgrading Enterprise PKS, you can choose one of the following approaches:

- Upgrade the Enterprise PKS tile and all Enterprise PKS-provisioned Kubernetes clusters simultaneously. This is the default behavior of the Enterprise PKS tile.
- Upgrade the Enterprise PKS tile only and then upgrade Enterprise PKS-provisioned Kubernetes clusters separately. If you want to upgrade your clusters separately, do the following:
  1. Review Limitations.
  2. Complete the steps listed in Prerequisites.
  3. Complete the steps listed in Upgrade Clusters.

> **Note:** When Enterprise PKS is set to upgrade the Enterprise PKS tile only and not the clusters, the Kubernetes cluster version falls behind the Enterprise PKS tile version. If the clusters fall more than one version behind the tile, Enterprise PKS cannot upgrade the clusters. The clusters must be upgraded to match the Enterprise PKS tile version before the next tile upgrade.

For more information about Enterprise PKS upgrades, see What Happens During Enterprise PKS Upgrades.

> **warning**: Do not change the number of master/etcd nodes for any plan that was used to create currently-running clusters. Enterprise PKS does not support changing the number of master/etcd nodes for plans with existing clusters.

Prerequisites

Before upgrading PKS-provisioned Kubernetes clusters:

1. Verify the cluster you are upgrading supports upgrading. For information, see Verify Your Clusters Support Upgrading in the *Upgrade Preparation Checklist for Enterprise PKS*.
2. Verify that your Kubernetes environment is healthy. For information, see Verifying Deployment Health.
3. Disable **Upgrade all clusters errand** in the Enterprise PKS tile. For more information, see Modify Errand Configuration in the Enterprise PKS Tile below.

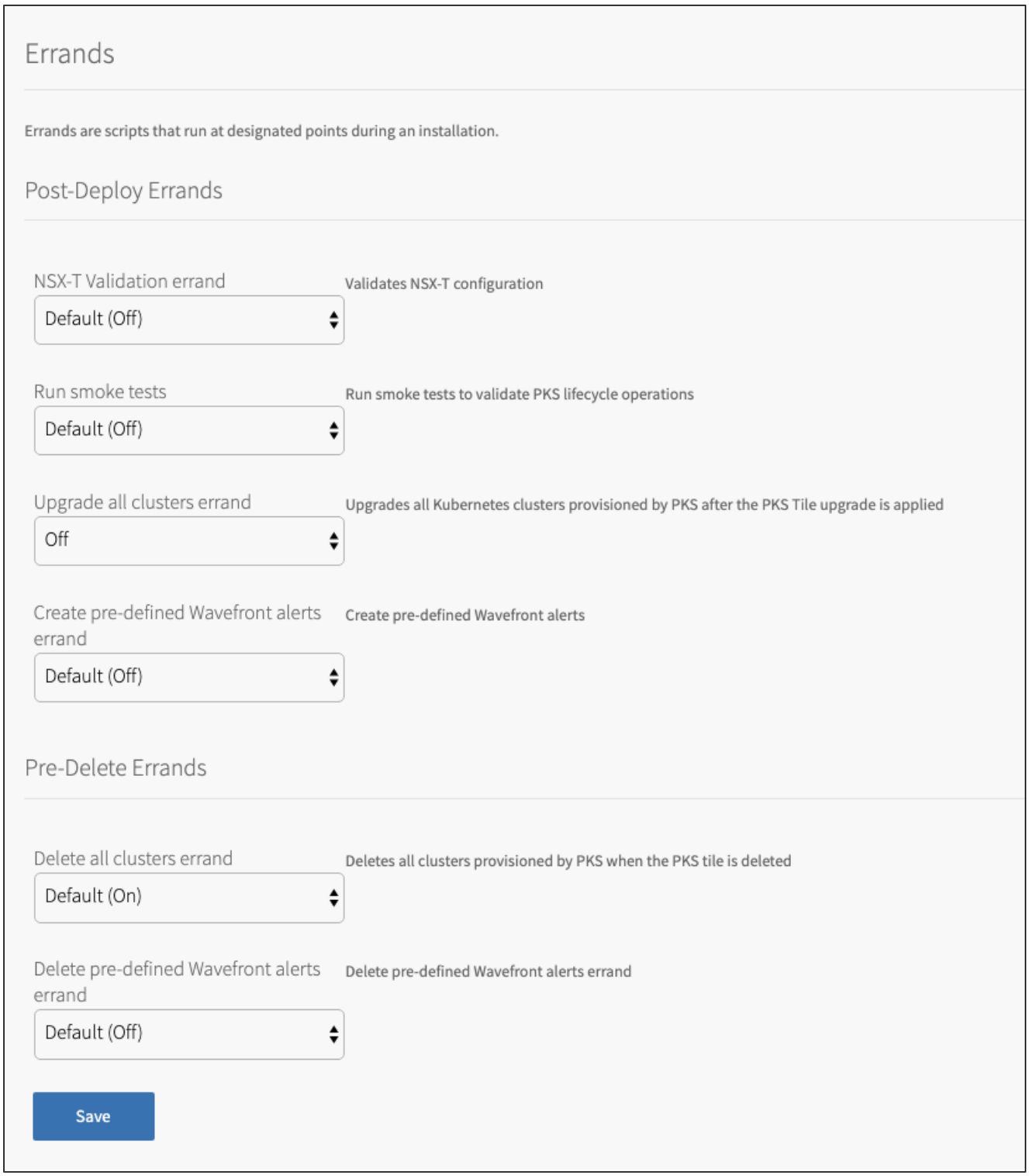
   > **warning:** If you disable **Upgrade all clusters errand** when upgrading the Enterprise PKS tile or updating your tile configuration, you must upgrade all your Kubernetes clusters before the next Enterprise PKS upgrade or tile configuration update.

4. Install the PKS command line interface (CLI) v1.6. For more information, see Installing the PKS CLI.
5. Log in to Enterprise PKS using `pks login`. For more information, see Logging in to Enterprise PKS.

Modify Errand Configuration in the Enterprise PKS Tile

To modify errand configuration when upgrading your Enterprise PKS version, perform the steps below. For more information about upgrading Enterprise PKS, see Upgrading Enterprise PKS.

1. If you have not already, import the product file for the new Enterprise PKS version to the Ops Manager **Installation Dashboard**.
2. In the Enterprise PKS tile, navigate to the **Errands** pane.
3. Disable **Upgrade all clusters errand** and click **Save**.



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

5. Select Enterprise PKS and any other products that you want to deploy and click **Apply Changes**.

Upgrade Clusters

You can upgrade your Enterprise PKS tile separately from upgrading your Enterprise PKS-provisioned Kubernetes clusters.

After upgrading your Enterprise PKS tile, upgrade individual or multiple clusters using the following procedures:

- Upgrade a Single Kubernetes Cluster
- Upgrade Multiple Kubernetes Clusters

To monitor or stop a cluster upgrade, follow the procedures in Manage Your Kubernetes Cluster Upgrade Job, below.

Upgrade a Single Kubernetes Cluster

The Enterprise PKS CLI provides `upgrade-cluster` for upgrading an individual Enterprise PKS-provisioned Kubernetes cluster.

To upgrade an individual Kubernetes cluster, run the following command:

```
pks upgrade-cluster CLUSTER-NAME
```

Where `CLUSTER-NAME` is the name of the Kubernetes cluster you want to upgrade.

To upgrade multiple clusters, see Upgrade Multiple Kubernetes Clusters, below.

For more information about the `pks upgrade-cluster` command, see pks upgrade-cluster in the *PKS CLI* documentation.

Note: The PKS CLI also provides the similar `pks update-cluster` command. Use `pks update-cluster` when modifying a cluster's configuration, such as to change a cluster's node count or to apply IaaS tagging to a cluster's VMs. For more information about `pks update-cluster`, see pks update-cluster in the *PKS CLI* documentation.

Upgrade Multiple Kubernetes Clusters

The Enterprise PKS CLI provides `upgrade-clusters` for upgrading multiple Enterprise PKS-provisioned Kubernetes clusters. You can upgrade clusters serially, serially with some clusters designated as canary clusters, or entirely in parallel.

To upgrade a single cluster, see Upgrade a Single Kubernetes Cluster above.

For more information about the `pks upgrade-clusters` command, see pks upgrade-clusters in the *PKS CLI* documentation.

Upgrade Multiple Kubernetes Clusters in Parallel

To upgrade multiple Kubernetes clusters, run the following command:

```
pks upgrade-clusters --clusters CLUSTER-NAMES --max-in-flight CLUSTER-COUNT --wait
```

Where:

- `CLUSTER-NAMES` is a comma-delimited list of the names of the Kubernetes clusters you want to upgrade.
- `CLUSTER-COUNT` is the maximum number of clusters to upgrade in parallel within an an AZ.
  - If the `CLUSTER-NAMES` list is longer than the `CLUSTER-COUNT`, the first set of clusters are upgraded in parallel and subsequent clusters are queued and then upgraded in parallel as the preceding cluster upgrades complete.
  - If an upgrade fails for a cluster in the `CLUSTER-NAMES` list, the job continues to a subsequent cluster in the list.
  - To run the cluster upgrade job as a background task, remove the `--wait` argument.

Note: Run `upgrade-clusters` with a `--max-in-flight` argument less than your BOSH Director > Director Config > **Director**

Workers value. For example, if your **Director Workers** value remains the default of `5`, run `upgrade-clusters` with a `--max-in-flight` argument value less than `5`.

Note: `max-in-flight` is an optional argument. If `max-in-flight` is not set, Enterprise PKS uses the default `max-in-flight` value of `1` and the clusters are upgraded serially.

For example:

```
$ pks upgrade-clusters --clusters k8-cluster-000,k8-cluster-001,k8-cluster-002 --max-in-flight 2  --wait


You are about to upgrade k8-cluster-000, k8-cluster-001 and k8-cluster-002.
Warning: This operation may be long running and may block further operations on the cluster(s) until complete


Continue? (y/n):y
Your taskID for the upgrade task is: d772aba0-2670-4fba-b26c-044b19d6ab60
Started upgrading cluster: k8-cluster-000
Started upgrading cluster: k8-cluster-001
Finished upgrading cluster: k8-cluster-000
Started upgrading cluster: k8-cluster-002
Finished upgrading cluster: k8-cluster-001
Finished upgrading cluster: k8-cluster-002
Upgrade task d772aba0-2670-4fba-b26c-044b19d6ab60 is done.
```

Upgrade Multiple Kubernetes Clusters With Canaries

To upgrade multiple clusters and automatically stop upgrading clusters if a cluster upgrade fails, specify your cluster list as canary clusters. You can specify one or more clusters as canary clusters.

To upgrade multiple clusters with one or more canary clusters, run the following command:

```
pks upgrade-clusters --canaries CANARY-CLUSTER-NAMES --clusters CLUSTER-NAMES --wait
```

Where:

- `CANARY-CLUSTER-NAMES` is a comma-delimited list of the names of the Kubernetes clusters you want to upgrade as canary clusters.
- `CLUSTER-NAMES` is a comma-delimited list of Kubernetes clusters to upgrade if all canary clusters successfully upgrade.
  - The specified canary clusters are upgraded prior to upgrading the clusters in your `CLUSTER-NAMES` list.
  - Canary clusters are always upgraded serially.
  - If an upgrade fails for a canary cluster, the entire `upgrade-clusters` job stops.
  - If an upgrade fails for a cluster in the `CLUSTER-NAMES` list, the `upgrade-clusters` job continues to a subsequent cluster in the list.
  - To run the cluster upgrade job as a background task, remove the `--wait` argument.

Note: `--clusters` is a required argument. To configure `upgrade-clusters` to stop for any cluster upgrade failure, specify only one cluster in your `CLUSTER-NAMES` list and the remaining clusters in your `CANARY-CLUSTER-NAMES` list.

Note: Canary clusters are always upgraded serially. Only the clusters specified in the `--clusters` list are upgraded in parallel when you run `upgrade-clusters` with both `--canaries` and `--max-in-flight` arguments.

For example:

```
$ pks upgrade-clusters --canaries k8-cluster-dev,k8-cluster-000,k8-cluster-001  --clusters k8-cluster-002  --wait


You are about to upgrade k8-cluster-dev k8-cluster-000, k8-cluster-001 and k8-cluster-002.
Warning: This operation may be long running and may block further operations on the cluster(s) until complete

Continue? (y/n):y
Your taskID for the upgrade task is: ce31a1bb-380a-453f-afa0-835ffa1ce6ac
Started upgrading cluster: k8-cluster-000
Upgrading cluster succeeded: k8-cluster-000
Started upgrading cluster: k8-cluster-001
Upgrading cluster succeeded: k8-cluster-001
Started upgrading cluster: k8-cluster-dev
Upgrading cluster failed: k8-cluster-dev
Upgrade task ce31a1bb-380a-453f-afa0-835ffa1ce6ac is done.
```

Manage Your Kubernetes Cluster Upgrade Job

You can use the PKS CLI to monitor and manage your Enterprise PKS-provisioned Kubernetes cluster upgrade jobs.

Monitor the Status of Your Kubernetes Clusters

To review the status of the actions being performed on your clusters, run the following command:

```
pks clusters
```

For example:

```
$ pks clusters


Upgrade is available to PKS Version: 1.6.0-build.14


PKS Version     Name            k8s Version  Plan Name  UUID                                   Status       Action
1.6.0-build.14  k8-cluster-000  1.14.5       small      9527ebaa-e2fa-422f-a52b-de3c3f0e39a4  succeeded    UPGRADE
1.6.0-build.14  k8-cluster-001  1.14.5       small      9527ebaa-e2fa-422f-a52b-de3c3f0e39a4  failed       UPGRADE
1.6.0-build.14  k8-cluster-002  1.14.5       small      9527ebaa-e2fa-422f-a52b-de3c3f0e39a4  in progress  UPGRADE
1.6.0-build.14  k8-cluster-003  1.14.5       small      9527ebaa-e2fa-422f-a52b-de3c3f0e39a4  queued       UPGRADE
```

Monitor the Status of Your Kubernetes Cluster Upgrade Job

To review the status of your `upgrade-clusters` job, run the following command:

```
pks task TASKID
```

Where `TASKID` is the ID of the task that was returned when you ran `pks upgrade-clusters`.

For example:

```
$ pks task ce31a1bb-380a-453f-afa0-835ffa1ce6ac

Your upgrade task is: done

Name          Status    Start time                  End time                    isCanary
k8-cluster-000 succeeded  Mon, 14 Oct 2019 12:00:00 PDT  Mon, 14 Oct 2019 12:19:54 PDT      true
k8-cluster-001 failed     Mon, 14 Oct 2019 12:20:00 PDT  ---                          true
```

Stop Your Kubernetes Cluster Upgrade Job

To cancel a running `upgrade-clusters` job, run the following PKS CLI command:

```
pks cancel-task TASKID
```

Where `TASKID` is the ID of the task that was returned when you ran `pks upgrade-clusters`.

warning: `pks cancel-task` does not cancel cluster upgrades currently in progress. This command only cancels a job's pending cluster upgrades.

Limitations

If you upgrade the Enterprise PKS tile only and then attempt to resize or update your existing Kubernetes clusters, these operations fail until you upgrade the clusters.

If you want to upgrade the Enterprise PKS tile and your existing Kubernetes clusters simultaneously, do not disable **Upgrade all clusters errand** in the Enterprise PKS tile.

After Upgrading Clusters

(Optional) Restore Cluster Sizing

If you scaled your cluster up for the upgrade and you prefer to restore your cluster to its original sizing, you can now scale the cluster back down to its previous configuration. VMware recommends that you not scale down your clusters and continue to run them with recommended configurations, reducing the chance of a future outage.

Deleting Clusters

In this topic

Page last updated:

This topic describes how to delete a Kubernetes cluster deployed by VMware Enterprise PKS. Running the `pks delete-cluster` command automatically deletes all cluster objects.

If you are using Enterprise 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** > **Enterprise PKS** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
   - `USERNAME` is your user name.

   See Logging in to Enterprise PKS for more information about the `pks login` command.

   > **Note:** If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

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 Enterprise 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 Enterprise PKS clusters.

   Note: If the cluster is not deleted, see Cluster Deletion Fails in *Troubleshooting*.

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.

Supporting Clusters

Page last updated:

This section describes how to support Kubernetes clusters provisioned by VMware Enterprise PKS.

See the following topics:

- Retrieving Cluster Credentials and Configuration
- Managing Cluster Access and Permissions
- Enabling, Disabling, and Using Admission Plugins for Enterprise PKS Clusters
- Configuring Ingress Routing
- Configuring Ingress Using the NSX-T Load Balancer
- Getting Started with VMware Harbor Registry
- Configuring Enterprise PKS Clusters with Private Docker Registry CA Certificates
- PersistentVolume Storage Options on vSphere
- Configuring and Using PersistentVolumes

Note: Enterprise PKS does not currently support the Kubernetes Service Catalog and the GCP Service Broker.

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

Retrieving Cluster Credentials and Configuration

In this topic

Page last updated:

This topic describes how to use the `pks get-credentials` command in VMware Enterprise 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** > **Enterprise PKS** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
   - `USERNAME` is your user name.

   See Logging in to Enterprise PKS for more information about the `pks login` command.

   > **Note:** If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML

Identity Provider

2. 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 <cluster-name>
```

Note: If you enable OpenID Connect (OIDC) in the Enterprise 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 Enterprise PKS for your IaaS.

Note: If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

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.

Managing Cluster Access and Permissions

In this topic

Page last updated:

This topic describes how to grant Kubernetes cluster access and namespace permissions to users in VMware Enterprise PKS.

Overview

Enterprise PKS admin users can grant other users permissions to specific clusters by using the PKS CLI and kubectl.

If you are an Enterprise PKS admin user, you can do the following:

- Grant user access to a cluster with a `ClusterRole` or a namespace within a cluster with a `Role`. See Grant Cluster Access to a User below.
- Grant LDAP or SAML group access to a cluster with a `ClusterRole` or a namespace within a cluster with a `Role`. See Grant Cluster Access to a Group below.

After you grant access, you must create a `ClusterRoleBinding` or a `RoleBinding` for the user you gave access to the cluster.

For more information, see RoleBinding and ClusterRoleBinding and Default Roles and Role Bindings in the Kubernetes documentation.

Prerequisites

Before setting up cluster access for users in Enterprise PKS, you must have the following:

- Access to an Enterprise PKS admin user account. For information on how to create PKS admin users, see Managing Enterprise PKS Users with UAA.
- A PKS API fully qualified domain name (FQDN) of your PKS deployment.
- A LDAP or SAML identity provider configured by the operator in the Enterprise PKS tile.

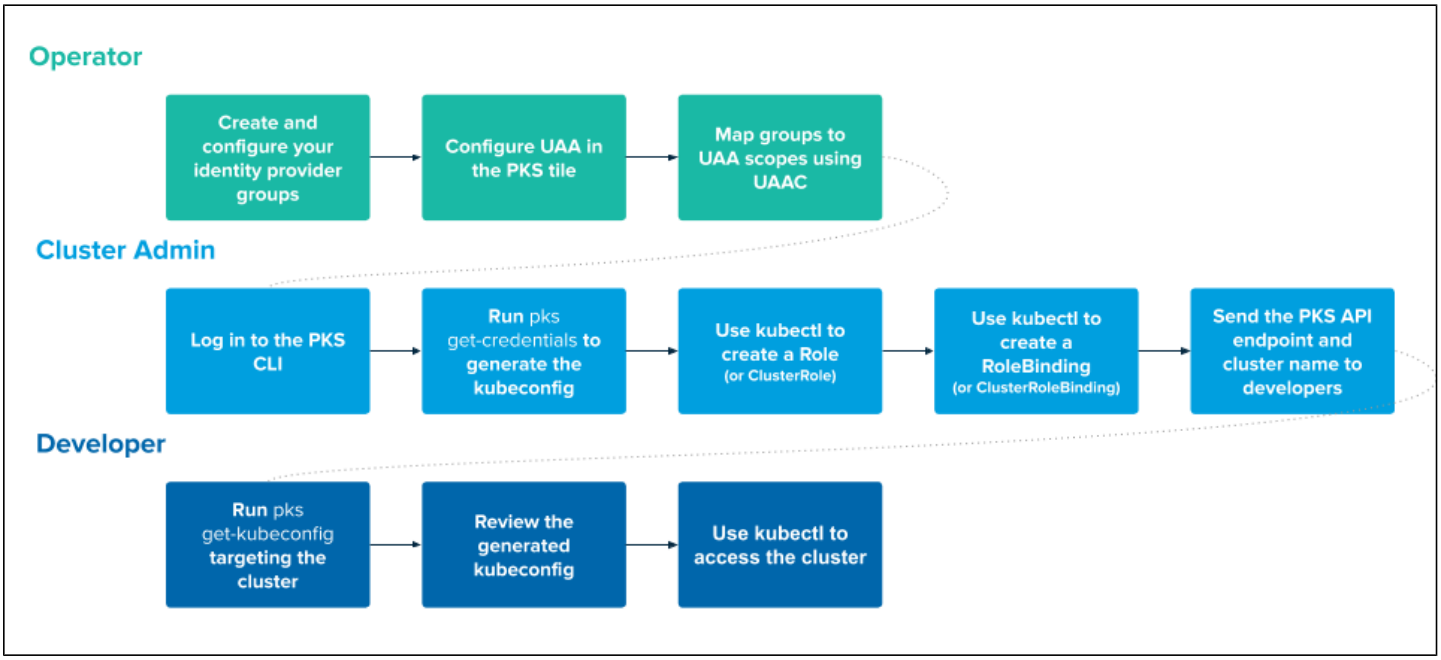
Grant Cluster Access to a User

After the cluster admin grants cluster access to end users, the Kubernetes end user can use the Kubernetes Command Line Interface (kubectl) to connect to the cluster. End users can only do actions that are permitted by the cluster admin. They cannot create, resize, or delete clusters.

Note: Before you can grant cluster access to Kubernetes end users, you must enable OpenID Connect (OIDC) by selecting

Enable UAA as OIDC provider in **Ops Manager Installation Dashboard** > **Enterprise PKS** > **Settings** > **UAA**. After you enable OIDC, you must run `pks get-credentials` to update your existing kubeconfig file.

The following diagram outlines the workflow to grant cluster access to a user who belongs to an identity provider group:



To grant cluster access to users, do the following:

1. Log in to PKS by running following command:

   ```
   pks login -u USERNAME -p PASSWORD -a PKS-API --ca-cert CERT-PATH
   ```

   Where:

   - `USERNAME` is your cluster admin username. This is the username created for your organization's LDAP or SAML identity provider.
   - `PASSWORD` is your cluster admin password.
   - `PKS-API` is the FQDN you use to access the PKS API.
   - `CERT-PATH` is the path to your root CA certificate. Provide the certificate to validate the PKS API certificate with SSL.

   Note: If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

2. Confirm that you can successfully connect to a cluster and use kubectl as a cluster admin by running the following command:

   ```
   pks get-credentials CLUSTER-NAME
   ```

   This step creates a `ClusterRoleBinding` for the cluster admin.

   Note: If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in

PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

3. When prompted, re-enter your password.

4. Create a YAML file with either the `Role` or `ClusterRole` for your Kubernetes end user. Use the following example as a template:

```
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. For example: `["pod-reader"]`
   - `API-REQUEST-VERB` is the request verb used to specify resource requests. For more information, see Determine the Request Verb in the Kubernetes documentation.

5. Create the `Role` or `ClusterRole` resource defined in your YAML file by running the following command:

```
kubectl create -f ROLE-CONFIGURATION.yml
```

   Where `ROLE-CONFIGURATION.yml` is the YAML file you created in the above step.

6. Create a YAML file containing either a `ClusterRoleBinding` or a `RoleBinding` for the Kubernetes end user. Use the following example as a template:

```
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 is given by the cluster admin.
   - `NAMESPACE` is the namespace within the cluster. This is omitted when creating a `ClusterRole`.
   - `USERNAME` is the Kubernetes end user username. This is the username created for your organization's LDAP or SAML

identity provider.
If you configured **UAA OIDC Username Prefix** in **Ops Manager Installation Dashboard** > **Enterprise PKS** > **Settings** > **UAA**, you must prepend `USERNAME` with the prefix you configured. For more information, see UAA in the *Installing* topic for your IaaS.
- `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. Create the `RoleBinding` or `ClusterRoleBinding` resource defined in your YAML file by running following command:

```
kubectl apply -f ROLE-BINDING-CONFIGURATION.yml
```

Where `ROLE-BINDING-CONFIGURATION.yml` is the YAML file you created in the above step.

8. Share the following information with your Kubernetes end users:
   - PKS API FQDN
   - Cluster name

Obtain Cluster Access as a User

To obtain cluster access, the end user must perform the following actions:

1. Fetch the kubeconfig file by running one of the following command:
   - If you want to validate the PKS API certificate with SSL, run the following command:

```
pks get-kubeconfig CLUSTER-NAME -u USERNAME -a PKS-API --ca-cert CERT-PATH
```

Where:

- `CLUSTER-NAME` is the cluster name provided by the cluster admin.
- `USERNAME` is the Kubernetes end user username. This is the username created for your organization's LDAP or SAML identity provider.
- `PKS-API` is the FQDN you use to access the PKS API.
- `CERT-PATH` is the path to your root CA certificate. Provide the certificate to validate the PKS API certificate with SSL.

For example:

```
$ pks get-kubeconfig my-cluster -u naomi -a api.pks.example.com \
  --ca-cert /var/tempest/workspaces/default/root_ca_certificate
```

   - If your CA is trusted and you want to skip SSL validation, run the following command:

```
pks get-kubeconfig CLUSTER-NAME -u USERNAME -a PKS-API -k
```

Where `-k` is the shortcut flag to skip SSL validation.
For example:

```
$ pks get-kubeconfig my-cluster -u naomi -a api.pks.example.com -k
```

Note: If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in

PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

2. When prompted, enter your password.

3. The PKS CLI generates a kubeconfig for the cluster you have access to. Review the following example kubeconfig file:

```
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: https://PROVIDED-BY-ADMIN:8443/oauth/token
        refresh-token:  PROVIDED-BY-USER
      name: oidc
```

4. Access the cluster using kubectl. For more information about kubectl commands, see Overview of kubectl in the Kubernetes documentation.

Grant Cluster Access to a Group

Cluster admins can grant access to an identity provider group by creating a `ClusterRoleBinding` or `RoleBinding` for that group. You can only grant access to an identity provider group if you use a LDAP or SAML identity provider for UAA. You can configure a LDAP or SAML identity provider in **Ops Manager Installation Dashboard** > **Enterprise PKS** > **Settings** > **UAA**.

Note: If you are using a LDAP group, you must confirm that the LDAP group you are giving access is in the allowlist in the Enterprise PKS tile. To do this, review **External Groups Whitelist** in **Ops Manager Installation Dashboard** > **Enterprise PKS** > **Settings** > **UAA**.

The procedure for granting cluster access to an identity provider group is similar to the procedure in Grant Cluster Access to a User above.

To grant cluster access to an identity provider group, do the procedure in Grant Cluster Access to a User above and replace step 6 with the following:

1. In the YAML file for a `ClusterRoleBinding` or a `RoleBinding`, replace the `subjects` section with the following:

```
subjects:
- kind: Group
  name: NAME-OF-GROUP
  apiGroup: rbac.authorization.k8s.io
```

Use the following example as a template:

```
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 the role binding. This is given by the cluster admin.
- `NAMESPACE` is the namespace within the cluster. This is omitted when creating a `ClusterRole`.
- `NAME-OF-GROUP` is the identity provider group name. This name is case sensitive.
  If you configured **UAA OIDC Groups Prefix** in **Ops Manager Installation Dashboard** > **Enterprise PKS** > **Settings** > **UAA**, you must prepend `NAME-OF-GROUP` with the prefix you configured. For more information, see UAA in the *Installing* topic for your IaaS.
- `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 binding the Group to.

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

Enabling, Disabling, and Using Admission Control Plugins for Enterprise PKS Clusters

Page last updated:

Topic provided by VMware

This section describes how to enable, use, and disable admission control plugins for VMware Enterprise PKS clusters.

For more information about Admission Controllers, see Using Admission Controllers in the Kubernetes documentation.

Enterprise PKS supports three admission control plugins. See the following topics for details on each:

- Enabling the PodSecurityPolicy Admission Plugin for Enterprise PKS Clusters and Using Pod Security Polices
- Enabling the DenyEscalatingExec Admission Plugin for Enterprise PKS Clusters
- Enabling the SecurityContextDeny Admission Plugin for Enterprise PKS Clusters

To disable one or more Admission Plugins, refer to the following topic:

- Disabling Admission Plugins for Enterprise PKS Clusters

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

Enabling and Configuring Pod Security Policies

In this topic

Page last updated:

This topic describes how to enable and use Pod Security Policies in VMware Enterprise PKS.

> **Note**: When the **PodSecurityPolicy** option is enabled for a Kubernetes cluster plan, a cluster administrator must define the policy, role, and role binding that gives cluster users (developers) permission to deploy pods to the cluster. See Workflow for Enabling PSPs.

About Pod Security Policies

In Kubernetes, a Pod Security Policy (PSP) is a cluster-level resource that controls security sensitive aspects of the pod specification. **PodSecurityPolicy** objects define a set of conditions that a pod must run with in order to be accepted into the system, as well as defaults for related fields. For more information, see Pod Security Policies in the Kubernetes documentation.

There are various ways to implement the use of PodSecurityPolicy objects in Kubernetes. One common approach is the use of role based access control (RBAC) objects: roles and role bindings. For a PodSecurityPolicy to take effect, the Kubernetes cluster User or ServiceAccount launching a workload must have the `use` permission on the desired PodSecurityPolicy object.

A role contains rules that represent a set of permissions. A role can be defined within a namespace with a **Role**, or cluster-wide with a **ClusterRole**. For more information, see Role and ClusterRole in the Kubenetes documentation.

A role binding grants the permissions defined in a role to a user or set of users. A role binding holds a list of subjects (users, groups, or service accounts), and a reference to the role being granted. Permissions can be granted within a namespace with a **RoleBinding**, or cluster-wide with a **ClusterRoleBinding**. For more information, see RoleBinding and ClusterRoleBinding in the Kubenetes documentation.

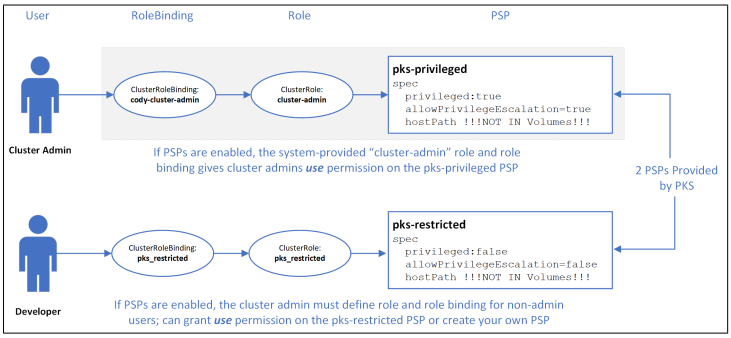
> **Note**: The most commonly used Kubernetes workloads (*deployment*, *replication controller*, for example) are spinning up pods using a service account. It is this entity that requires the *use* permission on the PSP. Just because the kubectl-user has the*use* permission on a PSP does not mean your workload will spin up (unless you are doing simple pod-level workloads using *kubectl run*). To grant a service account *use* permission for a PSP, the user attempting to create the role binding must also have*use* on that PSP.

Default Pod Security Policies in PKS

Enterprise PKS ships with two default PSPs: PKS Privileged and PKS Restricted.

PSP	Description
PKS Privileged	Allows privileged access to pod containers, which allows the container to do almost everything a host can do. See Privileged in the Kubernetes PSP documentation for more information.
PKS Restricted	Restricts privileged access to pod containers.

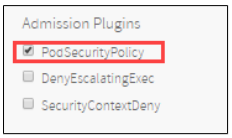
By default, when PSPs are enabled for a plan, the cluster administrator is bound to the PKS Privileged PSP. This policy grant gives the cluster administrator permission to deploy pods. Other users will not be able to deploy pods unless the cluster administrator creates and binds a PSP for such users. The PKS Restricted PSP is example for such purposes.



Note: The default PSPs will be updated and overwritten on upgrade. You should not edit the default PSPs.

Enabling Pod Security Policies in PKS

Enabling PSPs is done during configuration of Enterprise PKS in the Plan section of the tile configuration. Refer to the PKS tile configuration documentation for your IaaS for details.



A Kubernetes cluster created from a plan with the **PodSecurityPolicy** option enabled will require cluster users to have a binding that grants `use` on an appropriate PSP to deploy pods. Enabling the **PodSecurityPolicy** option is a security feature. The design goal is to make the Kubernetes cluster more secure. Once enabled, **PodSecurityPolicy** acts on creation and modification of pods in that cluster or namespace, and determines the actions that should be permitted based on the requested security context and the applied PSP. See PSP Policy Order for more information on PSP application order when multiple PSPs are in use.

WARNING: Enabling the **PodSecurityPolicy** option on install or upgrade of Enterprise PKS will prevent developers from using the cluster as they would expect unless the proper PSP, role, and role binding are configured by the cluster administrator before cluster deployment.

New Installations of PKS: PSPs Are Not Enabled for Any Plan

For new Enterprise PKS installations, the **PodSecurityPolicy** option is not enabled by default in a plan. If the **PodSecurityPolicy** option is enabled, the cluster administrator will be able to deploy pods, but developers without a binding to a PSP will not. For new deployments, the cluster administrator will need to create one or more PSPs, roles, and role bindings for developers to deploy pods. Once you enable PSPs, you will need to define the RBAC objects and PSP for cluster users. See Workflow for Enabling PSPs and Configuring the pks-restricted PSP for Developers to Use with PKS.

Upgrade of PKS: PSPs Must Be Enabled per Plan

On upgrade of Enterprise PKS, existing plans will not have the **PodSecurityPolicy** option enabled. If the PSP option is enabled for a plan that is in use and did not previously leverage PSPs, and the cluster is upgraded, the cluster administrator will need to create the appropriate PSPs, roles, and role bindings **BEFORE** upgrading the clusters associated to that plan. Cluster upgrades when enabling PSPs on an existing plan can have unpredictable results on workloads if the appropriate PSPs are not enabled. If you are considering enabling PSPs on an existing plan to apply to all associated clusters, the following set of tasks is recommended:

Workflow for Enabling PSPs

Before you select the **PodSecurityPolicy** check box and enable PSPs for a new or existing plan, complete the following tasks:

1. Review all existing pods to collect their security requirements for each cluster associated with that plan. There are some open-source tools to assist in this task, such as kube-psp-advisor.
2. Create the appropriate PSPs, roles, and role bindings for each cluster associated with that plan to allow the pods to run after upgrade. See Configuring PSP for Developers to Use as a starting point.
3. Enable PSPs by selecting the **PodSecurityPolicy** checkbox in the appropriate plan.
4. Review the pending changes and verify that **Upgrade all clusters errand** is enabled.
5. Apply the changes to update the clusters that use the plan with PSPs enabled.
6. Verify that the workloads are in desired state after upgrade.
7. If **Upgrade all clusters errand** was not enabled, run it manually and redeploy.

Alternatively, instead of enabling **Upgrade all clusters errand**, you can upgrade individual Kubernetes clusters through the PKS Command Line Interface (PKS CLI). For instructions on upgrading individual Kubernetes clusters, see Upgrading Clusters.

Configuring the pks-restricted PSP for Developers to Use with PKS

This section describes how to define and configure a pod security policy for developers to use to access a PKS provisioned cluster that has enabled PSPs.

At a high-level, the steps for configuring a PSP with appropriate RBAC controls are as follows:

1. Define the PSP.
2. Create a role.
3. Create a role binding.

Step 1: Define the PSP

The first step is to define the PSP. We provide the PSP named `pks-restricted` for general development work in Kubernetes. To onboard cluster users (developers), the best practice is to start with the `pks-restricted` PSP. To create your own PSP, refer to the Kubernetes documentation.

To view the `pks-restricted` PSP, run the following command:

```
kubectl get psp pks-restricted -o yaml
```

```
apiVersion: extensions/v1beta1
kind: PodSecurityPolicy
metadata:
  annotations:
    apparmor.security.beta.kubernetes.io/allowedProfileNames: runtime/default
    apparmor.security.beta.kubernetes.io/defaultProfileName: runtime/default
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"policy/v1beta1","kind":"PodSecurityPolicy","metadata":{"annotations":{"apparmor.security.beta.kubernetes.io/allowedProfileNames":"runtime/default","apparmor.security.beta.kubernetes.io/defaultProfileName":"runtime/default","seccomp.security.alpl
    seccomp.security.alpha.kubernetes.io/allowedProfileNames: docker/default
    seccomp.security.alpha.kubernetes.io/defaultProfileName: docker/default
  creationTimestamp: 2019-03-28T02:33:07Z
  name: pks-restricted
  resourceVersion: "261"
  selfLink: /apis/extensions/v1beta1/podsecuritypolicies/pks-restricted
  uid: d25bb0eb-5101-11e9-abc1-0a834658b12e
spec:
  allowPrivilegeEscalation: false
  fsGroup:
    ranges:
    - max: 65535
      min: 1
    rule: MustRunAs
  requiredDropCapabilities:
  - ALL
  runAsUser:
    rule: MustRunAsNonRoot
  seLinux:
    rule: RunAsAny
  supplementalGroups:
    ranges:
    - max: 65535
      min: 1
    rule: MustRunAs
  volumes:
  - configMap
  - emptyDir
  - projected
  - secret
  - downwardAPI
  - persistentVolumeClaim
```

Step 2: Create the Role

The following **ClusterRole** grants permission to use the `pks-restricted` PSP resource. You can use this role for onboarding cluster users and granting them cluster access.

Below is the system-provided **ClusterRole**.

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: psp:restricted
rules:
- apiGroups:
  - extensions
  resources:
  - podsecuritypolicies
  resourceNames:
  - pks-restricted # the psp we are giving access to
  verbs:
  - use
```

Note: Because this is a **ClusterRole** example, rather than a **Role** example, the namespace is omitted.

For more information about `Role` and `ClusterRole`, see Role and ClusterRole in the Kubernetes documentation.

Step 3: Create the Role Binding

For onboarding cluster users, you have two options based on the mode of authentication you have chosen for PKS, either internal or external:

- Internal authentication uses a service account mechanism for role binding, such as `SERVICE-UUID-cluster-admin`, for example.
- External authentication uses the OpenID Connect (OIDC) identity provider to interface with an external system such as LDAP or AD. In this case the role binding references the `user` or `group` name.

RoleBinding and **ClusterRoleBinding** configurations adhere to the following format:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ROLE-BINDING-TYPE
metadata:
  name: ROLE-BINDING-NAME
  namespace: NAMESPACE-NAME
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ROLE-REF-TYPE
  name: BIND-TO
subjects:
  - kind: ServiceAccount
    name: ACCOUNT-ID
```

Where:

- `ROLE-BINDING-TYPE` is the binding type, either `RoleBinding` or `ClusterRoleBinding`.
- `ROLE-BINDING-NAME` is your name for the role binding you are creating.
- `NAMESPACE-NAME` is the name of the namespace the binding grants access within. `namespace` is optional for a `ClusterRole` binding.

- `ROLE-REF-TYPE` is the `roleRef` type, either `Role` or `ClusterRole`.
  If the binding type is `Role`, the `namespace` property is required.
- `BIND-TO` is the name of the **ClusterRole** to bind to.
- `ACCOUNT-TYPE` is the type of account managed by the binding, either `ServiceAccount`, `User`, or `Group`.
- `ACCOUNT-ID` is the account managed by the binding. The `ACCOUNT-ID` is case sensitive.

For example, the following **ClusterRoleBinding** examples bind specific groups or users to the `psp:restricted` **ClusterRole**, which provides permissions to use the `pks-restricted` PSP:

- Example of binding a service account user:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: psp:restricted
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole # Must be Role or ClusterRole
  name: psp:restricted # The name of the ClusterRole to bind to
subjects:
  - kind: ServiceAccount
    name: cluster-user
```

- Example of binding an externally authenticated user named `jane`:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: psp:restricted
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole # This must be Role or ClusterRole
  name: psp:restricted # The name of the ClusterRole to bind to
subjects:
  - kind: User
    name: jane # Name is case sensitive
```

- Example of binding externally authenticated group members:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: psp:restricted
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole # This must be Role or ClusterRole
  name: psp:restricted # The name of the ClusterRole to bind to
subjects:
  - kind: Group
    name: developers # Name is case sensitive
```

Note: Because these are **ClusterRoleBinding** examples, rather than **RoleBinding** examples, the namespace property is omitted.

For more information about `RoleBinding` and `ClusterRoleBinding`, see RoleBinding and ClusterRoleBinding in the Kubernetes documentation.

Administering PSPs, Roles, and RoleBindings

This section lists common kubectl commands for administering and managing PSPs and related objects in Kubernetes.

To view the default PSPs:

```
kubectl get psp
```

```
NAME              PRIV    CAPS   SELINUX    RUNASUSER         FSGROUP     SUPGROUP    READONLYROOTFS   VOLUMES
pks-privileged     true    *     RunAsAny   RunAsAny          RunAsAny    RunAsAny    false            awsElasticBlockStore,azureDisk,azureFile,cephFS,configMap,csi,downwardAPI,emptyDir,fc,flexVolume,flocker,gcePersistentDisk,glusterfs,iscsi,nfs,persistentVolumeClai
pks-restricted     false          RunAsAny   MustRunAsNonRoot  MustRunAs   MustRunAs   false            configMap,emptyDir,projected,secret,downwardAPI,persistentVolumeClaim
```

To view the details of a particular PSP:

```
kubectl describe psp <psp-name>
```

For example:

```
kubectl describe psp pks-privileged
Name:  pks-privileged

Settings:
  Allow Privileged:                  true
  Allow Privilege Escalation:        true
  Default Add Capabilities:          <none>
  Required Drop Capabilities:        <none>
  Allowed Capabilities:              *
  Allowed Volume Types:              awsElasticBlockStore,azureDisk,azureFile,cephFS,configMap,csi,downwardAPI,emptyDir,fc,flexVolume,flocker,gcePersistentDisk,glusterfs,iscsi,nfs,persistentVolumeClaim,projected,portworxVolume,quobyte,rbd,scaleIO,secret,stora
  Allow Host Network:                true
  Allow Host Ports:                  0-65535
  Allow Host PID:                    true
  Allow Host IPC:                    true
  Read Only Root Filesystem:         false
  SELinux Context Strategy: RunAsAny
    User:                            <none>
    Role:                            <none>
    Type:                            <none>
    Level:                           <none>
  Run As User Strategy: RunAsAny
    Ranges:                          <none>
  FSGroup Strategy: RunAsAny
    Ranges:                          <none>
  Supplemental Groups Strategy: RunAsAny
    Ranges:                          <none>
```

```
kubectl describe psp pks-restricted
Name:  pks-restricted

Settings:
  Allow Privileged:                  false
  Allow Privilege Escalation:        false
  Default Add Capabilities:          <none>
  Required Drop Capabilities:        ALL
  Allowed Capabilities:              <none>
  Allowed Volume Types:              configMap,emptyDir,projected,secret,downwardAPI,persistentVolumeClaim
  Allow Host Network:                false
  Allow Host Ports:                  <none>
  Allow Host PID:                    false
  Allow Host IPC:                    false
  Read Only Root Filesystem:         false
  SELinux Context Strategy: RunAsAny
    User:                            <none>
    Role:                            <none>
    Type:                            <none>
    Level:                           <none>
  Run As User Strategy: MustRunAsNonRoot
    Ranges:                          <none>
  FSGroup Strategy: MustRunAs
    Ranges:                          1-65535
  Supplemental Groups Strategy: MustRunAs
    Ranges:                          1-65535
```

To view the role bindings:

```
kubectl get rolebinding
```

To view the cluster role bindings:

```
kubectl get clusterrolebindings
```

In the results, the `UUID-cluster-admin` is a service account.

```
NAME                                                   AGE
9a605c3d-b2f3-4961-b2a4-804885d85eba-cluster-admin     23s
9e495241-65b0-43bb-ac3a-120cb4ee6533-cluster-admin     2d1h
c990e9cc-29fc-4696-8673-fc69a93904f4-cluster-admin     2d1h
cluster-admin                                          8d
system:basic-user                                      8d
telemetry-agent                                        8d
```

To view service accounts:

```
kubectl get sa
```

```
NAME                                   SECRETS   AGE
2c92e23f-6480-4265-9456-4c81bffaa9c6   1         2d1h
38cb4886-8e19-4da3-a522-3ebaa3d6e553   1         44h
default                                1         8d
```

To view the YAML for a cluster role binding:

```
kubectl get clusterrolebinding 9a605c3d-b2f3-4961-b2a4-804885d85eba-cluster-admin -o yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  creationTimestamp: 2019-04-05T18:54:04Z
  labels:
    generated: "true"
  name: 9a605c3d-b2f3-4961-b2a4-804885d85eba-cluster-admin
  resourceVersion: "983229"
  selfLink: /apis/rbac.authorization.k8s.io/v1/clusterrolebindings/9a605c3d-b2f3-4961-b2a4-804885d85eba-cluster-admin
  uid: 2f24d7f0-57d4-11e9-abc1-0a834658b12e
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: 9a605c3d-b2f3-4961-b2a4-804885d85eba
  namespace: default
```

To view a cluster role:

```
kubectl get clusterrole cluster-admin
```

```
NAME          AGE
cluster-admin   8d
```

To view the YAML for a clusterrole:

```
kubectl get clusterrole cluster-admin -o yaml
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  annotations:
    rbac.authorization.kubernetes.io/autoupdate: "true"
  creationTimestamp: 2019-03-28T02:32:53Z
  labels:
    kubernetes.io/bootstrapping: rbac-defaults
  name: cluster-admin
  resourceVersion: "41"
  selfLink: /apis/rbac.authorization.k8s.io/v1/clusterroles/cluster-admin
  uid: c9c20648-5101-11e9-abc1-0a834658b12e
rules:
- apiGroups:
  - '*'
  resources:
  - '*'
  verbs:
  - '*'
- nonResourceURLs:
  - '*'
  verbs:
  - '*'
```

Troubleshooting PSP Configuration

This section provides some troubleshooting tips for working with PodSecurityPolicy.

Verify Policy Order

The source of many common problems related to **PodSecurityPolicy**–including mutating, workloads passing validation, and workloads failing at runtime–can usually be traced to proper policy ordering. If you are experiencing such issues, see Policy Order in the Kubernetes documentation.

Verify Use Permission Grants

The most commonly used Kubernetes workloads (`deployment` and `replication controller` , for example) are spinning up pods using a service account. It is this entity that requires the `use` permission on the PSP. To grant a service account `use` permission for a PSP, the user attempting to create the role binding must also have `use` on that PSP.

PSP Files and Demo

Enterprise PKS provides additional resources for implementing PSPs, including the two default PSPs provided with the product, example YAML files, and a PSP demo movie. Download the files at the following location: https://github.com/pivotal-cf/docs-pks/blob/master/demos/psp-demo.tar.gz.

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

Enabling the SecurityContextDeny Admission Plugin

In this topic

Page last updated:

Topic provided by VMware

This section describes how to enable the SecurityContextDeny admission controller for VMware Enterprise PKS clusters.

About the SecurityContextDeny Admission Plugin

The SecurityContextDeny admission controller plugin will deny any pod that attempts to set certain escalating Security Context fields.

In Kubernetes, a security context defines privilege and access control settings for a pod or container. The securityContext field is a PodSecurityContext object. For more information, see Set the security context for a Pod in the Kubernetes documentation.

When to Enable the SecurityContextDeny Admission Plugin

The SecurityContextDeny admission plugin should be enabled if a cluster does not use pod security policies (PSPs) to restrict the set of values a security context can take. See Enabling and Using Pod Security Policies for more information.

PSPs are the preferred method for providing a more secure Kubernetes environment. However, PSPs have administrative overhead. Enabling the SecurityContextDeny is a stopgap method of providing a more secure Kubernetes environment when it is not feasible to use PSPs. If you plan to use PSPs in the future, consider enabling the SecurityContextDeny admission plugin as an interim security measure.

Impact of Enabling the SecurityContextDeny Admission Plugin

This section describes the impact of enabling the SecurityContextDeny admission control plugin for new and existing cluster plans.

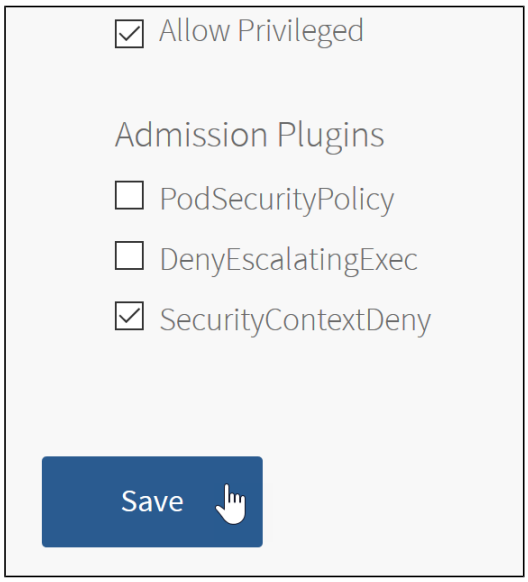
New Cluster. If you enable the SecurityContextDeny admission plugin in a plan and deploy a new Kubernetes cluster based on that plan, cluster users will not be able to create securityContext capabilities on that cluster.

Existing Cluster. If you enable the SecurityContextDeny admission plugin in a plan and update a Kubernetes cluster, cluster users will no longer be able to create securityContext capabilities on that cluster. This assumes you enable **Upgrade all clusters errand** or update your cluster individually through the PKS Command Line Interface (PKS CLI).

Enabling the SecurityContextDeny Admission Plugin

To enable the SecurityContextDeny admission plugin:

1. In the PKS tile, select the desired Plan, such as Plan 1.
2. At the bottom of the configuration panel, select the **SecurityContextDeny** option.



3. Click **Save**.
4. On the Installation Dashboard, click **Review Pending Changes**.
5. For Enterprise PKS, verify that **Upgrade all clusters errand** is enabled.
6. Click **Apply Changes** to deploy the cluster with the admission plugin enabled.

Alternatively, instead of enabling **Upgrade all clusters errand**, you can upgrade individual Kubernetes clusters through the PKS Command Line Interface (PKS CLI). For instructions on upgrading individual Kubernetes clusters, see Upgrading Clusters.

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

Enabling the DenyEscalatingExec Admission Plugin

In this topic

Page last updated:

Topic provided by VMware

This section describes how and when to enable the DenyEscalatingExec admission controller for VMware Enterprise PKS clusters.

About the DenyEscalatingExec Admission Plugin

The DenyEscalatingExec admission controller denies the “exec” and “attach” commands to pods that run with escalated privileges and allow host access. This includes pods that run as privileged, have access to the host Interprocess Communication (IPC) namespace, and have access to the host PID namespace.

See DenyEscalatingExec in the Kubernetes documentation for more information.

Note: The DenyEscalatingExec admission plugin is deprecated and is scheduled to be removed in a future Kubernetes release.

When to Enable the DenyEscalatingExec Admission Plugin

To provide better security when privileged containers are enabled, enable the DenyEscalatingExec admission controller or use PodSecurityPolicy. Privileged containers are enabled when **Allow Privileged** is selected.

Since the DenyEscalatingExec admission controller is being deprecated, the recommended approach is to use PodSecurityPolicy or a custom admission plugin that protects against the creation of overly privileged pods and that can be targeted at specific users or namespaces.

For more information, see Pod Security Policy.

Warning: If the DenyEscalatingExec admission plugin is enabled for a plan before upgrade, it remains enabled after upgrade.

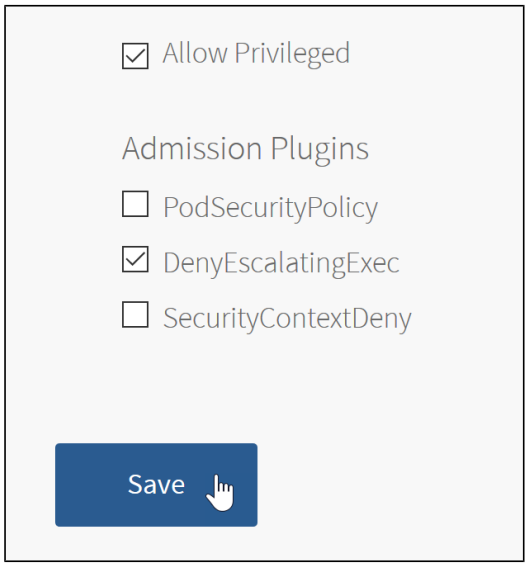
Impact of Enabling the DenyEscalatingExec Admission Controller

By selecting the **DenyEscalatingExec** checkbox, you make Kubernetes clusters deployed with the associated plan more secure.

Enabling the DenyEscalatingExec Admission Plugin

To enable the DenyEscalatingExec admission plugin, do the following:

1. In the Enterprise PKS tile, select the desired **Plan**, such as **Plan 1**.
2. At the bottom of the configuration panel, select the **DenyEscalatingExec** option.



3. Click **Save**.
4. On the Installation Dashboard, click **Review Pending Changes**.
5. For Enterprise PKS, verify that **Upgrade all clusters errand** is enabled.
6. Click **Apply Changes** to deploy clusters with the admission plugin enabled.

Alternatively, instead of enabling **Upgrade all clusters errand**, you can upgrade individual Kubernetes clusters through the PKS Command Line Interface (PKS CLI). For instructions on upgrading individual Kubernetes clusters, see Upgrading Clusters.

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

Disabling Admission Control Plugins for Enterprise PKS Clusters

In this topic

Page last updated:

Topic provided by VMware

This section describes how to disable one or more admission control plugins for VMware Enterprise PKS clusters. For more information, see Enabling and Using Admission Control Plugins for Enterprise PKS Clusters.

Disabling a Single Admission Control Plugin

To disable a single admission control plugin, do the following:

1. Log in to Ops Manager.
2. Click the Enterprise PKS tile.
3. Select the plan where you configured the admission control plugin, such as **Plan 1**.
4. Deselect the admission control plugin.
5. Click **Save**.
6. In the **Errands** pane, verify that **Upgrade all clusters errand** is enabled.
7. Return to **Installation Dashboard** and select **Review Pending Changes**.
8. Click **Apply Changes**.

Alternatively, instead of enabling **Upgrade all clusters errand**, you can upgrade individual Kubernetes clusters through the PKS Command Line Interface (PKS CLI). For instructions on upgrading individual Kubernetes clusters, see Upgrading Clusters.

Disabling an Orphaned Admission Control Plugin

The Ops Manager UI does not let you deselect (disable) all admission control plugins.

In other words, after an admission control plugin is enabled, the Ops Manager UI requires that at least one admission control plugin checkbox is selected (enabled).

To disable an orphaned Admission control Plugin, complete the following workflow:

1. Obtain the FQDN, user name and password of your Ops Manager.
2. Authenticate into the Ops Manager API and retrieve a UAA access token to access Ops Manager. For more information, see Using the Ops Manager API.

3. Obtain the BOSH deployment name for the Enterprise PKS tile by doing one of the following options:

   a. Option 1: Use the Ops Manager API:

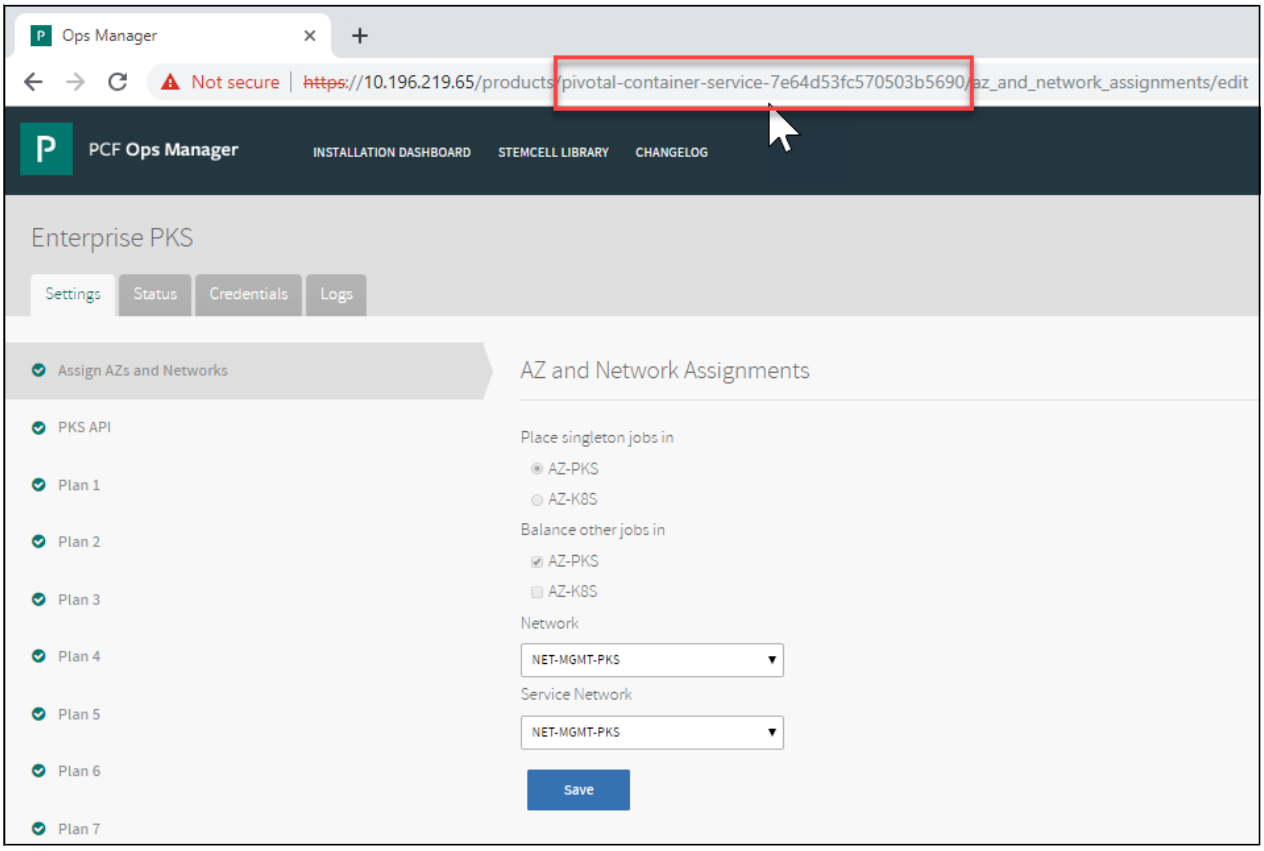
      i. In a terminal, run the following command:

```
curl -i "https://OPS-MAN-FQDN/api/v0/staged/products" -X GET -H "Authorization: Bearer UAA-ACCESS-TOKEN" -k
```

      ii. In the output, locate the `installation_name` that begins with `pivotal-container-service`.
      iii. Copy the entire BOSH deployment name, including the unique GUID. For example, `pivotal-container-service-4b48fc5b704d54c6c7de`.

   b. Option 2: Use the Ops Manager UI:

      i. In Ops Manager, click on the Enterprise PKS tile.
      ii. Copy the BOSH deployment name including the GUID from the URL:



      The deployment name contains “pivotal-container-service” and a unique GUID string. For example, `pivotal-container-service-4b48fc5b704d54c6c7de`.

4. To disable the orphaned admission control plugin, run the following Ops Manager API command:

```
curl -i "https://OPS-MAN-FQDN/api/v0/staged/pivotal-container-service-GUID/properties" \
-H "Authorization: Bearer UAA-ACCESS-TOKEN" \
-X PUT -d '{"properties": {".properties.PLAN-NUMBER_selector.active.admission_plugins":{"value":[]}}}' \
-H "Content-Type: application/json"
```

   Where:

   - `OPS-MAN-FQDN` is the URL of your Ops Manager.
   - `pivotal-container-service-GUID` is the BOSH deployment name of your Enterprise PKS that you retrieved earlier in this

procedure.

- `UAA-ACCESS-TOKEN` is the UAA token you retrieved earlier in this procedure.
- `PLAN-NUMBER` is the plan configuration you want to update. For example, `plan1` or `plan2`.

For example:

```
$ curl -i "https://pcf.example.com/api/v0/staged/products/pivotal-container-service-4b48fc5b704d54c6c7de/properties" \
-H "Authorization: Bearer aBcdEfg0hIJKlm123.e" \
-X PUT -d '{"properties": {".properties.plan1_selector.active.admission_plugins":{"value":[]}}}' \
-H "Content-Type: application/json"
```

5. From the output, verify that the command returns a `HTTP 200` status code.

6. Validate your manifest change in the Ops Manager UI. Do the following:

   a. Log in to Ops Manager.
   b. Select **Review Pending Changes**.
   c. On the Review Pending Changes pane, navigate to the Enterprise PKS section and select **SEE CHANGES**.
   d. Verify that the admission control plugins are displayed as removed in the **Manifest** section. For example:

7. Click **Apply Changes**.

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

Configuring Ingress Routing

In this topic

Page last updated:

This topic provides resources for configuring an ingress controller on VMware Enterprise 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. To set the kubectl context for the cluster where you want to deploy the ingress controller, run the following command:

   ```
   pks get-credentials CLUSTER-NAME
   ```

   Where `CLUSTER-NAME` is the name of your PKS-deployed Kubernetes 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 <cluster-name>
   ```

   Note: If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

2. To verify a DNS service is enabled for your Kubernetes cluster, run the following command:

   ```
   kubectl cluster-info
   ```

   If a DNS service is enabled, the DNS service's URL is included in the `kubectl cluster-info` output.

   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
   CoreDNS is running at https://104.197.5.247/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
   grafana is running at https://104.197.5.247/api/v1/namespaces/kube-system/services/monitoring-grafana/proxy
   ```

   The current default Kubernetes cluster DNS service is `CoreDNS`. The example output above includes the URL for this DNS service, indicating it is running.

   If a DNS service is not running for your cluster, enable the `CoreDNS` service:

   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.

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

Configuring Ingress Using the NSX-T Load Balancer

Page last updated:

This section provides topics for configuring the NSX-T load balancer used for ingress resources.

Layer 7 load balancing is implemented via a Kubernetes ingress resource. The ingress is allocated an IP from the Floating IP Pool specified in the NSX-T configuration. NCP exposes the ingress load balancer service on this IP address for both the HTTP and HTTPS ports (port 80 and 443).

Configuring Ingress Using the NSX-T Load Balancer

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

Configuring Ingress Resources and Load Balancer Services

In this topic

Page last updated:

This topic describes example configurations for ingress routing (Layer 7) and load balancing (Layer 4) for Kubernetes clusters deployed by VMware Enterprise PKS on vSphere with NSX-T integration.

Note: The examples in this topic are based on NCP v2.3.2.

About Ingress Routing

A Kubernetes ingress resource exposes HTTP and HTTPS routes from outside the cluster to services within the cluster. Traffic routing is controlled by rules defined on the ingress resource. See the Kubernetes documentation for more information about ingress resources.

The NSX-T load balancer that is automatically provisioned by Enterprise PKS provides two Layer 7 virtual servers for Kubernetes ingress resources, one for HTTP and the other for HTTPS. For more information, see Supported Load Balancer Features in the NSX-T documentation.

You define ingress resource configuration in the manifest of your Kubernetes deployment. Use wildcard DNS entries to route traffic to the exposed ingress resource.

How Kubernetes Applies Ingress Rules

When you define an ingress rule, the hostname and path values are both optional. It is common to define an ingress rule that specifies a hostname and no path, but defining an ingress rule without a hostname is uncommon.

When you define two ingress rules with the same hostname, include both the hostname and path in the ingress rule. If multiple ingress rules use the same hostname and the same path, the first rule you create takes priority.

For example, see the two ingress rules below.

Ingress Rule Example 1

In this example, the NSX ingress rule matches `host: test.com` and `path: /testpath` for the incoming request.

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: svc-ingress1
spec:
  rules:
  - host: test.com
    http:
      paths:
      - path: /testpath
        backend:
          serviceName: svc1
          servicePort: 80
```

Ingress Rule Example 2

In the following example, the NSX ingress rule matches `host: test.com` for the incoming request with no path.

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: svc-ingress2
spec:
  rules:
  - host: test.com
    http:
      paths:
      - path:
        backend:
          serviceName: svc1
          servicePort: 80
```

If you create **Ingress Rule Example 1** before **Ingress Rule Example 2**, then `svc-ingress1` serves the `test.com/testpath` URI because inbound requests hit the `host: test.com` and `path: /testpath` NSX ingress rule first.

If you create **Ingress Rule Example 2** before **Ingress Rule Example 1**, then `svc2-ingress2` serves the `test.com/testpath` URI because inbound requests hit the `host: test.com` NSX ingress rule first.

About the Load Balancer Service

When deploying a Kubernetes service `type: LoadBalancer`, NSX-T automatically creates a new virtual IP address (VIP) on the existing load balancer. NSX-T supports autoscaling, which spins up a new `LoadBalancer` service if the previous one has reached its scale limit.

Load Balancer Example

The Kubernetes `LoadBalancer` service definition supports an integer or a string value for the named `targetPort` attribute. For example, `8080` or `http`. For more information, see Type LoadBalancer in the Kubernetes documentation.

> **Note**: With NCP v2.3.2 and earlier, the named `targetPort` must be an integer, not a string. If you define a service `type: LoadBalancer` with NSX-T, the value of `targetPort` must be a port number, not a port name.

You must specify a listening port and a target port in the service. The port name is not mandatory for a single port service. For multi-port services, Kubernetes requires a port name. If the protocol is not specified it defaults to TCP.

The following example shows a `LoadBalancer` service definition for an Enterprise PKS-provisioned cluster with NSX-T:

```
kind: Service
apiVersion: v1
metadata:
  name: test-service
spec:
  type: LoadBalancer
  selector:
    app: testApp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8080
    name: web
```

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

Viewing and Troubleshooting the Health Status of Cluster Network Objects (NSX-T only)

In this topic

Page last updated:

This topic describes how cluster managers and users can troubleshoot NSX-T networking errors using the `kubectl nsxerrors` command.

About the NSX Errors CRD

The NSX Errors CRD gives you the ability to view errors related to NSX-T that may occur when applications are deployed to a PKS-provisioned Kubernetes cluster. Previously, NSX-T errors were logged in NCP logs on the master nodes, which cluster users do not have access to. The NSX Errors CRD improves visibility and troubleshooting for cluster managers and users.

The NSX Errors CRD creates a `nsxerror` object for each Kubernetes resource that encounters an NSX error during attempted creation. In addition, the Kubernetes resource is annotated with the `nsxerror` object name. The NSX Error CRD provides the command `kubectl nsxerrors` that lets you view the NSX errors encountered during resource creation. The `nsxerror` object is deleted once the NSX error is resolved and the Kubernetes resource is successfully created.

Errors Reported by the NSX Errors CRD

The following errors are reported by the NSX Errors CRD:

- Auto-scaler failed to allocate additional load balancer service due to Edge Node limit
- Number of pools exceed the load balancer service limit
- Number of pool members exceed the load balancer service and Edge Node limit
- Floating IP pool is exhausted when exposing the load balancer type service
- Pod IP Block is exhausted
- The number of available IP allocations is low
- The NSX manager is unavailable
- The NSX manager rate limit is exceeded

NSX Errors CRD Example

To illustrate how the NSX Errors CRD works and can be used, consider the following example: the NSX auto-scaler fails to allocate additional load balancer services due to Edge Node limits reached. In this case, the number of virtual switches exceed load balancer service limits with auto-scaling enabled.

The resource is fetched by name to check its status.


```
# kubectl get svc test-svc-3
test-svc-3     LoadBalancer 10.104.236.243   <pending> 80:32095/TCP,8080:32664/TCP   4
```

The status is pending so we look at the annotations. The `ncp/error` and `nsxerror` annotations are visible.

```
# kubectl get svc test-svc-3 –o yaml
annotations:
    ncp/error.loadbalancer: SERVICE_LOADBALANCER_UNREALIZED
    Nsxerror: services-1f48fa28c17d983bc73c33f005611e0c
```

We use the command `kubectl get nsxerror` to view the details of the error, revealing that the number of load balancer virtual server instances requested exceeds the limits of the Edge Node.

```
# kubectl get nsxerror services-1f48fa28c17d983bc73c33f005611e0c
- apiVersion: vmware.eng.com/v1
  kind: NSXError
  metadata:
    clusterName: ""
    creationTimestamp: 2019-01-22T03:17:16Z
    labels:
      error-object-type: services
    name: services-1f48fa28c17d983bc73c33f005611e0c
    namespace: ""
    resourceVersion: "1291084"
    selfLink: /apis/vmware.eng.com/v1/services-1f48fa28c17d983bc73c33f005611e0c
    uid: 386e60e5-1df4-11e9-abd8-000c29c02b4c
  spec:
    error-object-id: default.test-svc-1
    error-object-name: test-svc-1
    error-object-ns: default
    error-object-type: services
    message: [2019-01-21 19:17:16]10087: Number of loadbalancer requested exceed Edge node limit'
```

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

Scaling Ingress Resources (NSX-T only)

In this topic

Page last updated:

This topic describes how to monitor the health status of the NSX-T load balancer and scale ingress resources.

> **Note:** This feature requires NCP v2.5.1 or later.

About Load Balancer Monitoring and Ingress Scaling

For each Kubernetes cluster provisioned by Enterprise PKS with NSX-T, there are 2 layer 7 virtual servers (HTTP and HTTPS) attached to the cluster load balancer service that perform ingress routing.

Using custom resources, Enterprise PKS v1.6 gives you the ability to monitor the state of the NSX-T load balancer service and scale the virtual servers created for ingress.

Monitoring the NSX-T Load Balancer Service

You can use the NSXLoadBalancerMonitor CRD to monitor the NSX-T load balancer service, including traffic, usage and health score information.

For each load balancer service created by the NSX-T load balancer, NCP (by way of the CRD) creates a corresponding NSXLoadBalancerMonitor object:

- 1 layer 4 load balancer created by default for the Kubernetes master nodes
- 1 layer 4 auto-scaled load balancer created for **each** service resource
- 2 layer 7 ingress load balancers which can be interactively scaled

For each type of load balancer, the NSXLoadBalancerMonitor returns statistics showing the number of connections and throughput of the virtual servers. In addition, a **health score** is calculated that reflects the current performance of that load balancer.

If the health score is poor for one of the layer 4 load balancers, you can use a network profile to increase the size of the NSX-T load balancer service.

If the health score is poor for the layer 7 ingress load balancers, you can use the LoadBalancer CRD to manually scale ingress.

Refer to the NSXLoadBalancerMonitor Actions table for details you can take based on the results of the NSXLoadBalancerMonitor.

Using the NSXLoadBalancerMonitor CRD

To run the NSXLoadBalancerMonitor CRD, issue the following set of `kubectl` commands.

Issue the following command to view the NSXLoadBalancerMonitor CRD:

```
kubectl get crd
```

Issue the following command to get the UUID of the NSX-T load balancer deployed for the cluster:

```
kubectl get nsxlbmonitors
```

Issue the following command to view statistics, throughput, and health score for all virtual servers deployed by a specific load balancer service:

```
kubectl describe nsxlbmonitors UUID-OF-LOAD-BALANCER
```

Example Results Returned by the NSXLoadBalancerMonitor CRD

The following examples shows the results returned by the NSXLoadBalancerMonitor CRD for an example NSX-T load balancer service.

```
$ kubectl describe nsxlbmonitor f61a8cec-28eb-4b0c-bf4a-906f3ce2d8e6
Name:         f61a8cec-28eb-4b0c-bf4a-906f3ce2d8e6
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  vmware.com/v1alpha1
Health:
  Metrics:
    Cpu Usage Percentage:         0
    Poolmember Usage Percentage:  1
  Service Pressure Index:         0,LOW
  Infra Pressure Index:           0,LOW
  Metrics:
    Cpu Usage Percentage:         0
    Lb Service Usage Percentage:  0
    Memory Usage Percentage:      0
    Poolmember Usage Percentage:  0
Kind:                             NSXLoadBalancerMonitor
Metadata:
  Creation Timestamp:  2019-11-13T19:37:10Z
  Generation:          914
  Resource Version:    17139
  Self Link:           /apis/vmware.com/v1alpha1/nsxlbmonitors/f61a8cec-28eb-4b0c-bf4a-906f3ce2d8e6
  UID:                 f56d3cf5-748d-44c3-8026-c6c569fde954
Traffic:
  Bytes In Rate:         0
  Bytes Out Rate:        0
  Current Session Rate:  0
  Ip Address:            192.168.160.102
  Max Sessions:          0
  Packets In Rate:       0
  Packets Out Rate:      0
  Protocol:              TCP
  Total Sessions:        0
  Virtual Server Name:   pks-042bccde-2197-4e06-863e-55129bf2e195-http
  Bytes In Rate:         0
  Bytes Out Rate:        0
  Current Session Rate:  0
  Ip Address:            192.168.160.102
  Max Sessions:          0
  Packets In Rate:       0
  Packets Out Rate:      0
  Protocol:              TCP
  Total Sessions:        0
  Virtual Server Name:   pks-042bccde-2197-4e06-863e-55129bf2e195-https_terminated
Usage:
  Current Server Pool Count:     1
  Current Virtual Server Count:  3
Events:                          <none>
```

NSXLoadBalancerMonitor Actions

The NSXLoadBalancerMonitor CRD returns two health scores:

- `servicePressureIndex` which represents an overall health score for the NSX-T load balancer service
- `infraPressureIndex` which represents the heath score of the NSX-T Edge Node that is running the load balancer and associated virtual servers

Based on the health score the user can decide what action to take. The table below summarizes the actions that you can take based on the health scores.

servicePressureIndex	infraPressureIndex	Cluster Manager	Infrastructure Admin
LOW or WARM	LOW or WARM	NONE	NONE
LOW or WARM	HIGH	Alert infra admin	Move the LBS from the CRITICAL Edge Node to another Edge Node.
HIGH	LOW or WARM	Resolve the LBS health score by scaling the ingress LB and, if necessary, by increasing the size of the LBS using network profile.	NONE
HIGH	HIGH	Alert infra admin; Resolve the LBS health score by scaling the ingress LB and, if necessary, by increasing the size of the LBS using network profile.	Move the LBS from the CRITICAL Edge Node to another Edge Node.

Scaling Ingress Using the LoadBalancer CRD

The LoadBalancer CRD provides you with an interactive method to scale the load balancer for ingress routing. After you have used the NSXLoadBalancerMonitor CRD to check the health status of the load balancer service, you can use the LoadBalancer CRD to create a new ingress load balancer. Then you annotate the Kubernetes K8S ingress resource with the newly created ingress load balancer. NCP will attach the ingress rules to the scaled out load balancer.

LoadBalancer CRD Example

The following example creates a new ingress load balancer.

```
# kubectl apply –f lb.yaml
apiVersion: vmware.com/v1alpha1
kind: LoadBalancer
metadata:
  name: cluster1-lbs0                    # display name of the loadBalancer
spec:
  httpConfig:                            # optional config to support http/https route on the loadBalancer. Set as httpConfig: {} to apply default settings.
    virtualIP:                           # optional, defaults to auto_allocate
    port: 233                            # optional, defaults to 80
    tls:
      port: 2333                         # optional, defaults to 443
      secretName: default_secret         # optional, defaults to nil
      secretNamespace: default           # optional, defaults to nil
    xForwardedFor: INSERT                # optional, available values are INSERT, REPLACE, defaults to nil
    affinity:
      type: source_ip                    # optional, available values are sourceIP, cookie
      timeout: 100                       # optional, defaults to 10800
  size: MEDIUM                           # optional, defaults to SMALL
  virtualNetwork: virtualnetwork1        # optional, defaults to nil
status:
  httpVirtualIP: <realized external ip>   # auto-allocated or user desired external ip for http/https virtual server
```

Note: You must deploy the new ingress load balancer in the same namespace where you deploy the ingress resource.

The following example annotates the Kubernetes ingress resource with the newly created ingress load balancer.

```
# kubectl apply –f ingress.yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: svc-ingress1
  annotations:
    nsx/loadbalancer: cluster1-lbs0       # loadbalancer-name-that-you-gave-in-crd
spec:
  rules:
  - host: test.com
    http:
      paths:
      - path: /testpath
          backend:
            serviceName: svc1
            servicePort: 80
```

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

Ingress Support for URL Rewrite (NSX-T only)

In this topic

Page last updated:

This topic describes how to perform URL rewrite for Kubernetes ingress resources.

About Support for URL Rewrite for Ingress Resources

Enterprise PKS with NSX-T supports ingress URL path rewrite using NSX-T v2.5.1+ and NCP v2.5.1+.

All the ingress paths will be rewritten to the provided value. If an ingress has annotation `ingress.kubernetes.io/rewrite-target: /` and has path `/tea`, for example, the URI `/tea` will be rewritten to `/` before the request is sent to the backend service. Numbered capture groups are supported.

URL Rewrite Example

The following example shows how to implement URL rewrite.

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: cafe-ingress
  annotations:
    kubernetes.io/ingress.class: "nsx"
    ncp/use-regex: "True"
    #/tea/cup will be rewritten to /cup before sending request to endpoint
    ingress.kubernetes.io/rewrite-target: /$1
spec:
  rules:
  - host: cafe.example.com
    http:
      paths:
      - path: /tea/(.*)
        backend:
          serviceName: tea-svc
          servicePort: 80
      - path: /coffee/(.*)
        backend:
          serviceName: coffee-svc
          servicePort: 80
```

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

Getting Started with VMware Harbor Registry

In this topic

Page last updated:

This topic describes VMware Harbor Registry, an enterprise-class image registry server that stores and distributes container images for VMware Enterprise PKS.

Overview

Harbor allows you to store and manage container images for your Enterprise PKS deployment. Deploying an image registry alongside Enterprise PKS improves image transfer speed.

As an enterprise private registry, Harbor also offers enhanced performance and improved security. By configuring Harbor with Enterprise PKS, you can apply enterprise features to your image registry, such as security, identity, and management.

You can install Harbor alongside Enterprise PKS on vSphere, Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure.

Install Harbor

To install Harbor, do the following:

1. Install Enterprise PKS. See the *Installing Enterprise 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 Managing Users in the Harbor documentation.
2. Create a Harbor project that contains all repositories for your app. See Managing projects in the Harbor documentation.

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 in the Docker Client in the Harbor documentation.
- Manage Helm charts in your Harbor project using either the Harbor portal or the Helm CLI. See Managing Helm Charts in the Harbor documentation.

- 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 Working with Images, Tags, and Helm Charts in the Harbor documentation.

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 Harbor Administration in the Harbor documentation.

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

Configuring Enterprise PKS Clusters with Private Docker Registry CA Certificates (Beta)

In this topic

Page last updated:

This topic describes how to configure VMware Enterprise PKS Kubernetes clusters with private Docker registry SSL Certificate Authority (CA) certificates.

Overview

Docker allows you to store Docker images in private registries and secures the registries with SSL CA certificates. You can enable your Enterprise PKS Kubernetes clusters to authenticate into your private Docker registries by configuring your clusters with SSL CA certificates.

You can configure both new and existing Enterprise PKS clusters to have Docker registry CA certificates.

Note: Only Linux clusters can be configured to have Docker registry CA certificates.

To create a new cluster configured with Docker registry SSL CA certificates, complete the following procedures:

1. Set up Your API Access Token
2. Create a Cluster with SSL CA Certificates

To update an existing cluster with Docker registry SSL CA certificates, complete the following procedures:

1. Set up Your API Access Token
2. Update a Cluster with SSL CA Certificates

Note: The procedures documented in this topic configure an individual Enterprise PKS Kubernetes cluster with a Docker Registry SSL CA certificate. See Import the CA Certificate Used to Sign the Harbor Certificate and Key to BOSH in *Integrating VMware Harbor Registry with Enterprise PKS* if you want to apply a single Harbor Registry certificate to all of your Enterprise PKS clusters.

warning: Configuring Enterprise PKS clusters with private Docker registry CA certificates is currently in beta and is intended for evaluation and test purposes only. Do not use this product in a PKS production environment.

Prerequisites

Before configuring Enterprise PKS Kubernetes clusters to have Docker registry CA certificates, you must have the following:

- A private Docker registry configured to use SSL CA certificates. For more information about securing a private Docker registry, see Use self-signed certificates in the *Docker Registry* manual.

warning: The FQDN for the private Docker registry cannot contain a hyphen, dash, or semi-colon. If such a character is included in the registry name the PKS API will reject it as not a valid character.

Set up Your API Access Token

The curl commands in this topic use an access token environment variable to authenticate to the PKS API endpoints.

1. To export your access token into an environment variable, run the following command:

```
pks login -a PKS-API -u USER-ID -p 'PASSWORD' -k; \
export YOUR-ACCESS-TOKEN=$(bosh int ~/.pks/creds.yml --path /access_token)
```

   Where:

   - `PKS-API` is the FQDN of your PKS API endpoint. For example, `api.pks.example.com`.
   - `USER-ID` is your Enterprise PKS user ID.
   - `PASSWORD` is your Enterprise PKS password.
   - `YOUR-ACCESS-TOKEN` is the name of your access token environment variable.

   For example:

```
$ pks login -a pks.my.lab -u alana -p 'psswrdabc123...!' -k; \
export my_token=$(bosh int ~/.pks/creds.yml --path /access_token)
```

   Note: If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

Create a Cluster with SSL CA Certificates

You can create a new cluster configured with one or more SSL CA certificates by using the PKS API `create-cluster` endpoint.

1. To create a cluster configured with one or more SSL CA certificates, run the following command:

```
curl -X POST \
  https://PKS-API:9021/v1/clusters \
  -H 'Accept: application/json' \
  -H "Authorization: Bearer $YOUR-ACCESS-TOKEN" \
  -H 'Content-Type: application/json' \
  -H 'Host: PKS-API:9021' \
  -d '{
  "name": "CLUSTER-NAME",
  "plan_name": "PLAN-NAME",
  "parameters": {
    "kubernetes_master_host": "KUBERNETES-MASTER-HOST",
    "custom_ca_certs": [
      {
        "domain_name": "DOMAIN-NAME",
        "ca_cert": "CA-CERTFICATE"
      }
    ]
  }
}'
```

Where:

- `PKS-API` is the FQDN of your PKS API endpoint. For example, `api.pks.example.com`.
- `YOUR-ACCESS-TOKEN` is the name of your access token environment variable.
- `CLUSTER-NAME` is the name of your cluster.
- `PLAN-NAME` is the name of your plan.
- `KUBERNETES-MASTER-HOST` is your Kubernetes master host.
- `DOMAIN-NAME` is a Docker Registry URL. You cannot remove an existing Docker Registry URL from a cluster. If you specify a URL that is already registered with your cluster, the cluster's existing CA certificate for that URL is overwritten.
- `CA-CERTFICATE` is the CA certificate that corresponds to `DOMAIN-NAME`. For more information about using a CA certificate in a PKS API command, see Prepare a Certificate String for Command Line Use, below.

  You can configure your cluster with additional certificates by including the certificates in the `custom_ca_certs` array as additional `domain_name`, `ca_cert` pairs.

  > **Note:** You can include wildcard characters in your `domain_name` URLs. For example, `*.docker.com`.

Update a Cluster with SSL CA Certificates

You can update an existing cluster with one or more SSL CA certificates by using the PKS API `update-cluster` endpoint.

1. To configure an existing cluster with one or more SSL CA certificates, run the following command:

```
curl -X PATCH \
  https://PKS-API:9021/v1/clusters/CLUSTER-NAME \
  -H 'Accept: application/json' \
  -H "Authorization: Bearer $YOUR-ACCESS-TOKEN" \
  -H 'Content-Type: application/json' \
  -H 'Host: PKS-API:9021' \
  -d '{
      "custom_ca_certs": [
        {
          "domain_name": "DOMAIN-NAME",
          "ca_cert": "CA-CERTFICATE"
        }
      ]
    }'
```

Where:

- `PKS-API` is the FQDN of your PKS API endpoint. For example, `api.pks.example.com`.
- `YOUR-ACCESS-TOKEN` is the name of your access token environment variable.
- `CLUSTER-NAME` is the name of your cluster.
- `DOMAIN-NAME` is a Docker Registry URL. You cannot remove an existing Docker Registry URL from a cluster. If you specify a URL that is already registered with your cluster, the cluster's existing CA certificate for that URL is overwritten.
- `CA-CERTFICATE` is the CA certificate that corresponds to `DOMAIN-NAME`. For more information about using a CA certificate in a PKS API command, see Prepare a Certificate String for Command Line Use, below.

  You can configure your cluster with additional certificates by including the certificates in the `custom_ca_certs` array as additional `domain_name`, `ca_cert` pairs.

  Note: You can include wildcard characters in your `domain_name` URLs. For example, `*.docker.com`.

SSL CA Certificate Formats

SSL CA certificates are unique CA-issued ASCII text strings.

The CAs issue most certificates as a PEM formatted ASCII text files. PEM certificate files typically have the extensions `.pem`, `.crt`, `.cer`, or `.key`.

PEM files start with the string `-----BEGIN CERTIFICATE-----`, terminate with `-----END CERTIFICATE-----`, and are Base64-encoded.

Certificate strings are long and are frequently stored within a certificate file with newline wrapping every 64 characters.

Prepare a Certificate String for Command Line Use

When you provide a certificate string on a command line or PKS API command, as in the PKS API commands above, your certificate string must be provided without newline wrapping.

Note: The PKS API does not validate certificate strings for correctness. Ensure your certificate string is free of newline characters before using the certificate string in a PKS API command.

To prepare your certificate string for command line use:

1. To remove newline wrapping from a certificate string, run the following command:

   ```
   awk 'NF {sub(/\r/, ""); printf "%s\\n",$0;}' CA-PEM
   ```

   Where `CA-PEM` is the filename of your PEM-formatted CA certificate file.

   This command returns your certificate string without newline wrapping.

PersistentVolume Storage Options on vSphere

In this topic

Page last updated:

This topic describes options for configuring VMware Enterprise 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 Enterprise 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 Enterprise 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 Enterprise 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.

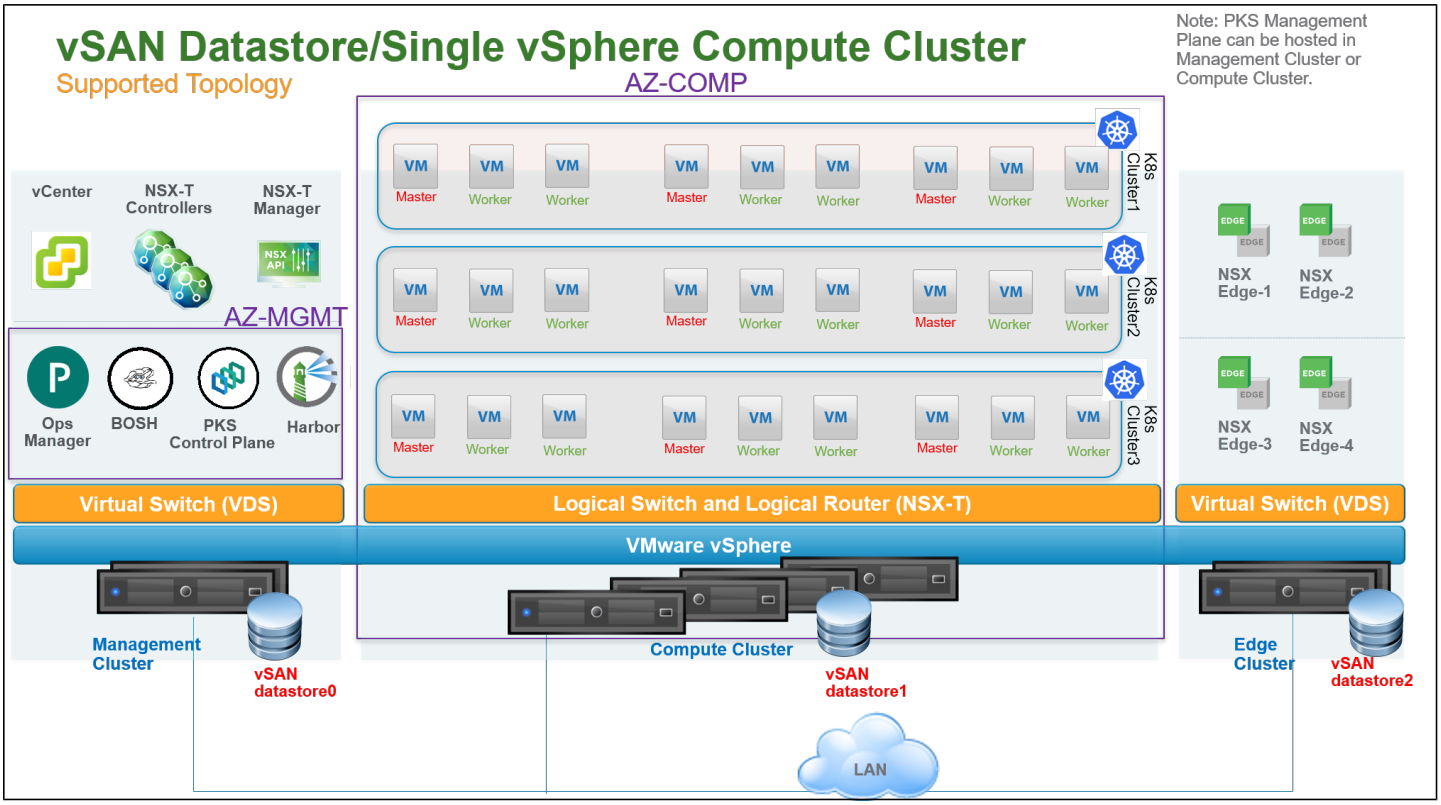
The table below summarizes Enterprise PKS support for PVs in Kubernetes when deployed on vSphere:

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, Enterprise 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 Enterprise PKS foundation, create an AZ that is mapped directly to the single cluster.
- If you use multiple Enterprise PKS foundations, create an AZ that is mapped to this single cluster and a Resource Pool.

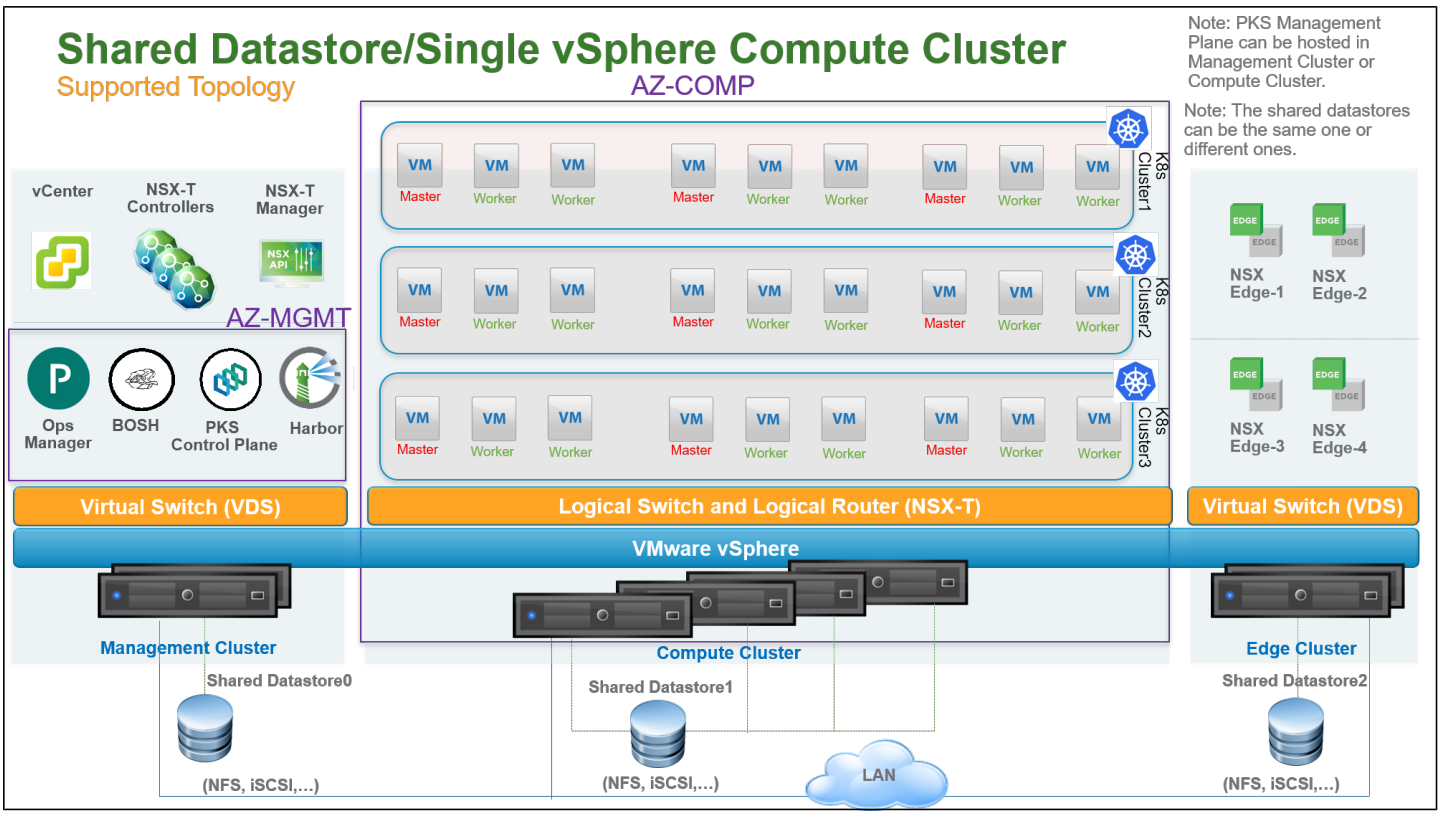
With this topology, you can create multiple vSAN datastores on the same compute cluster using different disk groups on each ESXi host. PVs, backed by respective VMDK files, can be dispatched across the datastores to mitigate the impact of datastore failure. For StatefulSets, all PVs used by different 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, Enterprise 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.

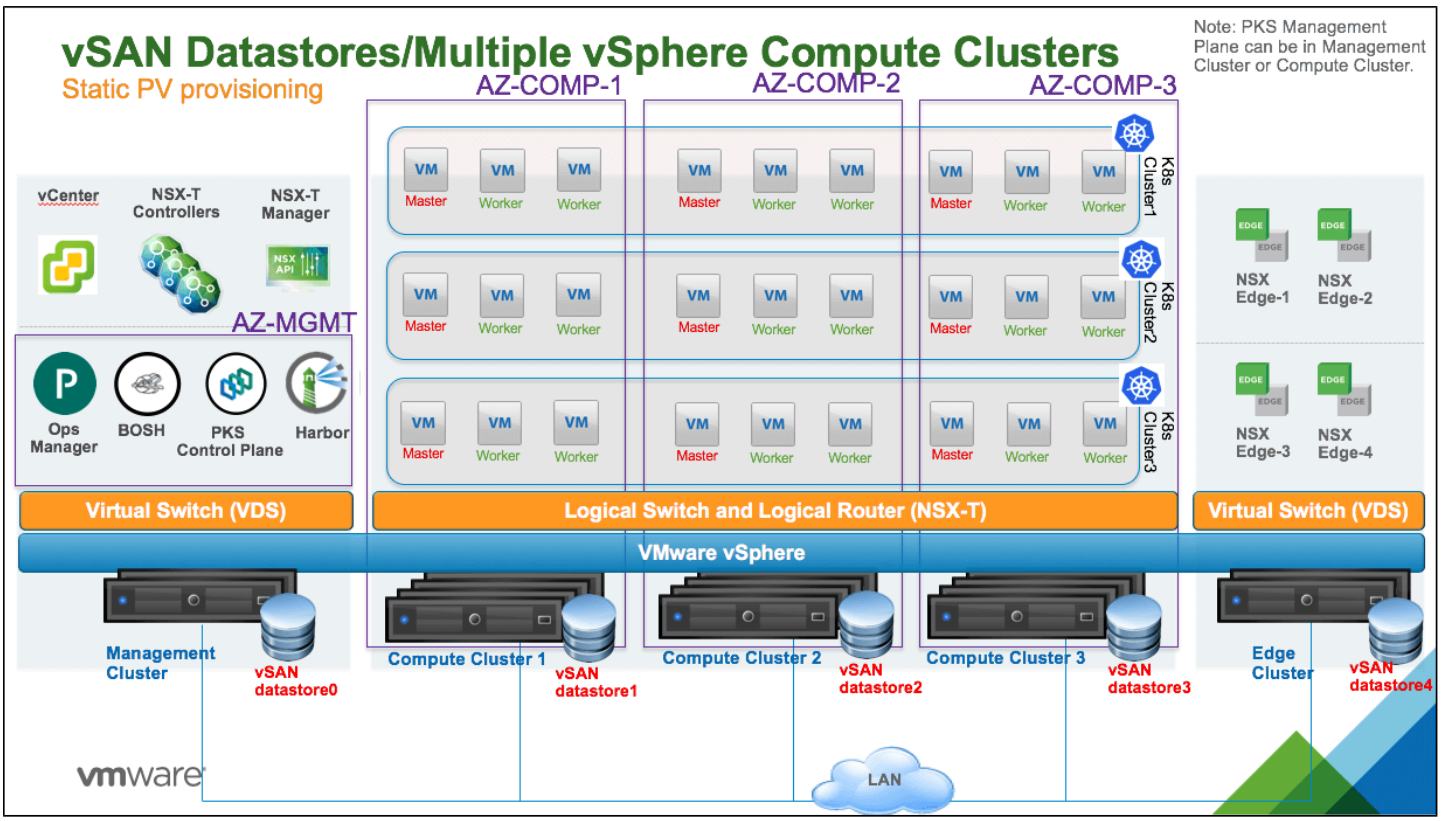
With this topology, you can create multiple shared datastores connected to the same compute cluster. PVs, backed by respective VMDK files, can be dispatched across the datastores to mitigate the impact of datastore failure. For StatefulSets, all PVs used by different instances of the replica land in 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.

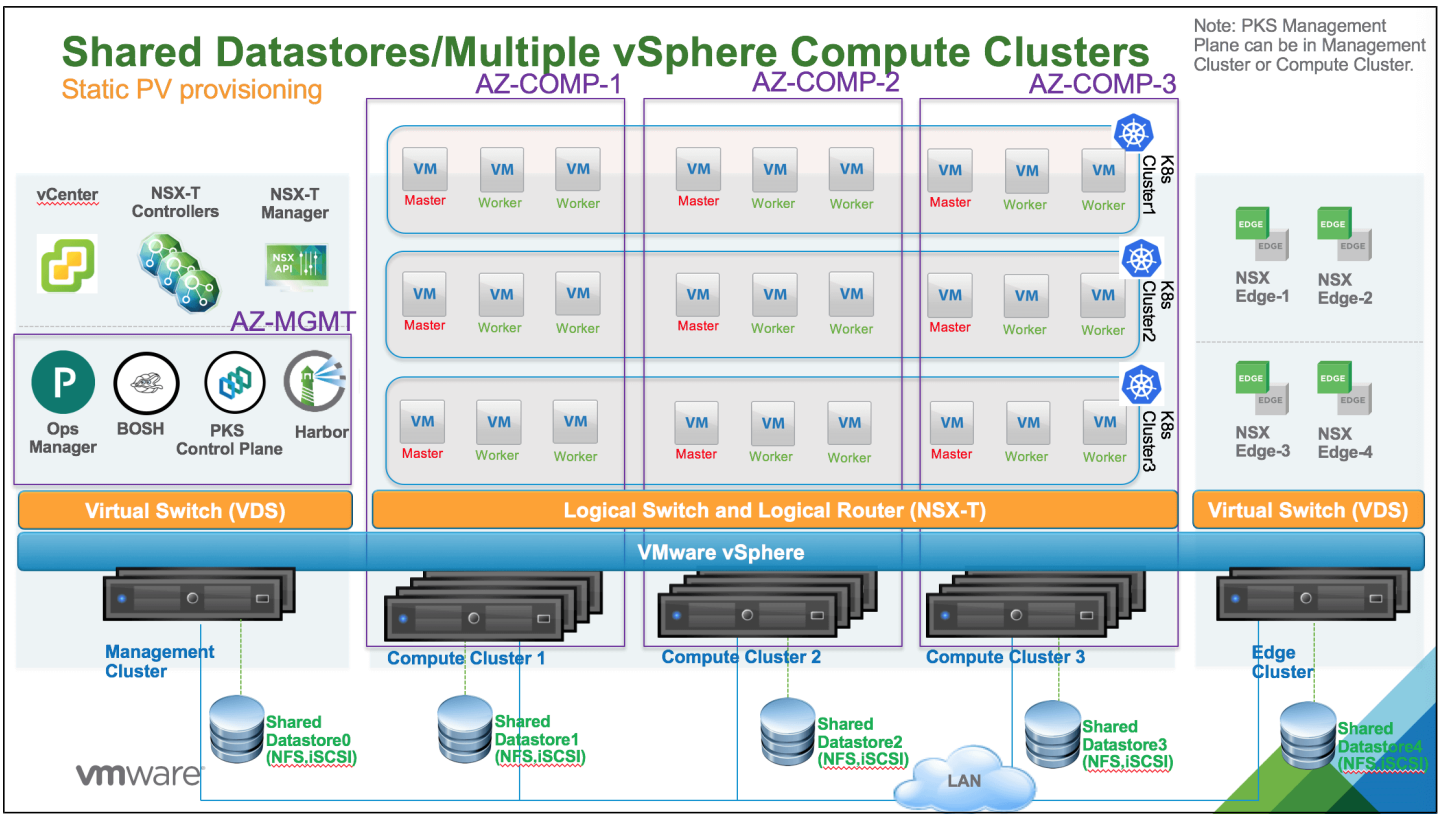


In this topology, vSAN is enabled on each compute cluster. There is one local vSAN datastore per compute cluster. For example, in the above diagram, vSAN datastore1 is provisioned for Compute Cluster 1 and vSAN datastore2 is provisioned for Compute Cluster 2.

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.

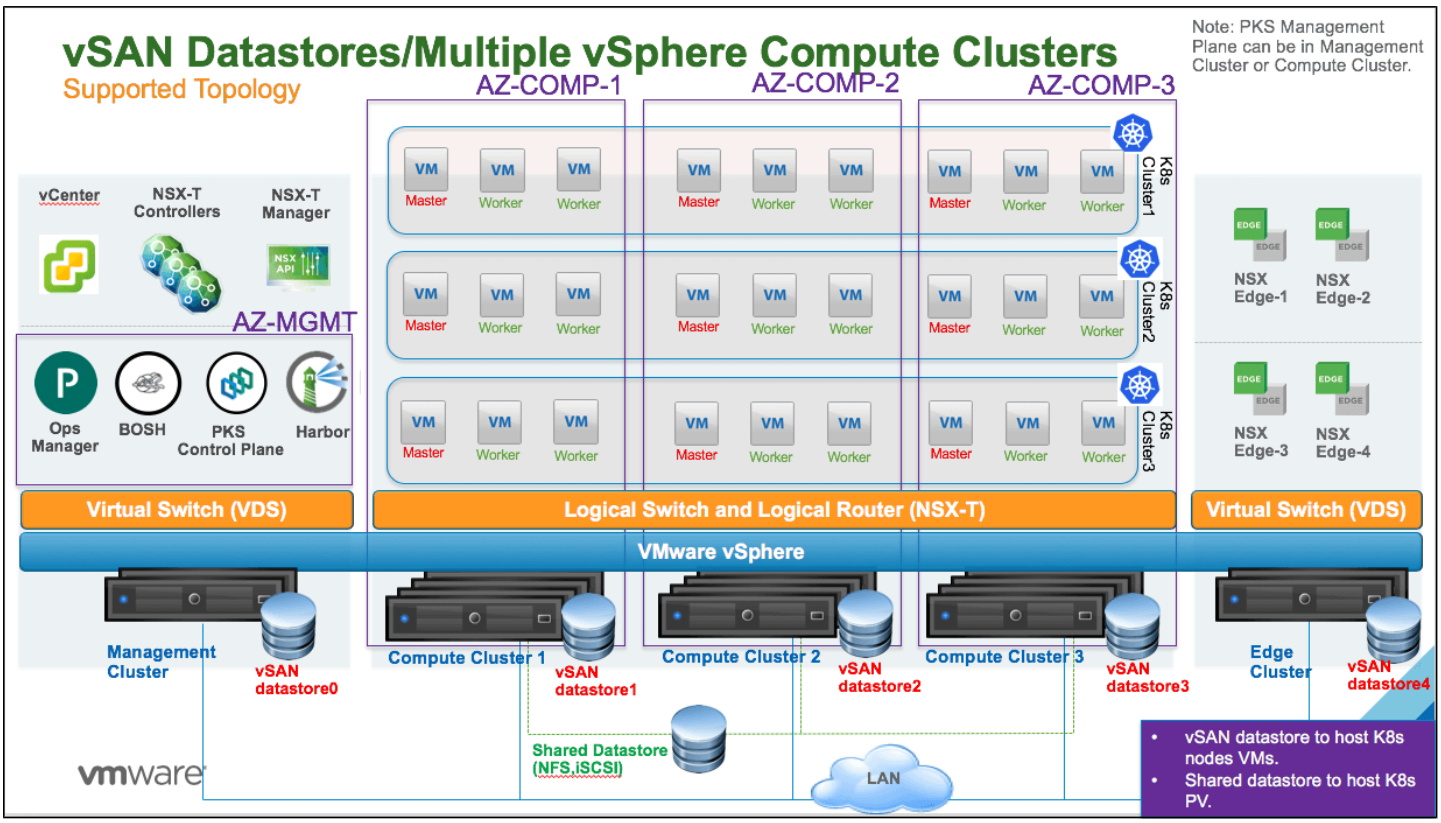
One or more AZs can be instantiated. Each AZ is mapped to a vSphere compute cluster. The AZ is bound to a failure domain which is typically the physical rack where the compute cluster is hosted.

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.

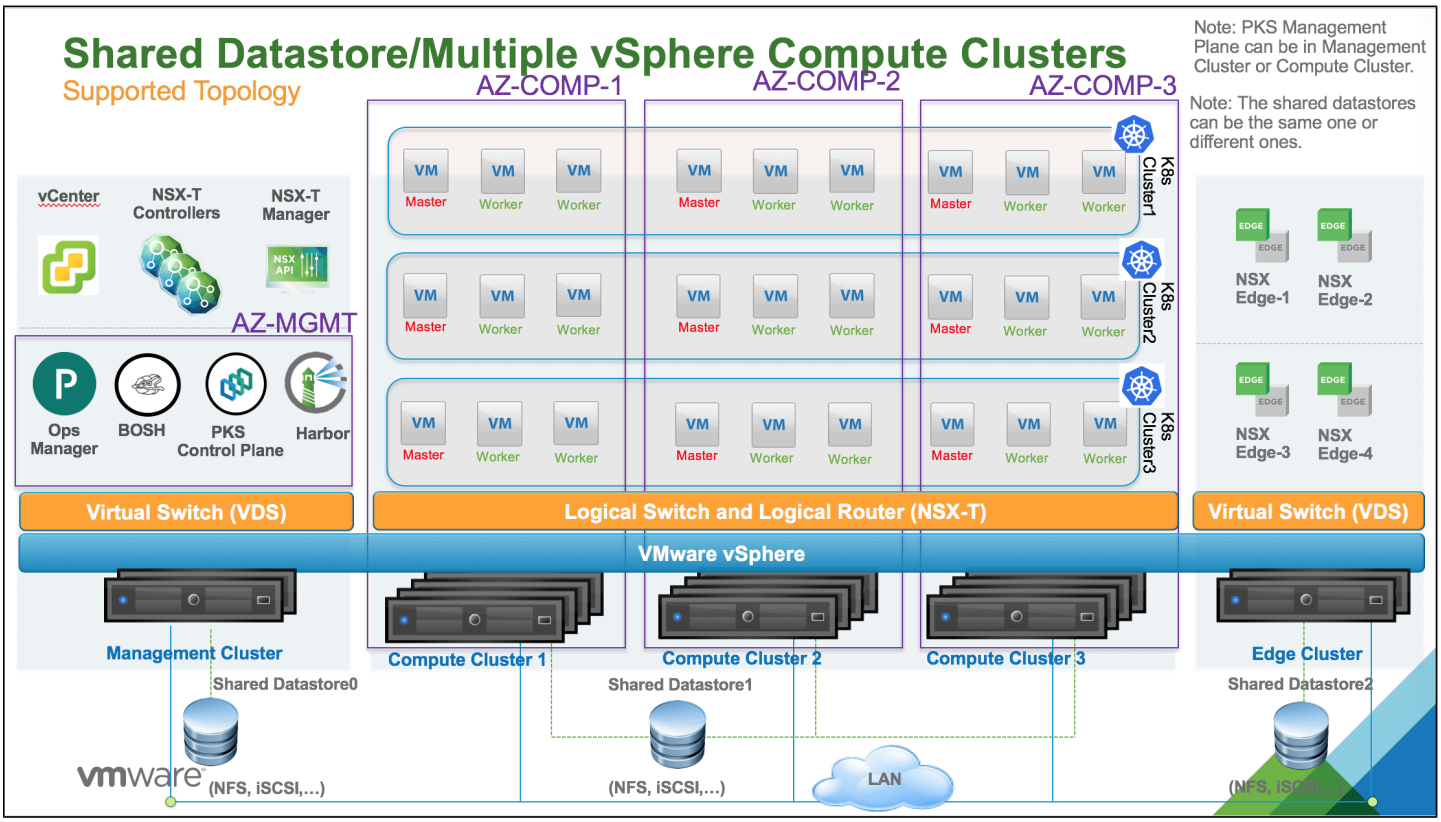
You can insert a shared NFS, iSCSI (VMFS), or FC (VMFS) datastore across all vSAN-based vSphere compute clusters to support both static and dynamic PV provisioning.

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, Enterprise 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.

Configuring and Using PersistentVolumes

In this topic

Page last updated:

This topic describes how to provision static and dynamic PersistentVolumes (PVs) for VMware Enterprise 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, see Persistent Volumes in the *Kubernetes Concepts* 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 StatefulSets 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.

Enterprise 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
```

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
```

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
```

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 Enterprise 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 by running the command 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://raw.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
     ```

2. Apply the spec by running the following command:

   ```
   kubectl create -f STORAGE-CLASS-SPEC.yml
   ```

   Where `STORAGE-CLASS-SPEC` is the name of the file that you downloaded in the previous step.
   For example:

   ```
   $ kubectl create -f storage-class-gcp.yml
   ```

3. Download the example PVC by running the following command:

   ```
   wget https://raw.githubusercontent.com/cloudfoundry-incubator/kubo-ci/master/specs/persistent-volume-claim.yml
   ```

4. Apply the PVC by running the following command:

   ```
   kubectl create -f persistent-volume-claim.yml
   ```

5. Confirm that you applied the PVC by running the following command:

   ```
   kubectl get pvc -o wide
   ```

6. To use the dynamic PV, create a pod that uses the PVC. For an example, see the pv-guestbook.yml configuration file in the kubo-ci repository in GitHub.

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

Deploying Workloads

Page last updated:

This section describes how to deploy workloads to Kubernetes clusters provisioned by VMware Enterprise PKS.

See the following topics:

- Deploying and Exposing Basic Linux Workloads
- Deploying and Exposing Basic Windows Workloads (Beta)
- Adding Custom Linux Workloads
- Using Helm with Enterprise PKS

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

Deploying and Exposing Basic Linux Workloads

In this topic

Page last updated:

This topic describes how to configure, deploy, and expose basic workloads in VMware Enterprise 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 Enterprise 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 the Kubernetes service configuration file for your workload 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 that the Kubernetes service configuration of the workload is set to `type: LoadBalancer`.
4. Confirm that the `type` property of the Kubernetes service for each workload is similarly configured.

> **Note**: For an example of a fully configured Kubernetes service, see the `type: LoadBalancer` configuration for the nginx app example in the kubo-ci repository in GitHub.

For more information about configuring the `LoadBalancer` Service type see Type LoadBalancer in the *Service* section of 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 the load balancer IP address and port number of your exposed workload, run the following command:

   ```
   kubectl get svc SERVICE-NAME
   ```

   Where `SERVICE-NAME` is the specified service `name` of your workload configuration.
   For example:

   ```
   $ kubectl get svc nginx
   ```

2. Retrieve the external IP address and port of the load balancer 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 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 the Kubernetes service configuration file for your workload 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 that the Kubernetes service configuration of the workload is set to `type: LoadBalancer`.
4. Confirm that the `type` property of the Kubernetes service for each workload is similarly configured.

> **Note**: For an example of a fully configured Kubernetes service, see the `type: LoadBalancer` configuration for the nginx app example in the kubo-ci repository in GitHub.

For more information about configuring the `LoadBalancer` Service type see Type LoadBalancer in the *Service* section of 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 the load balancer IP address and port number of your exposed workload, run the following command:

   ```
   kubectl get svc SERVICE-NAME
   ```

   Where `SERVICE-NAME` is the specified service `name` of your workload configuration.
   For example:

   ```
   $ kubectl get svc nginx
   ```

2. Retrieve the external IP address and port of the load balancer 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 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 the Kubernetes service configuration file for your workload 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 Kubernetes service configuration for the workload is set to `type: LoadBalancer`.
5. Confirm that the `annotations` and `type` properties of the Kubernetes service for each workload are similarly configured.

Note: For an example of a fully configured Kubernetes service, see the `type: LoadBalancer` configuration for the nginx app example in the kubo-ci repository in GitHub.

For more information about configuring the `LoadBalancer` Service type see Type LoadBalancer in the *Service* section of 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 the Kubernetes service configuration of your workload.
   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 the load balancer IP address and port number of your exposed workload, run the following command:

   ```
   kubectl get svc SERVICE-NAME
   ```

   Where `SERVICE-NAME` is the specified service `name` of your workload configuration.
   For example:

```
$ kubectl get svc nginx
```

2. Retrieve the external IP and port of the load balancer 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: Run this command 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 must expose your workloads with a `NodePort`.

Configure Your Workload

To expose a static port on your workload, perform the following steps:

1. Open the Kubernetes service configuration file for your workload 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 Kubernetes service configuration of the workload is set to `type: NodePort`.

4. Confirm that the `type` property of the Kubernetes service for each workload is similarly configured.

Note: For an example of a fully configured Kubernetes service, see the `type: LoadBalancer` configuration for the nginx app

example in the kubo-ci repository in GitHub.

For more information about configuring the `NodePort` Service type see Type NodePort in the *Service* section of 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 the Kubernetes service configuration of your workload.
   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 all 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 command:

     ```
     kubectl get nodes -L spec.ip
     ```

   - On the Ops Manager command line, run the following command 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 the specified service `name` of your workload configuration.
   For example:

   ```
   $ kubectl get svc nginx
   ```

3. Find the node port number in the `3XXXX` range. You use this port number when configuring your external load balancer.

4. Configure your external load balancer to map your application Uri to the IP and port number that you collected above. 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 the Kubernetes service configuration file for your workload 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 Kubernetes service configuration of the workload is set to `type: NodePort`.

4. Confirm that the `type` property of the Kubernetes service for each workload is similarly configured.

Note: For an example of a fully configured Kubernetes service, see the `type: LoadBalancer` configuration for the nginx app example in the kubo-ci repository in GitHub.

For more information about configuring the `NodeP{ort` Service type see Type NodePort in the *Service* section of 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 the Kubernetes service configuration of your workload.
   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 command:

   ```
   kubectl get nodes -L spec.ip
   ```

   - On the Ops Manager command line, run the following command 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 the specified service `name` of your workload configuration.
   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:

   ```
   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.

Deploying and Exposing Basic Windows Workloads (Beta)

In this topic

Page last updated:

This topic describes deploying Windows worker-based Kubernetes clusters in VMware Enterprise PKS.

Overview

In Enterprise PKS, you can deploy Windows-based workloads to Kubernetes clusters on vSphere with Flannel.

To deploy a new Windows-based workload to a new pod, do the following:

1. Access Your Windows-Based Cluster
2. Configure a Pod Deployment Manifest
3. Deploy the Pod
4. Configure a Service Manifest
5. Deploy the Service

warning: Support for Windows-based Kubernetes clusters is in beta and supports only vSphere with Flannel.

Do not enable this feature if you are using Enterprise PKS v1.5 with vSphere with NSX-T, Google Cloud Platform (GCP), Azure, or Amazon Web Services (AWS).

We are actively looking for feedback on this beta feature. To submit feedback, send an email to pcf-windows@pivotal.io.

Prerequisites

You can deploy Windows workloads to only Windows-based clusters. Before you can use Windows-based clusters, you must configure the Enterprise PKS tile. For instructions on configuring the Enterprise PKS tile, see Configuring Windows Worker-Based Clusters (Beta).

Access Your Windows-Based Cluster

Your command line must have access to your Windows-based cluster to deploy Windows VMs and workloads to the cluster.

1. To determine which of your existing clusters is Windows-based, use the following command:

   ```
   pks clusters
   ```

   For example:

```
$ pks clusters


Name              Plan Name           UUID                                  Status     Action
windows-k8s        Plan-11-Windows-Beta  881543kd-64fg-7826-hea6-3h7g1o04kh0e  succeeded  Create
second-windows-k8s  Plan-11-Windows-Beta  951547dl-67kg-9631-bju8-7h9s3o98br0q  succeeded  Create
```

Only clusters configured on Plans 11, 12, or 13 are Windows-based.

2. To access your Windows-based cluster, run the following command:

```
pks get-credentials CLUSTER-NAME
```

Where `CLUSTER-NAME` is the name of your Windows-based cluster.

For example:

```
$ pks get-credentials windows-k8s


Fetching credentials for cluster windows-k8s.
Context set for cluster windows-k8s.


You can now switch between clusters by using:
$kubectl config use-context <cluster-name>
```

The `pks get-credentials` command creates a local `kubeconfig`, allowing you to manage the cluster from the command line. For more information about the `pks get-credentials` command, see Retrieving Cluster Credentials and Configuration.

3. To verify you have established access to the correct cluster, run the following command:

```
kubectl cluster-info
```

4. (Optional) To review the existing pods in the cluster, run the following command:

```
kubectl get pods
```

Deploy a Windows Worker Pod

A pod deployment manifest file configures the VMs deployed to a pod.

Configure a Pod Deployment Manifest

You must create a Windows worker deployment manifest before deploying your new Windows worker pod.

1. To create a Windows worker deployment manifest, create a new YAML file containing the following:

```
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata
  labels:
      app: POD-NAME
  name: POD-NAME
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: POD-NAME
      name: POD-NAME
    spec:
      containers:
      - name: CONTAINER-NAME
        image: CONTAINER-FILE:latest
        env:
        - name: PORT
          value: "80"
        ports:
        - name: http
          containerPort: 80
      nodeSelector:
        kubernetes.io/os: windows
      tolerations:
      - key: "windows"
        operator: "Equal"
        value: "2019"
        effect: "NoSchedule"
```

Where:

- `POD-NAME` is the name of your pod.
- `CONTAINER-NAME` is the internal name of your container.
- `CONTAINER-FILE` is the filename of your container.

For example:

```
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata
  labels:
      app: win-webserver
  name: win-webserver
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: win-webserver
      name: win-webserver
    spec:
      containers:
      - name: windowswebserver
        image: stefanscherer/webserver-windows:latest
        env:
        - name: PORT
          value: "80"
        ports:
        - name: http
          containerPort: 80
      nodeSelector:
        kubernetes.io/os: windows
      tolerations:
      - key: "windows"
        operator: "Equal"
        value: "2019"
        effect: "NoSchedule"
```

Deploy the Pod

1. To deploy a new Windows worker pod, run the following command:

```
kubectl apply -f POD-CONFIG-FILE
```

   Where `POD-CONFIG-FILE` is the filename of the Windows worker deployment manifest created above.

2. To confirm the status of the new pod, and the creation of new Windows worker nodes, run the following commands:

```
kubectl get pods
kubectl get nodes -o wide
```

   For example:

```
$ kubectl apply -f first-k8s.yml
deployment.extensions/win-webserver created


$ kubectl get pods  -o wide
NAME                          READY  STATUS   RESTARTS  AGE  IP           NODE                                NOMINATED NODE   READINESS GATES
win-webserver-795g866cd7-58oct   1/1    Running  0         88s  10.200.42.4   0983934a-6d69-8e5g-g3k1-98r8r56l345j  <none>           <none>


$ kubectl get nodes -o wide
NAME                                  STATUS   ROLES    AGE    VERSION    INTERNAL-IP   EXTERNAL-IP   OS-IMAGE                        KERNEL-VERSION      CONTAINER-RUNTIME
0983934a-6d69-8e5g-g3k1-98r8r56l345j    Ready    <none>   19d    v1.14.1    10.85.41.118  10.85.41.118  Windows Server 2019 Datacenter    10.0.17763.503      docker://18.9.0
6d69934a-7d43-9g3g-h4d1-54r9r97l395j    Ready    <none>   19d    v1.14.1    10.85.41.115  10.85.41.115  Ubuntu 16.04.6 LTS                4.15.0-50-generic   docker://18.9.0
7636d69a-2e75-5l0g-k6m1-76r3r37l729k    Ready    <none>   19d    v1.14.1    10.85.41.117  10.85.41.117  Windows Server 2019 Datacenter    10.0.17763.503      docker://18.9.0
406d694a-9g96-2d3g-f3j1-32r1r44l342x    Ready    <none>   19d    v1.14.1    10.85.41.116  10.85.41.116  Windows Server 2019 Datacenter    10.0.17763.503      docker://18.9.0
```

   In the preceding example a new pod is created, and creation and status of the new pod and new nodes verified.

Deploy a Service to a Windows Worker Pod

A service deployment manifest file configures your service, defining how your service will run and how it will be exposed.

Configure a Service Manifest

You must create a Windows service deployment manifest before deploying your Windows worker workload.

1. To create a Windows service deployment manifest, create a new YAML file containing the following:

```
---
apiVersion: v1
kind: Service
metadata:
  name: APP-NAME
  labels:
    app: APP-NAME
spec:
  ports:
    # the port that this service should serve on
  - port: 80
    targetPort: 80
  selector:
    app: APP-NAME
  type: NodePort
```

   Where `APP-NAME` is the name of your Windows service.

   For example:

```
---
apiVersion: v1
kind: Service
metadata:
  name: win-webserver
  labels:
    app: win-webserver
spec:
  ports:
    # the port that this service should serve on
  - port: 80
    targetPort: 80
  selector:
    app: win-webserver
  type: NodePort
```

Deploy the Service

1. To expose the specified service on a NodePort, run the following command:

```
kubectl apply -f SERVICE-CONFIG-FILE
```

Where `SERVICE-CONFIG-FILE` is the filename of your Windows service deployment manifest created above.

For example:

```
$ kubectl get services
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP    PORT(S)    AGE
kubernetes   ClusterIP   10.100.200.1   <none>         443/TCP    20d


$ kubectl apply -f first-k8s-service.yml
service/win-webserver created


$ kubectl get services
NAME         TYPE       CLUSTER-IP       EXTERNAL-IP  PORT(S)       AGE
kubernetes   ClusterIP  10.100.200.1     <none>       443/TCP       20d
win-webserver NodePort  10.100.200.221   <none>       80:32073/TCP  5s


$ curl 10.85.41.118:32073
<pre>
   <a href="License.txt">License.txt</a>
   <a href="ProgramData/">ProgramData/</a>
   <a href="Users/">Users/</a>
   <a href="WcSandboxState/">WcSandboxState/</a>
   <a href="Windows/">Windows/</a>
   <a href="var/">var/</a>
   <a href="webserver.exe">webserver.exe</a>
</pre>
```

In the preceding example a new service is created, verified, and validated.

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

Adding Custom Linux Workloads

In this topic

Page last updated:

This topic describes how to add custom workloads to VMware Enterprise 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 Enterprise PKS tile.

For more information, see the *Plans* section of the *Installing Enterprise PKS* topic for your IaaS. For example, Plans in *Installing Enterprise PKS on vSphere*.

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

Using Helm with Enterprise PKS

In this topic

Page last updated:

This topic describes how to use the package manager Helm 3 or its predecessor Helm 2 for your Kubernetes apps running on VMware Enterprise PKS.

Helm 3 requires less configuration than Helm 2.

Overview

Helm 3 and Helm 2 include the following components:

Component	Role	Location
`helm`	Client	Runs on your local workstation
`tiller` (Helm 2 only)	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.

Install and Configure Helm 3

To install and configure Helm 3, follow the Step 1: Install And Configure Helm instructions in the Bitnami PKS documentation.

Install and Configure Helm 2

To use Helm 2 with Enterprise PKS, you must first configure the Tiller component to give it access to the Kubernetes API. Tiller runs inside the Kubernetes cluster.

To grant API access to Tiller and install Helm 2:

1. Create a role-based access control (RBAC) configuration file named `rbac-config.yaml` that contains the following:


```
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 latest v2 patch release of the Helm CLI.

4. Deploy Helm 2 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.

For more information about securing Helm 2, see the Bitnami article Exploring the Security of Helm.

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

Enterprise PKS Management Console (vSphere Only)

Page last updated:

VMware Enterprise PKS Management Console provides a unified installation experience for deploying VMware Enterprise PKS to vSphere. The management console is provided as a virtual appliance that you deploy to vSphere by using an OVA template. The management console provides a graphical user interface that assists you with the configuration when deploying Enterprise PKS to vSphere:

- Configures networking for Enterprise PKS
- Deploys Ops Manager
- Generates and registers SSL certificates
- Deploys BOSH Director
- Deploys Enterprise PKS
- Deploys Harbor Registry

Enterprise PKS Management Console is easy to use. If you are experienced with installing Enterprise PKS on vSphere, the help and the tool tips in the installer UI should be enough to complete the process. If you are new to Enterprise PKS, refer to this documentation as needed to assist with the installation.

After you have deployed Enterprise PKS on vSphere, you can use the management console to monitor and manage the operation of your Enterprise PKS deployments, as well as perform upgrades and patch the individual components.

If you have a previous installation of Enterprise PKS Management Console, you also use the management console to upgrade and patch your deployment.

See the following topics:

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

Prerequisites for Enterprise PKS Management Console Deployment

In this topic

Page last updated:

VMware Enterprise PKS Management Console is provided as an OVA template that requires at a minimum the vSphere resources described in Virtual Infrastructure Prerequisites.

Network Configurations

Enterprise PKS Management Console provides 3 network configuration options for your Enterprise PKS deployments. Each network configuration option has specific prerequisites.

- **Bring your own topology**: Deploy Enterprise PKS to an existing NSX-T Data Center network that you have fully configured yourself. See Prerequisites for a Bring Your Own Topology Deployment to NSX-T Data Center
- **Automated NAT deployment**: Deploy Enterprise PKS to an existing NSX-T Data Center network that you have not fully set up, that Enterprise PKS Management Console helps to configure for you. See Prerequisites for an Automated NAT Deployment to NSX-T Data Center.
- **Flannel**: Deploy Enterprise PKS to a Flannel network that Enterprise PKS Management Console provisions for you. See Prerequisites for a Flannel Network.

For the list of firewall ports that must be open for an Enterprise PKS Management Console deployment, see Firewall Ports and Protocols Requirements for Enterprise PKS Management Console.

When your environment meets the prerequisites for vSphere and for your chosen type of networking, you can Deploy the Enterprise PKS Management Console Appliance.

When Should I Use Enterprise PKS Management Console?

Enterprise PKS Management Console greatly simplifies the process of deploying Enterprise PKS, especially in less complex environments. However, if you require more flexibility in in configuring your deployment, especially in complex NSX-T Data Center deployments, it might be more appropriate to perform the installation manually. For information about the supported topologies for a manual installation, see NSX-T Deployment Topologies for Enterprise PKS.

Before using Enterprise PKS Management Console to deploy Enterprise PKS, consider the following factors:

- Deployments to a No-NAT topology with an NSX-T Data Center logical switch is supported in BYOT deployments only.
- Deployments to a Multi-Tier-0 topology are supported in BYOT deployments only and require additional configuration. For information about the additional configuration required, see Enterprise PKS Management Console Cannot Retrieve Cluster Data in a Multi-Tier0 Topology in *Troubleshooting Enterprise PKS Management Console*.
- Deployments to a No-NAT topology with a vSphere Standard Switch or a vSphere Distributed Switch is not supported in any case.
- Multi-Foundation deployments are not supported in any case.

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

Virtual Infrastructure Prerequisites

Page last updated:

The vSphere environment to which you deploy the appliance OVA requires the following configuration:

- CPU: 2
- RAM: 8 GB
- Disk: 40 GB
- Virtual NIC (vNIC) should be assigned to a network with connectivity to vCenter and NSX Datacenter Manager, if you are using NSX-T Data Center as the container networking interface for Enterprise PKS.

Note: The OVA requirements described here are the minimum supported configuration.

The following vSphere clusters must exist in the target vCenter Server datacenter before you can deploy Enterprise PKS from the management console:

- Management cluster for PKS Management Plane components.
- At least one compute cluster for Kubernetes Cluster nodes, with the recommendation being to deploy more than one, for high-availability purposes.

For information about the supported versions of vSphere, see the release notes.

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

Firewall Ports and Protocols Requirements for Enterprise PKS Management Console

Page last updated:

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 Enterprise PKS, is is recommended to disable security policies that filter traffic between the networks supporting the system. To secure the environment and grant access between system components with Enterprise 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.

If you are unable to implement your security policy using these methods, refer to the table below, which identifies the flows between the system components in an Enterprise PKS Management Console deployment.

Notes: The Source Component is IP address of the Enterprise PKS Management Console appliance VM.

In a standard Enterprise PKS deployment, it is assumed that Ops Manager and BOSH are already deployed before you deploy Enterprise PKS. This is not the case with Enterprise PKS deployments from the management console, in which you do not know the IP addresses in the deployment network that will be assigned to PKS API VM, BOSH VM, and Ops Manager VM. As a consequence, it is recommended to create a firewall rule that allows access by the management console appliance VM to the entire deployment subnet.

Source Component	Destination Component	Destination Protocol	Destination Port	Service
Management Console Appliance VM	All System Components	TCP	22	ssh
Management Console Appliance VM	All System Components	TCP	80	http
Management Console Appliance VM	All System Components	TCP	443	https
Management Console Appliance VM	Cloud Foundry BOSH Director	TCP	25555	bosh director rest api
Management Console Appliance VM	DNS validation for Ops Manager	TCP	53	netcat
Management Console Appliance VM	Kubernetes Cluster API Server - LB VIP	TCP	8443	httpsca
Management Console Appliance VM	Pivotal Cloud Foundry Operations Manager	TCP	22	ssh

Source Component	Destination Component	Destination Protocol	Destination Port	Service
Management Console Appliance VM	Pivotal Cloud Foundry Operations Manager	TCP	443	https
Management Console Appliance VM	PKS Controller	TCP	9021	pks api server
Management Console Appliance VM	vCenter Server	TCP	443	https

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

Prerequisites for a Bring Your Own Topology Deployment to NSX-T Data Center

In this topic

Page last updated:

A bring your own topology environment is an NSX-T Data Center instance that you have fully configured yourself for use with Enterprise PKS. For example, an NSX-T Data Center instance that you have used in a previous deployment of Enterprise PKS. The following objects must be in place before you start a production deployment.

- 3 NSX Manager Nodes deployed
- NSX Management Cluster formed
- Virtual IP address assigned for Management Cluster or load balancer

For information about the supported versions of NSX-T Data Center, see the release notes.

General Requirements

- An active/standby Tier-0 Router created
- A logical switch on an NSX-T Virtual Distributed Switch (N-VDS) is prepared under the Tier-1 router, for use by the PKS management plane
- Edge Cluster with at least 2 NSX-T Data Center Edge Nodes deployed in active/standby mode, with connectivity to an uplink network configured
- Overlay Transport Zone created, with the edge nodes included
- VLAN Transport Zone created, with the edge nodes included
- MTU of all transport nodes and physical interfaces configured to 1600 or more
- If your NSX-T Data Center environment uses custom certificates, obtain the CA certificate for NSX Manager

Notes: Do not use the network on which you deploy the Enterprise PKS Management Console appliance VM as the network for the management plane when you deploy Enterprise PKS. Using the same network for the appliance VM and the management plane requires additional NSX-T Data Center configuration and is not recommended.

If NSX-T Data Center uses custom certificates and you do not provide the CA certificate for NSX Manager, Enterprise PKS Management Console automatically generates one and registers it with NSX Manager. This can cause other services that are integrated with NSX Manager not to function correctly.

In BYOT mode, Enterprise PKS Management Console automatically retrieves the tier0 HA mode from your NSX-T Data Center environment and creates NAT rules on the tier 0 or tier 1 router.

If you are deploying Enterprise PKS in a multiple-tier0 topology, additional post-deployment configuration of the management console VM is required. For information, see Enterprise PKS Management Console Cannot Retrieve Cluster Data in a Multi-Tier0 Topology in *Troubleshooting Enterprise PKS Management Console*.

NSX-T Data Center Configuration Requirements

- Virtual IP for the Tier-0 Router configured
- Floating IP Pool configured
- Pod IP Block ID created
- Node IP Block ID created
- Logical Switch configured for PKS Management Plane
- Tier-1 Router configured and connected to the Tier-0 Router
- Routing for PKS Floating IPs configured to point to the Tier-0 HA Virtual IP

Proof-of-Concept Deployments

The requirements above are for production environments. In proof-of-concept deployments one NSX Manager node is sufficient. The NSX management cluster and load balancer are also optional for proof-of-concept deployments.

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

Prerequisites for an Automated NAT Deployment to NSX-T Data Center

In this topic

Page last updated:

An unprepared environment is an NSX-T Data Center instance that you have not already configured for use with Enterprise PKS. Enterprise PKS Management Console helps you to complete the configuration of an unprepared environment, but the environment must meet certain infrastructure prerequisites.

- 3 NSX Manager Nodes deployed
- NSX Management Cluster formed
- Virtual IP address assigned for the Management Cluster or load balancer

For information about the supported versions of NSX-T Data Center, see the release notes.

General Requirements

- Edge Cluster with at least 2 NSX-T Data Center Edge Nodes deployed and connectivity to an uplink network configured and verified
- Overlay Transport Zone created, with the edge nodes included
- VLAN Transport Zone created, with the edge nodes included
- MTU of all transport nodes and physical interfaces configured to 1600 or more
- Obtain the following IP addresses for the uplink network to use:
  - Subnet, subnet mask, gateway, and VLAN ID of the uplink network
  - Addresses within the uplink subnet for the Tier 0 uplinks
  - Address to use for the HA Virtual IP on the Tier-0 router
- Obtain the following IP additional addresses:
  - CIDR ranges to use for deployment, pods, and nodes. This range of IP addresses must not be in conflict with any other workloads.
  - IP addresses of DNS and NTP servers
  - A range of 5 available floating IP addresses
- If your NSX-T Data Center environment uses custom certificates, obtain the CA certificate for NSX Manager

> **Note**: If NSX-T Data Center uses custom certificates and you do not provide the CA certificate for NSX Manager, Enterprise PKS Management Console automatically generates one and registers it with NSX Manager. This can cause other services that are integrated with NSX Manager not to function correctly.

Proof-of-Concept Deployments

The requirements above are for production environments. In proof-of-concept deployments one NSX Manager node is sufficient. The NSX management cluster and load balancer are also optional for proof-of-concept deployments. One NSX-T Data Center Edge node is sufficient for proof-of-concept deployments.

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

Prerequisites for a Flannel Network

Page last updated:

You can select the option for Enterprise PKS Management Console to provision a Flannel container networking interface for you during Enterprise PKS deployment.

Obtain the following IP addresses to use for deployment to a Flannel network:

- DNS server, subnet, subnet mask, and gateway of the network on which to deploy Enterprise PKS
- DNS server, subnet, subnet mask, and gateway of the Flannel service network
- Subnet range and subnet mask for the Kubernetes pod and Kubernetes service networks

---

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

Deploy the Enterprise PKS Management Console Appliance

In this topic

Page last updated:

This topic describes how to deploy the VMware Enterprise PKS Management Console appliance from the OVA template.

If you have deployed a previous version of VMware Enterprise PKS Management Console, you can use the management console to upgrade it to a newer version. For information about upgrading, see Upgrade Enterprise PKS Management Console.

Prerequisites

- Download the OVA template from https://downloads.vmware.com.
- Use an account with vSphere administrator privileges to log in to vSphere using the vSphere Client.
- The vCenter Server instance must be correctly configured for Enterprise PKS Management Console deployment. For information about the vCenter Server requirements, see Virtual Infrastructure Prerequisites.

Step 1: Deploy the OVA Template

To deploy the Enterprise PKS Management Console appliance to vSphere, the procedure is as follows:

1. In the vSphere Client, right-click an object in the vCenter Server inventory, select **Deploy OVF template**, select **Local file**, and click **Browse** to navigate to your download of the OVA template.

2. Follow the installer prompts to perform basic configuration of the appliance and to select the vSphere resources for it to use.

   - Accept or modify the appliance name
   - Select the destination datacenter or folder
   - Select the destination cluster or resource pool for the appliance VM
   - Accept the end user license agreements (EULA)
   - Select the disk format and destination datastore for the appliance VM

3. On the **Select Networks** page, select a network port group to which to connect the appliance.

   > **important:** If you intend to deploy Enterprise PKS in a bring your own topology NSX-T Data Center environment, do not use the network on which you deploy the Enterprise PKS Management Console appliance VM as the network for the management plane when you deploy Enterprise PKS. Using the same network for the appliance VM and the management plane requires additional NSX-T Data Center configuration and is not recommended.

4. On the **Customize template** page, expand **Appliance Configuration**.

   - Set the root password for the appliance VM. Setting the root password for the appliance is mandatory.

- Optionally uncheck the **Permit Root Login** checkbox.

> **Note**: If you uncheck the checkbox, you can permit root login later by editing the settings of the appliance VM.

The root password is the only mandatory option. If you want to use auto-generated certificates, DHCP networking, and you do not want to integrate with VMware vRealize Log Insight, click **Next** to start the OVA deployment. Otherwise, complete the remaining steps in this procedure.

5. Configure the appliance certificate, that is used by all of the services that run in the appliance to authenticate connections.
   To use auto-generated, self-signed certificates, leave the **Appliance TLS Certificate**, **Appliance TLS Certificate Key**, and **Certificate Authority Certificate** text boxes blank.
   To use a custom certificate:
   Paste the contents of the server certificate PEM file in the **Appliance TLS Certificate** text box.

```
-----BEGIN CERTIFICATE-----
appliance_certificate_contents
-----END CERTIFICATE-----
```

   Paste the contents of the certificate key in the **Appliance TLS Certificate Key** text box. The appliance supports unencrypted PEM encoded formats for TLS private keys.

```
-----BEGIN PRIVATE KEY-----
appliance_private_key_contents
-----END PRIVATE KEY-----
```

   Paste the contents of the Certificate Authority (CA) file in the **Certificate Authority Certificate** text box.

```
-----BEGIN CERTIFICATE-----
root_CA_certificate_contents
-----END CERTIFICATE-----
```

   To use a certificate that uses a chain of intermediate CAs, paste into the **Certificate Authority Certificate** text box the contents of a certificate chain PEM file. The PEM file must include a chain of the intermediate CAs all the way down to the root CA.

```
-----BEGIN CERTIFICATE-----
intermediate_CA_certificate_contents
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
intermediate_CA_certificate_contents
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
root_CA_certificate_contents
-----END CERTIFICATE-----
```

6. Expand **Networking Properties** and optionally configure the networking for the appliance VM.

   To use DHCP, leave these properties blank.

   - To set a static IP address on the appliance, set the **Network IP Address**, **Network Netmask**, and **Default Gateway** settings.
   - To configure DNS servers, set the **Domain Name Servers**, and **Domain Search Path** settings.
   - To specify a fully qualified domain name (FQDN) for the appliance, set the **FQDN** setting.
   - If necessary, update **Docker Container Network Subnet** and **Docker Container Network Gateway**.

Services in the management console appliance are deployed as Docker containers on a Docker bridge network. Update these values if the default subnet CIDR 172.18.0.0/16 and gateway address 172.18.0.1 for this bridge network conflict with existing networks.

7. Optionally enter the host name and port for VMware vRealize Log Insight in the **Log Insight Server Host/IP** and **Log Insight Server Port** text boxes.
   vRealize Log Insight gathers logs from the Enterprise PKS Management Console appliance itself. For vRealize Log Insight to gather logs from your Enterprise PKS deployments, you must configure the connection when you deploy Enterprise PKS from Enterprise PKS Management Console.

8. Click **Next** to review the settings that you have made.

9. Click **Finish** to deploy the Enterprise PKS Management Console appliance.

Use the Recent Tasks panel at the bottom of the vSphere Client to check the status of the OVA import and deployment of the appliance VM. The appliance VM takes a few minutes to deploy.

If the appliance VM fails to deploy, see Troubleshooting.

Step 2: Log In to Enterprise PKS Management Console

When the OVA deployment has completed successfully, you can access the management console.

1. In the vSphere Client, right-click the appliance VM and select **Power** > **Power On**.

2. When the appliance VM has booted, go to the **Summary** tab for the VM and copy its IP address.

3. Enter the appliance IP address in a browser.

4. At the VMware Enterprise PKS log in page, enter username `root` and the root password that you set when you deployed the OVA template.

Next Steps

You can now use Enterprise PKS Management Console to deploy or upgrade Enterprise PKS instances, either by using the configuration wizard or by importing an existing YAML configuration file.

- Deploy Enterprise PKS by Using the Configuration Wizard
- Deploy Enterprise PKS by Importing a YAML Configuration File
- Upgrade Enterprise PKS Management Console

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

Deploy Enterprise PKS by Using the Configuration Wizard

In this topic

Page last updated:

You can deploy a new VMware Enterprise PKS instance either by using the VMware Enterprise PKS Management Portal configuration wizard to guide you through the configuration process, or by importing an existing YAML configuration file into the YAML editor.

This topic describes how to use the configuration wizard to deploy Enterprise PKS.

- For information about how to deploy Enterprise PKS from a YAML, see Deploy Enterprise PKS by Importing a YAML Configuration File.
- For information about how to upgrade an existing deployment to this version, see Upgrade Enterprise PKS Management Console.

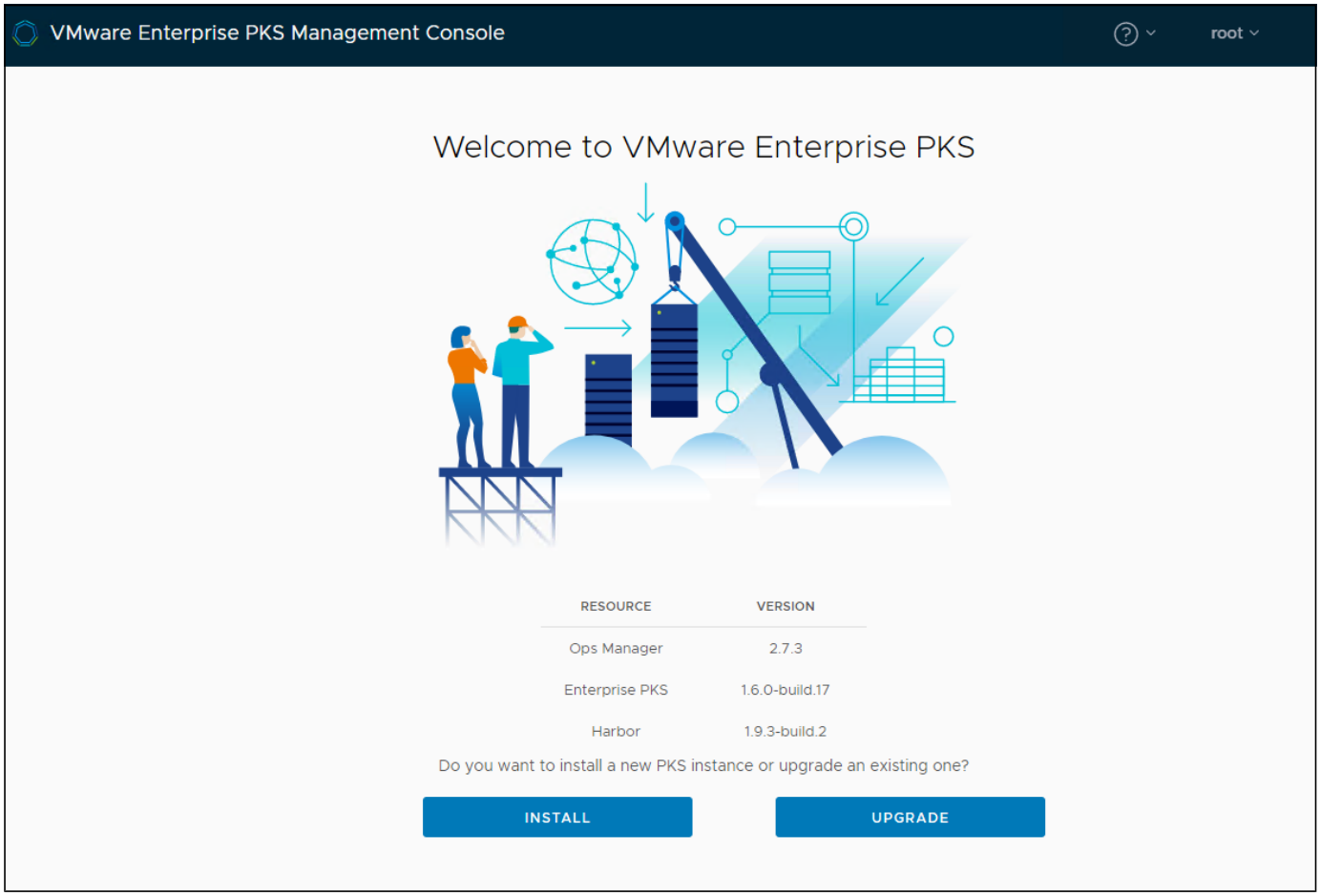
Prerequisites

- Deploy the Enterprise PKS Management Console Applianceto vCenter Server.
- The vCenter Server instance must be correctly configured for Enterprise PKS deployment. For information about the vCenter Server requirements, see Virtual Infrastructure Prerequisites.

- Depending on the type of networking you want to use, your infrastructure must meet the appropriate prerequisites. For information about networking prerequisites, see the following topics:
  - Prerequisites for an Automated NAT Deployment to NSX-T Data Center
  - Prerequisites for a Bring Your Own Topology Deployment to NSX-T Data Center
  - Prerequisites for a Flannel Network
- Log in to Enterprise PKS Management Console.

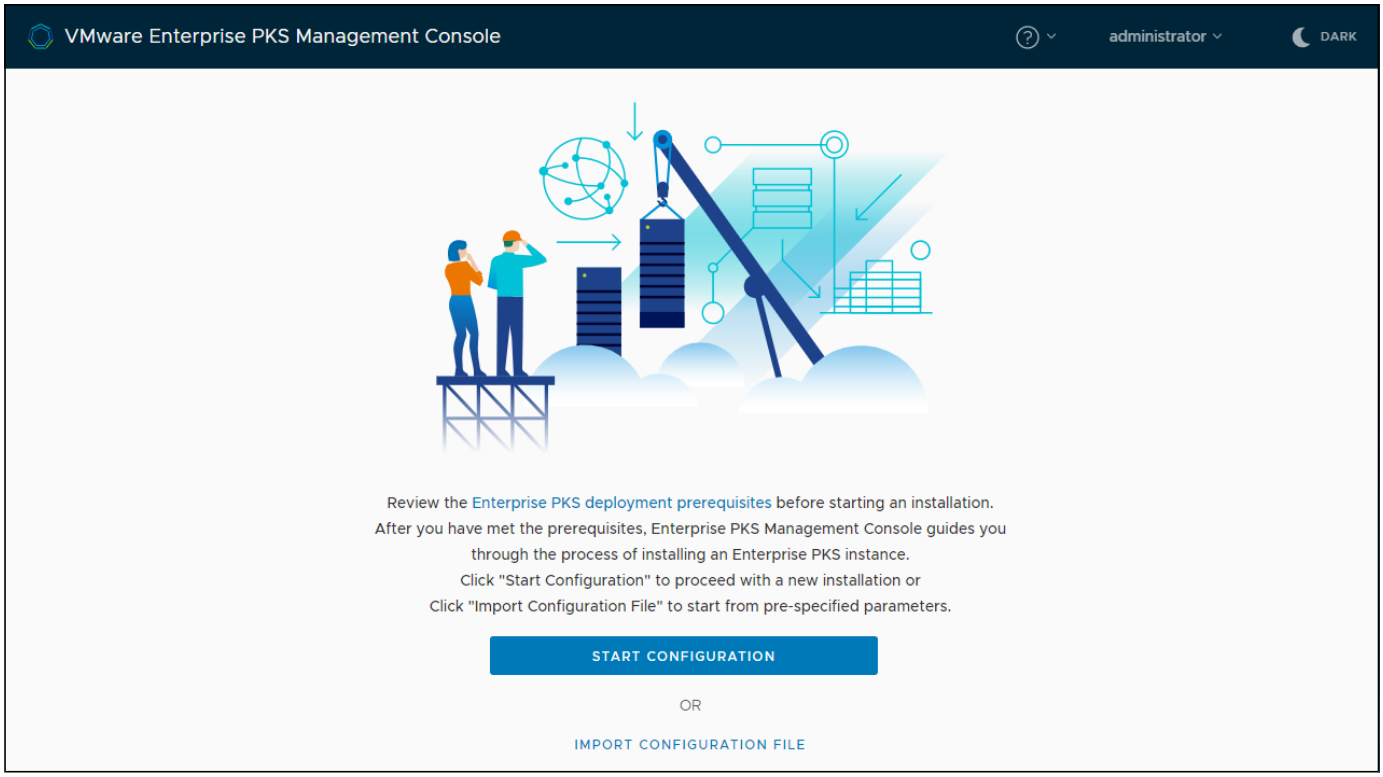
Step 0: Launch the Configuration Wizard

1. On the VMware Enterprise PKS landing page, click **Install**.



View a larger version of this image

2. Click **Start Configuration**.



View a larger version of this image

To get help in the wizard at any time, click the **?** icon at the top of the page and select **Help**, or click the **More Info...** links in each section to see help topics relevant to that section. Click the **i** icons for tips about how to fill in specific fields.

Step 1: Connect to vCenter Server

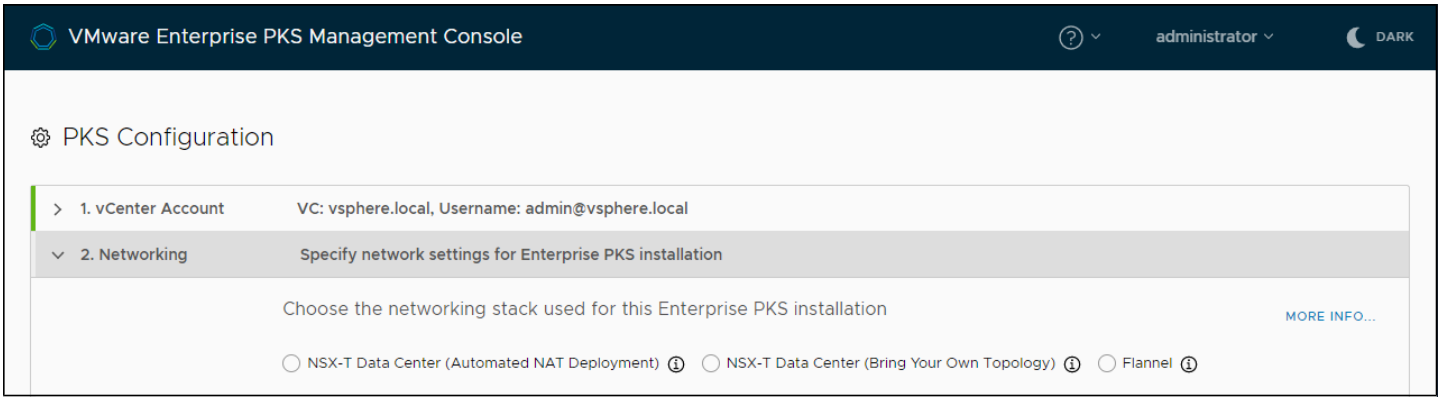
1. Enter the IP address or FQDN for the vCenter Server instance on which to deploy Enterprise PKS.
2. Enter the vCenter Single Sign On username and password for a user account that has vSphere administrator permissions.
3. Click **Connect**.
4. Select the datacenter in which to deploy Enterprise PKS from the drop-down menu.
5. Click **Next** to configure networking.

Step 2: Configure Networking

Provide connection information for the container networking interface to use with Enterprise PKS. Enterprise PKS Management Console provides 3 network configuration options for your Enterprise PKS deployments. Each network configuration option has specific prerequisites.

- **Automated NAT deployment**: Deploy Enterprise PKS to an existing NSX-T Data Center network that you have not fully set up, that Enterprise PKS Management Console configures for you. See Configure an Automated NAT Deployment to NSX-T Data Center below for instructions.
- **Bring your own topology**: Deploy Enterprise PKS to an existing NSX-T Data Center network that you have fully configured yourself. See Configure a Bring Your Own Topology Deployment to NSX-T Data Centerbelow for instructions.
- **Flannel**: Deploy Enterprise PKS to a Flannel network that Enterprise PKS Management Console provisions for you. See Configure

a Flannel Network below for instructions.



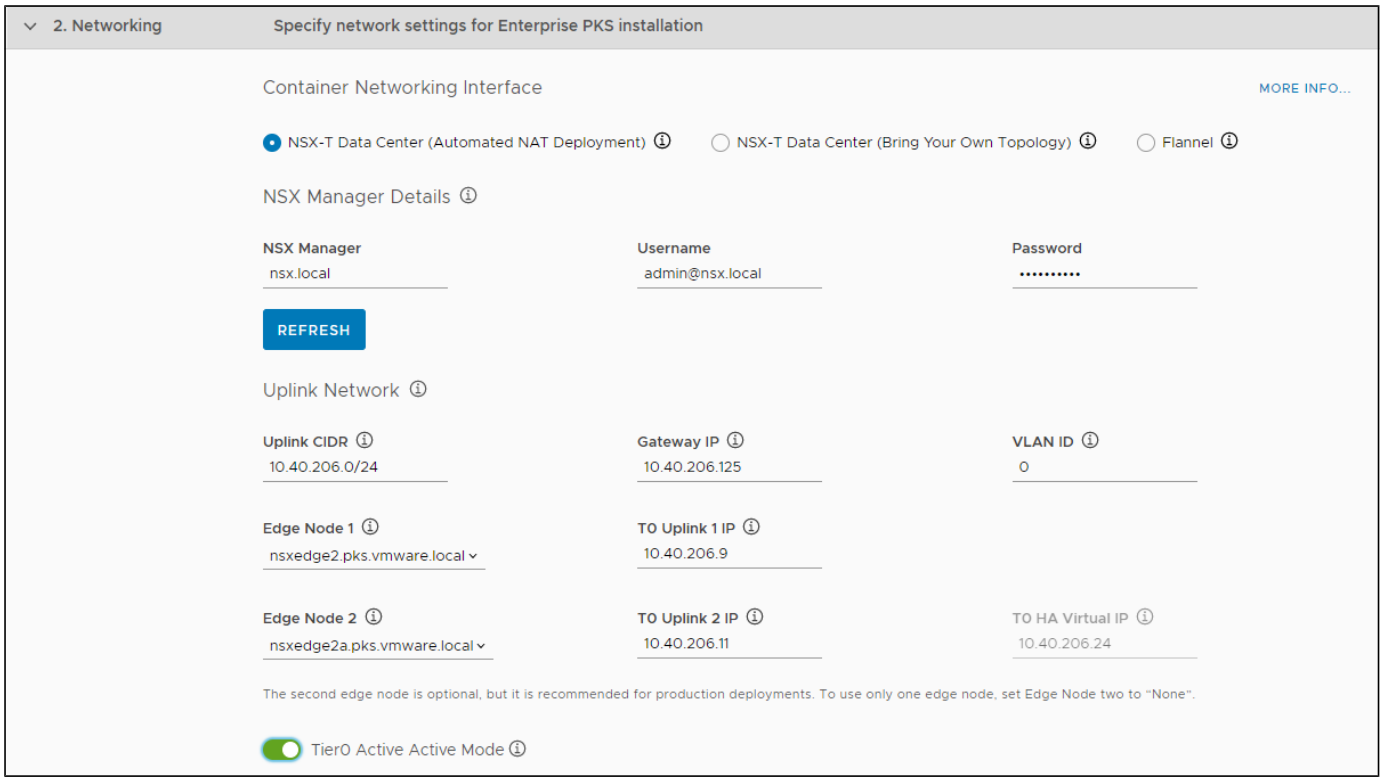
View a larger version of this image.

> **Important**: You cannot change the type of networking after you deploy Enterprise PKS.
>
> You cannot configure plans to use Windows worker nodes with NSX-T Data Center networking in this release. You can currently only use Windows worker nodes if you implement Flannel networking.

Configure an Automated NAT Deployment to NSX-T Data Center

Provide information about an NSX-T Data Center network that you have not already configured for use with Enterprise PKS. You provide information about your NSX-T Data Center setup, and Enterprise PKS Management Console creates the necessary objects and configures them for you. Make sure that your NSX-T Data Center setup satisfies the Prerequisites for an Automated NAT Deployment to NSX-T Data Center before you begin.

1. Select the **NSX-T Data Center (Automated NAT Deployment)** radio button.
2. Configure the connection to NSX Manager.
   - Enter the IP address or FQDN of NSX Manager.
   - Enter the user name and password for an NSX administrator account.
3. Click **Connect**.
4. Enter information about the uplink network.
   - **Uplink CIDR**: Enter a CIDR range within the uplink subnet for the Tier 0 uplinks, for example 10.40.206.0/24.
   - **Gateway IP**: Enter the IP address for the gateway, for example 10.40.206.125.
   - **VLAN ID**: Enter the VLAN ID within the range 0 to 4095, for example 1206.
   - **Edge Node 1**: Select an Edge Node from the drop-down menu, for example `nsx-edge-1`.
   - **T0 Uplink 1 IP**: Enter the IP address of the Tier 0 uplink 1, for example 10.40.206.9.
   - **Edge Node 2**: Select an Edge Node from the drop-down menu, for example `nsx-edge-2`. The second edge node is optional for proof-of-concept deployments, but it is strongly recommended for production deployments. To use only one edge node, set Edge Node two to “None”.
   - **T0 Uplink 2 IP**: Enter the IP address of the Tier 0 uplink 1, for example 10.40.206.11.
   - **T0 HA Virtual IP**: Enter the IP address for the HA Virtual IP, for example 10.40.206.24.
5. Optionally enable **Tier0 Active Active Mode**.
   By default, the management console sets the high availability (HA) mode of the tier 0 router to active-standby. You can optionally enable active-active mode on the tier 0 router, so that all NAT configuration moves from the tier 0 to the tier 1 router.



View a larger version of this image.

6. Enter information about the network resources for the Enterprise PKS deployment to use.
   - **Deployment CIDR**: Enter a CIDR range to use for Enterprise PKS components, for example 10.192.182.1/22.
   - **Deployment DNS**: Enter the IP address of the DNS server to use for deploying Enterprise PKS components, for example 192.168.111.155.
   - **NTP Server**: Enter the IP address of an NTP server.
   - **Pod IP Block CIDR**: Enter a CIDR range to use for pods, with a maximum suffix of 24. For example 11.192.183.1/22.
   - **Node IP Block CIDR**: Enter a CIDR range to use for nodes, with a maximum suffix of 22. For example 11.192.184.1/22.
   - **Nodes DNS**: Enter the Domain Name Server used by the Kubernetes nodes.
   - **Deployment Network Reserved IP Range**: Optionally enter a range of IP addresses in the **From** and **To** text boxes. No VMs are deployed in this range. You cannot modify reserved IP ranges after the initial deployment.
   - **Usable range of floating IPs**: Enter the floating IP range, for example **From** 192.168.160.100 **To** 192.168.160.199. Click **Add Range** to add more IP ranges.

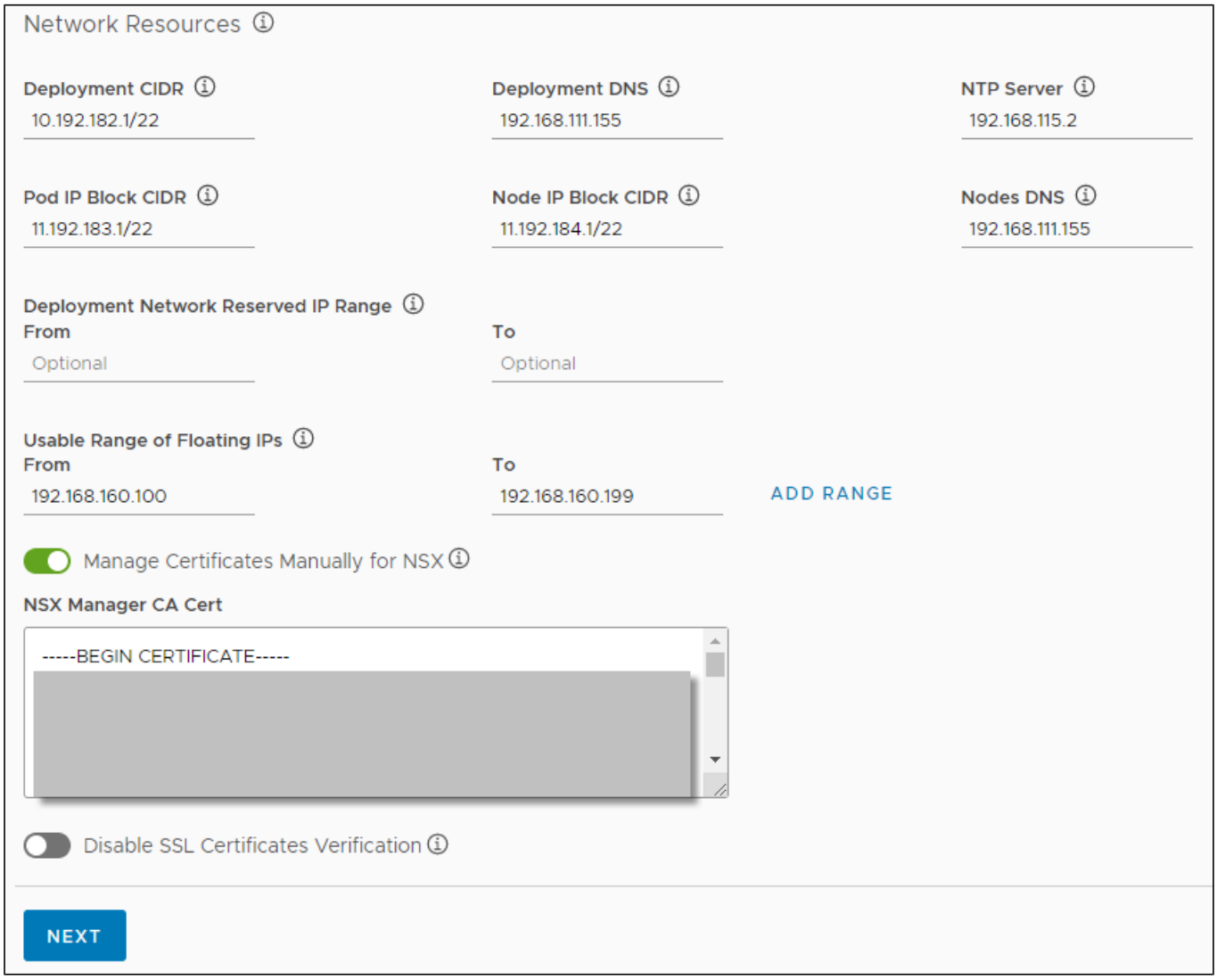
7. Optionally enable **Manage certificates manually for NSX** if NSX Manager uses a custom CA certificate.

   > **Note**: If NSX-T Data Center uses custom certificates and you do not provide the CA certificate for NSX Manager, Enterprise PKS Management Console automatically generates one and registers it with NSX Manager. This can cause other services that are integrated with NSX Manager not to function correctly.

   Enter the contents of the CA certificate in the **NSX Manager CA Cert** text box:

   ```
   -----BEGIN CERTIFICATE-----
   nsx_manager_CA_certificate_contents
   -----END CERTIFICATE-----
   ```

   If you do not select **Manage certificates manually for NSX**, the management console generates a certificate for you.



View a larger version of this image.

8. Optionally enable **Disable SSL certificates verification** to allow unsecured connections to NSX Manager.

9. Click **Next** to configure identity management.

For the next steps, see Configure Identity Management.

Configure a Bring Your Own Topology Deployment to NSX-T Data Center

Provide information about an NSX-T Data Center network that you have already fully configured for use with Enterprise PKS. Make sure that your NSX-T Data Center setup satisfies the Prerequisites for a Bring Your Own Topology Deployment to NSX-T Data Center before you begin.

1. Select the **NSX-T Data Center (Bring Your Own Topology)** radio button.

2. Configure the connection to NSX Manager.

   - Enter the IP address or FQDN of the NSX Manager.
   - Enter the user name and password for an NSX administrator account.

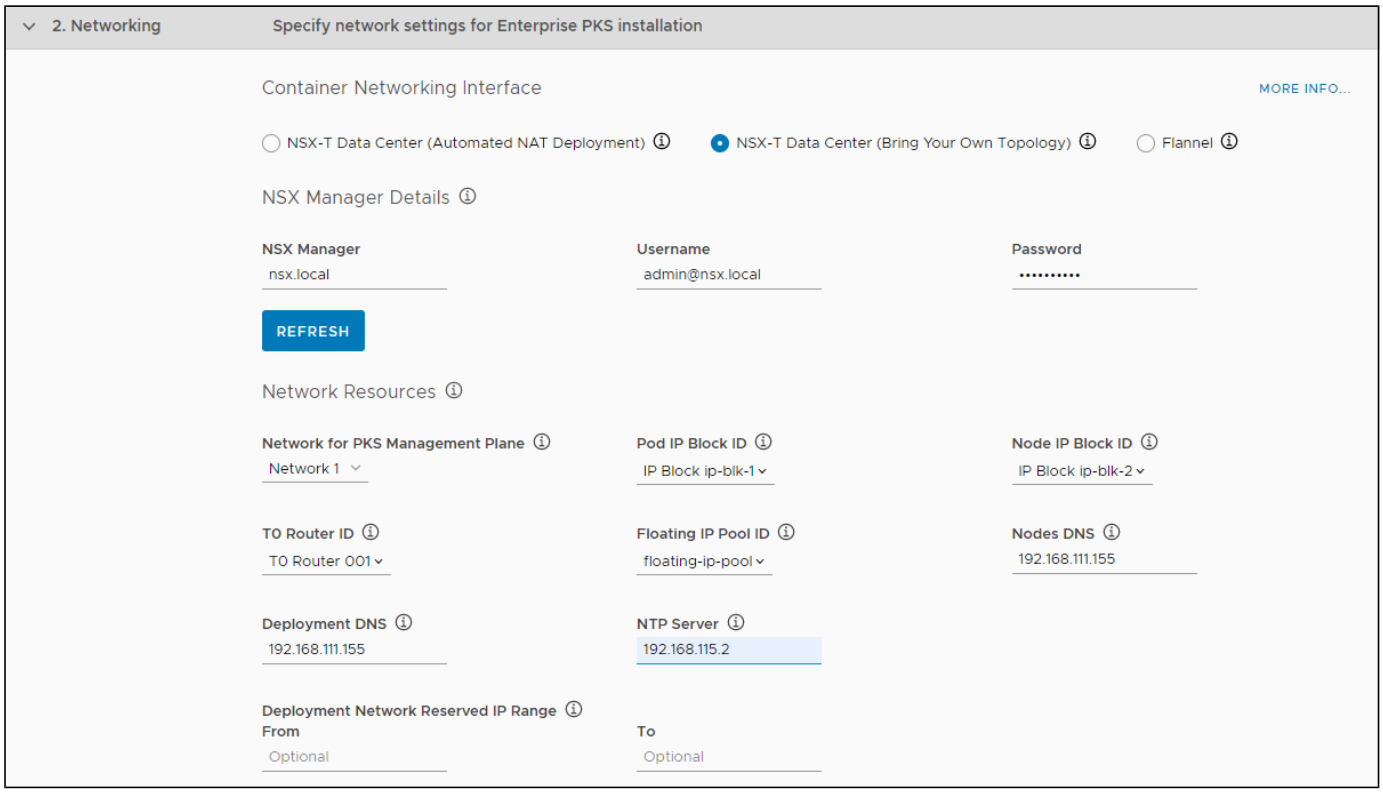
3. Click **Connect**.

4. Use the drop-down menus to select existing network resources for each of the following items.
   - **Network for PKS Management Plane**: Select the name of an opaque network on an NSX-T Virtual Distributed Switch (N-VDS).

     > **important**: Do not use the network on which you deployed the Enterprise PKS Management Console appliance VM as the network for the management plane. Using the same network for the appliance VM and the management plane requires additional NSX-T Data Center configuration and is not recommended.

   - **Pod IP Block ID**: Select the UUID for the IP block to use for Kubernetes pods.
   - **Node IP Block ID**: Select the UUID for the IP block to use for Kubernetes nodes.
   - **T0 Router ID**: Select the UUID for the Tier-0 Logical Router configured in NSX-T Data Center.
   - **Floating IP Pool ID**: Select the UUID for the Floating IP Pool.

5. Enter IP addresses for the following resources.
   - **Nodes DNS**: Enter the IP address for the DNS server to use for Kubernetes nodes and pods.
   - **Deployment DNS**: Enter the IP address for the DNS server to use for the PKS control plane VMs, for example 192.168.111.155.
   - **NTP Server**: Enter the IP address of an NTP server.
   - **Deployment Network Reserved IP Range**: Optionally enter a range of IP addresses in the **From** and **To** text boxes. No VMs are deployed in this range. You cannot modify reserved IP ranges after the initial deployment.



View a larger version of this image.

6. Optionally disable **NAT Mode** to implement a routable (No-NAT) topology.
   Enterprise PKS supports NAT topologies, No-NAT with logical switch (NSX-T) topologies, and multiple tier-0 routers for tenant isolation. For information about implementing a routable topology, see No-NAT Topology in *NSX-T Deployment Topologies for Enterprise PKS*.

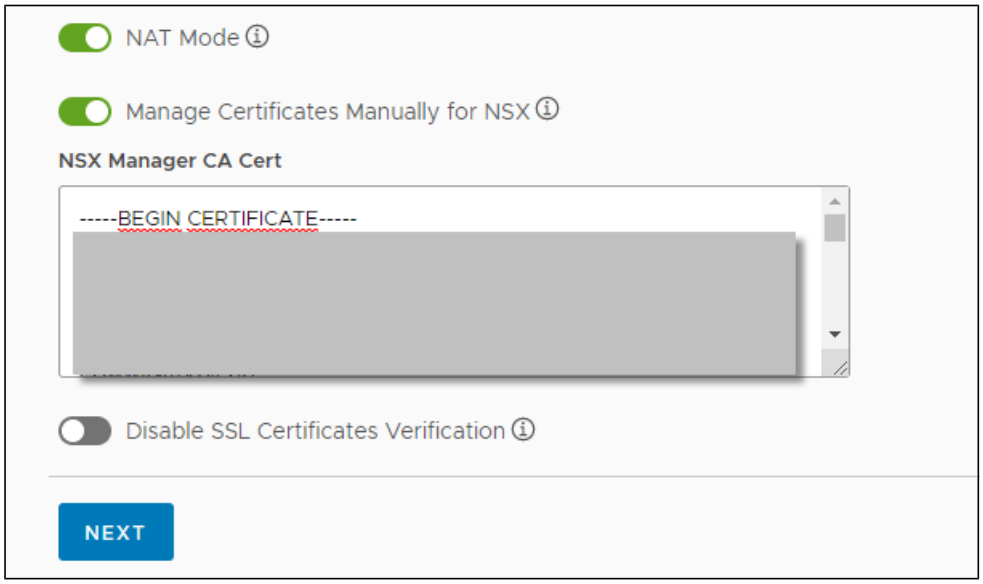
7. Optionally enable **Manage certificates manually for NSX** if NSX Manager uses a custom CA certificate.

   Enter the contents of the CA certificate in the **NSX Manager CA Cert** text box:

   ```
   -----BEGIN CERTIFICATE-----
   nsx_manager_CA_certificate_contents
   -----END CERTIFICATE-----
   ```

   If you do not select **Manage certificates manually for NSX**, the management console generates a certificate for you.

8. Optionally enable **Disable SSL certificates verification** to allow unsecured connections to NSX Manager.



9. Click **Next** to configure identity management.

For the next steps, see Configure Identity Management.

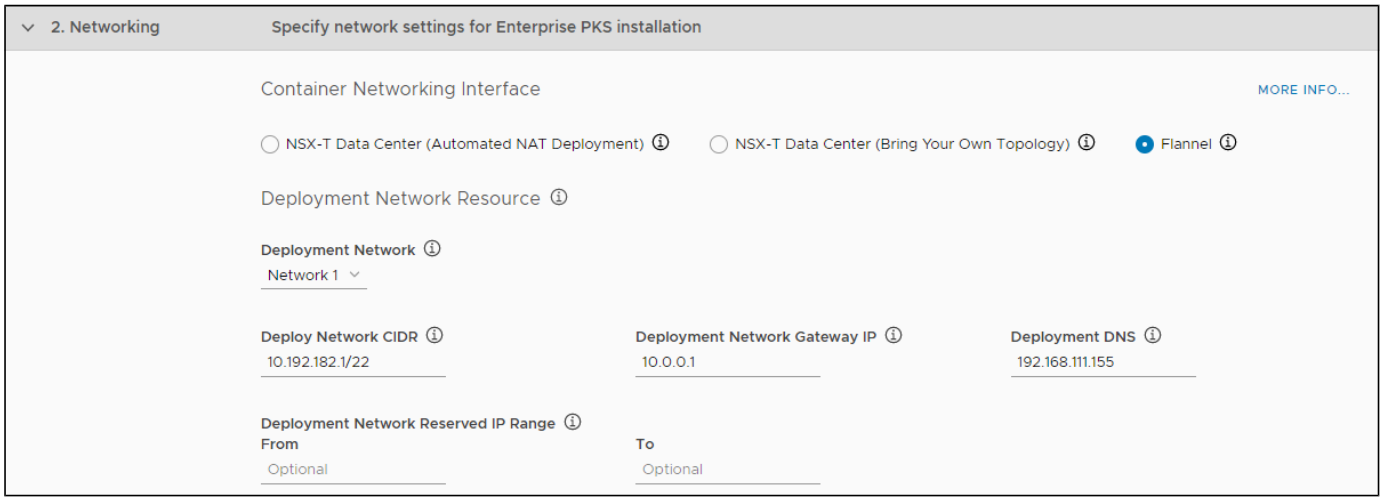
Configure a Flannel Network

Provide networking information so that Enterprise PKS Management Console can provision a Flannel network for you during deployment. Make sure that you have the information listed in Prerequisites for a Flannel Network before you begin.

1. Select the **Flannel** radio button.

2. Configure the Deployment Network Resource options.

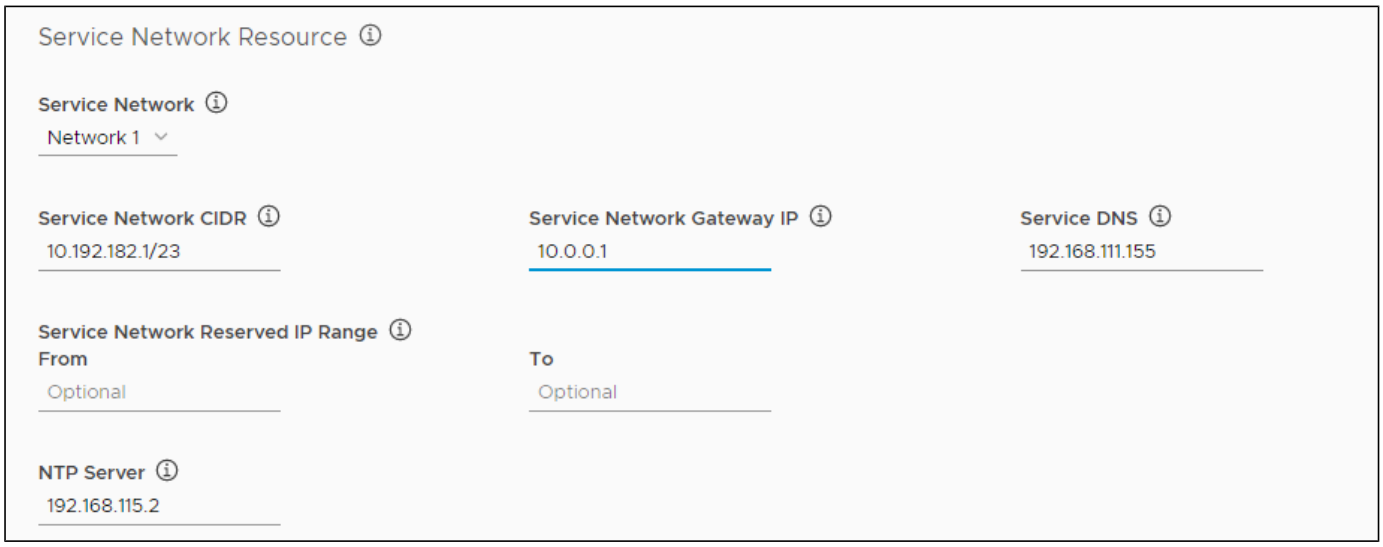
   - **Deployment Network**: Select a vSphere network on which to deploy Enterprise PKS.
   - **Deployment Network CIDR**: Enter a CIDR range to use for Enterprise PKS components, for example 10.192.182.1/22.
   - **Deployment Network Gateway IP**: Enter the IP address for the gateway for the deployment network, for example 10.0.0.1.
   - **Deployment DNS**: Enter the IP address for the deployment network DNS server, for example 192.168.111.155.
   - **Deployment Network Reserved IP Range**: Optionally enter a range of IP addresses in the **From** and **To** text boxes. No VMs are deployed in this range. You cannot modify reserved IP ranges after the initial deployment.



View a larger version of this image.

3. Configure the Service Network Resource options.

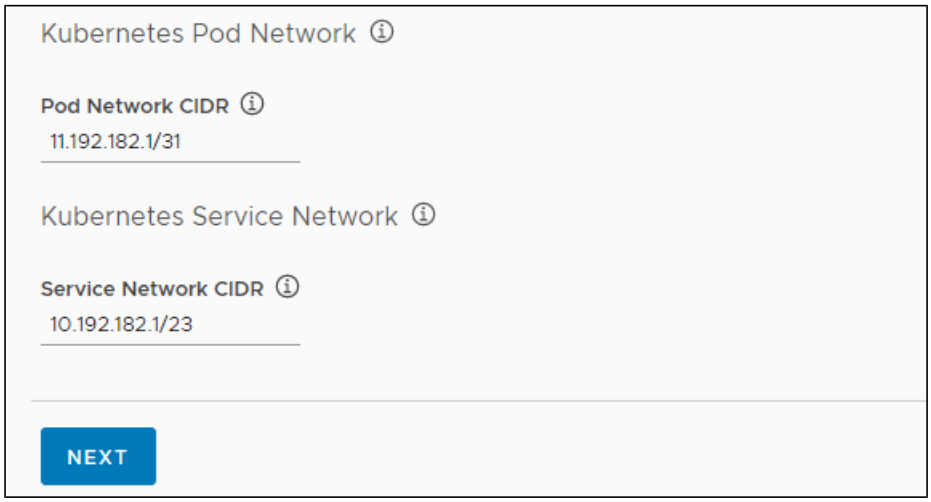
- **Service Network**: Select a vSphere network to use as the service network.
- **Service Network CIDR**: Enter a CIDR range to use for the service network, for example 10.192.182.1/23.
- **Service Network Gateway IP**: Enter the IP address for the gateway for the service network.
- **Service DNS**: Enter the IP address for the service network DNS server, for example 192.168.111.155.
- **Service Network Reserved IP Range**: Optionally enter a range of IP addresses in the **From** and **To** text boxes. No VMs are deployed in this range. You cannot modify the reserved IP range after the initial deployment.
- **NTP Server**: Enter the IP address of an NTP server.



View a larger version of this image.

4. Configure the Kubernetes network options.

- **Pod Network CIDR**: Enter a CIDR range to use for pods, for example 11.192.182.1/31.
- **Service Network CIDR**: Enter a CIDR range to use for the Kubernetes services, for example 10.192.182.1/23.



5. Click **Next** to configure identity management.

Step 3: Configure Identity Management

Enterprise PKS Management Console provides 3 identity management options for your Enterprise PKS deployments.

- A local database of users. See Use a Local Database below for instructions.
- Connect to an external Active Directory or LDAP server. See Use an External LDAP Server below for instructions.
- Connect to a SAML identity provider. See Use a SAML Identity Provider below for instructions.

Use a Local Database

You can manage users by using a local database that is created during Enterprise PKS deployment. After deployment, you can add users and groups to the database and assign roles to them in the Identity Management view of the Enterprise PKS Management Console.

1. Select the **Local user database** radio button.
2. In the **PKS API FQDN** text box, enter an address for the PKS API Server VM, for example `api.pks.example.com`.

For the next steps, see Optionally Configure UAA and Custom Certificates.

Use an External LDAP Server

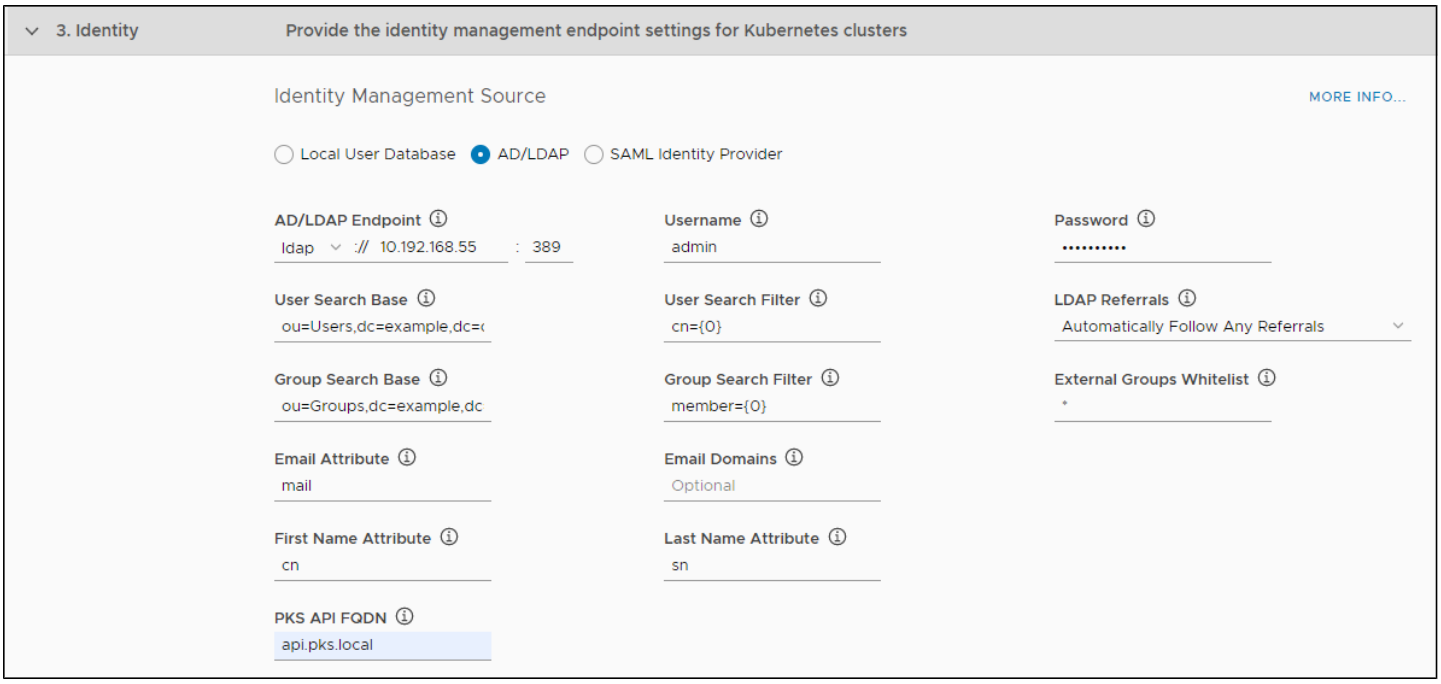
Provide information about an existing external Active Directory or LDAP server.

1. Select the **AD/LDAP** radio button.
2. For **AD/LDAP Endpoint**, select **ldap** or **ldaps** from the drop-down menu and enter the IP address and port of the AD or LDAP server.
3. Enter the username and password to use to connect to the server.
4. Enter the remaining details for your server:
    - **User Search Base**: Enter the location in the AD/LDAP directory tree where user search begins. For example, a domain

named `cloud.example.com` might use `ou=Users,dc=example,dc=com`.

- **User Search Filter**: Enter a string to use for user search criteria. For example, the standard search filter `cn=Smith` returns all objects with a common name equal to `Smith`. Use `cn={0}` to return all LDAP objects with the same common name as the username.
- **LDAP Referrals**: Select how to handle references to alternate locations in which AD/LDAP requests can be processed:
  - Automatically follow referrals
  - Ignore referrals
  - Abort authentication
- **Group Search Base**: Optionally enter the location in the AD/LDAP directory tree where group search begins. For example, a domain named `cloud.example.com` might use `ou=Groups,dc=example,dc=com`.
- **Group Search Filter**: Enter a string that defines AD/LDAP group search criteria, such as `member={0}`.
- **External Groups Whitelist**: Optionally enter a comma-separated list of group patterns to be populated in the user's `id_token`.
- **Email Attribute**: Enter the attribute name in the AD/LDAP directory that contains user email addresses. For example, `mail`.
- **Email Domains**: Optionally enter a comma-separated list of the email domains for external users who can receive invitations to Enterprise PKS.
- **First Name Attribute**: Optionally enter the attribute name in the AD/LDAP directory that contains user first names, for example `cn`.
- **Last Name Attribute**: Optionally enter the attribute name in the AD/LDAP directory that contains user last names. for example `sn`.

5. In the **PKS API FQDN** text box, enter an address for the PKS API Server VM, for example `api.pks.example.com`.



View a larger version of this image.

For the next steps, see Optionally Configure UAA and Custom Certificates.

Use a SAML Identity Provider

You can configure Enterprise PKS so that Kubernetes authenticates users against a SAML identity provider. Before you configure a

SAML identity provider, you must configure your identity provider to designate Enterprise PKS as a service provider. For information about how to configure Okta and Azure Active Directory, see the following topics:

- Configuring Okta as a SAML Identity Provider
- Configuring Azure Active Directory as a SAML Identity Provider

After you have configured your identity provider, enter information about the provider in Enterprise PKS Management Console.

1. Select the **SAML Identity Provider** radio button.

2. For **Provider Name**, enter a unique name you create for the Identity Provider.
   This name can include only alphanumeric characters, `+`, `_`, and `-`. You must not change this name after deployment because all external users use it to link to the provider.

3. For **Display Name**, enter a display name for your provider.
   The display name appears as a link on your login page.

4. Enter the metadata from your identity provider either as XML or as a URL.

   - Download your identity provider metadata and paste the XML into **Provider Metadata**.
   - If your identity provider exposes a metadata URL, enter it in **Provider Metadata URL**.

5. For **Name ID Format**, select the name identifier format for your SAML identity provider.
   This translates to `username` on Enterprise PKS. The default is `Email Address`.

6. For **First Name Attribute** and **Last Name Attribute**, enter the attribute names in your SAML database that correspond to the first and last names in each user record.
   These fields are case sensitive.

7. For **Email Attribute**, enter the attribute name in your SAML assertion that corresponds to the email address in each user record, for example, `EmailID`.
   This field is case sensitive.

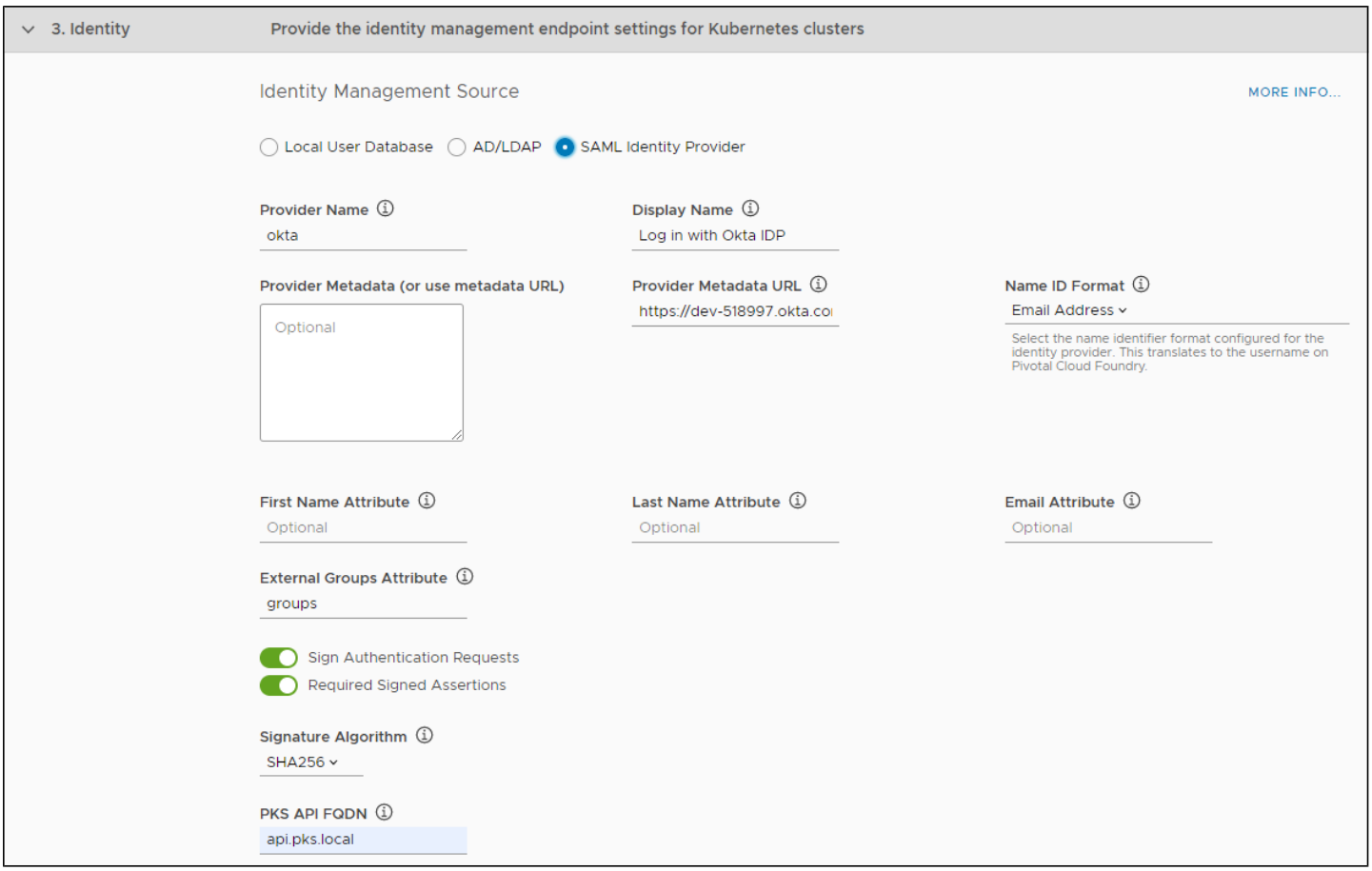
8. For **External Groups Attribute**, enter the attribute name in your SAML database for your user groups.
   This field is case sensitive. To map the groups from the SAML assertion to admin roles in Enterprise PKS, see Grant Enterprise PKS Access to an External LDAP Group.

9. By default, all SAML authentication requests from Enterprise PKS are signed, but you can optionally disable **Sign Authentication Requests**.
   If you disable this option, you must configure your identity provider to verify SAML authentication requests.

10. To validate the signature for the incoming SAML assertions, enable **Required Signed Assertions**.
    If you enable this option, you must configure your Identity Provider to send signed SAML assertions.

11. For **Signature Algorithm**, choose an algorithm from the drop down to use for signed requests and assertions.
    The default value is SHA256.

12. In the **PKS API FQDN** text box, enter an address for the PKS API Server VM, for example `api.pks.example.com`.



View a larger version of this image.

For the next steps, see Optionally Configure UAA and Custom Certificates.

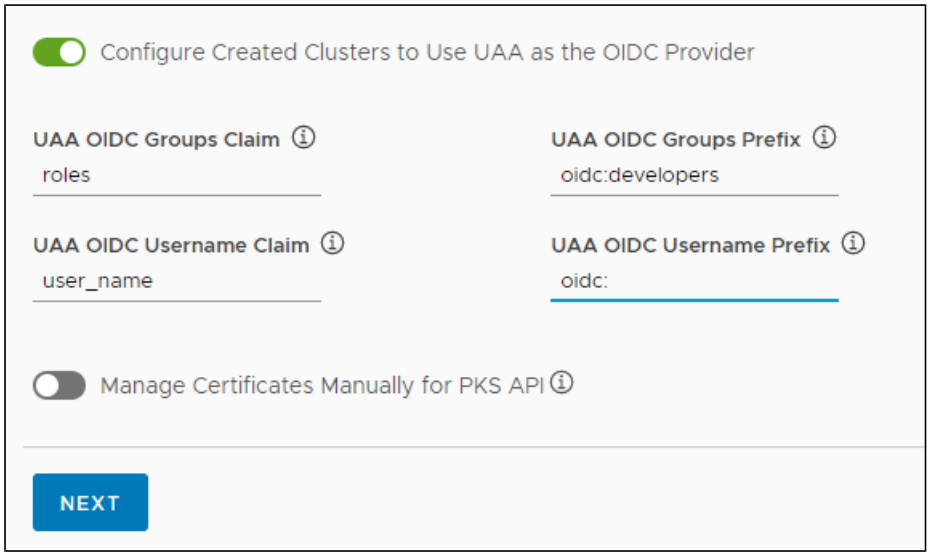
Optionally Configure UAA and Custom Certificates

However you manage identities, you can use OpenID Connect (OIDC) to instruct Kubernetes to verify end-user identities based on authentication performed by a User Account and Authentication (UAA) server. Using OIDC lets you set up an external IDP, such as Okta, to authenticate users who access Kubernetes clusters with `kubectl`. If you enable OIDC, administrators can grant namespace-level or cluster-wide access to Kubernetes end users. If you do not enable OIDC, you must use service accounts to authenticate `kubectl` users.

> **Note**: You cannot enable OIDC if you intend to integrate Enterprise PKS with VMware vRealize Operations Management Pack for Container Monitoring.

1. Optionally select **Configure created clusters to use UAA as the OIDC provider** and provide the following information.
   - **UAA OIDC Groups Claim**: Sets the `--oidc-groups-claim` flag on the kube-api server. Enter the name of your groups claim. This is used to set a user's group in the JSON Web Token (JWT) claim. The default value is `roles`.
   - **UAA OIDC Groups Prefix**: Sets the `--oidc-groups-prefix` flag. Enter a prefix for your groups claim. This prevents conflicts with existing names. For example, if you enter the prefix `oidc:`, UAA creates a group name like `oidc:developers`.
   - **UAA OIDC Username Claim**: Sets the `--oidc-username-claim` flag. Enter the name of your username claim. This is used to set a user's username in the JWT claim. The default value is `user_name`. Depending on your provider, admins can enter claims besides `user_name`, such as `email` or `name`.
   - **UAA OIDC Username Prefix**: Sets the `--oidc-username-prefix` flag. Enter a prefix for your username claim. This prevents conflicts with existing names. For example, if you enter the prefix `oidc:`, UAA creates a username like

`oidc:admin`.



2. Optionally select **Manage Certificates Manually for PKS API** to generate and upload your own certificates for the PKS API Server.

   If you do not select this option, the management console creates auto-generated, self-signed certificates.

   Enter the contents of the certificate in the **PKS API Certificate** text box:

   ```
   -----BEGIN CERTIFICATE-----
   pks_api_certificate_contents
   -----END CERTIFICATE-----
   ```

   Enter the contents of the certificate key in the **Private Key PEM** text box:

   ```
   -----BEGIN PRIVATE KEY-----
   pks_api_private_key_contents
   -----END PRIVATE KEY-----
   ```

3. Click **Next** to configure availability zones.

Step 4: Configure Availability Zones

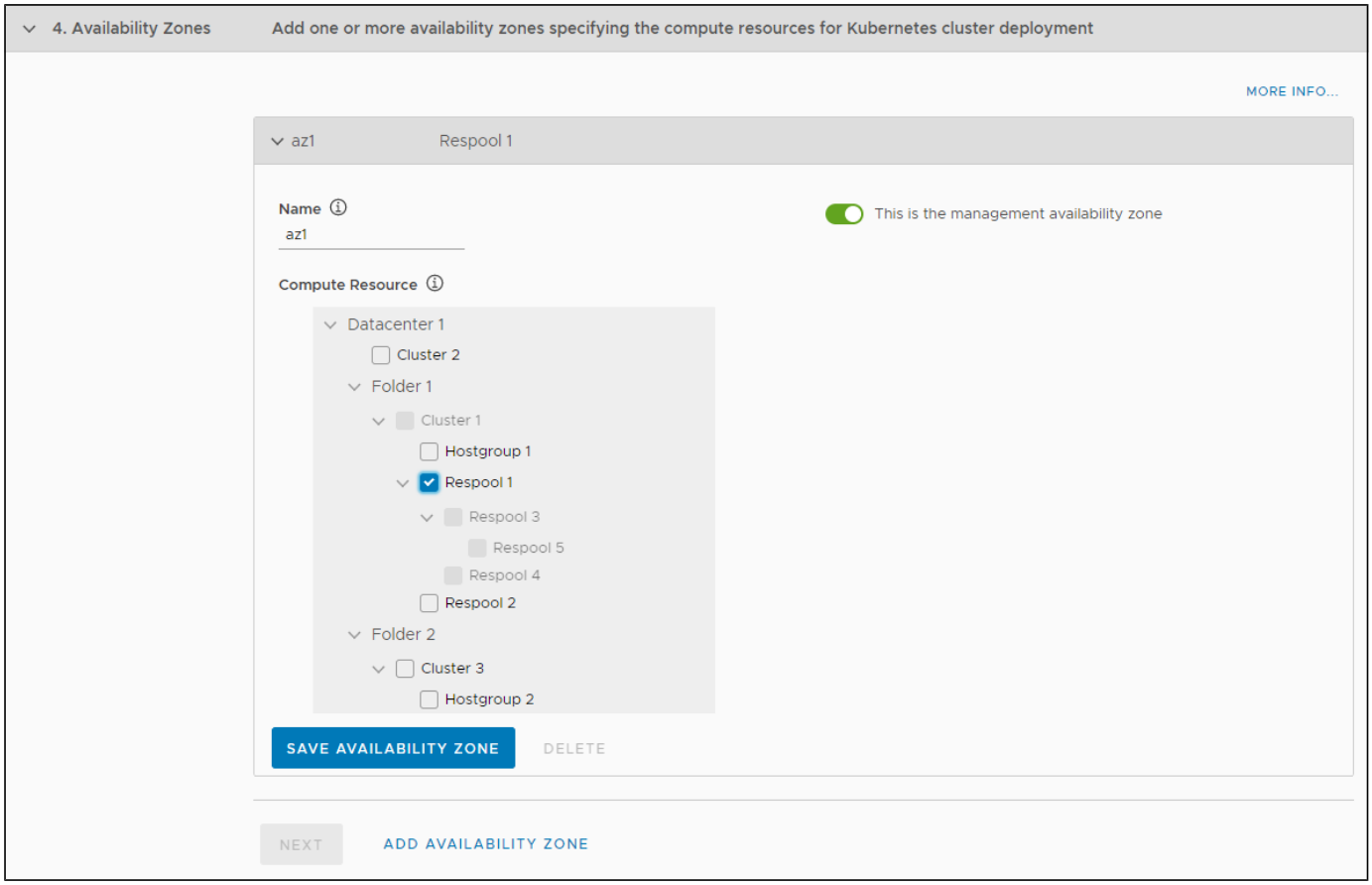
Availability zones specify the compute resources for Kubernetes cluster deployment. Availability zones are a BOSH construct, that in Enterprise PKS deployments to vSphere correspond to vCenter Server clusters, host groups, and resource pools. Availability zones allow you to provide high-availability and load balancing to applications. When you run more than one instance of an application, those instances are balanced across all of the availability zones that are assigned to the application. You must configure at least one availability zone. You can configure multiple additional availability zones.

1. In the **Name** field, enter a name for the availability zone.

2. Optionally select **This is the management availability zone**.
   The management availability zone is the availability zone in which to deploy the PKS Management Plane. The management plane consists of the PKS API VM, Ops Manager, BOSH Director, and Harbor Registry. You can only designate one availability zone as the management zone. If you do not designate an availability zone as the management zone, Enterprise PKS Management Console selects the first one.

3. In the **Compute Resource** tree, select clusters, host groups, or resource pools for this availability zone to use.

4. Click **Save Availability Zone**.



View a larger version of this image.

5. Optionally click **Add Availability Zone** to add another zone.
   You can only select resources that are not already included in another zone. You can create multiple availability zones.

6. Click **Save Availability Zone** for every additional availability zone that you create.

7. Click **Next** to configure storage.

Step 5: Configure Resources and Storage

Resource Settings allow you to configure the resources that are allocated to the Enterprise PKS API VM. Allocate resources according to the workloads that Enterprise PKS will run.

You must also designate the datastores to use for the different types of storage required by your Enterprise PKS deployment.

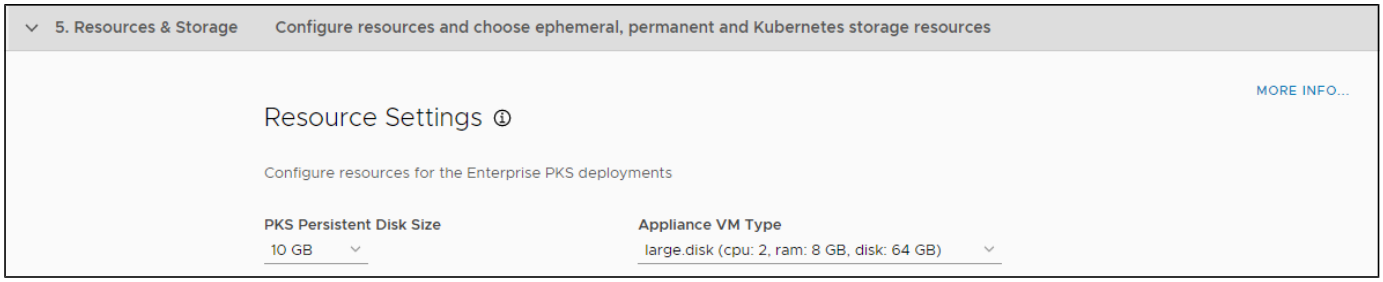
- Ephemeral storage is used to contain the files for ephemeral VMs that Enterprise PKS creates during installation, upgrade, and operation. Ephemeral VMs are automatically created and deleted as needed.
- Permanent storage is used for permanent Enterprise PKS data.
- Kubernetes persistent volume storage is used to store Kubernetes persistent volumes, for use in stateful applications.

You can use different datastores for the storage of permanent and ephemeral data. If you disable the permanent storage option, Enterprise PKS uses the ephemeral storage for permanent data. For information about when it is appropriate to share the ephemeral,

permanent, and persistent volume datastores or use separate ones, see PersistentVolume Storage Options on vSphere.

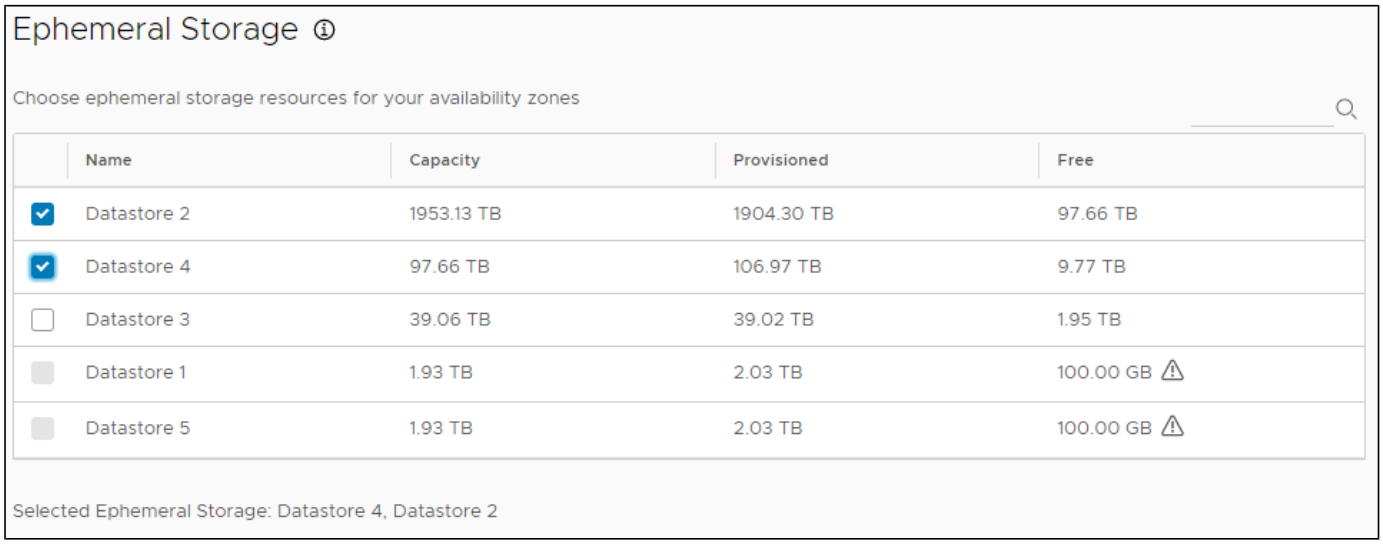
You can use VMware vSAN, Network File Share (NFS), or VMFS storage for ephemeral, permanent, and Kubernetes persistent storage. Datastores can only be selected if their minimum capacity is greater than 250GB.

1. For **PKS Persistent Disk Size**, select the size of the persistent disk for the Enterprise PKS API VM.
   Set the PKS persistent disk size according to the number of pods that you expect the cluster workload to run continuously. It is recommended to allocate 10GB for every 500 pods. For example:
   - For 1000 pods, allocate 20GB
   - For 10,000 pods, allocate 200GB
   - For 50,000 pods, allocate 1TB
2. Use the **Appliance VM Type** drop-down menu to select from different combinations of CPU, RAM, and storage for the Enterprise PKS API VM.
   Choose the configuration for the API VM depending on the expected CPU, memory, and storage consumption of the workloads that it will run. For example, some workloads might require a large compute capacity but relatively little storage, while others might require a large amount of storage and less compute capacity.



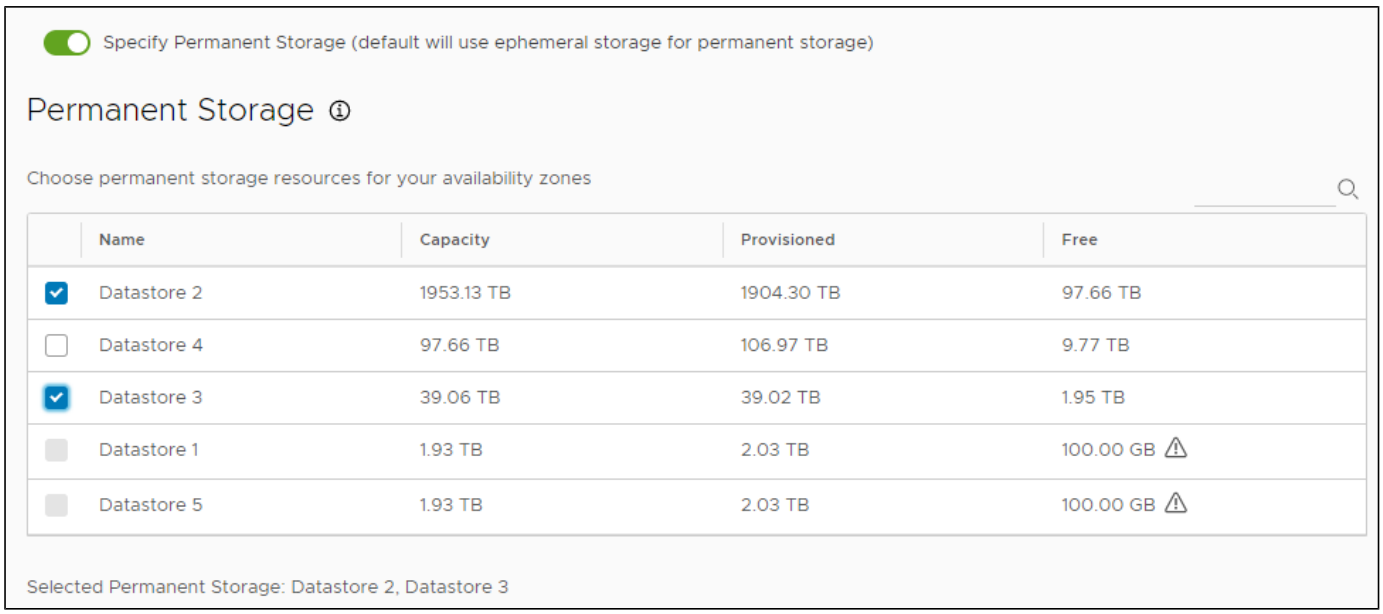
View a larger version of this image.

3. Under **Ephemeral Storage**, select one or more datastores for use as ephemeral storage, or use the search field on the right to find datastores by name.



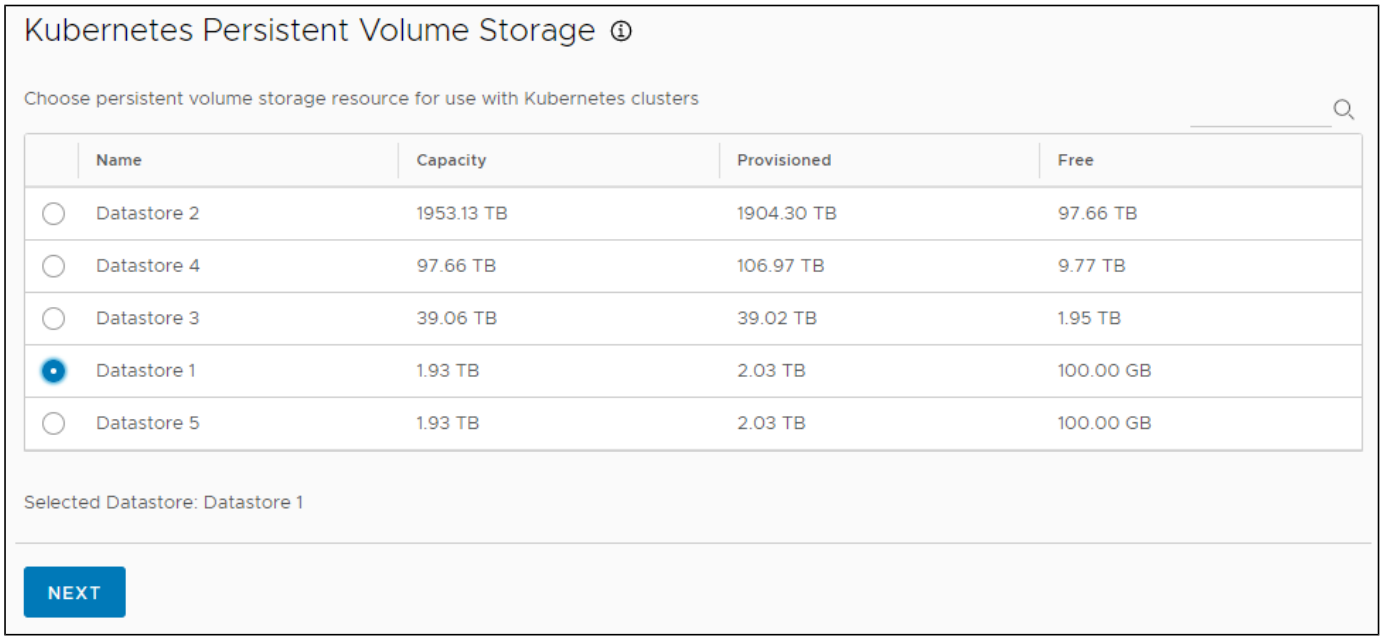
View a larger version of this image.

4. Optionally enable **Specify Permanent Storage** to designate different datastores for ephemeral and permanent data.
5. If you enabled permanent storage, under **Permanent Storage**, select one or more datastores for permanent storage, or use the search field to find datastores by name.



View a larger version of this image.

6. Under **Kubernetes Persistent Volume Storage**, select one datastore in which to store Kubernetes volumes, or use the search field to find datastores by name.



View a larger version of this image.

7. Click **Next** to configure plans.

Step 6: Configure Plans

A plan is a cluster configuration template that defines the set of resources for Enterprise PKS to use when deploying Kubernetes clusters. A plan allows you to configure the numbers of master and worker nodes, select between Linux and Windows OS for worker nodes, specify the configuration of the master and worker VMs, set disk sizes, select availability zones for master and node VMs, and configure advanced settings.

notes about windows worker nodes: In this release, using Windows worker nodes is a beta feature and is intended for

evaluation and test purposes only. Using Windows worker nodes is subject to the following limitations and requires additional configuration after you deploy Enterprise PKS:

You can only use Windows worker nodes if you implement Flannel networking. You cannot use Windows worker nodes with NSX-T Data Center networking.

You can create a maximum of 3 plans that implement Windows worker nodes in a given Enterprise PKS deployment.

If you use Windows worker nodes, certain options are not available, and the default values of other options change. See the option descriptions below for more information.

If you use Windows worker nodes, after you deploy Enterprise PKS, you must use Operations Manager to manually install a Windows Server Stemcell in BOSH. For information about how to install a Windows Server Stemcell and other steps to perform after you deploy Enterprise PKS with Windows worker nodes, see Enable Plans with Windows Worker Nodes.

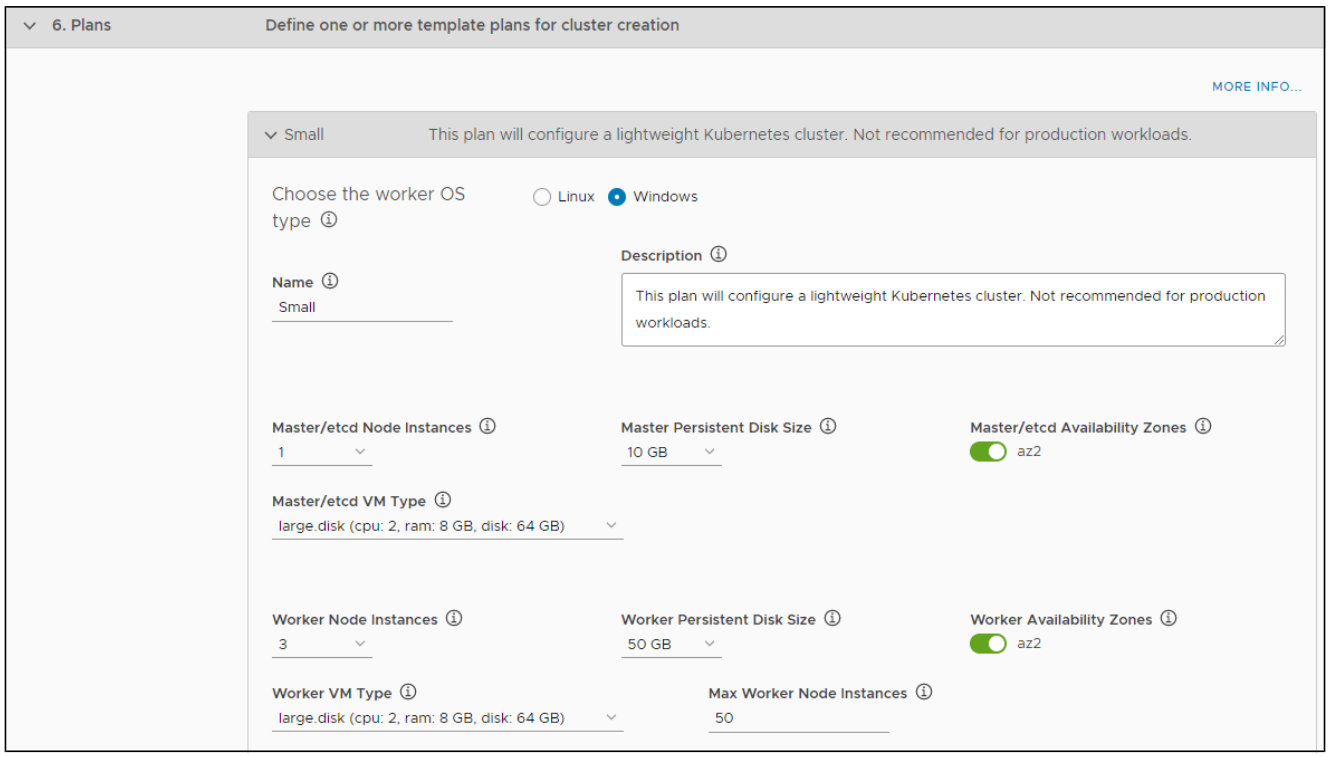
Enterprise PKS Management Console provides preconfigured default plans, for different sizes of Kubernetes clusters. You can change the default configurations, or you can enable the plans as they are. You must enable at least one plan configuration because when you use the PKS CLI to create a Kubernetes cluster, you must specify the plan on which you are basing the Kubernetes cluster. If no plans are enabled, you cannot create Kubernetes clusters.

Enterprise PKS plans support privileged containers and three admission control plugins. For information about privileged containers and the supported admission plugins, see Privileged mode for pods in the Kubernetes documentation. For information about admission plugins, see Enabling, Disabling, and Using Admission Control Plugins for Enterprise PKS Clusters.

1. To use the preconfigured plans as they are, click **Save Plan** for each of the `small`, `medium`, and `large` plans.

2. Optionally use the drop-down menus and buttons to change the default configurations of the preconfigured plans.

   - If you are deploying Enterprise PKS to a Flannel network, for **Choose the worker OS type**, select **Linux** or **Windows**.
   - Enter a name and a description for the plan in the **Name** and **Description** text boxes.
   - **Master/etcd Node Instances**: Select **1** (small), **3** (medium), or **5** (large).
   - **Master Persistent Disk Size**: Select the size of the master persistent disk.
   - **Master/etcd Availability Zones**: Enable one or more availability zones for the master nodes.
   - **Master/etcd VM Type**: Select the size of the Master VM. If you use Windows worker nodes, this option defaults to **large.disk**.
   - **Worker Node Instances**: Specify the number of worker nodes. For a small deployment, 3 is suggested.
   - **Worker Persistent Disk Size**: Select the size of the worker node persistent disk.
   - **Worker Availability Zones**: Enable one or more available availability zones for the worker nodes.
   - **Worker VM Type**: Select a configuration for worker nodes. If you use Windows worker nodes, this option defaults to **large.disk**.
   - **Max Worker Node Instances**: Select the maximum number of worker nodes.

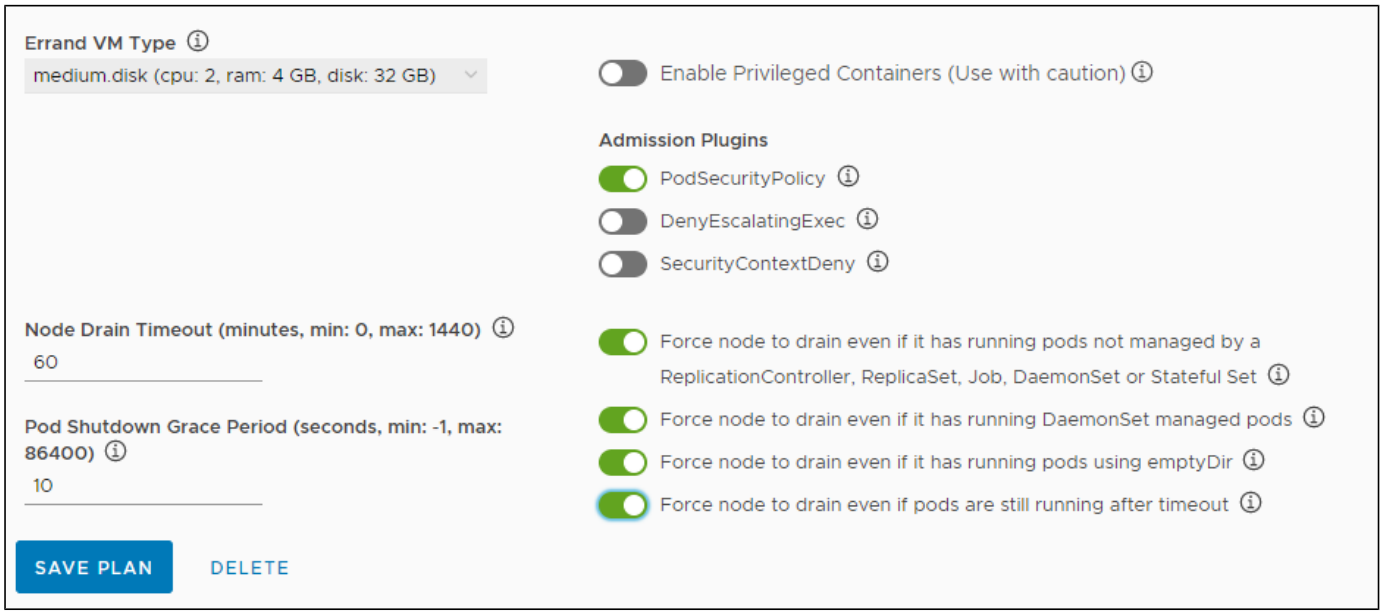


View a larger version of this image.

- **Errand VM Type**: Select the size of the VM to run BOSH errand tasks.
- **Enable Privileged Containers**: Optionally enable privileged container mode. Use with caution. If you use Windows worker nodes, this option is not available.
- **Admission Plugins**: Optionally enable admission plugins. Admission plugins, provide a higher level of access control to the Kubernetes API server and should be used with caution.
  - `PodSecurityPolicy`
  - `DenyEscalatingExec`. If you use Windows worker nodes, this option is not available.
  - `SecurityContextDeny`
- **Node Drain Timeout**: Enter the timeout in minutes for the node to drain pods. If you set this value to 0, the node drain does not terminate. If you use Windows worker nodes, the node drain options are not available. To configure when the nodes drain, optionally enable the following:
  - **Force node to drain even if it has running pods not managed by a ReplicationController, ReplicaSet, Job, DaemonSet or Stateful Set**
  - **Force node to drain even if it has running DaemonSet managed pods**
  - **Force node to drain even if it has running pods using emptyDir**
  - **Force node to drain even if pods are still running after timeout**
- **Pod Shutdown Grace Period**: Enter a timeout in seconds for the node to wait before it forces the pod to terminate. If you set this value to -1, the default timeout is set to the one specified by the pod. If you use Windows worker nodes, this option is not available.

3. Click **Save Plan** for each plan that you edit.

4. Optionally click **Add Plan** to create a new plan, configure it as described above, and click **Save Plan**.
   You can create a maximum of 10 Linux plans and a maximum of 3 Windows plans.



View a larger version of this image.

5. Optionally delete any plans that you do not need.

6. Click **Next** to configure integrations.

Step 7: Configure Integrations

If your infrastructure includes existing deployments of VMware Tanzu Mission Control, Wavefront by VMware, VMware vRealize Operations Management Pack for Container Monitoring, or VMware vRealize Log Insight, you can configure Enterprise PKS to connect to those services. You can also configure Enterprise PKS to forward logs to a Syslog server.

Configure a Connection to VMware Tanzu Mission Control

Participants in the VMware Tanzu Mission Control beta program can use Enterprise PKS Management Console to integrate their Enterprise PKS deployment with Tanzu Mission Control.

Tanzu Mission Control integration lets you monitor and manage Enterprise PKS clusters from the Tanzu Mission Control console, making the Tanzu Mission Control console a single point of control for all Kubernetes clusters.

> **important:** VMware Tanzu Mission Control is currently experimental Beta software and is intended for evaluation and test purposes only. For more information about Tanzu Mission Control, see the VMware Tanzu Mission Control home page.

1. Select the **Enable** toggle to enable the Tanzu Mission Control Integration.

2. For **API URL**, enter the API URL of your Tanzu Mission Control subscription, without a trailing slash `/`).

3. For **Cluster Group Name**, enter the name of a Tanzu Mission Control cluster group.

   - The name can be `default` or another value, depending on your role and access policy:

     - `Org Member` users in VMware cloud services have a `service.admin` role in Tanzu Mission Control. These users:

       - By default, can only create and attach clusters in the `default` cluster group.

        - Can create new cluster groups after an `organization.admin` user grants them the `clustergroup.admin` or `clustergroup.edit` role.

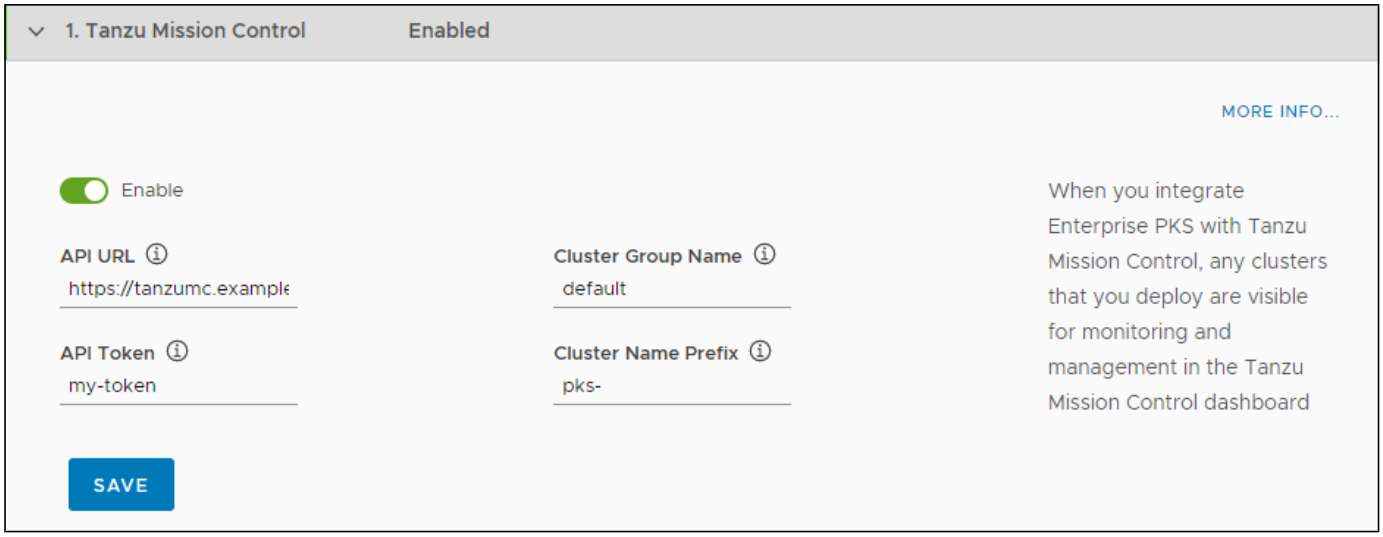
    - VMware cloud services `Org Owner` users have `organization.admin` permissions in Tanzu Mission Control. These users:

        - Can create cluster groups.
        - Can grant `clustergroup` roles to `service.admin` users through the Tanzu Mission Control Access Policy view.

  - **Tanzu Mission Control Cluster Name Prefix**: Enter a name prefix for identifying the PKS clusters in Tanzu Mission Control.

4. For **API token**, Enter your API token to authenticate with VMware Cloud Services APIs. Retrieve this token by logging into VMware Cloud Services and viewing your account information.

5. For **Cluster Name Prefix**, enter a name prefix for identifying the PKS clusters in Tanzu Mission Control. This name prefix cannot contain uppercase letters. For more information, see the see Cluster Group Name Limitation for Tanzu Mission Control Integration in the *Known Issues*.



View a larger version of this image.

6. Click **Save**.

7. Configure integrations with other applications, or click **Next** to install Harbor.

Configure a Connection to Wavefront

By connecting your Enterprise PKS deployment to an existing deployment of Wavefront by VMware, you can obtain detailed metrics about Kubernetes clusters and pods. Before you configure Wavefront integration, you must have an active Wavefront account and access to a Wavefront instance. For more information, including about how to generated a Wavefront access token, see VMware PKS Integration and VMware Enterprise PKS Integration Details in the Wavefront by VMware documentation.

1. Select the **Enable** toggle to enable a connection to Wavefront.

2. Enter the address of your Wavefront instance in the **Wavefront URL** text box.

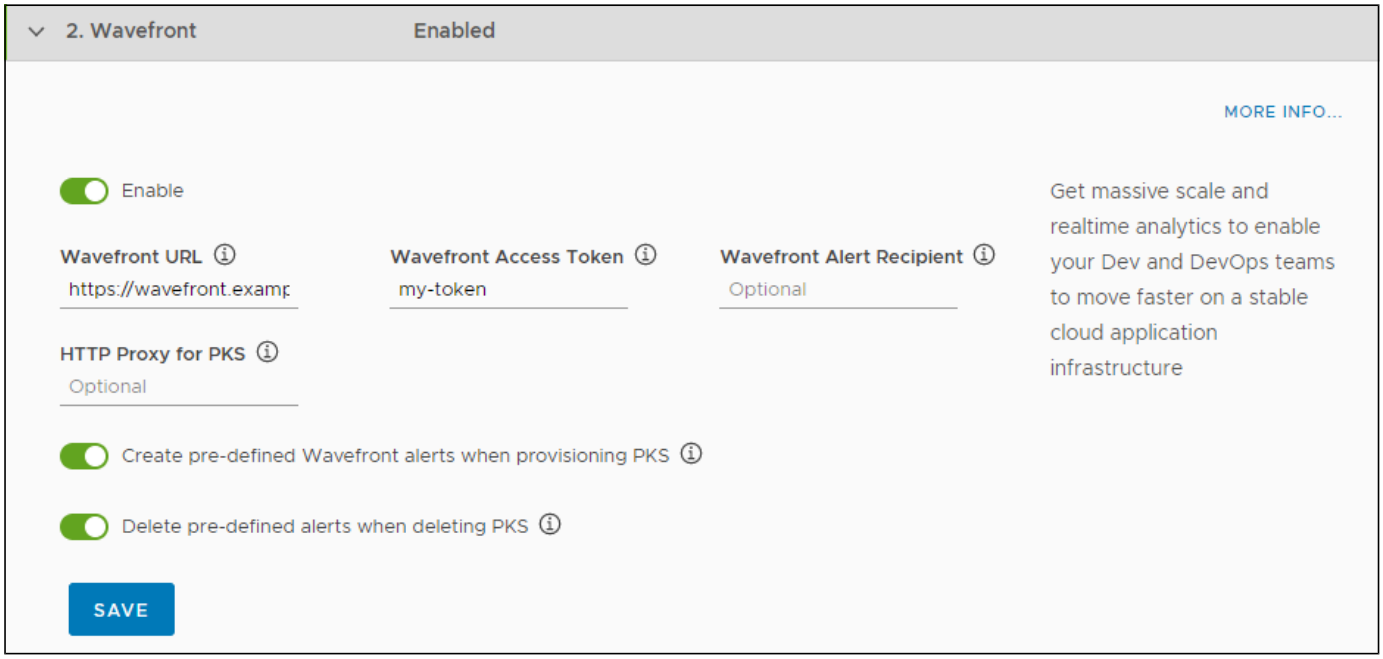
3. Enter the Wavefront API token in the **Wavefront Access Token** text box.

4. Enter an email address to which Wavefront sends alerts in the **Wavefront Alert Recipient** text box.

5. Optionally disable **Create pre-defined Wavefront alerts when provisioning PKS**.

6. In the **HTTP Proxy for PKS** text box, enter the address of the proxy server to use when it is not possible for the Enterprise PKS Wavefront component to connect to an outside address over HTTP. For example, http://your.proxy.com:8080 or https://your.proxy.com:443.

7. Optionally disable **Delete pre-defined alerts when deleting PKS**.



View a larger version of this image.

8. Click **Save**.

9. Configure integrations with other applications, or click **Next** to install Harbor.

Configure a Connection to VMware vRealize Operations Management Pack for Container Monitoring

You can connect your Enterprise PKS deployment to an existing instance of VMware vRealize Operations Management Pack for Container Monitoring. vRealize Operations Management Pack for Container Monitoring provides detailed monitoring of your Kubernetes clusters. vRealize Operations Management Pack for Container Monitoring must be installed, licensed, running, and available in your environment before you enable the option. For more information, see the vRealize Operations Management Pack for Container Monitoring documentation.

If you enable the option to integrate Enterprise PKS with VMware vRealize Operations Management Pack for Container Monitoring, the management console creates a `cAdvisor` container in your Enterprise PKS deployment.

1. Select the **Enable** toggle to enable a connection to vRealize Operations Management Pack for Container Monitoring.

2. Click **Save**.

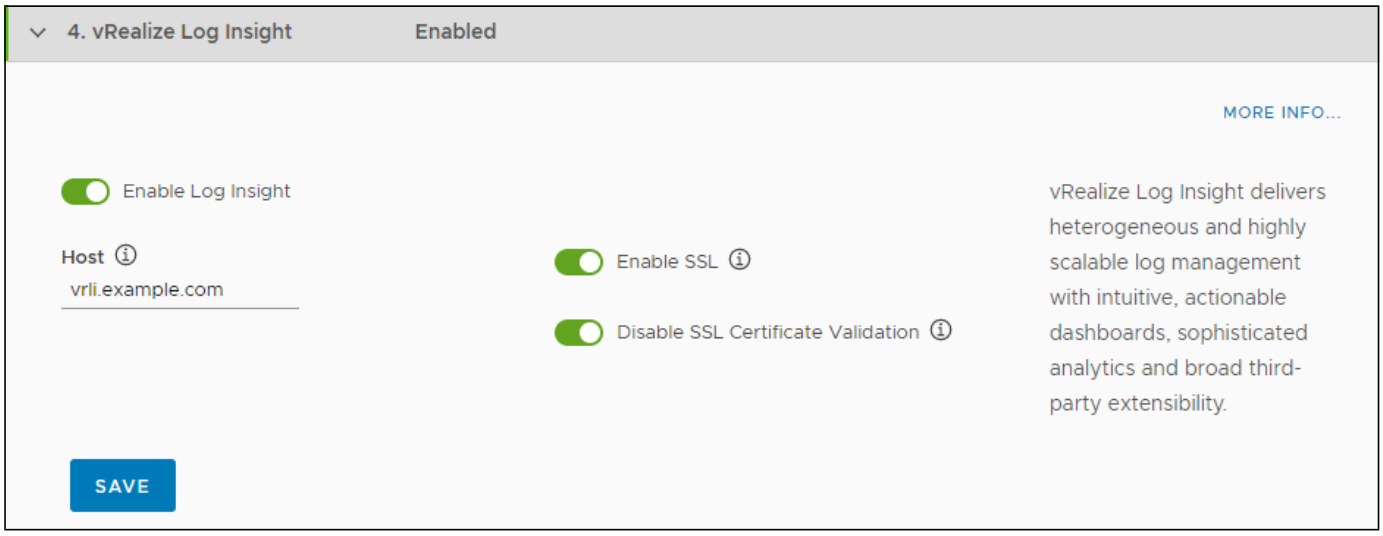
3. Configure integrations with other applications, or click **Next** to install Harbor.

Configure a Connection to VMware vRealize Log Insight

You can configure Enterprise PKS deployment so that an existing deployment of VMware vRealize Log Insight 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.

vRealize Log Insight must be installed, licensed, running, and available in your environment before you enable the option. For instructions and additional information, see the vRealize Log Insight documentation.

1. Select the **Enable** toggle to enable a connection to vRealize Log Insight.
2. Enter the address of your vRealize Log Insight instance in the **Host** text box.
3. Optionally disable **Enable SSL**.
4. Optionally disable **Disable SSL certificate validation**.



View a larger version of this image.

5. Click **Save**.
6. Configure integrations with other applications, or click **Next** to install Harbor.

Note: If you enable integration with vRealize Log Insight, Enterprise PKS Management Console generates a unique vRealize Log Insight agent ID for the appliance. You must provide this agent ID to vRealize Log Insight so that it can pull the appropriate logs from the appliance. For information about how to obtain the agent ID, see Obtain the VMware vRealize Log Insight Agent ID for Enterprise PKS Management Console in *Troubleshooting Enterprise PKS Management Console*.

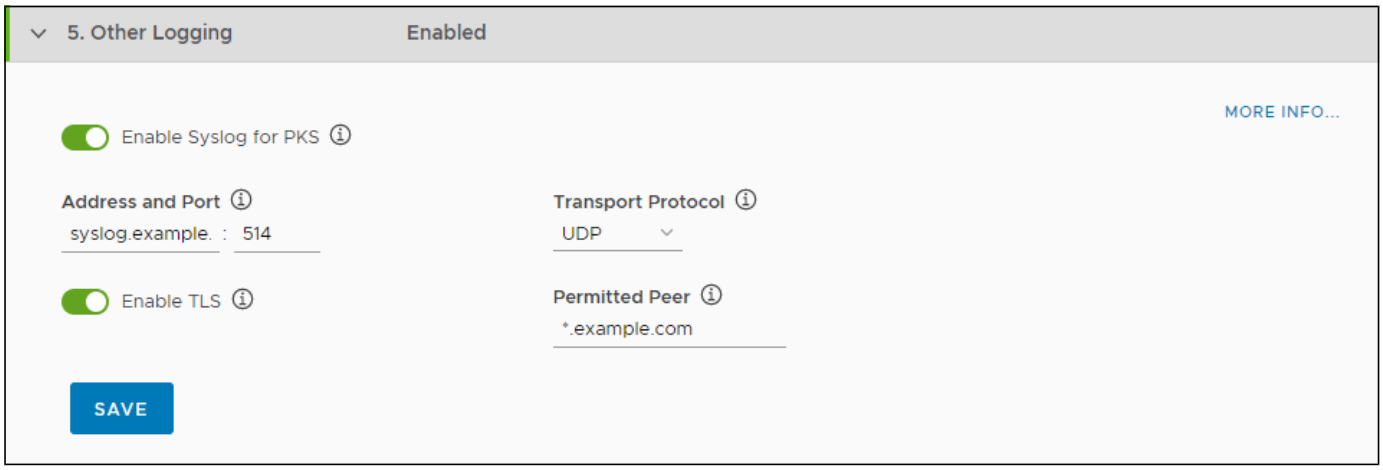
Configure a Connection to Syslog

You can configure your Enterprise PKS deployment so that it sends logs for BOSH-deployed VMs, Kubernetes clusters, and namespaces to an existing Syslog server.

1. Select the **Enable** toggle to enable a connection to Syslog.
2. Enter the address of your Syslog server in the **Address and port** text boxes.
3. Select **TCP**, **UDP**, or **RELP** from the **Transport protocol** drop-down menu.

4. Optionally select **Enable TLS**.

5. Enter a permitted peer ID.



View a larger version of this image.

6. Click **Save**.

7. Click **Next** to install Harbor.

Step 8: Configure Harbor

Harbor is an enterprise-class registry server that you can use to store and distribute container images. Harbor allows you to organize image repositories in projects, and to set up role-based access control to those projects to define which users can access which repositories. Harbor also provides rule-based replication of images between registries, optionally implements Content Trust with Notary and vulnerability scanning of stored images with Clair, and provides detailed logging for project and user auditing.

Harbor provides a Notary server that allows you to implement Content Trust by signing and verifying the images in the registry. When Notary content trust is enabled, users can only push and pull images that have been signed and verified to or from the registry.

Harbor uses Clair to perform vulnerability and security scanning of images in the registry. You can set thresholds that prevent users from running images that exceed those vulnerability thresholds. Once an image is uploaded into the registry, Harbor uses Clair to check the various layers of the image against known vulnerability databases and reports any issues found.

1. Optionally select the **Enable** toggle to deploy Harbor when you deploy Enterprise PKS.

2. In the **Harbor FQDN** text box, enter a name for the Harbor VM, for example `harbor.pks.example.com`.
   This is the address at which you access the Harbor administration UI and registry service.

3. Enter and confirm a password for the Harbor VM.

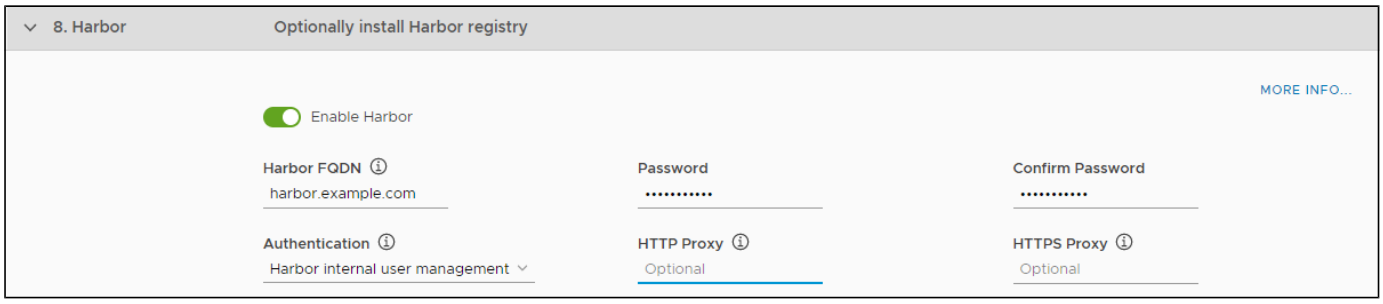
4. Select the method to use for authenticating connections to Harbor.

   - **Harbor internal user management**: Create a local database of users in the Harbor VM.
   - **Log in Harbor with LDAP users**: Use AD or LDAP to manage users. You configure the connection to the LDAP server in Harbor after deployment.
   - **UAA in Pivotal Container Service**: Use the same UAA as you use for Enterprise PKS.

5. If your environment does not allow Harbor components to access the external network on which Enterprise PKS Management Console is running, provide proxy addresses.

- In the **HTTP Proxy** field, enter the proxy server to use when it is not possible for Harbor to connect to an outside address over HTTP. For example, http://your.proxy.com:8080 or https://your.proxy.com:443.
- In the **HTTPS Proxy** field, enter the proxy server to use when it is not possible for Harbor to connect to an outside address over HTTPS. For example, http://your.proxy.com:8080 or https://your.proxy.com:443.

These proxies allow Clair to obtain updates from its vulnerability database.



View a larger version of this image.

6. Optionally select **Manage Certificates Manually for Harbor** to use custom certificates with Harbor.
   To use a custom certificate, paste the contents of the server certificate PEM file in the **SSL Certificate PEM** text box.

```
-----BEGIN CERTIFICATE-----
ssl_certificate_contents
-----END CERTIFICATE-----
```

   Paste the contents of the certificate key in the **SSL Key PEM** text box.

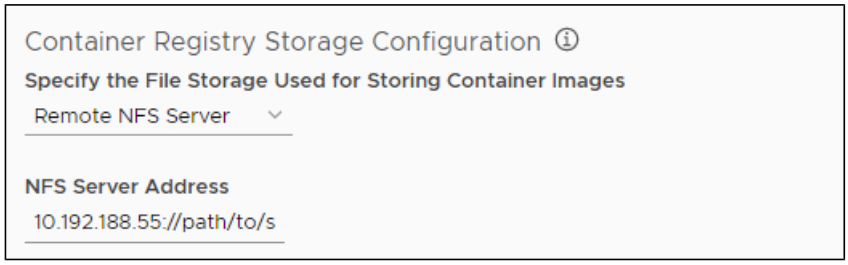
```
-----BEGIN PRIVATE KEY-----
ssl_private_key_contents
-----END PRIVATE KEY-----
```

   Paste the contents of the Certificate Authority (CA) file in the **Certificate Authority** text box.

```
-----BEGIN CERTIFICATE-----
CA_certificate_contents
-----END CERTIFICATE-----
```

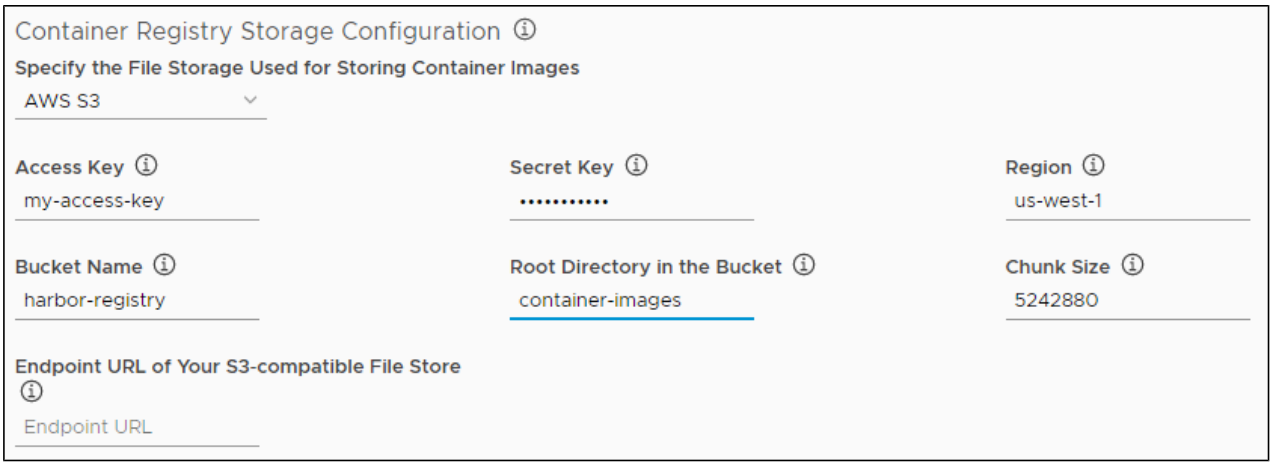
7. Select the location in which to store image repositories.

   - **Local file system**: Stores images in the Harbor VM. No configuration required.
   - **Remote NFS server**: Provide the IP address and path to an NFS share point.



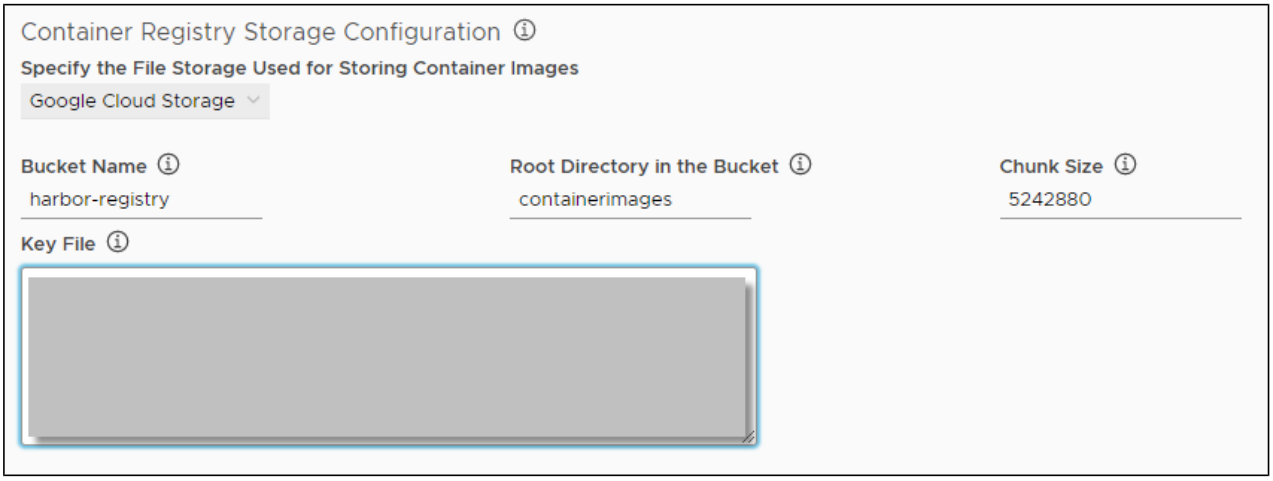
   - **AWS S3**: Provide the connection details for your Amazon S3 account.
     - **Access Key**: Enter your access key ID.
     - **Secret Key**: Enter the secret access key for your access key ID.
     - **Region**: The region in which your bucket is located.

- **Bucket Name**: Enter the name of your S3 bucket.
- **Root Directory in the Bucket** Enter the root directory of the bucket.
- **Chunk Size**: The default is 5242880 but you can change it if necessary.
- **Endpoint URL of your S3-compatible file store** Enter the URL of your S3-compatible filestore.
- **Enable v4auth**: Access to the S3 bucket is authenticated by default. Deselect this checkbox for anonymous access.
- **Secure mode**: Access to your S3 bucket is secure by default. Deselect this checkbox to disable secure mode.



View a larger version of this image.

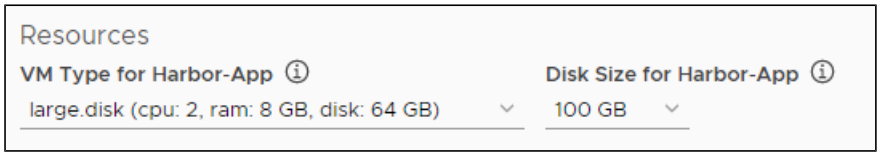
- **Google Cloud Storage**: Provide the connection details for your Google Cloud Storage account.
  - **Bucket Name**: Enter the name of the GCS bucket.
  - **Root Directory in the Bucket** Enter the root directory of the bucket.
  - **Chunk Size**: The default is 5242880 but you can change it if necessary.
  - **Key File**: Enter the service account key for your bucket.



View a larger version of this image.

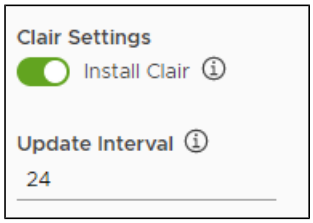
8. Select the configuration for the Harbor VM from the **VM Type for Harbor-App** drop-down menu.

9. Select the size of the disk for the Harbor VM from the **Disk Size for Harbor-App** drop-down menu.



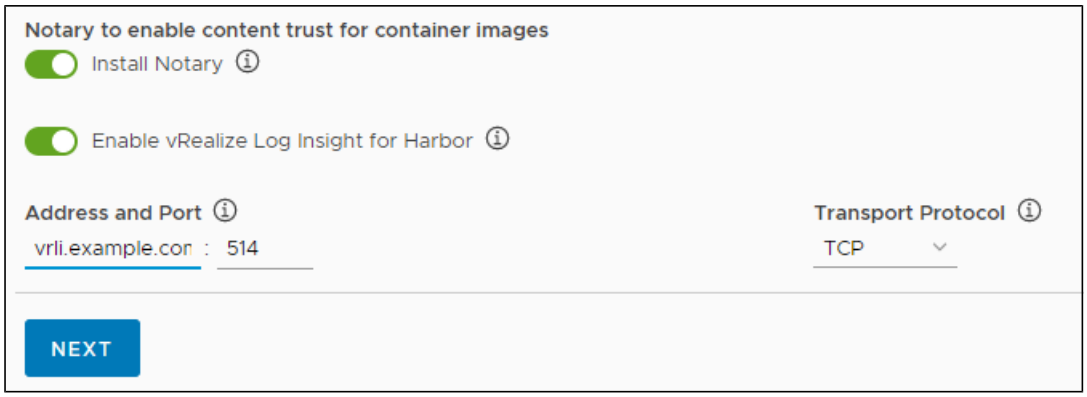
10. Optionally enable Clair by enabling the **Install Clair** toggle.

11. In the **Update Interval** field, specify when Clair will update its CVE databases for the registered sources.
    When the updater interval expires, Clair will update its CVE databases. The default updater interval is 0, which means Clair will never update its CVE databases. If you set the updater interval to 24, Clair updates its CVE databases every 24 hours.



12. Optionally enable Notary by enabling the **Install Notary** toggle.

13. Optionally send Harbor logs to vRealize Log Insight by enabling the **Enable vRealize Log Insight for Harbor**toggle.
    If you enable vRealize Log Insight, provide the address and port of your vRealize Log Insight service, and select either UDP or TCP for the transport protocol.



14. Click **Next** to complete the configuration wizard.

Step 9: Configure CEIP and Telemetry

VMware's Customer Experience Improvement Program (CEIP) and the Pivotal Telemetry Program provide VMware and Pivotal with information 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 programs, VMware and Pivotal collect technical information about your organization's use of Enterprise PKS Management Console.

To configure VMware's Customer Experience Improvement Program (CEIP) and the Pivotal Telemetry Program (Telemetry), do the following:

1. Click **CEIP and Telemetry**.

2. Review the information about the CEIP and Telemetry.

CEIP and Telemetry

About the CEIP and Telemetry Program

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.

Customers who participate (at the enhanced tier) are eligible for several benefits, including Proactive Support.

Additional information regarding the data collected through CEIP or Telemetry, and the purposes for which it is used by VMware is set forth in the Trust & Assurance Center and for Pivotal on the Pivotal Telemetry pages. If you prefer not to participate in CEIP and Telemetry for PKS, you should not join below. You may join or leave CEIP and Telemetry for PKS at any time.

Levels of Participation

No personally identifiable information (PII) is collected at either level of participation. Please refer to the data dictionary for more information on the data we collect.

Standard: This participation tier is anonymous. Your data will be used to improve PKS, but is not identifiable to your organization

Enhanced: This participation tier allows us to identify your organization so we may provide proactive support and other benefits. See sample reports for more details.

Please Note

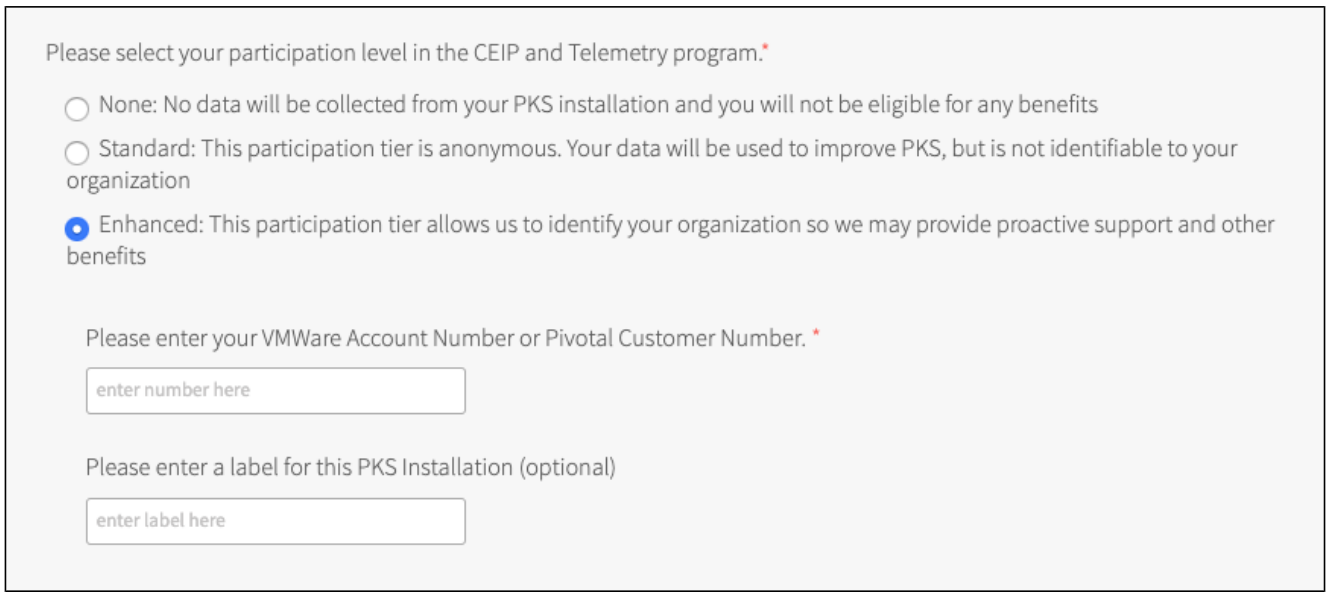
- If you are opting in on behalf of an organization (and not for you as an individual), you represent and warrant that you have legal authority to bind that organization, and you hereby join CEIP/Telemetry on behalf of your organization.
- If you are opting in to the enhanced tier, in the event a term or condition of CEIP or the Telemetry program conflicts with a term or condition of a previously executed license procurement agreement between you and Licensor (Pivotal or VMware), the CEIP or Telemetry program terms supersede solely for purposes of CEIP and Telemetry
- If you are running PKS on a private network, you will need to enable outgoing internet access by opening your firewall to allow traffic to `https://vcsa.vmware.com/ph` on port `443`

Resources

- Data Dictionary
- Participation Benefits
- Sample Reports
- Trust and Assurance Center
- Pivotal Telemetry

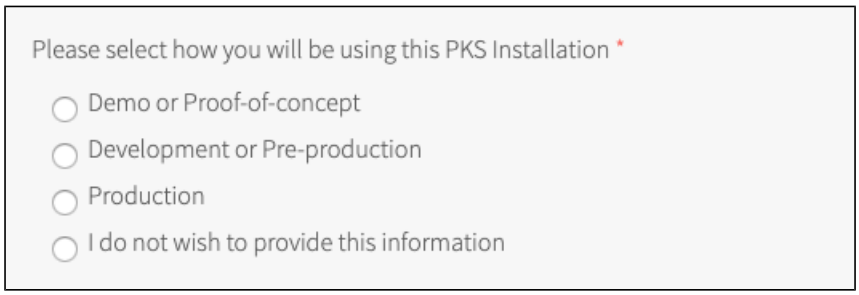
View a larger version of this image.

3. To specify your level of participation in the CEIP and Telemetry program, select one of the **Participation Level** options:
   - **None:** If you select this option, data is not collected from your Enterprise PKS installation.
   - (Default) **Standard:** If you select this option, data is collected from your Enterprise PKS installation to improve Enterprise PKS. This participation level is anonymous and does not permit the CEIP and Telemetry to identify your organization.
   - **Enhanced:** If you select this option, data is collected from your Enterprise PKS installation to provide you proactive support and other benefits. This participation level permits the CEIP and Telemetry to identify your organization.



For more information about the CEIP and Telemetry participation levels, see Participation Levels in *Telemetry*.

4. If you selected the **Enhanced** participation level, complete the following:
   - Enter your VMware account number or Pivotal customer number in the **VMware Account Number or Pivotal Customer Number** field. If you are a VMware customer, you can find your VMware Account Number in your **Account Summary** on my.vmware.com. If you are a Pivotal customer, you can find your Pivotal Customer Number in your Pivotal Order Confirmation email.
   - (Optional) Enter a descriptive name for your PKS installation in the **PKS Installation Label** field. The label you assign to this installation will be used in telemetry reports to identify the environment.
5. To provide information about the purpose for this installation, select an option in the **PKS Installation Type** list.



6. Click **Save**.

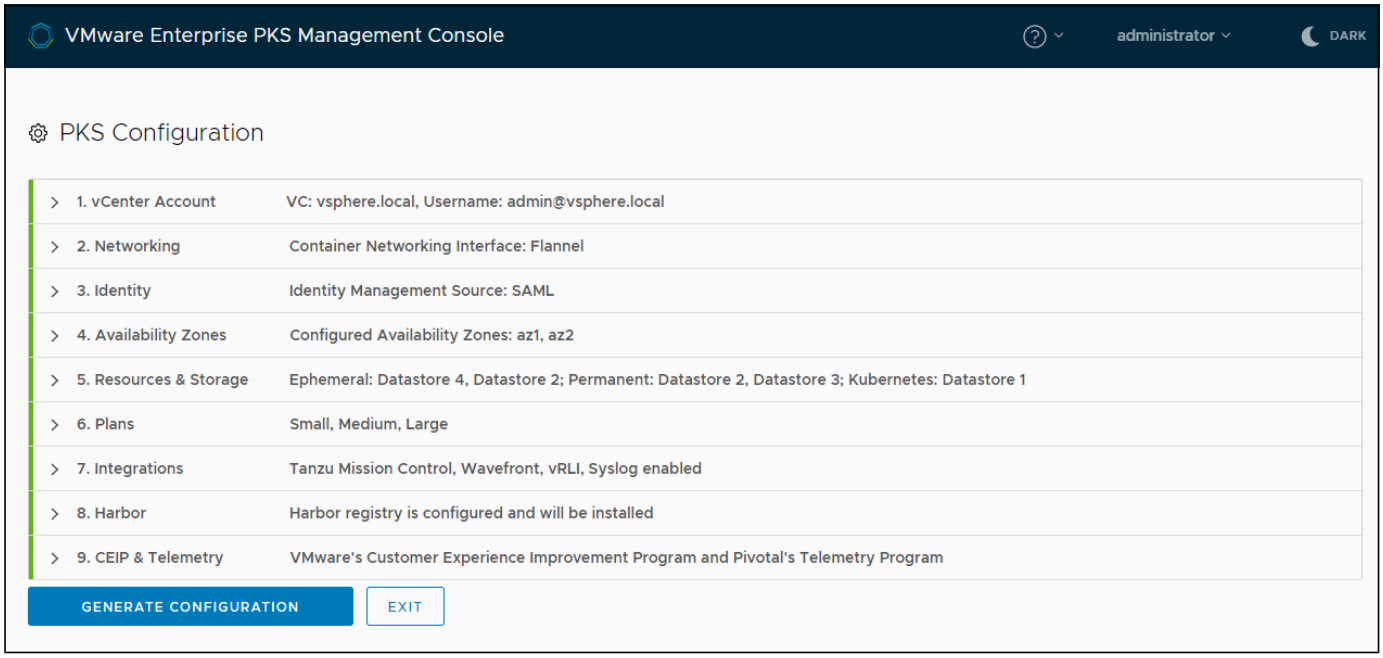
Note: If you join the CEIP and Telemetry Program for Enterprise PKS, open your firewall to allow outgoing access to `https://vcsa.vmware.com/ph` on port `443`.

Note: Even if you select **None**, Enterprise PKS-provisioned clusters send usage data to the PKS control plane. However, this data is not sent to VMware or Pivotal and remains on your Enterprise PKS installation.

Step 10: Generate Configuration File and Deploy Enterprise PKS

When all of the sections of the wizard are green, you can generate a YAML configuration file and deploy Enterprise PKS.

1. Click **Generate Configuration** to see the generated YAML file.

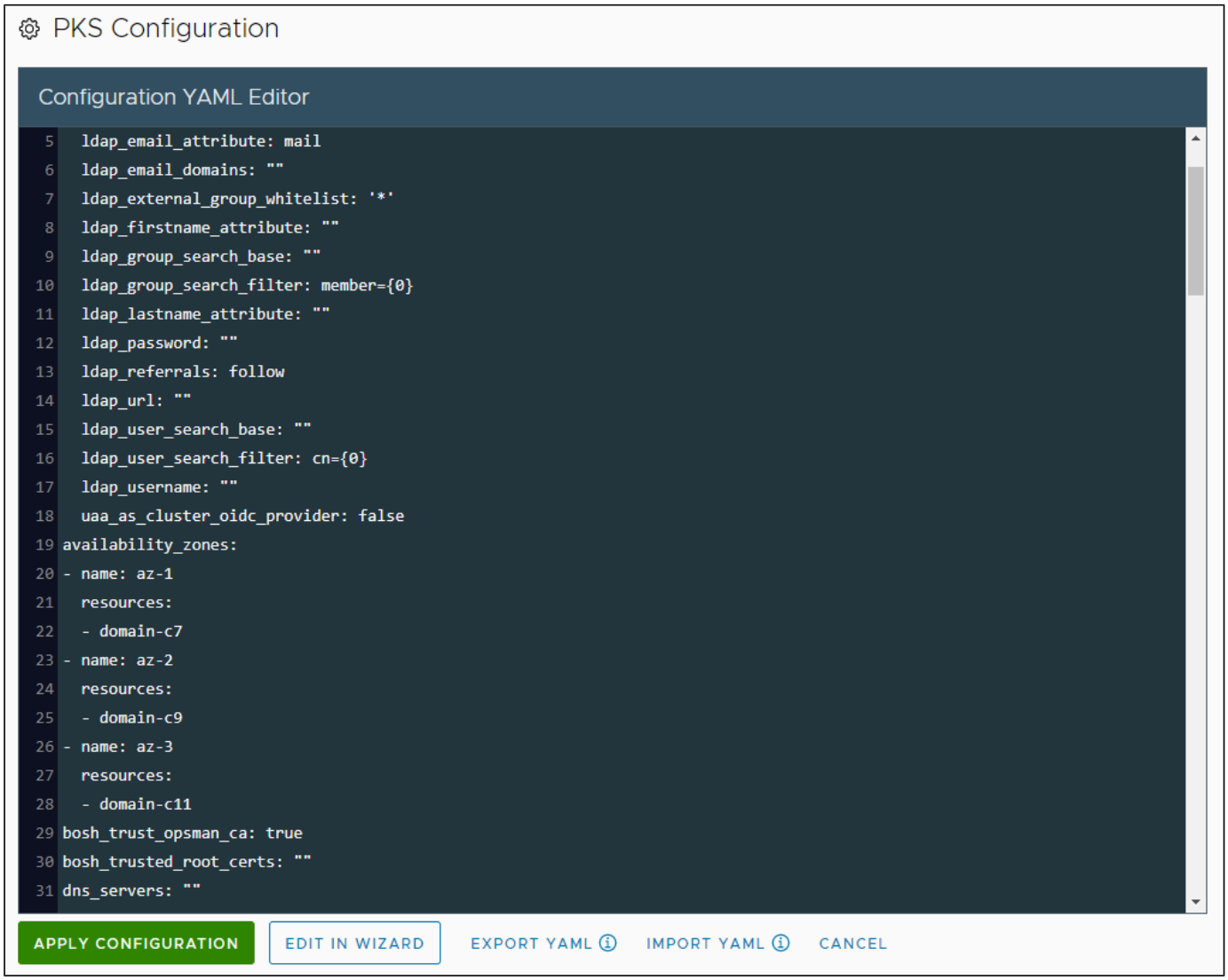


View a larger version of this image

2. Optionally click **Export YAML** to save a copy of the YAML file for future use.
   This is recommended. The manifest is exported as the file `PksConfiguration.yaml`.

3. Optionally edit the YAML directly in the YAML editor.

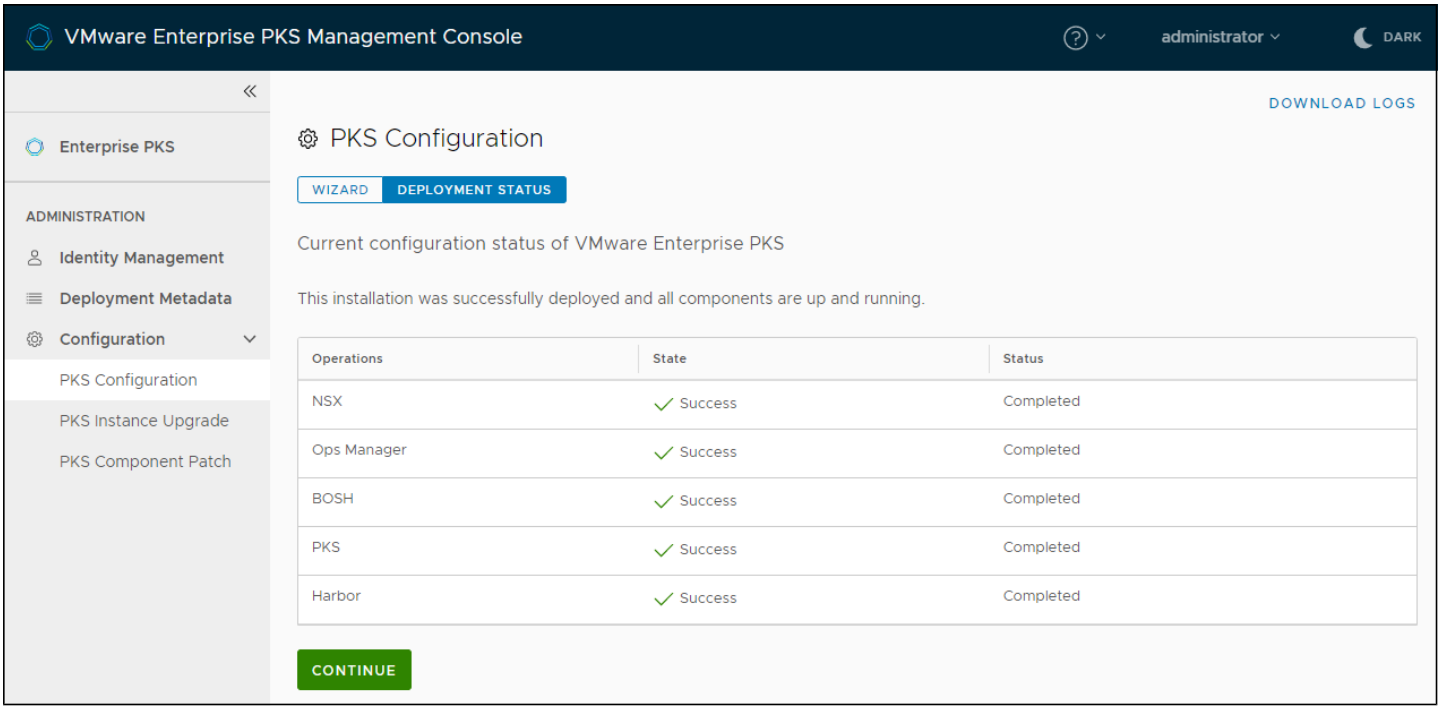
4. Click **Apply Configuration** then **Continue** to deploy Enterprise PKS.



View a larger version of this image.

5. On the PKS Configuration page, follow the progress of the deployment.

6. When the deployment has completed successfully, click **Continue** to monitor and manage your deployment.



View a larger version of this image.

Next Steps

For information about how you can use Enterprise PKS Management Console to monitor and manage your deployment, see Monitor and Manage Enterprise PKS in the Management Console.

You can now access the Enterprise PKS control plane and begin deploying Kubernetes clusters.

> **Important**: If you deployed Enterprise PKS with plans that use Windows worker nodes, see Enable Plans with Windows Worker Nodes for information about how to install a Windows Server stemcell and other necessary configuration actions that you must perform. Plans that use Linux worker nodes are available immediately, but plans that use Windows worker nodes are ignored until you install the Windows Server stemcell.

If Enterprise PKS fails to deploy, see Troubleshooting.

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

Deploy Enterprise PKS by Importing a YAML Configuration File

In this topic

Page last updated:

If you have an existing YAML configuration file from a previous deployment of VMware Enterprise PKS, you can use the VMware Enterprise PKS Management Console to deploy a new Enterprise PKS instance from that file.

> **Notes**: You can only use the management console to deploy Enterprise PKS from a YAML file if that YAML file was generated by Enterprise PKS Management Console version 1.0 or later. YAML files from beta releases of the management console are not compatible with this release.
>
> You can import a YAML from an earlier supported version of Enterprise PKS Management Console. In this case, after you import the YAML, open the configuration in the wizard and configure any missing settings that are new in this version.

For information about how to deploy Enterprise PKS by using the configuration wizard, see Deploy Enterprise PKS by Using the Configuration Wizard.

For information about how to upgrade an existing deployment, see Upgrade Enterprise PKS Management Console.

YAML Files and Passwords

When Enterprise PKS Management Console generates the content of the YAML file for the YAML editor, it masks the passwords for NSX Manager, vCenter Server, and Harbor so that they do not appear in plain text. In the generated YAML files, the password fields look like the following example:

```
admin_password: <hidden:f065be51-84e9-4ca7-972d-ed46f7273123>
```

The `<hidden>` tag includes a GUID that refers to a database entry for the password that was entered into the configuration wizard. If you import a YAML file from an instance of Enterprise PKS Management Console that is deployed in a different vSphere environment, the GUID provided in the hidden tag will not correspond to an entry in the database of the environment in which you are importing the YAML. As a consequence, if you import a YAML from a different vSphere environment, you must manually update the passwords for NSX Manager, vCenter Server, and Harbor in the YAML editor. If you are importing a YAML file from the same environment, the correct passwords are held in the database and no action is required.

Prerequisites

- Deploy the Enterprise PKS Management Console Appliance to vCenter Server.
- The vCenter Server instance must be correctly configured for Enterprise PKS deployment. For information about the vCenter Server requirements, see Virtual Infrastructure Prerequisites.
- Depending on the type of networking to use, your infrastructure must meet the appropriate prerequisites. For information about

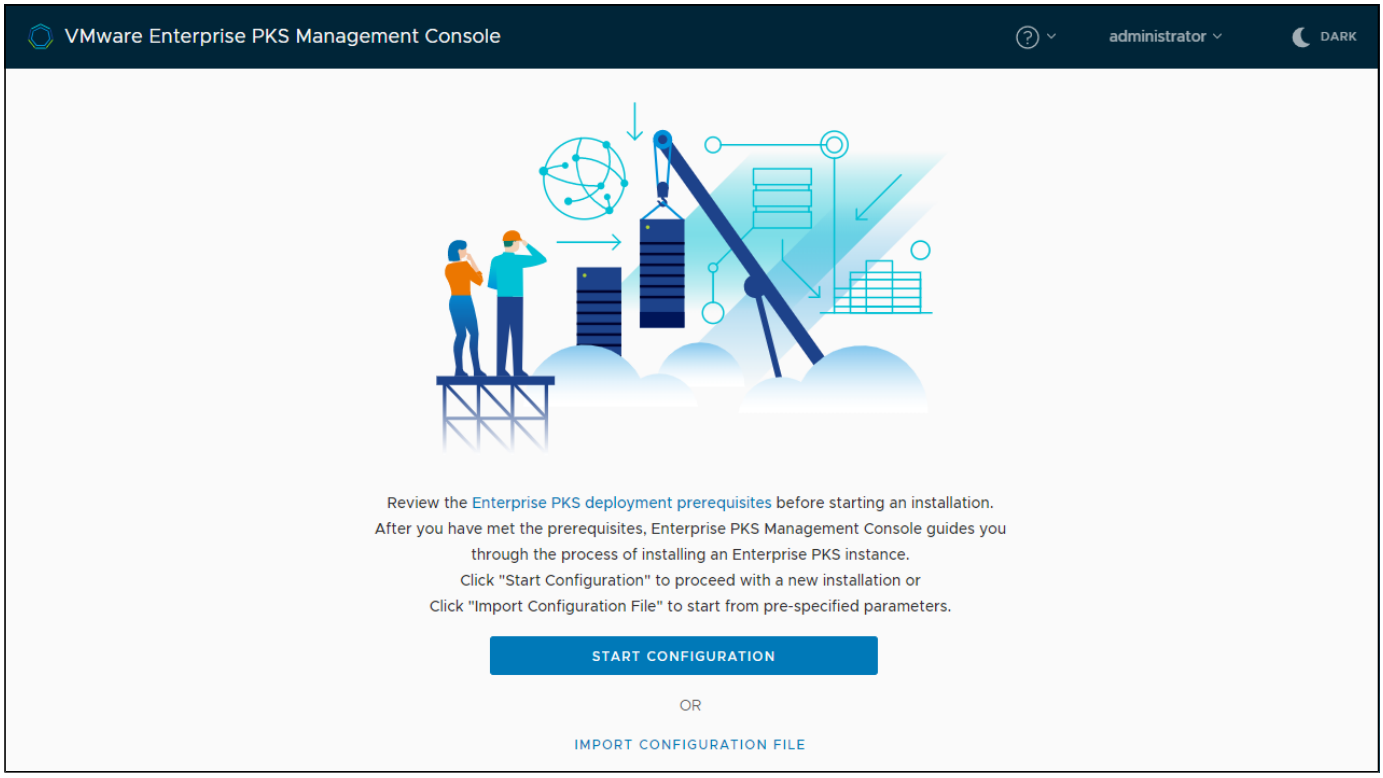
networking prerequisites, see the following topics:

- Prerequisites for a Bring Your Own Topology Deployment to NSX-T Data Center
- Prerequisites for an Automated NAT Deployment to NSX-T Data Center
- Prerequisites for a Flannel Network

- Log in to Enterprise PKS Management Console.
- You have an existing YAML configuration file that you exported during a previous Enterprise PKS deployment from Enterprise PKS Management Console.
- For information about how to set the networking parameters in the YAML file, see Networking Options in the YAML File below.

Import a YAML Configuration File

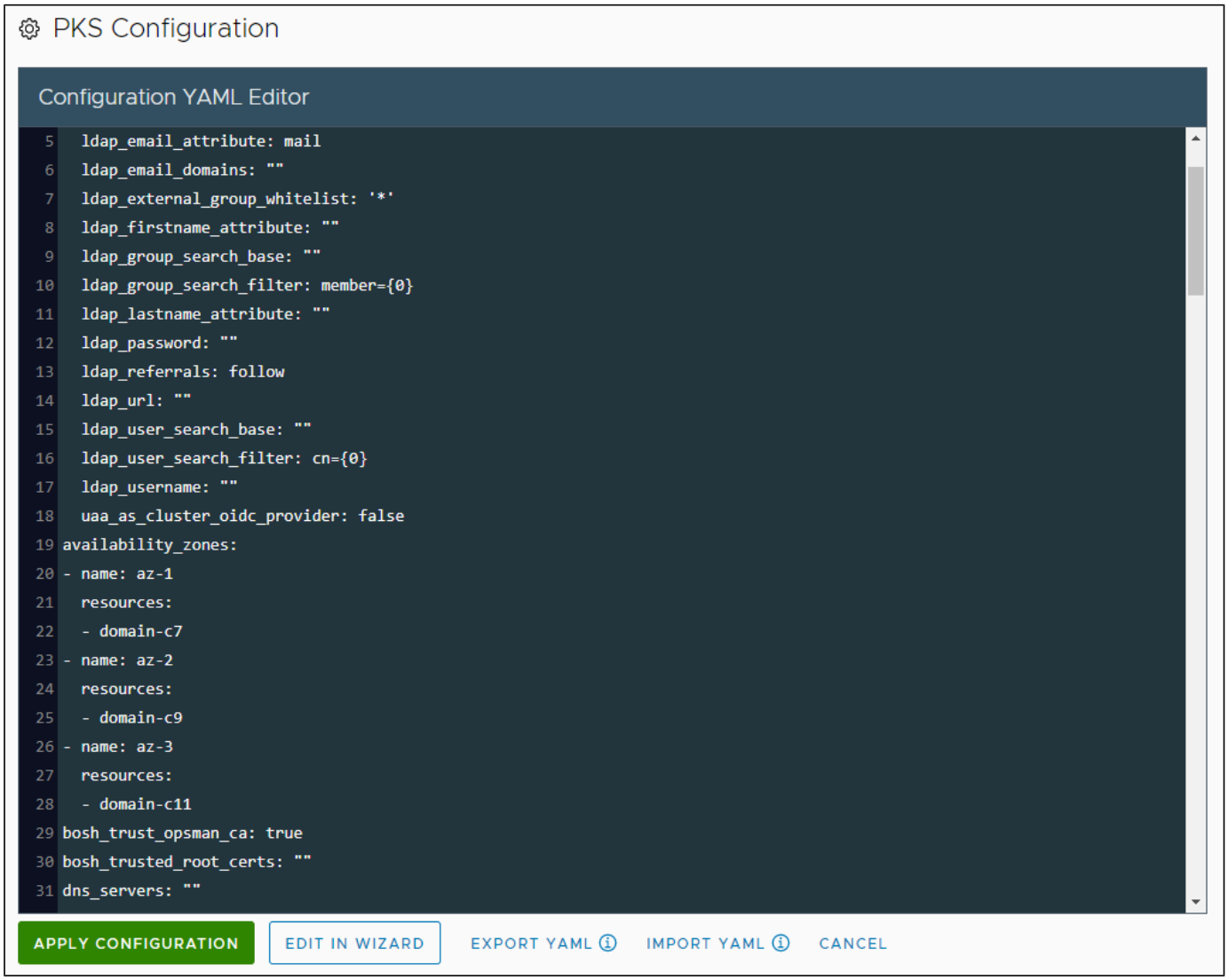
1. On the VMware Enterprise PKS landing page, click **Install** then **Import Configuration File**.



View a larger version of this image

2. Drag the YAML file into the Import Configuration File window, or click **Browse** to navigate to it.

3. In the Configuration File editor, modify the contents of the YAML file appropriately for the new instance of Enterprise PKS that you want to deploy.



View a larger version of this image

If the YAML was generated by an instance of management console that is running in a different vSphere environment, update the passwords for NSX Manager, vCenter Server, and Harbor.

You can click the **Edit in Wizard** button, to open the imported configuration in the wizard and modify it there. For example, if you have imported a YAML that was generated by a previous version of Enterprise PKS Management Console, open it in the wizard so that you can configure any options that are new in this version.

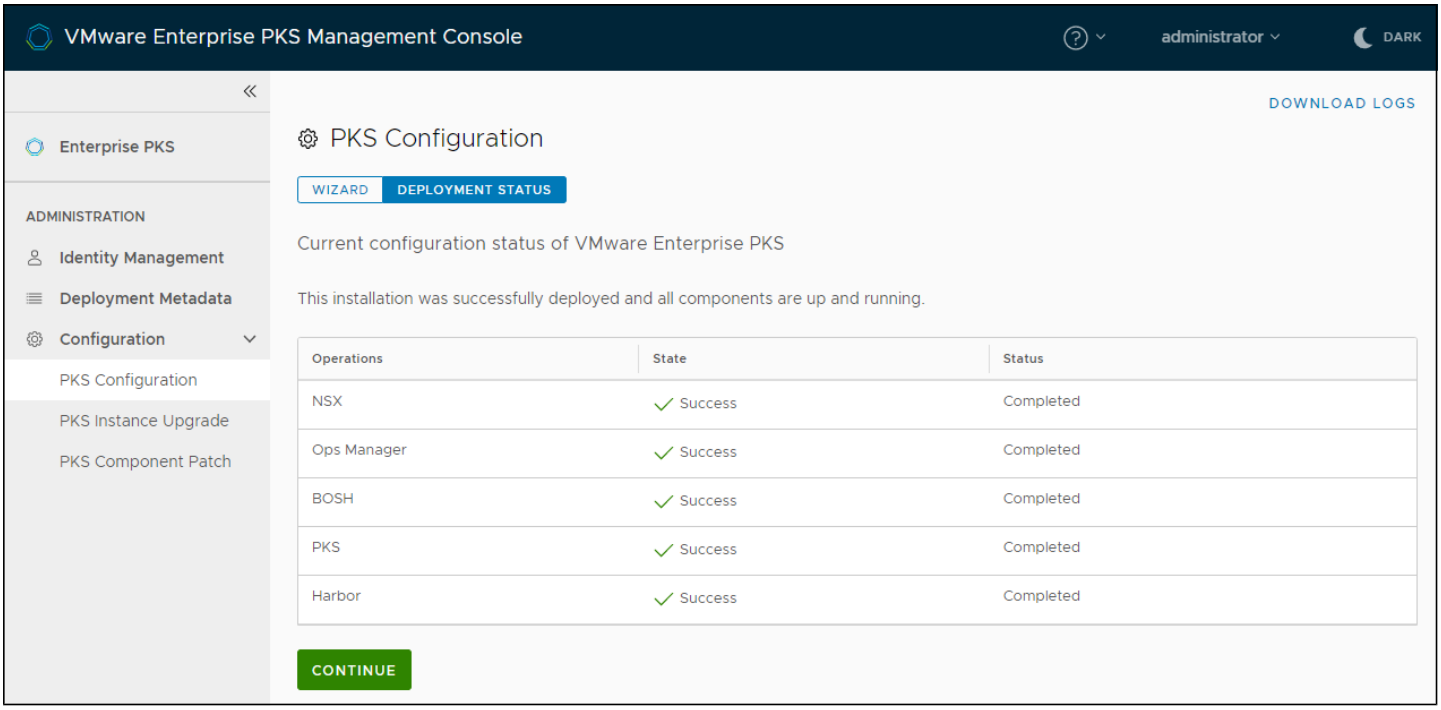
To abandon this YAML and start again, click **Import YAML** to upload the YAML again or to import a new one.

4. When you have finished editing the YAML in the Configuration File editor, click **Export YAML** to save a copy of your updated YAML configuration.

5. Click **Apply Configuration** and **Continue** to deploy Enterprise PKS from this configuration file.

6. On the PKS Configuration page, follow the progress of the deployment.

7. When the deployment has completed successfully, click **Continue** to monitor and manage your deployment.



View a larger version of this image

Networking Options in the YAML File

The networking parameters for the three types of Enterprise PKS networking are all included in the `network:` section of the YAML file. When you edit the YAML file, you only need to set those parameters that apply to your type of networking.

The following table lists the parameters to set for each type of networking.

Unprepared NSX-T Data Center	Prepared NSX-T Data Center	Flannel

use_nsx autoprovision_nsx external_portgroup_gateway external_portgroup_netmask external_portgroup_subnet external_vlan_id active_t0_edge_node active_t0_edge_node_ip standby_t0_edge_node standby_t0_edge_node_ip t0_edge_node_lb_ip t0_ha_mode_active_active dep_network_cidr dep_dns nsx_pod_cidr nsx_node_cidr nsx_dns floating_ips_range nsx_manual_ssl_certs nsx_ca_crt nsx_verify_ssl_certs nsx_host nsx_password nsx_username ntp_servers	use_nsx autoprovision_nsx dep_network_name nsx_pod_ip_block_id nsx_node_ip_block_id nsx_t0_id nsx_fip_id dep_dns dep_reserved_ip_range_from dep_reserved_ip_range_to nsx_dns nsx_manual_ssl_certs nsx_ca_crt nsx_verify_ssl_certs nsx_nat_mode nsx_host nsx_password nsx_username ntp_servers	use_nsx dep_network_name dep_network_cidr dep_network_gateway dep_dns dep_reserved_ip_range_from dep_reserved_ip_range_to svc_network_name svc_network_cidr svc_network_gateway svc_dns svc_reserved_ip_range_from svc_reserved_ip_range_to flannel_pod_network_cidr flannel_service_network_cidr ntp_servers

Next Step

For information about how you can use Enterprise PKS Management Console to monitor and manage your deployment, see Monitor and Manage Enterprise PKS in the Management Console.

If Enterprise PKS fails to deploy, see Troubleshooting.

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

Enable Plans with Windows Worker Nodes

In this topic

Page last updated:

Page last updated:

If you used Enterprise PKS Management Console to deploy Enterprise PKS with plans that implement Windows worker nodes, you must use Operations Manager to provide BOSH with a vSphere stemcell 2019.7 or later for Windows Server version 2019.

Enterprise PKS Management Console does not provide a mechanism for the automatic upload and installation of the Windows stemcell. Because Operations Manager and BOSH Director for vSphere are deployed when you deploy Enterprise PKS from Enterprise PKS Management Console, you can only install the stemcell after you have deployed Enterprise PKS.

After you deploy Enterprise PKS from the management console, any plans that use Windows worker nodes are ignored until you install a Windows Stemcell and configure the management console to use it.

> **important:** Support for Windows-based Kubernetes clusters is in beta and supports only vSphere with Flannel. You cannot use Windows worker nodes with NSX-T Data Center networking.

Prerequisites

- You used Enterprise PKS Management Console to deploy Enterprise PKS with Flannel networking. For information about deploying Enterprise PKS from the management console, see Deploy Enterprise PKS by Using the Configuration Wizard. For information about how to configure Flannel networking, see Prerequisites for a Flannel Network.
- During Enterprise PKS deployment, you configured a plan that implements Windows worker nodes. For information about creating a plan with Windows worker nodes, see Configure Plans in *Deploy Enterprise PKS by Using the Configuration Wizard*.
- If you did not create a plan that uses Windows worker nodes when you deployed Enterprise PKS, or if you have upgraded Enterprise PKS Management Console from a version that did not support Windows worker nodes, you can use Enterprise PKS Management Console to reconfigure the plans of your existing deployment. For information about reconfiguring Enterprise PKS in the management console, see Reconfigure Your Enterprise PKS Deployment.

Step 1: Create a Windows Server Stemcell

vSphere stemcells for Windows Server version 2019 are not available on the Pivotal Network. You must create Windows Server stemcells for vSphere by using Stembuild and your own Windows Server ISO.

Create a Windows stemcell by following the instructions in Creating a Windows Stemcell for vSphere Using stembuild.

Step 2: Install the Windows Server Stemcell in Operations Manager

After you have created your stemcell, you must upload it to Operations Manager and install it in BOSH. You can use either the Operations Manager interface or the Operations Manager CLI. These instructions describe how to upload and install the stemcell by using the Operations Manager interface.

1. Log in to Operations Manager. For information about how to log in to Operations Manager and the credentials to use, see Log In to the Operations Manager UI.
2. Select **Stemcell Library**.
3. Click **Import Stemcell** and navigate to the location on your local machine where you saved the stemcell `.tgz` file in Step 1 above.
4. Click **Save**.

Step 3: Reconfigure Your Enterprise PKS Deployment to Use the Stemcell

After you have created a Windows stemcell and used Operations Manager to install it in BOSH Director for vSphere, you must redeploy your Enterprise PKS with the Windows stemcell installed.

1. In Enterprise PKS Management Console, expand **Configuration** and select **PKS Configuration**.
2. If you did not already create a plan that uses Windows worker nodes, or if you have upgraded this Enterprise PKS instance from a version that did not support Windows worker nodes, expand the **Plans** section and create or reconfigure a plan that uses Windows worker nodes.
3. If necessary, reconfigure any other options that show a red status bar or that you want to change.
4. Click **Generate Configuration** to view the generated YAML file.
5. Click **Apply Configuration** to redeploy this Enterprise PKS instance.
6. When the deployment finishes, go to the **Enterprise PKS** view and verify that the Windows Stemcell Status is **INSTALLED**.

You can now deploy Kubernetes clusters in plans that implement Windows worker nodes.

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

Monitor and Manage Enterprise PKS in the Management Console

In this topic

Page last updated:

After you have deployed VMware Enterprise PKS, you can use VMware Enterprise PKS Management Console to perform the following operations:

- View the status of and general information about the components of your deployment, its clusters, and its nodes.
- View important metadata for the components of your deployment.
- Add and remove users and user groups, and assign roles to users and groups. For information about user management, see Identity Management in the Management Console.
- Edit the configuration of your deployment, either in the configuration wizard or by editing the YAML file. For information about reconfiguring your deployment, see Reconfigure Your Enterprise PKS Deployment.
- Upgrade your deployment to a new version. For information about upgrading deployments, see Upgrade Enterprise PKS Management Console.
- Patch the individual components of your deployment. For information about patching components, see Patch Enterprise PKS Management Console Components.
- Delete your deployment.

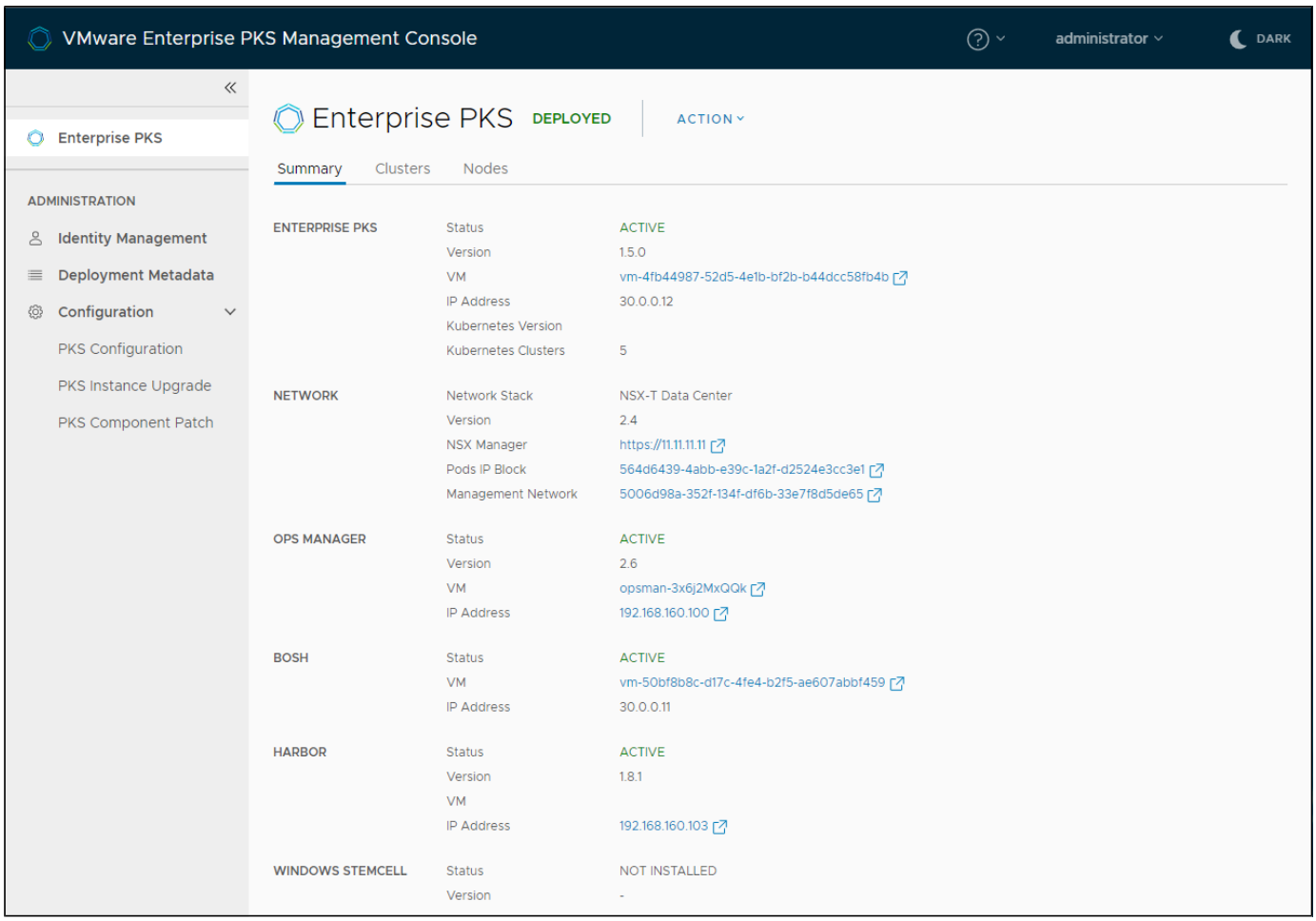
View Status, Clusters, and Nodes

You can find general information about your deployment, and information about all of the clusters and nodes running it, in the Enterprise PKS view of Enterprise PKS Management Console.

Obtain General Status Information

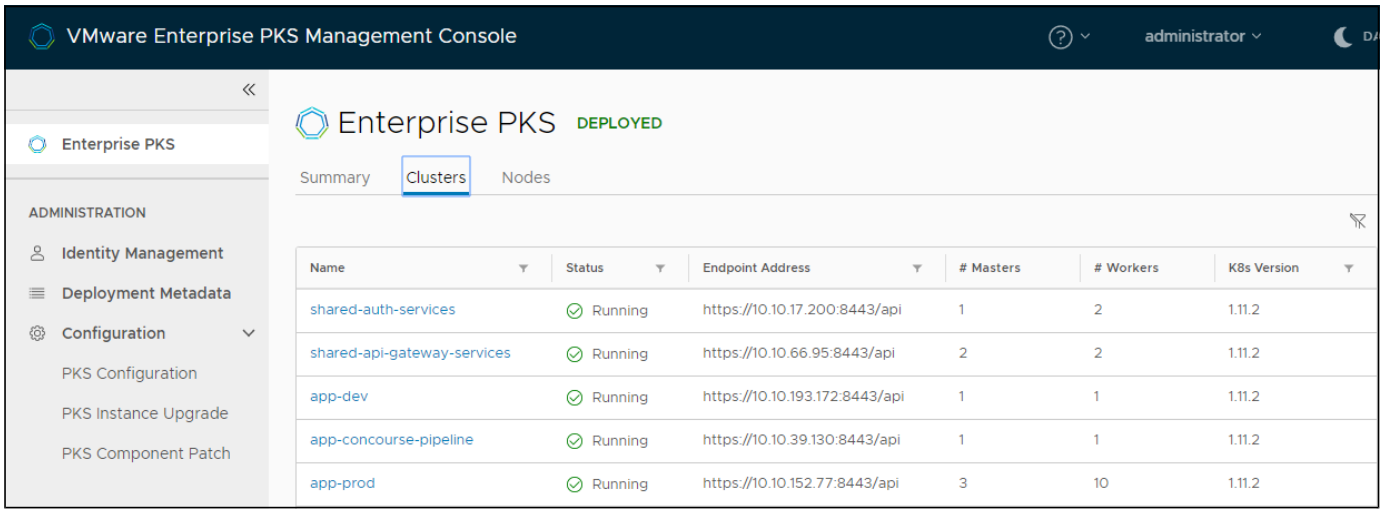
1. Go to the **Enterprise PKS** view of the management console.

2. Select the **Summary** tab for your Enterprise PKS instance.
   You see general information about your deployment, including the status and version of each component, as well as the names and addresses of the VMs that run those services.



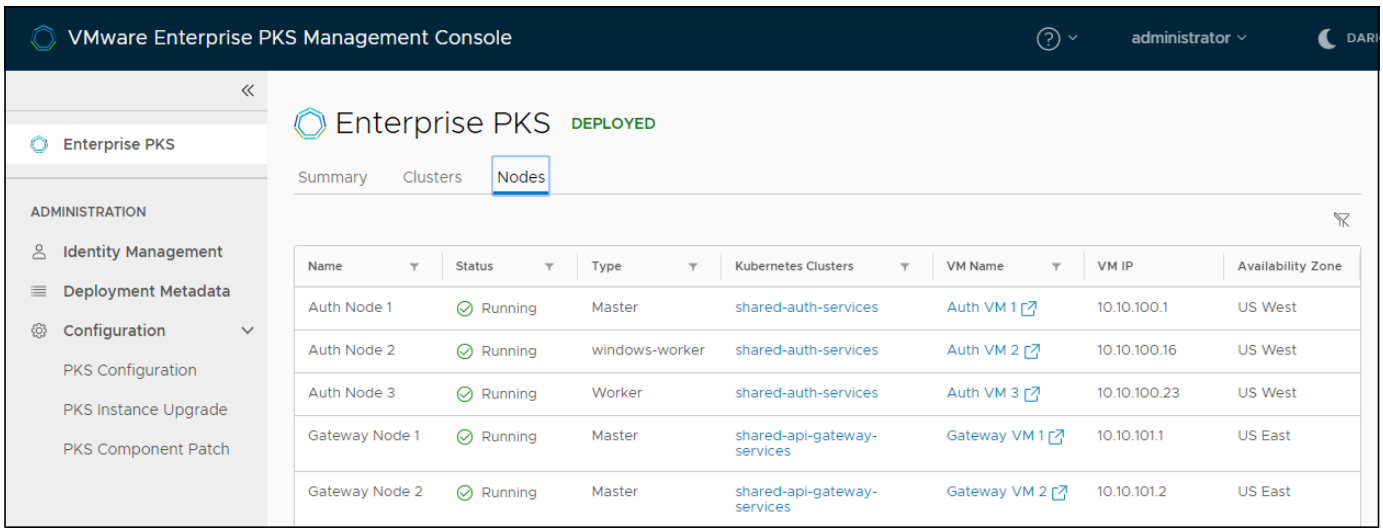
View a larger version of this image

3. Select the **Clusters** tab to see detailed information about all of the clusters running in this instance.



View a larger version of this image

4. Select the **Nodes** tab to see detailed information about all of the nodes running in this instance.



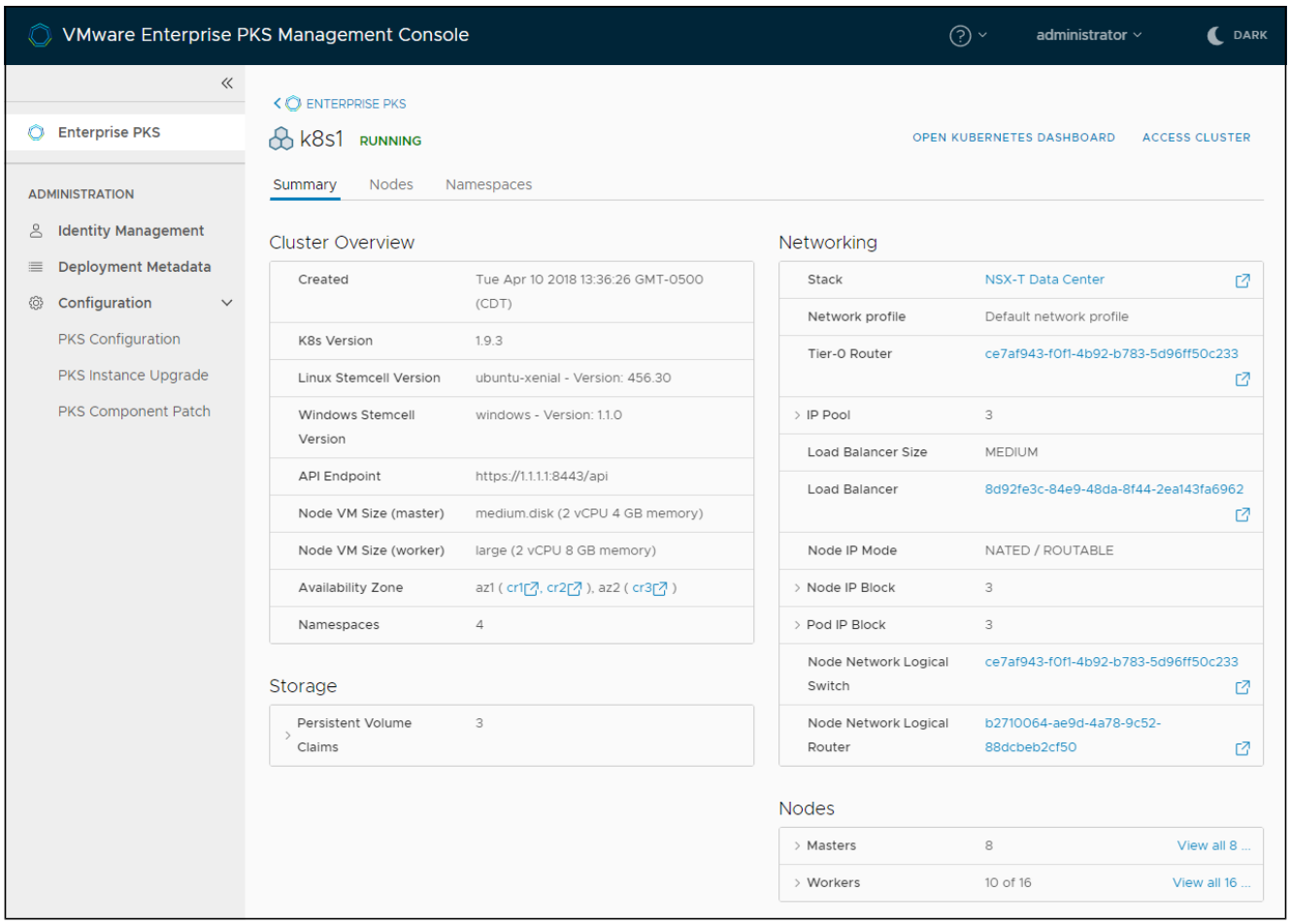
View a larger version of this image

Obtain Cluster Information

1. Go to the Enterprise PKS view of the management console.

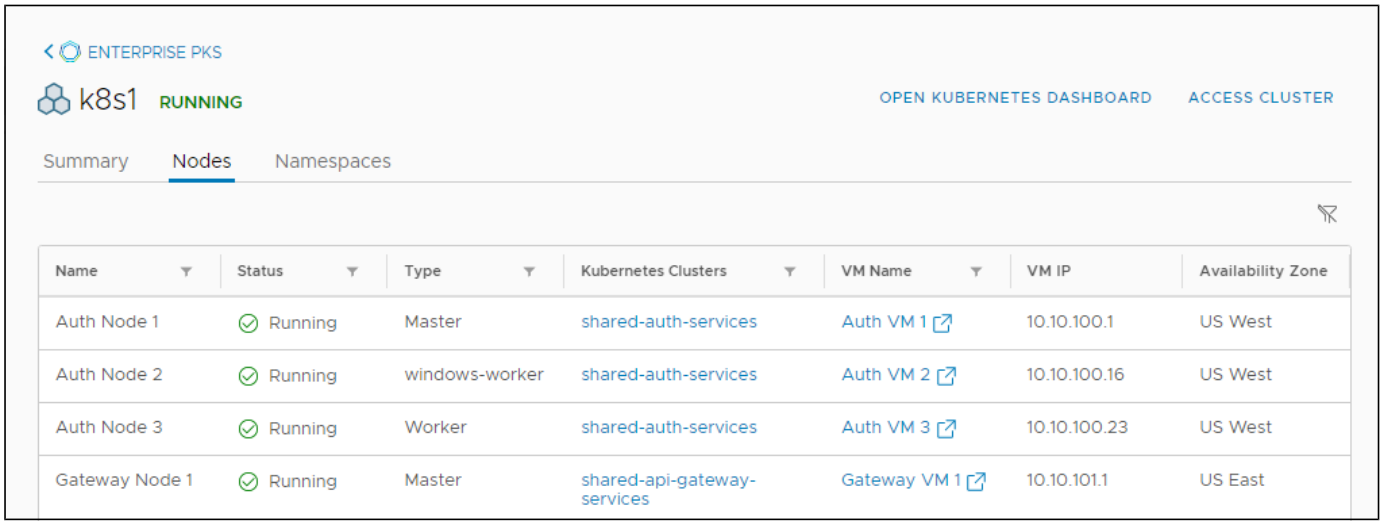
2. Select the **Clusters** tab for your Enterprise PKS instance and select a cluster.
   On the **Summary** tab for the cluster, you see general information about that cluster, as well as networking, and the nodes in that cluster.

   - In the Cluster Overview panel, select the Availability Zone links to be taken the vSphere cluster, host group, or resource pool that contains the cluster.
   - In the Networking panel, select the links to be taken to each of the different components that comprise the network stack for the cluster.
   - In the Nodes panel, expand **Masters** and **Workers** and select the VM links to go to those VMs in the vSphere inventory.



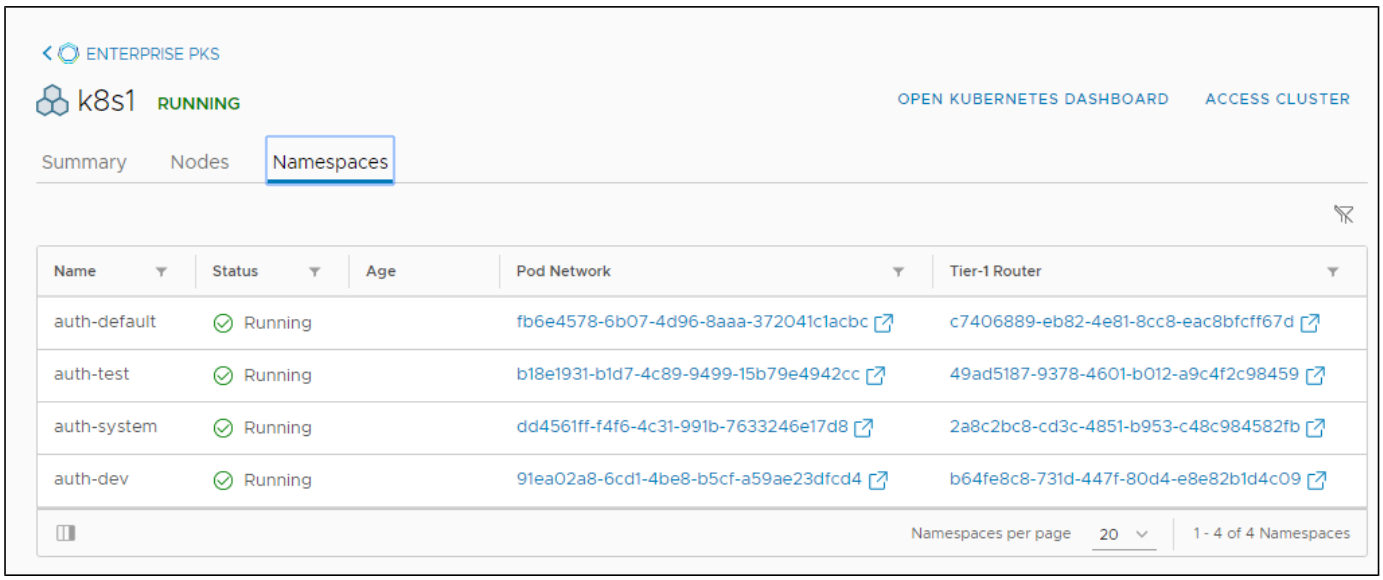
View a larger version of this image

3. Select the **Nodes** tab to see details of all of the nodes that are running in that cluster.



View a larger version of this image

4. Select the **Namespaces** tab to see the status and networking details of all of the namespaces that are running in that cluster.



View a larger version of this image

Connect to Clusters in Kubernetes Dashboard

1. Go to the Enterprise PKS view of the management console.
2. Select the **Clusters** tab for your Enterprise PKS instance.
3. Select a cluster.
4. Select **Open Kubenetes Dashboard** for instructions about how to access the cluster by using Kubernetes Dashboard.

View a larger version of this image

Connect to Clusters with kubectl

1. Go to the Enterprise PKS view of the management console.
2. Select the **Clusters** tab for your Enterprise PKS instance.
3. Select a cluster.
4. Select **Access Cluster** for instructions about how to access the cluster by using `kubectl`.

View a larger version of this image

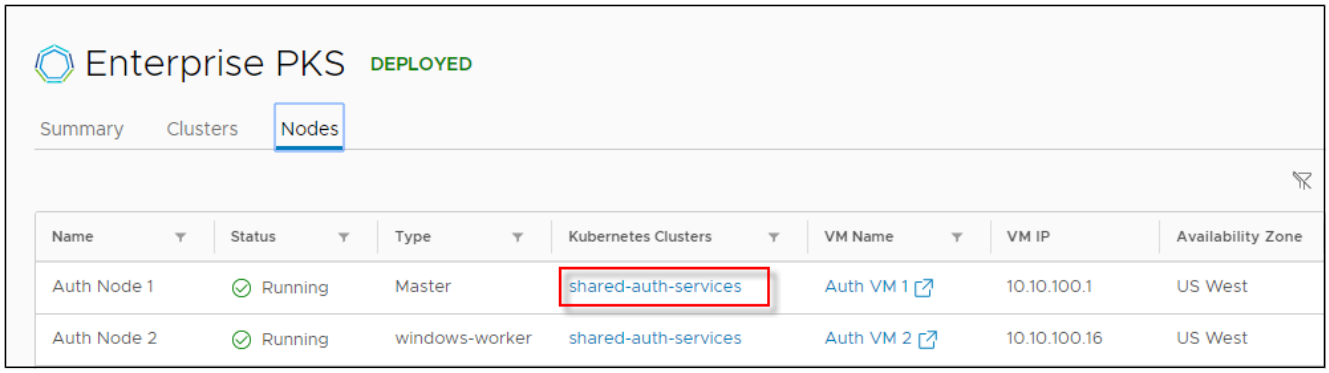
5. Click **Email** to send the instructions to users who need to use `kubectl` to connect to this cluster.

Obtain Node Information

1. Go to the Enterprise PKS view of the management console.

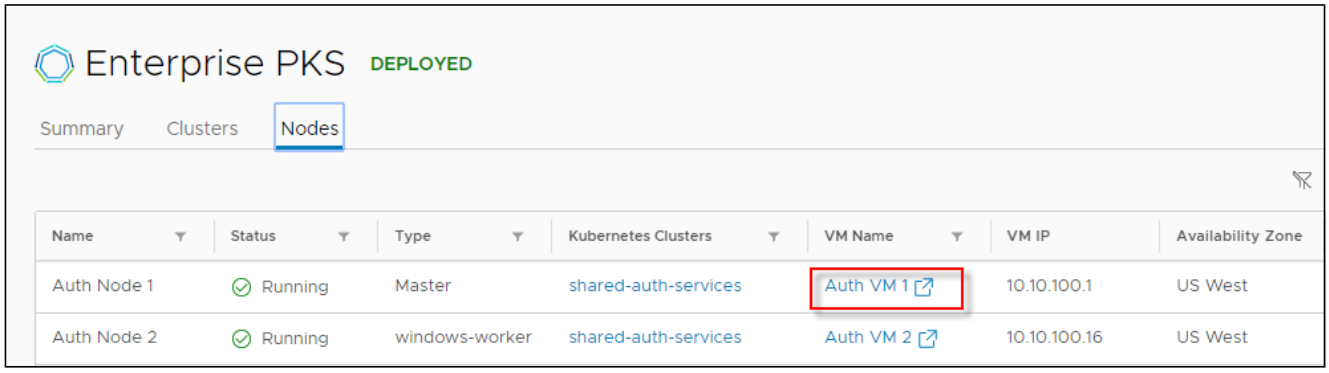
2. Select the **Nodes** tab for your Enterprise PKS instance.
   This tab shows the general status, type, name, IP, and availability zone for all of the nodes that are running in your Enterprise PKS instance.

   - Click the links in the Kubernetes Clusters columns to go the **Summary** tab of that cluster.



View a larger version of this image

   - Click the links in the VM Name column to be taken to the node VMs in the vSphere inventory.

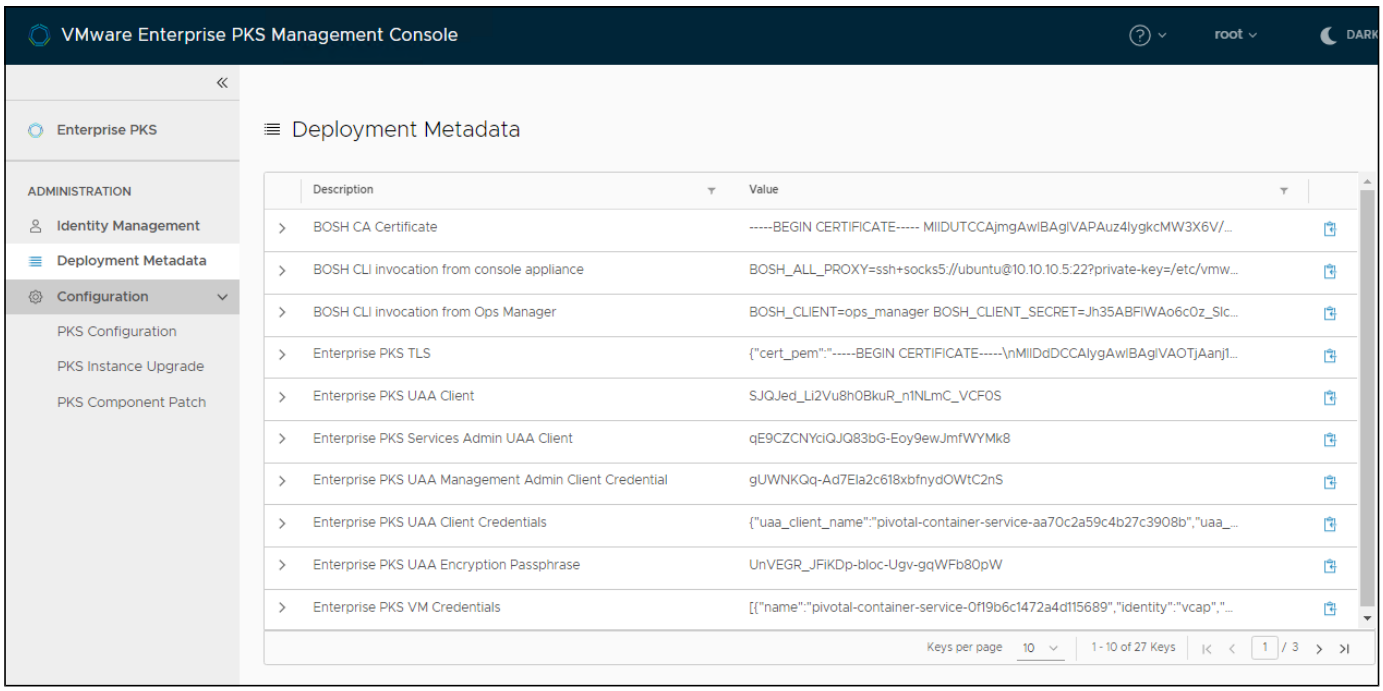


View a larger version of this image

Obtain Deployment Metadata

The deployment metadata provides credentials, certificates, and other metadata about your Enterprise PKS deployment.

1. Select **Deployment Metadata**.



View a larger version of this image

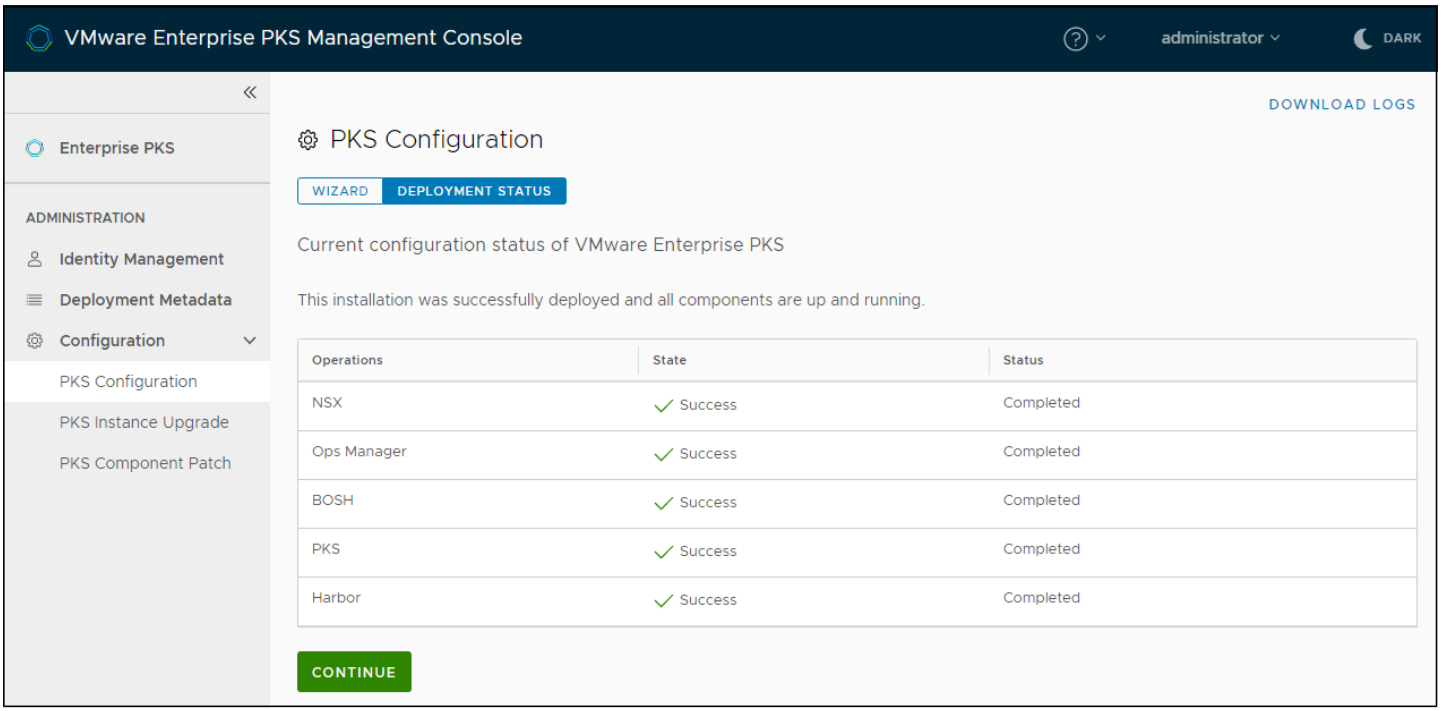
2. Select the clipboard icon at the end of each row to copy the relevant value.

View Component Deployment Status

You can see the status of the individual components of your deployment.

1. Expand **Configuration** and go to the **PKS Configuration** view of the management console.

2. Select **Deployment Status** to see the status of the components.



View a larger version of this image

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

Identity Management in the Management Console

In this topic

Page last updated:

You can add individual users or user groups to Enterprise PKS Management Console. You can assign the Cluster Manager and PKS Administrator roles to individual users or to groups. If you assign a role to a group, all of the users in that group have that role.

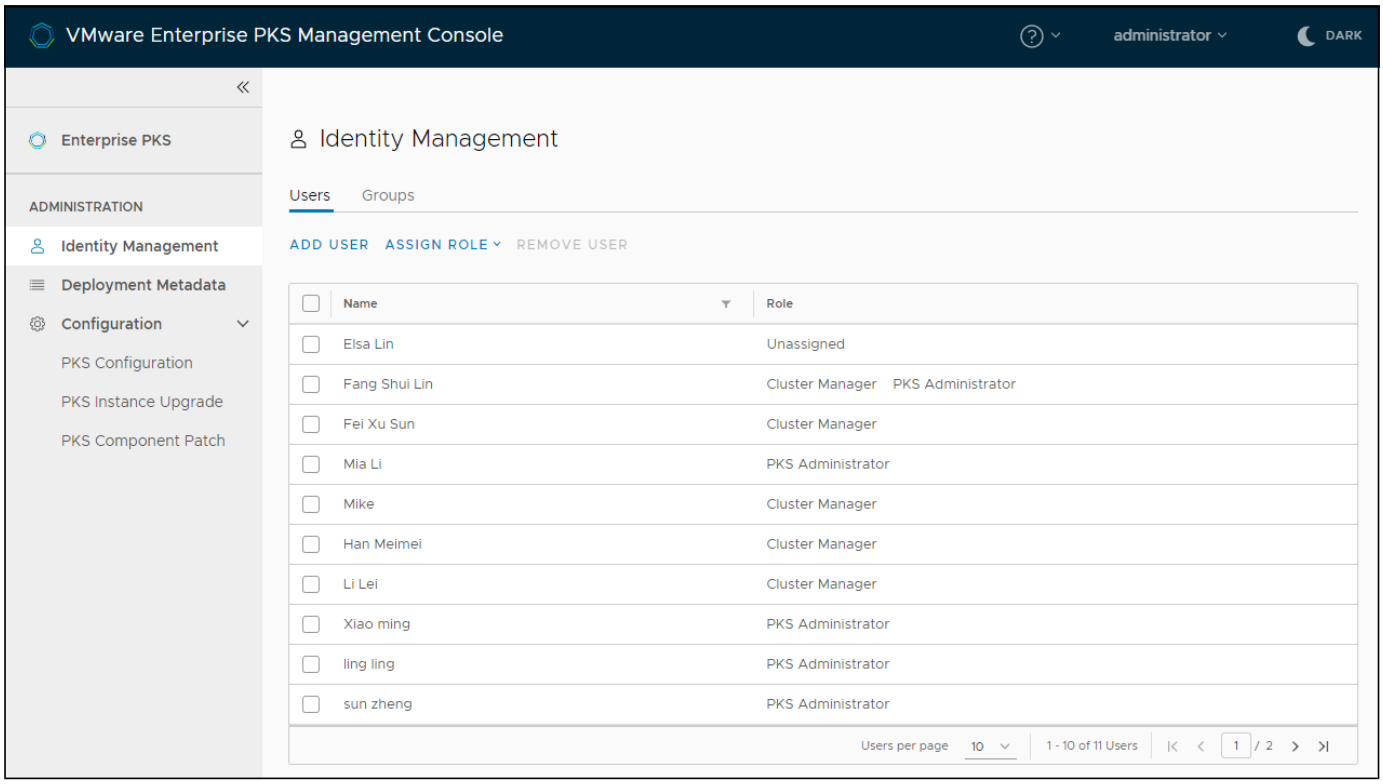
For information about the tasks that Cluster Managers can perform, see Enterprise PKS Cluster Management. The PKS Administrator role allows users to manage the Enterprise PKS infrastructure.

Add Individual Users

The procedure to add individual users to Enterprise PKS Management Console is as follows.

> **Note**: This release of Enterprise PKS Management Console does not support assigning roles to individual LDAP or SAML users. To assign roles to LDAP or SAML users, use user groups.

1. Go to the **Identity Management** view of the management console.
2. Select the **Users** tab.

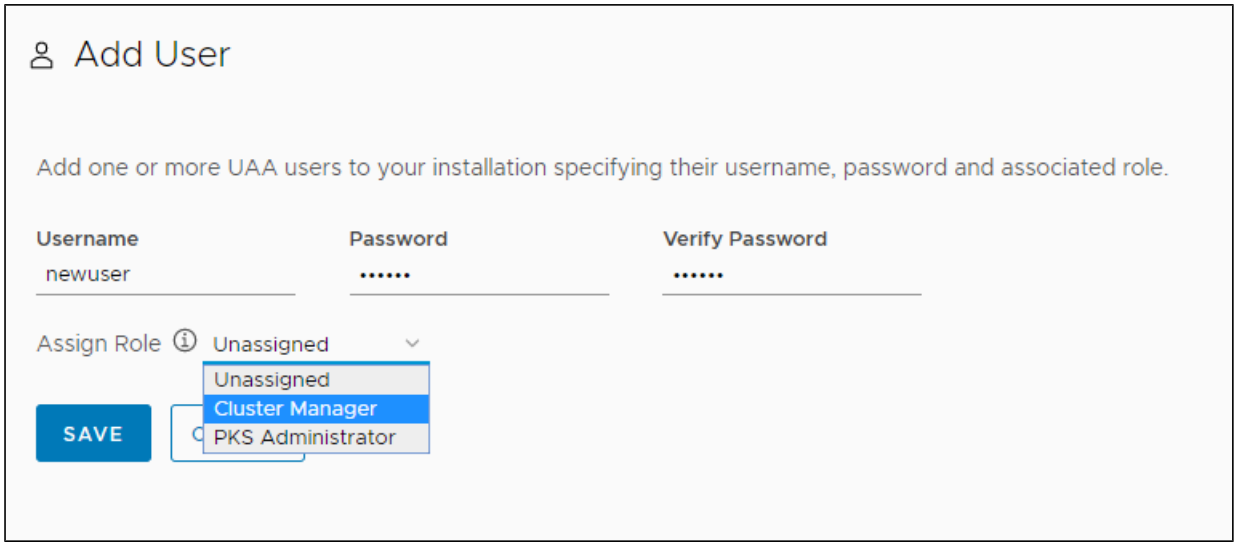


View a larger version of this image

3. Click **Add User**.

4. Select **LDAP** or **UAA** depending on the configuration that you set when you deployed Enterprise PKS.

   - If you select **LDAP**, enter a comma-separated list of user identities. The user identities must be configured in the user search filter. The default is `CN`.
   - If you select **UAA**, enter a user name and enter and verify a password to create a new user account.



5. Assign a role to the user.

   - **Cluster Manager**
   - **PKS Administrator**

6. Click **Save**.

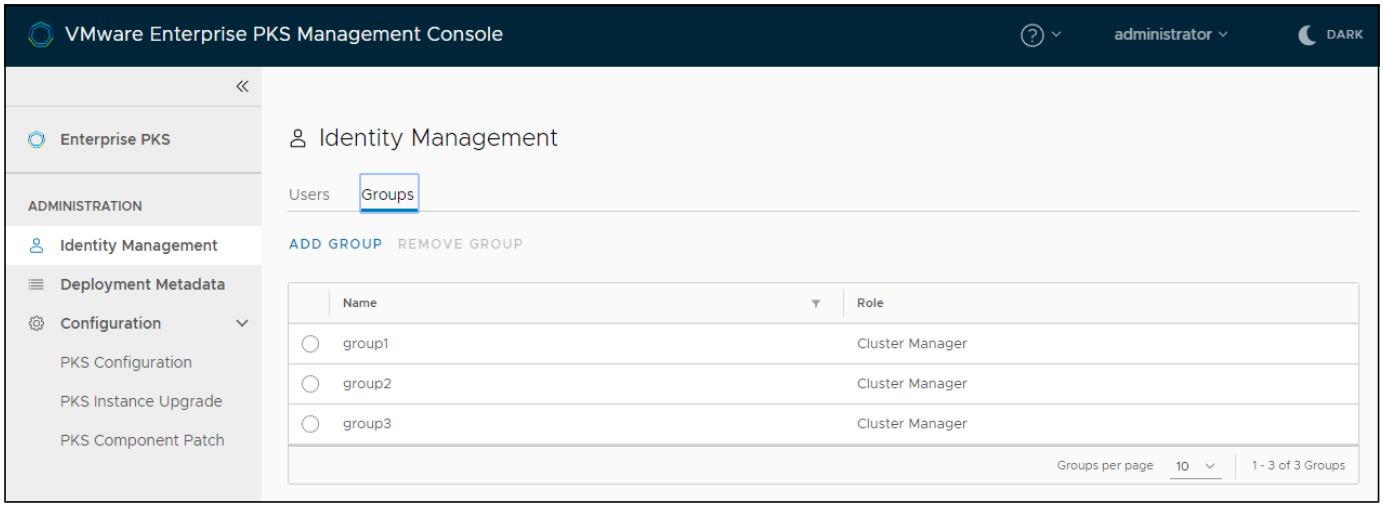
7. If you do not assign a role to a user when you create or add the account, or to change a user's role, select the user in the **Users** tab, and select **Assign Role**.

Add User Groups

The procedure to add user groups to Enterprise PKS Management Console is as follows.

1. Go to the **Identity Management** view of the management console.

2. Select the **Groups** tab.



View a larger version of this image

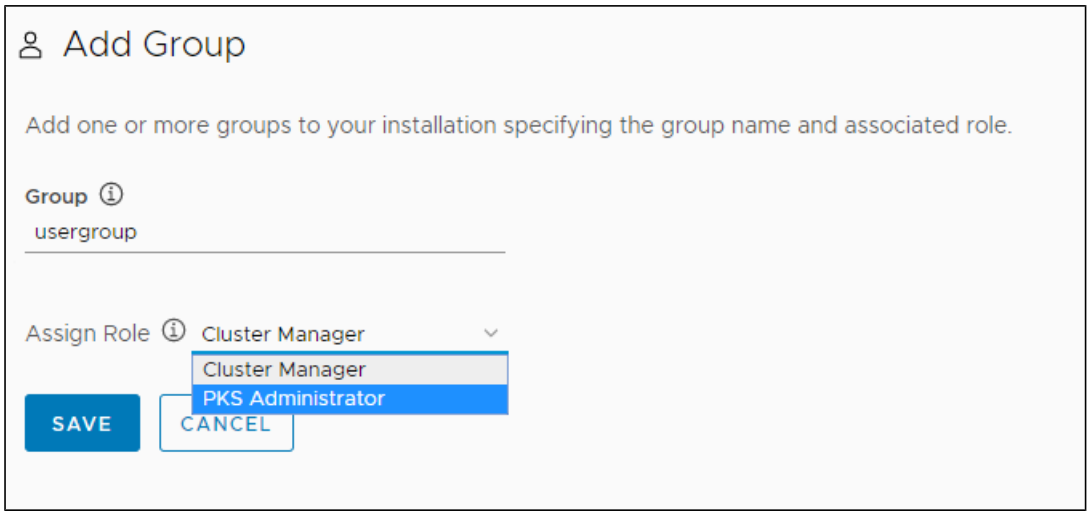
3. Click **Add Group**.

4. Enter an existing LDAP or SAML user group.

   - **LDAP**: Enter the distinguished name of an existing LDAP group under the configured group search base, for example `cn=admins,ou=engineering,dc=username,dc=local`.
   - **SAML**: Enter the name of your SAML identity provider group.

5. Assign a role to the group.

   - **Cluster Manager**
   - **PKS Administrator**



6. Click **Save**.

Note: You must assign a role to a group when you add it. You cannot assign, change, or revoke a group role after you have added the group.

Remove Individual Users

The procedure to remove individual users from Enterprise PKS Management Console is as follows.

1. Go to the **Identity Management** view of the management console.
2. Select the **Users** tab.
3. Select a user.
4. Click **Remove User**.

Remove User Groups

The procedure to remove individual users from Enterprise PKS Management Console is as follows.

1. Go to the **Identity Management** view of the management console.
2. Select the **Groups** tab.
3. Select a group.
4. Click **Remove Group**.

Connect to Operations Manager

In this topic

Page last updated:

When you use Enterprise PKS Management Console to deploy Enterprise PKS, it deploys Pivotal Cloud Foundry Operations Manager. The **Deployment Metadata** view of the management console displays the credentials that you need to log in to the deployed Operations Manager instance.

Connect to Operations Manager with SSH

Enterprise PKS Management Console generates an SSH private key to control SSH access to the Operations Manager VM when you deploy Enterprise PKS.

1. Go to **Deployment Metadata** in the management console.
2. Click the clipboard icon at the end of the **Ops Manager VM SSH Private Key** row to copy its contents.
3. Paste the contents of the SSH key into a text file, for example `~/pks_om.key`.
4. Go to the **Summary** tab of the **Enterprise PKS** view in the management console.
5. Copy the **Ops Manager IP Address**.
6. In a terminal run the following command to use SSH to connect to the Operations Manager VM:

```
ssh -i ~/pm_om.key ubuntu@ops_manager_endpoint_address
```

Log In to the Operations Manager UI

Enterprise PKS Management Console generates a random password for the Operations Manager admin account when you deploy Enterprise PKS.

1. Go to **Deployment Metadata** in the management console.
2. Click the clipboard icon at the end of the **Ops Manager VM Password** row to copy the password.
3. Go to the **Summary** tab of the **Enterprise PKS** view in the management console.
4. Click the **Ops Manager IP Address** to open the Operations Manager UI.
5. Log in to Operations Manager with user name `admin` and the password that you copied from the Deployment Metadata view.

Using the BOSH CLI

In this topic

Page last updated:

After you deploy Enterprise PKS from Enterprise PKS Management Console, you can use the BOSH CLI from both the appliance VM and the Ops Manager VM.

Using the BOSH CLI from the Enterprise PKS Management Console Appliance VM

You can use the BOSH CLI from the Enterprise PKS Management Console appliance VM.

1. In Enterprise PKS Management Console, go to the **Deployment Metadata** view.
2. Expand the row for **BOSH CLI invocation from console appliance**.
3. Click the clipboard icon at the end of the row to copy the BOSH CLI invocation information.
4. Connect to the management console appliance VM by using SSH.
   `ssh <console_vm_address>`
5. Export the value that you copied from Deployment Metadata view to use BOSH CLI from the management console appliance VM.
   `export <bosh_cli_invocation_value>`

Using BOSH SSH

The management console appliance does not support the use of the `bosh ssh` command to connect to the BOSH VM from the appliance VM. To connect to the BOSH VM by using `bosh ssh`, you must use the BOSH CLI from the Ops Manager VM.

1. In Enterprise PKS Management Console, go to the **Deployment Metadata** view.
2. Expand the row for **BOSH CLI invocation from Ops Manager**.
3. Click the clipboard icon at the end of the row to copy the BOSH CLI invocation command.
4. Connect to the Ops Manager VM by using SSH.
   For information about how to connect to the Ops Manager VM, see Connect to Operations Manager with SSH.
5. Export the value that you copied from Deployment Metadata view to use BOSH CLI from Ops Manager.
   `export <bosh_cli_invocation_value>`

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

Reconfigure Your Enterprise PKS Deployment

In this topic

Page last updated:

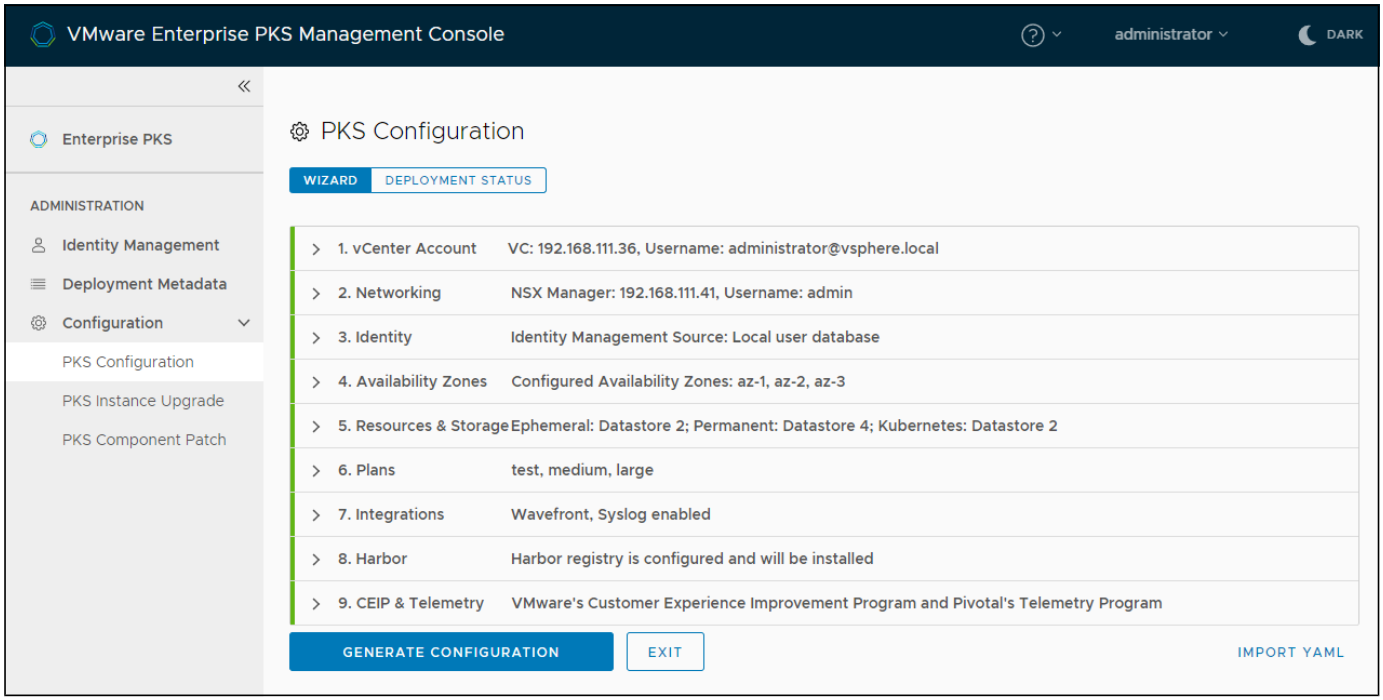
After deployment, you can reconfigure your your Enterprise PKS installation in VMware Enterprise PKS Management Console, either by using the wizard or by importing an updated YAML file.

Reconfigure Your Enterprise PKS Deployment in the Wizard

The procedure to reconfigure Enterprise PKS deployments in the wizard is as follows.

1. Expand **Configuration** and go to the **PKS Configuration** view of the management console.

2. Select **Wizard** to be taken to the configuration wizard for Enterprise PKS.



View a larger version of this image

3. Expand the different sections of the wizard and change the configurations as necessary.

   For information about which configuration options you can change, see Which Options Can I Reconfigure?below.

   For the options that you can modify, refer to Deploy Enterprise PKS by Using the Configuration Wizardfor instructions about how to fill in each section.

4. When you have finished reconfiguring, click **Generate Configuration**.

5. Optionally export the `PksConfiguration.yaml` file to save a copy of your configuration.

6. Click **Apply Configuration** and **Continue** to complete the reconfiguration of Enterprise PKS.

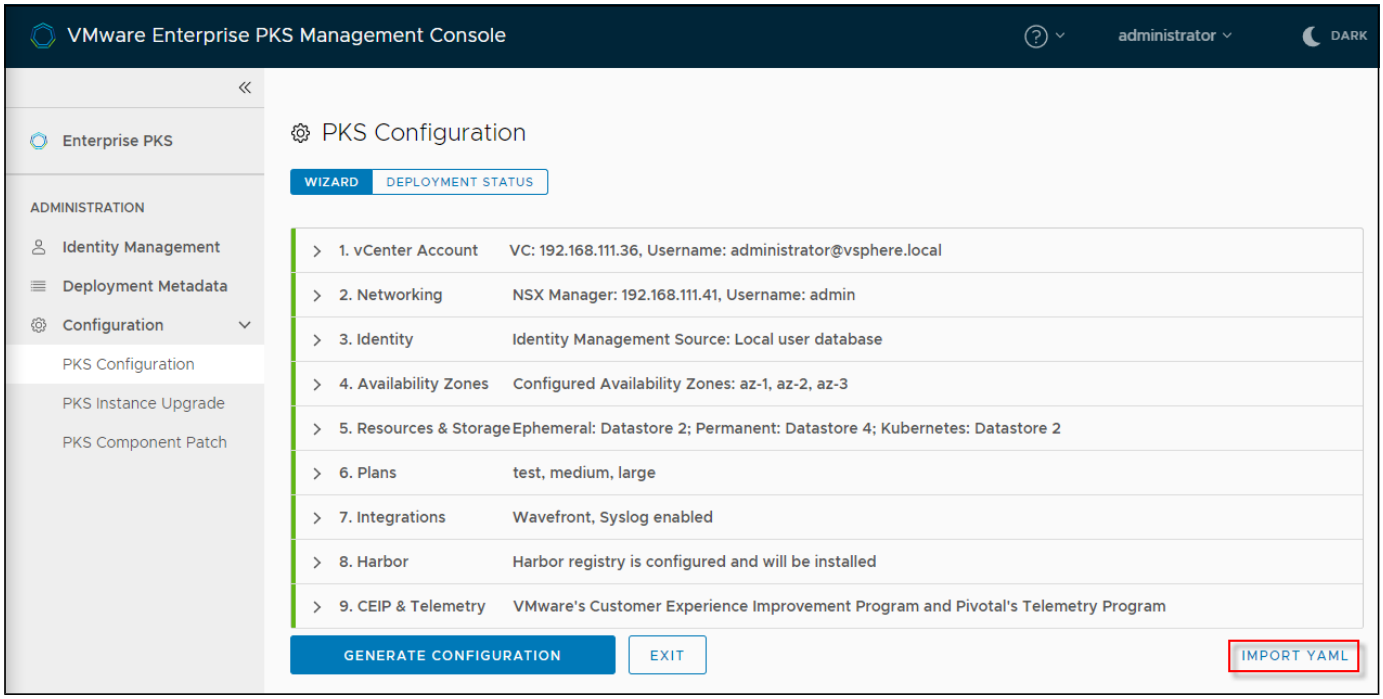
Reconfigure Your Enterprise PKS Deployment by Importing a YAML File

The procedure to reconfigure Enterprise PKS deployments by importing an updated YAML file is as follows.

1. Expand **Configuration** and go to the **PKS Configuration** view of the management console.

2. Select **Wizard** to be taken to the configuration wizard for Enterprise PKS.

3. Scroll to the bottom of the screen and click **Import YAML**.



4. Drag the YAML file into the Import Configuration File window, or click **Browse** to navigate to it.

5. In the Configuration File editor, modify the contents of the YAML file appropriately for the new instance of Enterprise PKS that you want to deploy.

   For information about which configuration options you can change, see Which Options Can I Reconfigure?below.

   If the YAML was generated by an instance of management console that is running in a different vSphere environment, update the passwords for NSX Manager, vCenter Server, and Harbor. For more information see Deploy Enterprise PKS by Importing a YAML Configuration File.

   To abandon this YAML and start again, click **Import** to upload the YAML again or to import a new one.

   You can also click the **Edit in Wizard** button, to open the imported configuration in the wizard.

6. Click **Apply Configuration** and **Continue** to complete the reconfiguration of Enterprise PKS.

Which Options Can I Reconfigure?

After the initial deployment of Enterprise PKS, there are certain options that you cannot reconfigure. In particular, you cannot make significant infrastructure changes.

The table below lists what you can and cannot modify on an existing Enterprise PKS deployment.

Section	Cannot Modify	Can Modify
vCenter Account	vCenter Server instance Datacenter	vCenter Server address, if it changes Username Password
Networking: Container Networking Interface	You cannot change between NSX-T Data Center (Automated NAT Deployment), NSX-T Data Center (Bring Your Own Topology), and Flannel.	None
Networking: NSX Manager Details	You cannot change the deployment to a different NSX Manager instance	NSX Manager address, if it changes Username Password
Networking: NSX-T Data Center (Automated NAT Deployment)	Tier0 Active Active Mode Deployment CIDR Deployment Network Reserved IP Range Usable Range of Floating IPs	All other options
Networking: NSX-T Data Center (Bring Your Own Topology)	Network for PKS Management Plane Floating IP Pool ID Deployment Network Reserved IP Range NAT mode	All other options
Networking: Flannel	Deployment Network Reserved IP Range Service Network Reserved IP Range	All other options
Identity	None	All options. Note that if you previously used a local user database and change to AD/LDAP or SAML, the local user database is deleted.
Availability Zones	Management availability zone	All other options, including adding and deleting availability zones

Plans	None	All options, including adding and deleting plans
Integrations	None	All options
Harbor	Harbor FQDN Authentication mode	All other options.
CEIP and Telemetry	None	All options

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

Upgrade Enterprise PKS Management Console

In this topic

Page last updated:

To upgrade a previous installation of VMware Enterprise PKS Management Console, you download and deploy a new version of the Enterprise PKS Management Console appliance. You then use the management console of the new appliance to migrate the configuration of the old installation to the new one.

You can only use the management console to upgrade an Enterprise PKS installation that was deployed from a previous version of the management console. You cannot use the console to upgrade an instance of Enterprise PKS that you installed manually.

Prerequisites

- You have deployed and configured an older version of Enterprise PKS Management Console.
- Download the new version of the Enterprise PKS Management Console OVA template from https://downloads.vmware.com.
- Use an account with vSphere administrator privileges to log in to vSphere using the vSphere Client.
- (Optional) If you deployed the old version of the Enterprise PKS Management Console appliance with a static IP address, and you want the new appliance to retain the same IP address after the upgrade, reconfigure the old appliance to use a temporary IP address before you start the upgrade procedure:

  1. Shut down the previous version of the appliance by selecting **Shut Down Guest OS**.

     > **warning:** Do not select **Power Off**.

  2. Access the vApp options for the appliance VM.

     - vSphere Client 6.5: Right-click the appliance VM, and select **Edit Settings** and select **vApp Options**.
     - vSphere Client 6.7: Select the appliance VM, select the **Configure** tab > **vApp Options** and scroll to the Properties section.

  3. Set a temporary IP address on the appliance VM.

     - vSphere Client 6.5: Edit the **2.1 Network IP Address** setting directly.
     - vSphere Client 6.7: Select the row for **2.1. Network IP Address** and click **Set Value**.

  4. Save your changes and power the appliance VM back on.

Step 1: Deploy the New OVA Template

Follow the instructions in Deploy the Enterprise PKS Management Console Appliance to deploy and power on the new version of

the appliance from the new OVA template.

> **Notes**: If you want to reuse the same IP address as before, and you assigned a temporary IP address to the old version of the appliance, configure the network settings of the new appliance to use the same static IP address as previously.
>
> If you used custom certificates when you deployed the previous version of the appliance, you must use the same certificates when you deploy the new version of the appliance. If you do not provide the certificate details when you deploy the new version, self-signed certificates are generated.

Step 2: Log In to the New Version of Enterprise PKS Management Console

When the OVA deployment has completed successfully, you can access the new version of the management console.

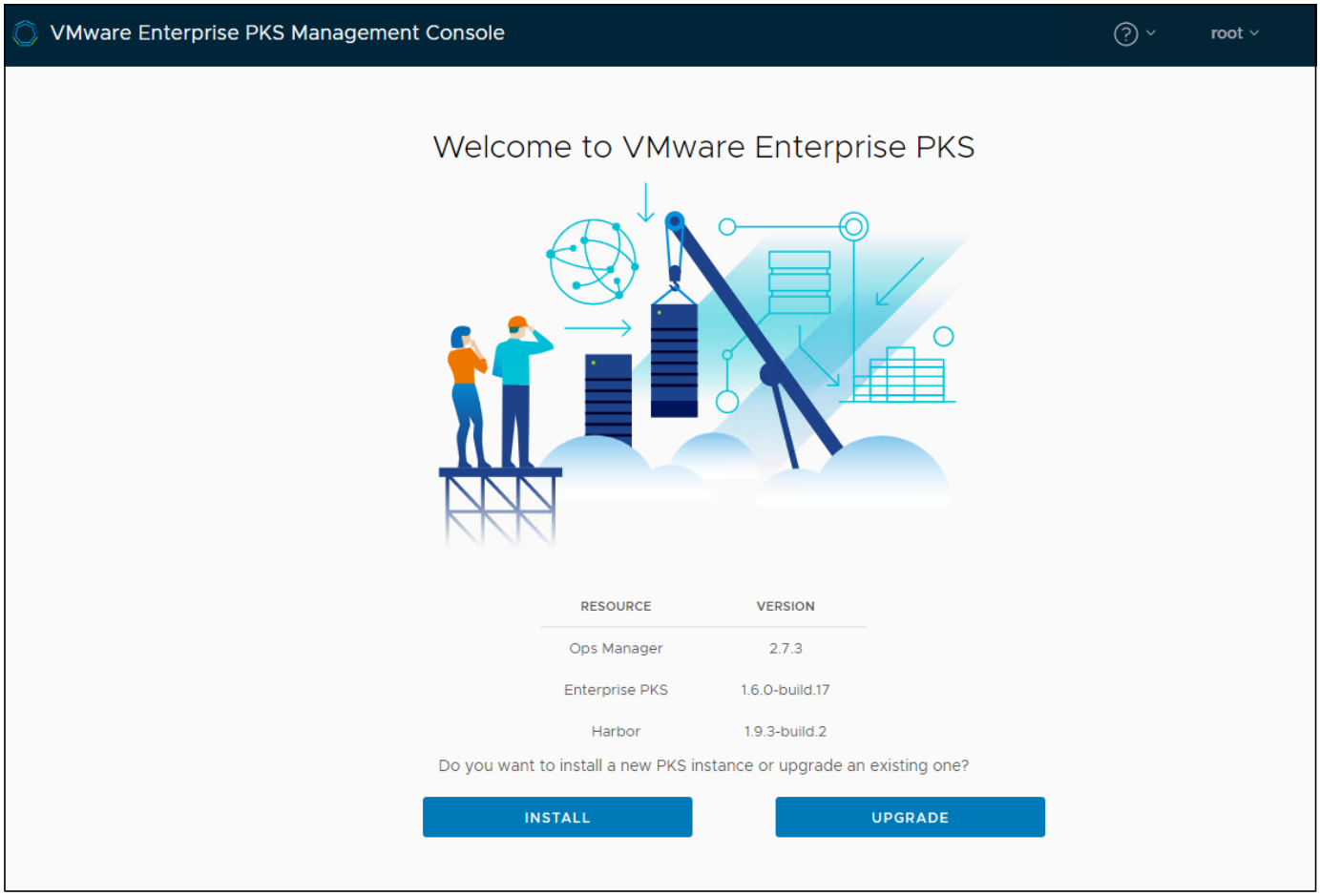
1. In the vSphere Client, right-click the new appliance VM and select **Power** > **Power On**.
2. When the new appliance VM has booted, go to the **Summary** tab for the VM and copy its IP address, if you do not know it already.
3. Enter the IP address of the new appliance in a browser.
4. At the VMware Enterprise PKS log in page, enter username `root` and the root password that you set when you deployed the new version of the OVA template.

Step 3: Migrate the Configuration from the Old Appliance to the New Version

Enterprise PKS Management Console provides an upgrade wizard to help you to migrate the configuration of your old deployment to the new version.

To get help in the wizard at any time, click the **?** icon at the top of the page, or click the **More Info...** links in each section to see help topics relevant to that section. Click the **i** icons for tips about how to fill in specific fields.

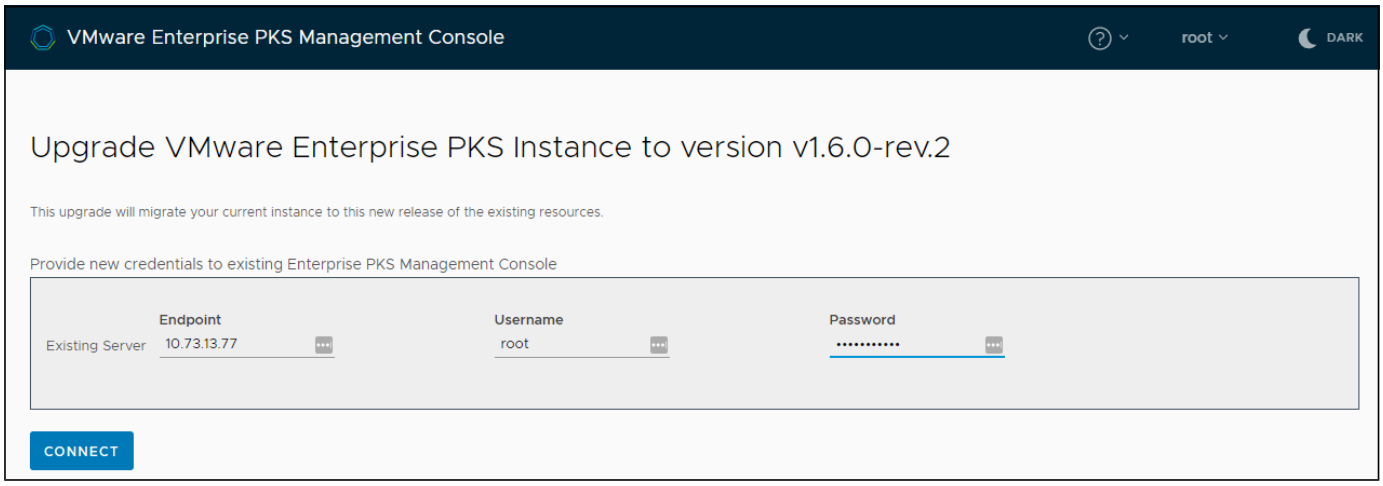
1. On the VMware Enterprise PKS Management Console landing page for the new appliance, click **Upgrade**.



View a larger version of this image

2. Enter the IP address of the old version of the Enterprise PKS Management Console appliance in the **Endpoint** text box.

3. Enter the username and password for the old version of the appliance VM and click **Connect**.



View a larger version of this image

4. Under Resources, verify the list of components that will be upgraded and click **Next**.

Resources

Resource	Previous Version	This Version	Description
harbor	1.8.4-build.2	1.9.3-build.2	harbor component upgrade
windowsStemcell			
opsman	2.6.11	2.7.3	opsman component upgrade
pks	1.5.1-build.8	1.6.0-build.17	pks component upgrade

NEXT CANCEL

View a larger version of this image

5. If any sections of the configuration wizard are marked in red, expand and reconfigure them.
   Sections might appear in red because they are in an error state, or because they relate to new configuration parameters that were not present in the previous version. For information about how to configure each section, see Deploy Enterprise PKS by Using the Configuration Wizard.

6. When all of the sections of the configuration wizard are green, click **Generate Configuration** to see the generated YAML file.

7. Optionally click **Export YAML** to save a copy of the YAML file for future use.
   This is recommended. The manifest is exported as the file `PksConfiguration.yaml`.

8. Click **Apply Configuration** then **Continue** to upgrade Enterprise PKS.

9. On the VMware Enterprise PKS Upgrade page, follow the progress of the upgrade.

Next Steps

You can now access the upgraded Enterprise PKS control plane and continue deploying Kubernetes clusters. Any new clusters that you deploy from the upgraded Enterprise PKS control plane will use the new version of Kubernetes.

> **important**: Existing clusters are not upgraded during the upgrade of Enterprise PKS Management Console. You must manually upgrade any existing clusters.

For information about how you can use Enterprise PKS Management Console to monitor and manage your upgraded deployment, see Monitor and Manage Enterprise PKS in the Management Console

You can decommission the old version of Enterprise PKS by deleting the previous version of the Enterprise PKS Management Console appliance VM from the vSphere inventory.

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

Patch Enterprise PKS Management Console Components

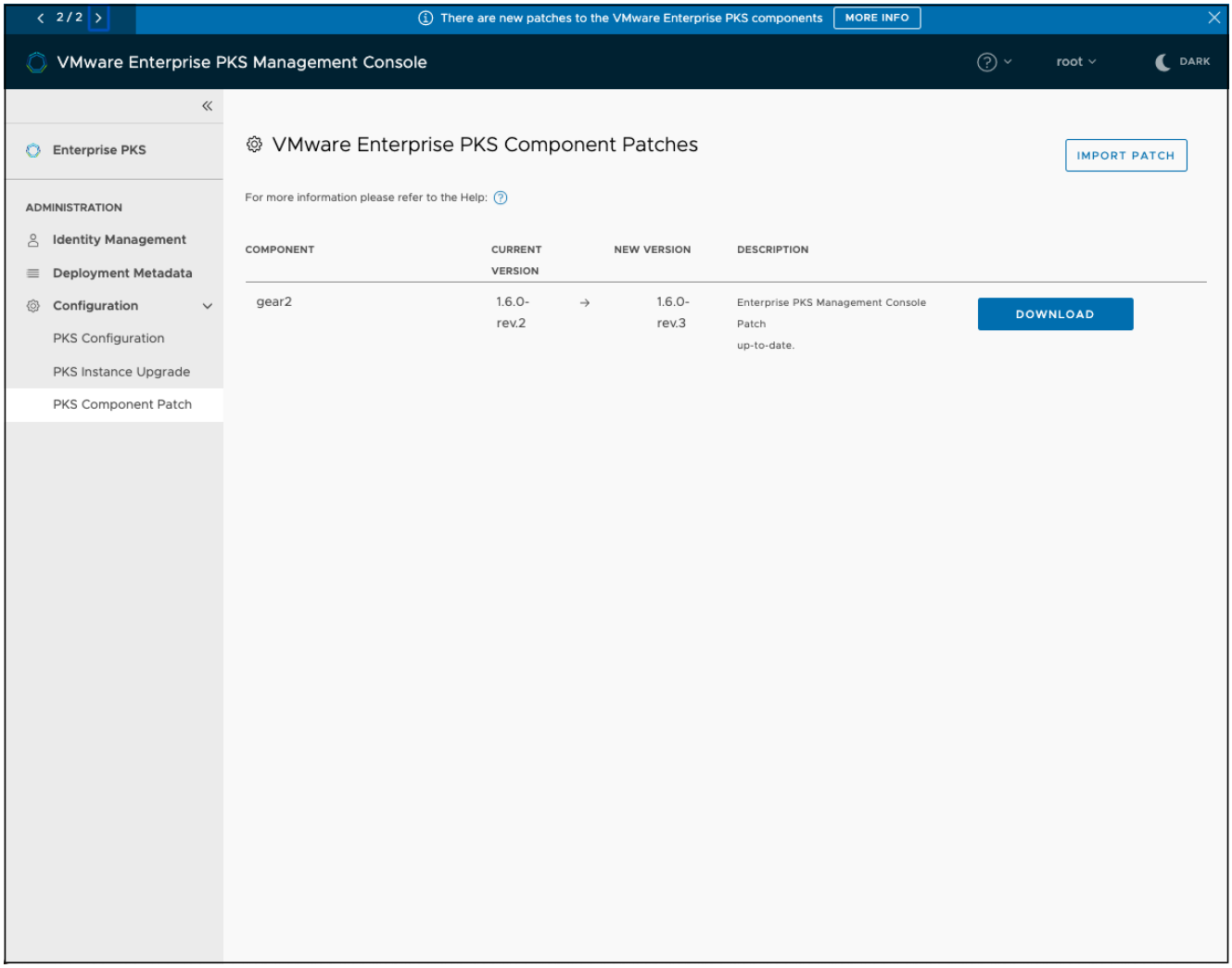
Page last updated:

You can use VMware Enterprise PKS Management Console to update some of the components of your deployment individually when a new minor version of those components is available.

1. In Enterprise PKS Management Console, go to **Configuration** > **PKS Component Patch** to view the list of components that are ready for patching.

2. Obtain the patch installers.

   - In air-gapped environments, download the patch installer from https://downloads.vmware.com/ to a local location. Click the **Import Patch** button to upload the installer to the management console.
   - In environments with access to the internet, click the **Download** button next to the relevant components to import the patch installers directly.



3. When the patch imports are complete, select **Install Patch** to patch a component.

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

Troubleshooting Enterprise PKS Management Console

In this topic

Page last updated:

The following sections describe how to troubleshoot failures to deploy of the VMware Enterprise PKS Management Console appliance and of Enterprise PKS instances from the management console.

Deployment of the Enterprise PKS Management Console Appliance Fails

Problem

Enterprise PKS Management Console appliance fails to deploy from the OVA template.

Solution

1. Use SSH to log in to the Enterprise PKS Management Console appliance VM as `root` user.
   Use the password that you specified when you deployed the OVA.
2. Run the following command to obtain the server logs:

   ```
   journalctl -u pks-mgmt-server > server.log
   ```

3. If the logs do not provide the solution, delete the appliance VM from vCenter Server and attempt to deploy it again.

Deployment of Enterprise PKS from the Management Console Fails

Problem

Enterprise PKS fails to deploy from the management console.

Solution

1. Follow the procedure in Delete Your Enterprise PKS Deployment to cleanly remove all Enterprise PKS components from vSphere and to clean up related objects in the management console appliance.
2. Attempt to deploy Enterprise PKS again.

Enterprise PKS Management Console Cannot Retrieve Cluster Data in a Multi-Tier0 Topology

Problem

In a deployment to a multiple-tier0 topology, Enterprise PKS Management Console cannot display cluster information when you go to **Enterprise PKS** > **Clusters** and select a cluster. You see errors of the following type:

```
Failed to retrieve current K8s Cluster summary. cannot get cluster details: cannot get cluster namespaces: Get https://<address>:8443/api/v1/namespaces: dial tcp <address>:8443: i/o timeout
Failed to retrieve current K8s Cluster Volumes. cannot get namespaces of cluster 0116663b-f27b-4026-87e3-cddd01af41f2: Get https://<address>:8443/api/v1/namespaces: dial tcp <address>:8443: i/o timeout
```

Cause

In a single tier0 topology, Enterprise PKS Management Console is deployed to the same infrastructure network as vSphere and NSX-T Data Center. In a multiple-tier0 topology, due to tenant isolation, the infrastructure network is not routable to tenant tier0 uplink networks. In a multiple-tier0 topology, data from the Kubernetes API is exposed by floating IP addresses on tenant tier0 routers. Consequently, the management console cannot retrieve cluster data from the Kubernetes API because it is not on the same network as the tenants.

Solution

Make sure that the Enterprise PKS Management Console can connect to tenant floating IP addresses.

1. Connect to the management console VM by using `ssh`.
2. Configure a route on the management console appliance VM. For example, run the following command:
   `route add -net <destination_subnet> gw <gateway_address>`
   - **Destination subnet**: The network CIDR of the tenant floating IP addresses.
   - **Gateway**: A VM that can reach the tenant floating IP addresses and the management console.

Because the gateway can reach both the management console and the tenant floating IP addresses, the management console can reach the tenants and retrieve cluster data from the Kubernetes API.

Obtain the vRealize Log Insight Agent ID for Enterprise PKS Management Console

If you enabled integration with VMware vRealize Log Insight, Enterprise PKS Management Console generates a unique vRealize Log Insight agent ID for the appliance. You must provide this agent ID to vRealize Log Insight so that it can pull the appropriate logs from the appliance.

You obtain the vRealize Log Insight agent ID as follows:

1. Use SSH to log in to the Enterprise PKS Management Console appliance VM as `root` user.
2. Run the following command to obtain the ID:

   ```
   grep LOGINSIGHT_ID /etc/vmware/environment | cut -d= -f2
   ```

   The resulting ID will be similar to `59debec7-daba-4770-9d21-226ffd743843`.
3. Log in to the vRealize Log Insight Web user interface as administrator and add the agent ID to your list of agents.

Delete Your Enterprise PKS Deployment

Page last updated:

If you no longer require an Enterprise PKS deployment, you can use VMware Enterprise PKS Management Console to delete it. The deletion process removes all objects from the vSphere inventory, and cleans up the objects in the Enterprise PKS Management Console appliance related to your deployment.

If Enterprise PKS fails to deploy correctly, always use Enterprise PKS Management Console to cleanly remove the failed deployment.

> **Note**: You must use the PKS CLI to delete any existing clusters and nodes before you can use the management console to delete your Enterprise PKS deployment.

1. Go to the **Enterprise PKS** view of the management console.
2. Click the **Action** drop-down menu and select **Delete Enterprise PKS Deployment**
3. Click **Delete** to confirm the deletion of the deployment.

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

Logging and Monitoring Enterprise PKS

Page last updated:

This section describes how to monitor VMware Enterprise PKS (PKS) environments.

See the following topics:

- Monitoring PKS and PKS-Provisioned Clusters
- Monitoring Workers and Workloads
- Accessing Dashboard
- Viewing Usage Data

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

Monitoring PKS and PKS-Provisioned Clusters

In this topic

Page last updated:

This topic lists VMware Enterprise PKS (PKS) components and integrations you can use to capture logs and metrics about PKS and PKS-provisioned cluster VMs.

For information about monitoring Kubernetes workloads, see Monitoring Workers and Workloads.

Overview

To monitor PKS and PKS-provisioned cluster VMs, you can enable one or more of the following components and integrations in the **Enterprise PKS** tile > **Host Monitoring**:

Name	Link
Syslog	See Syslog below.
VMware vRealize Log Insight	See vRLI Integration (vSphere Only) below.
Telegraf (metrics)	See Telegraf below.

These components and integrations are visible only to PKS admins. They are not visible to cluster users, such as developers.

To enable Syslog, VMware vRealize Log Insight (vRLI) Integration, or Telegraf, follow the instructions in *Host Monitoring* for your IaaS:

- Installing Enterprise PKS on vSphere
- Installing Enterprise PKS on vSphere with NSX-T
- Installing Enterprise PKS on GCP
- Installing Enterprise PKS on AWS
- Installing Enterprise PKS on Azure

Syslog

Syslog sends log messages from all BOSH-deployed VMs in a PKS environment to a syslog endpoint. To configure Syslog, see Syslog in the *Installing* topic for your IaaS.

If you do not use Syslog, you can retrieve logs from BOSH-deployed VMs by downloading them as described in Downloading Logs from VMs. However, retrieving these logs through Syslog is recommended.

vRLI Integration (vSphere Only)

The vRLI 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`.

To configure the vRLI integration, see VMware vRealize Log Insight Integration in the *Installing* topic for vSphere with Flannel or vSphere with NSX-T.

For information about vRLI, see vRealize Log Insight.

Inspecting Syslog and vRLI Logs

You might need to inspect Syslog or vRLI logs when troubleshooting or auditing your PKS environment. For information about key PKS events and the log entries they generate, see Auditing Enterprise PKS Logs.

Telegraf

The Telegraf agent sends metrics from master node and worker node VMs to a monitoring service, such as Wavefront or Datadog.

You can configure the Telegraf agent to collect metrics from one or both of the following sources:

Source	Includes metrics from...
Master nodes (not visible to cluster users)	One or both of the following: • Node Exporter (Prometheus) • etcd
Worker nodes	• kubelet

To configure the Telegraf agent, see Telegraf in the *Installing* topic for your IaaS.

For more information about Node Exporter, see About Node Exporter.

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

Auditing Enterprise PKS Logs

In this topic

Page last updated:

This topic summarizes key auditable events in PKS, and the content of the log entries that the events generate. Operators can use this information to audit event logs to see what users took what actions at what times. This is helpful for security, compliance, and troubleshooting.

Log content can either be downloaded or configured to be transported via syslog.

PKS API events

The following log entry examples are produced by PKS API events and correspond to key actions taken by a user logged into the PKS CLI.

Cluster Creation

create-cluster	
Description	A user has issued a create cluster command.
Identifying String	`Action 'create-cluster'`

Example Log Entries	`2019-05-16 14:59:34.897 INFO 7594 --- [nio-9021-exec-7] io.pivotal.pks.cluster.ClusterService : Action 'create-cluster' by user 'admin', cluster name: 'logs', plan name: 'small'. Details: class ClusterParameters { kubernetesMasterHost: logs.lathrop.cf-app.com kubernetesMasterPort: 8443 workerHaproxyIpAddresses: null kubernetesWorkerInstances: 3 authorizationMode: null nsxtNetworkProfile: null } 2019-05-16 14:59:34.911 INFO 7594 --- [nio-9021-exec-7] io.pivotal.pks.telemetry.Agent : Telemetry - addCluster: cluster request: class ClusterRequest { name: logs planName: small networkProfileName: null parameters: class ClusterParameters { kubernetesMasterHost: logs.lathrop.cf-app.com kubernetesMasterPort: 8443 workerHaproxyIpAddresses: null kubernetesWorkerInstances: 3 authorizationMode: null nsxtNetworkProfile: null } }, cluster entity: ClusterEntity{name='logs', uuid='f4e2b775-8be3-41b8-abe8-67f2265b957e', owner='admin', brokerOperationId='{"BoshTaskID":479,"BoshContextID":"256c3b65-2eae-48f7-81f0-caed7472fa5f","OperationType":"create","PostDeployErrand":{},"PreDeleteErrand":{},"Errands":[{"Name":"apply-addons","Instances":null},{"Name":"vrops-errand","Instances":null},{"Name":"telemetry-agent","Instances":null}]}', lastActionDescription='Creating cluster', planId='8A0E21A8-8072-4D80-B365-D1F502085560', lastAction='CREATE', lastActionState='in progress', masterIps='[In Progress]', parameters=io.pivotal.pks.cluster.data.ClusterParametersEntity@6efbedb6', networkProfileUuid=null', computeProfileUuid=null', taskStartedAt=2019-05-16T14:59:34.804}, plan: class Plan { id: 8A0E21A8-8072-4D80-B365-D1F502085560 name: small description: Example: This plan will configure a lightweight kubernetes cluster. Not recommended for production workloads. workerInstances: 3 masterInstances: 1 allowPrivilegedContainers: false }`

Cluster Deletion

delete-cluster	
Description	A user has issued a delete cluster command.
Identifying String	`delete deployment for instance`
Example Log Entries	`2019-06-04T14:16:52-06:00 10.0.10.10 broker/rs2 [on-demand-service-broker] [2f71a161-5755-4a0d-9c21-5b8405209594] 2019/06/04 20:16:52.493286 BOSH task ID 132 status: processing delete deployment for instance 67f77801-3d15-4d65-b501-38a643055e69: Description: delete deployment service-instance_67f77801-3d15-4d65-b501-38a643055e69 Result:`

Successful Login

UserAuthenticationSuccess	
Description	A user has successfully logged into Enterprise PKS.
Identifying String	`UserAuthenticationSuccess`

Example Log Entries	`[2019-05-16 17:12:48.833] uaa - 7777 [https-jsse-nio-8443-exec-2] .... INFO --- Audit: UserAuthenticationSuccess ('admin'): principal=0074aab6-6ff7-4b4c-b821-49526a96ebcb, origin=[remoteAddress=207.126.127.114, clientId=pks_cli], identityZoneId=[uaa] [2019-05-16 17:12:48.873] uaa - 7777 [https-jsse-nio-8443-exec-2] .... INFO --- Audit: TokenIssuedEvent ('["pks.clusters.admin"]'): principal=0074aab6-6ff7-4b4c-b821-49526a96ebcb, origin=[client=pks_cli, user=admin], identityZoneId=[uaa]`

Unsuccessful Login

UserAuthenticationFailure	
Description	A user has failed a login attempt into Enterprise PKS.
Identifying String	`UserAuthenticationFailure`
Example Log Entries	`[2019-05-16 17:15:31.363] uaa - 7777 [https-jsse-nio-8443-exec-8] .... INFO --- Audit: UserAuthenticationFailure ('admin'): principal=0074aab6-6ff7-4b4c-b821-49526a96ebcb, origin=[remoteAddress=207.126.127.114, clientId=pks_cli], identityZoneId=[uaa] [2019-05-16 17:15:31.371] uaa - 7777 [https-jsse-nio-8443-exec-8] .... INFO --- Audit: PrincipalAuthenticationFailure ('null'): principal=admin, origin=[207.126.127.114], identityZoneId=[uaa] [2019-05-16 17:15:33.387] uaa - 7777 [https-jsse-nio-8443-exec-6] .... INFO --- Audit: ClientAuthenticationSuccess ('Client authentication success'): principal=pks_client, origin=[remoteAddress=127.0.0.1, cl`

Successful Cluster Credential Retrieval

ClientAuthenticationSuccess	
Description	A user has successfully gained access to a cluster in Enterprise PKS.
Identifying String	`ClientAuthenticationSuccess`
Example Log Entries	`[2019-05-16 17:15:31.363] uaa - 7777 [https-jsse-nio-8443-exec-8] .... INFO --- Audit: UserAuthenticationFailure ('admin'): principal=0074aab6-6ff7-4b4c-b821-49526a96ebcb, origin=[remoteAddress=207.126.127.114, clientId=pks_cli], identityZoneId=[uaa] [2019-05-16 17:15:31.371] uaa - 7777 [https-jsse-nio-8443-exec-8] .... INFO --- Audit: PrincipalAuthenticationFailure ('null'): principal=admin, origin=[207.126.127.114], identityZoneId=[uaa] [2019-05-16 17:15:33.387] uaa - 7777 [https-jsse-nio-8443-exec-6] .... INFO --- Audit: ClientAuthenticationSuccess ('Client authentication success'): principal=pks_client, origin=[remoteAddress=127.0.0.1, cl`

User Creation

UserCreatedEvent	
Description	An administrator has successfully created a new user for Enterprise PKS.
Identifying String	`UserCreatedEvent`

Example Log Entries	`Jun 04 16:00:07 10.0.10.10 uaa/rs2: [2019-06-04 22:00:07.293] uaa - 18840 [https-jsse-nio-8443-exec-6] .... INFO --- Audit: UserCreatedEvent ('["user_id=dc803130-15dc-4279-8b42-868fc80b8ca1","username=USERNAME2"]'): principal=dc803130-15dc-4279-8b42-868fc80b8ca1, origin=[client=admin, details=(remoteAddress=35.192.67.34, tokenType=bearertokenValue=, sub=admin, iss=https://api.pks.hawthorne.cf-app.com:8443/oauth/token)], identityZoneId=[uaa]`

User Deletion

UserDeletedEvent	
Description	An administrator has successfully deleted a user for Enterprise PKS.
Identifying String	`UserDeletedEvent`
Example Log Entries	`Jun 04 16:00:07 10.0.10.10 uaa/rs2: [2019-06-04 22:00:07.293] uaa - 18840 [https-jsse-nio-8443-exec-6] .... INFO --- Audit: UserCreatedEvent ('["user_id=dc803130-15dc-4279-8b42-868fc80b8ca1","username=USERNAME2"]'): principal=dc803130-15dc-4279-8b42-868fc80b8ca1, origin=[client=admin, details=(remoteAddress=35.192.67.34, tokenType=bearertokenValue=, sub=admin, iss=https://api.pks.hawthorne.cf-app.com:8443/oauth/token)], identityZoneId=[uaa]`

Telemetry Collection

Telemetry Ping	
Description	The optional telemetry system has successfully reached an external host for collecting product data for Enterprise PKS. To learn more about the Enterprise PKS telemetry program, see Telemetry.
Identifying String	`telemetry-server`
Example Log Entries	`2019-06-04T15:41:05-06:00 10.0.10.10 telemetry-server/rs2 2019-06-04 21:41:05 +0000 [debug]: #0 generating helo 2019-06-04T15:41:05-06:00 10.0.10.10 telemetry-server/rs2 2019-06-04 21:41:05 +0000 [debug]: #0 checking ping 2019-06-04T15:41:05-06:00 10.0.10.10 telemetry-server/rs2 2019-06-04 21:41:05 +0000 [debug]: #0 generating pong 2019-06-04T15:41:05-06:00 10.0.10.10 telemetry-server/rs2 2019-06-04 21:41:05 +0000 [debug]: #0 connection established address="10.0.11.21" port=33366`

Kubernetes Audit Log Events

The Kubernetes control plane emits a standard log format every time a user takes action to query or change the state of the Kubernetes API. An example audit event log entry is below.

```
{
  "kind": "Event",
  "apiVersion": "audit.k8s.io/v1",
  "level": "Request",
  "auditID": "dc2bb4e9-4b85-42da-82a3-5ee47091207d",
  "stage": "ResponseStarted",
  "requestURI": "/apis/policy/v1beta1/poddisruptionbudgets?resourceVersion=370506\u0026timeout=7m54s\u0026timeoutSeconds=474\u0026watch=true",
  "verb": "watch",
  "user": {
    "username": "system:kube-scheduler",
    "uid": "system:kube-scheduler",
    "groups": ["system:authenticated"]
  },
  "sourceIPs": ["10.0.11.10"],
  "userAgent": "kube-scheduler/v1.15.4 (linux/amd64) kubernetes/67d2fcf/scheduler",
  "objectRef": {
    "resource": "poddisruptionbudgets",
    "apiGroup": "policy",
    "apiVersion": "v1beta1"
  },
  "responseStatus": {
    "metadata": {},
    "code": 200
  },
  "requestReceivedTimestamp": "2019-12-11T21:47:28.097065Z",
  "stageTimestamp": "2019-12-11T21:47:28.097491Z",
  "annotations": {
    "authorization.k8s.io/decision": "allow",
    "authorization.k8s.io/reason": "RBAC: allowed by ClusterRoleBinding \"system:kube-scheduler\" of ClusterRole \"system:kube-scheduler\" to User \"system:kube-scheduler\""
  }
}
```


For more information about Kubernetes Audit Event Log format see the Kubernetes documentation.

Related Links

- For information about configuring syslog log transport, see Installing Enterprise PKS.
- For information about downloading PKS logs, see Downloading Logs from VMs.
- For information about Kubernetes Audit Log format, see Kubernetes documentation

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

Downloading Logs from VMs

In this topic

Page last updated:

This topic explains how to download logs from BOSH-deployed VMs in your VMware Enterprise PKS environment using the BOSH Command Line Interface (CLI).

Overview

In Enterprise PKS, you can download logs from any BOSH-deployed VM, such as the PKS API VM or Kubernetes cluster VMs.

You might need to download these logs when troubleshooting or auditing your PKS environment.

Download Logs

To download logs from a BOSH-deployed VM:

1. Gather credential and IP address information for your BOSH Director, SSH into the Ops Manager VM, and use the 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. After logging in to the BOSH Director, list the names of your BOSH deployments by running:

   ```
   bosh -e ENVIRONMENT deployments
   ```

   Where `ENVIRONMENT` is your BOSH environment alias. For example:

   ```
   $ bosh -e pks deployments
   ```

3. Identify the names of the VMs that you want to retrieve logs from by listing the VMs in your target BOSH deployment:

   ```
   bosh -e ENVIRONMENT -d DEPLOYMENT vms
   ```

   Where:

   - `ENVIRONMENT` is the BOSH environment alias.
   - `DEPLOYMENT` is your target BOSH deployment name.

   For example, the following command lists the VMs in a Kubernetes cluster:

   ```
   $ bosh -e pks -d service-instance_ae681cd1-7ff4-4661-b12c-49a5b543f16f vms
   ```

   Kubernetes cluster deployment names begin with `service-instance_` and include a unique identifier.

4. Download logs from a VM:

```
bosh -e ENVIRONMENT -d DEPLOYMENT logs VM-NAME
```

For example:

```
$ bosh -e pks \
-d service-instance_ac681cd1-7ff4-4661-b12c-49a5b543f16f \
logs master/000a1111-222b-3333-4cc5-de66f7a8899b
```

For more information about log files, see View Log Files in *Using BOSH Diagnostic Commands in Enterprise PKS*.

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

Monitoring Master/etcd Node VMs

In this topic

Page last updated:

This topic describes how platform operators can monitor and retrieve metrics from master/etcd node VMs in VMware Enterprise PKS deployments.

Overview

You can configure the Telegraf agent to collect metrics from master/etcd node VMs and send the metrics to a third-party monitoring service. The Telegraf agent can read metrics from the following:

- the `/metrics` endpoint exposed by etcd
- Node Exporter

Node Exporter provides additional infrastructure metrics on the localhost of the master node. For more information, see node_exporter in GitHub.

Collect Metrics Using Telegraf

To collect metrics using Telegraf, do the following:

1. Create a configuration file for your output plugin. See Create a Configuration File.
2. Configure Telegraf in the Enterprise PKS tile. See Configure Telegraf in the Tile.

Create a Configuration File

To connect a third-party monitoring service to Enterprise PKS, you must create a configuration file for the service. The configuration file is written in a TOML format and consists of key-value pairs. After you create your configuration file, you can enter the file into the Enterprise PKS tile to connect the service.

To create a configuration file your monitoring service, do the following:

1. Locate the required format for your monitoring service in the `README.md` file for your service in telegraf in GitHub. The etcd documentation recommends using the open source Prometheus monitoring service.

   For more information about using Prometheus, see Overview in the Prometheus documentation.

2. Create your configuration file using the required format of your monitoring service. For example, if you want to create a configuration file for an HTTP output plugin, create a file similar to the following:

```
[[outputs.http]]
  url="https://example.com"
  method="POST"
  data_format="json"
[[processors.override]]
 [processors.override.tags]
   director = "bosh-director-1"
```

Note: You can add tags to your configuration file to label etcd metrics. For example, the above code snippet adds a `bosh-director-1` tag to the etcd metrics. If you have multiple BOSH Directors, Pivotal recommends adding tags to filter your metrics in your monitoring service.

Configure Telegraf in the Tile

To configure Telegraf in the Enterprise PKS tile, follow the instructions in the *Installing* topic for your IaaS. For example, if your are installing Enterprise PKS on vSphere, follow the instructions in the Telegraf section of *Installing Enterprise PKS on vSphere*.

Troubleshoot etcd

Pivotal recommends working with Pivotal or VMware Support to troubleshoot master/etcd node VMs. The monitoring and metrics data you retrieve 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.

Monitoring Workers and Workloads

In this topic

Page last updated:

This topic lists VMware Enterprise PKS components and integrations you can use to capture logs and metrics about your Kubernetes worker nodes and workloads.

For information about monitoring PKS and PKS-provisioned cluster VMs, see Monitoring PKS and PKS-Provisioned Clusters.

Overview

To monitor Kubernetes worker nodes and workloads in your PKS deployment, you can enable one or more of the following components and integrations in the **Enterprise PKS** tile > **In-Cluster Monitoring**:

Name	Type	Link
Sink resources	PKS component	See Sink Resources below.
Wavefront	Integration	See Supported Integrations below.
cAdvisor	Integration	See Supported Integrations below.

When running on worker nodes, these components and integrations are visible to both PKS admins and cluster users, such as developers.

To enable sink resources, Wavefront, or cAdvisor integration, follow the instructions in *In-Cluster Monitoring* for your IaaS:

- Installing Enterprise PKS on vSphere
- Installing Enterprise PKS on vSphere with NSX-T
- Installing Enterprise PKS on GCP
- Installing Enterprise PKS on AWS
- Installing Enterprise PKS on Azure

Sink Resources

In PKS, you can deploy log sinks and metric sinks to monitor your Kubernetes worker nodes and workloads that are running on them.

To deploy a log or a metric sink:

1. Enable sink resources in the **Enterprise PKS** tile > **In-Cluster Monitoring**. You can enable both log and metric sink resources or only one of them.

2. (Optional) Enable Node Exporter on worker nodes by selecting the **Enable node exporter on workers** checkbox.

3. Create sink resources. For instructions, see Creating and Managing Sink Resources.

For more information about sink resources, see:

- Conceptual information: Sink Architecture in Enterprise PKS
- Sink resource types, outputs, and identifying strings: Monitoring Clusters with Log Sinks

Supported Integrations

PKS supports the following integrations:

- **Wavefront**. For more information, see VMware Tanzu Kubernetes Grid Integration Details in the Wavefront documentation.

  > **Note:** You can also create a metric sink to send metrics to Wavefront.

- **cAdvisor**. For more information, see cAdvisor on GitHub and VMware vRealize Operations Management Pack for Container Monitoring.

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

Sink Architecture in Enterprise PKS

In this topic

Page last updated:

This topic describes how VMware Enterprise PKS (PKS) implements sinks for collecting logs and metrics from Kubernetes worker nodes and workloads.

For step-by-step instructions on creating sinks in PKS, see Creating and Managing Sink Resources.

Overview

A sink collects logs or metrics about Kubernetes worker nodes in a PKS deployment and workloads that are running on them.

For more information, see:

- Sink Types below
- Sink Architecture below

Sink Types

You can create two types of sinks:

- Log sinks
- Metric sinks

See the table below for information about these sink types.

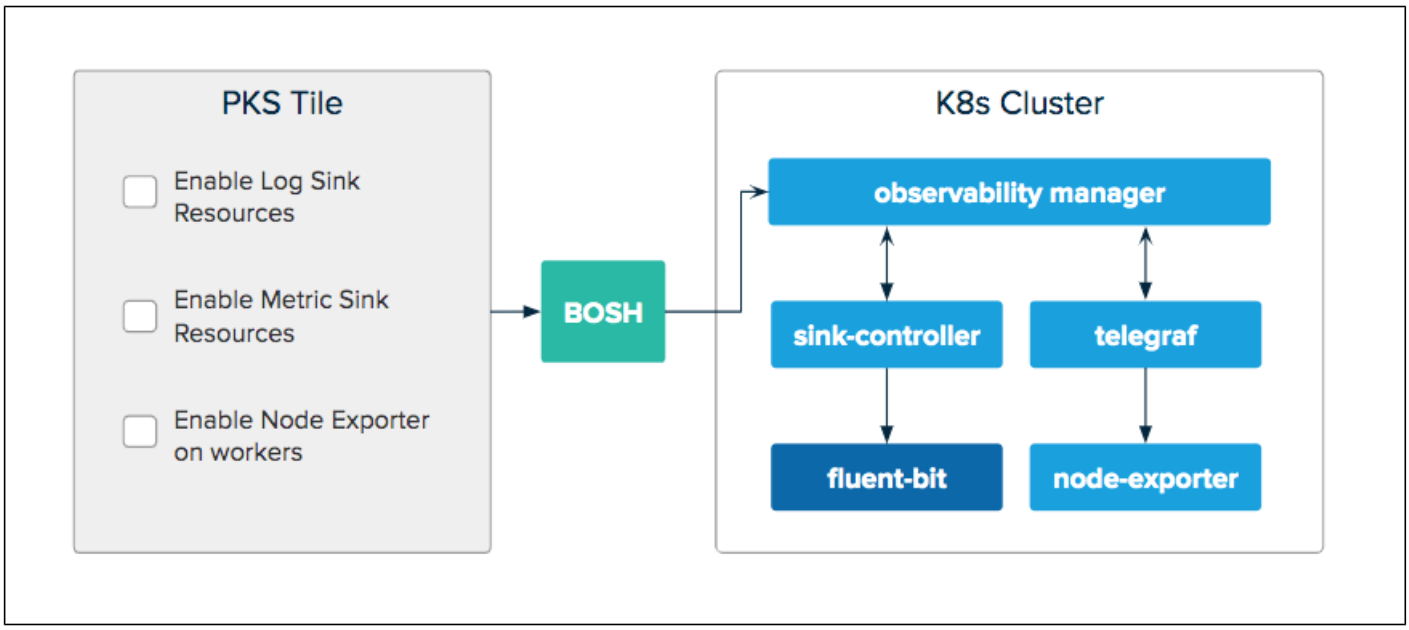
Sink Type	Sink Resource	Description
Log sink	`ClusterLogSink`	Forwards logs from a cluster to a log destination. Logs are transported using one of the following: • The Syslog Protocol defined in RFC 5424 • WebHook • Fluent Bit output plugins

Log sink	`LogSink`	Forwards logs from a namespaced subset within a `ClusterLogSink` resource to a log destination. Logs are transported using one of the following: • The Syslog Protocol defined in RFC 5424 • WebHook • Fluent Bit output plugins
Metric sink	`ClusterMetricSink`	Collects and writes metrics from a cluster to specified outputs using input and output plugins.
Metric sink	`MetricSink`	Collects and writes metrics from a namespace within a cluster to specified outputs using input and output plugins.

Sink Architecture

PKS-provisioned Kubernetes clusters include an observability manager that manages log sink and metric sink configurations within a cluster.

The following diagram details PKS cluster observability architecture:



In the **Enterprise PKS** tile > **In-Cluster Monitoring**:

- **Enable Metric Sink Resources** enables metric sinks.
- **Enable Log Sink Resources** enables log sinks.
- **Enable node exporter on workers** forwards additional infrastructure metrics.

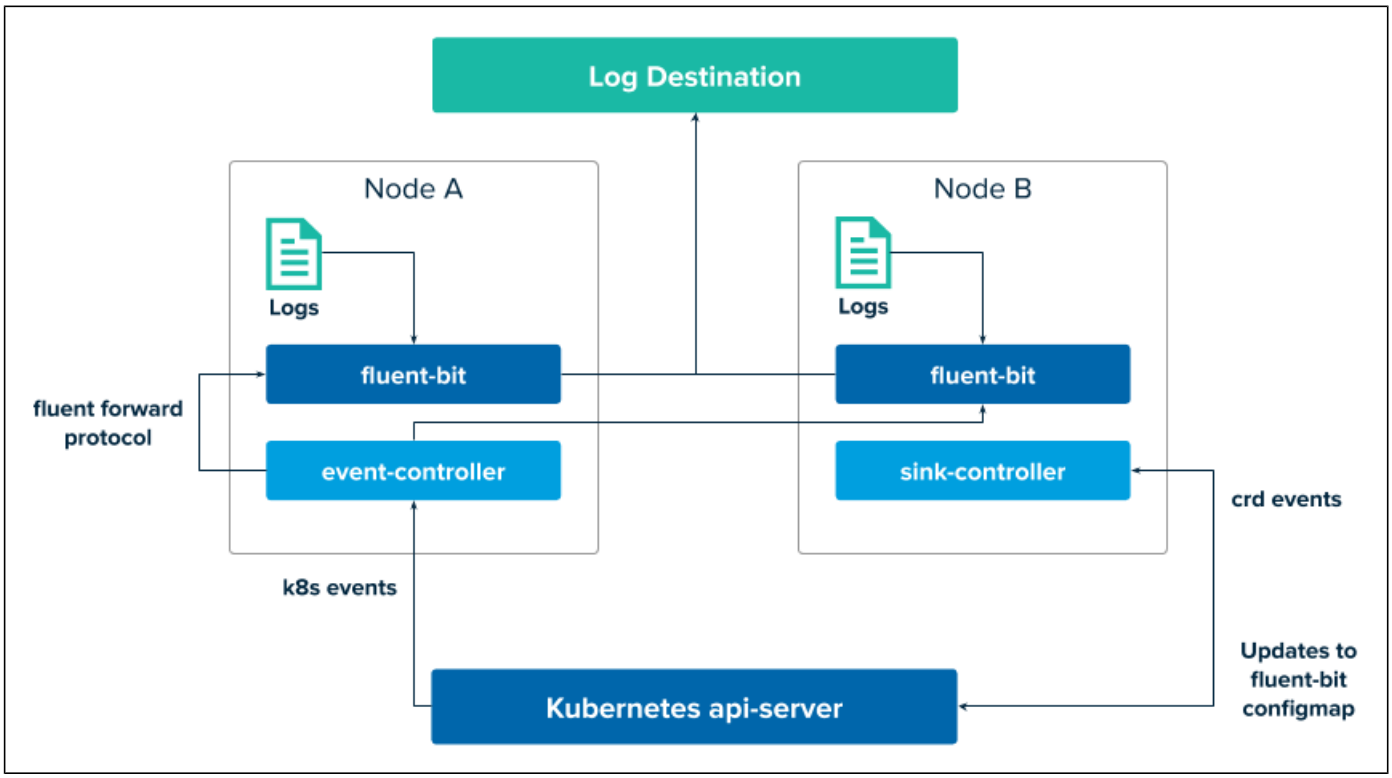
Setting these checkboxes in Ops Manager directs how BOSH configures the observability manager.

For more information about enabling log sinks and metrics sinks, see (Optional) In-Cluster Monitoring in the *Installing* topic for your IaaS.

Log Sink Architecture

The PKS log sink aggregates workload logs and forwards them to a common log destination.

The following diagram details PKS log sink architecture:



Logs are monitored and aggregated by a Fluent Bit `DaemonSet` running as a pod on each worker node.

An event-controller collects Kubernetes API events and sends them to a second Fluent Bit daemon pod for aggregation.

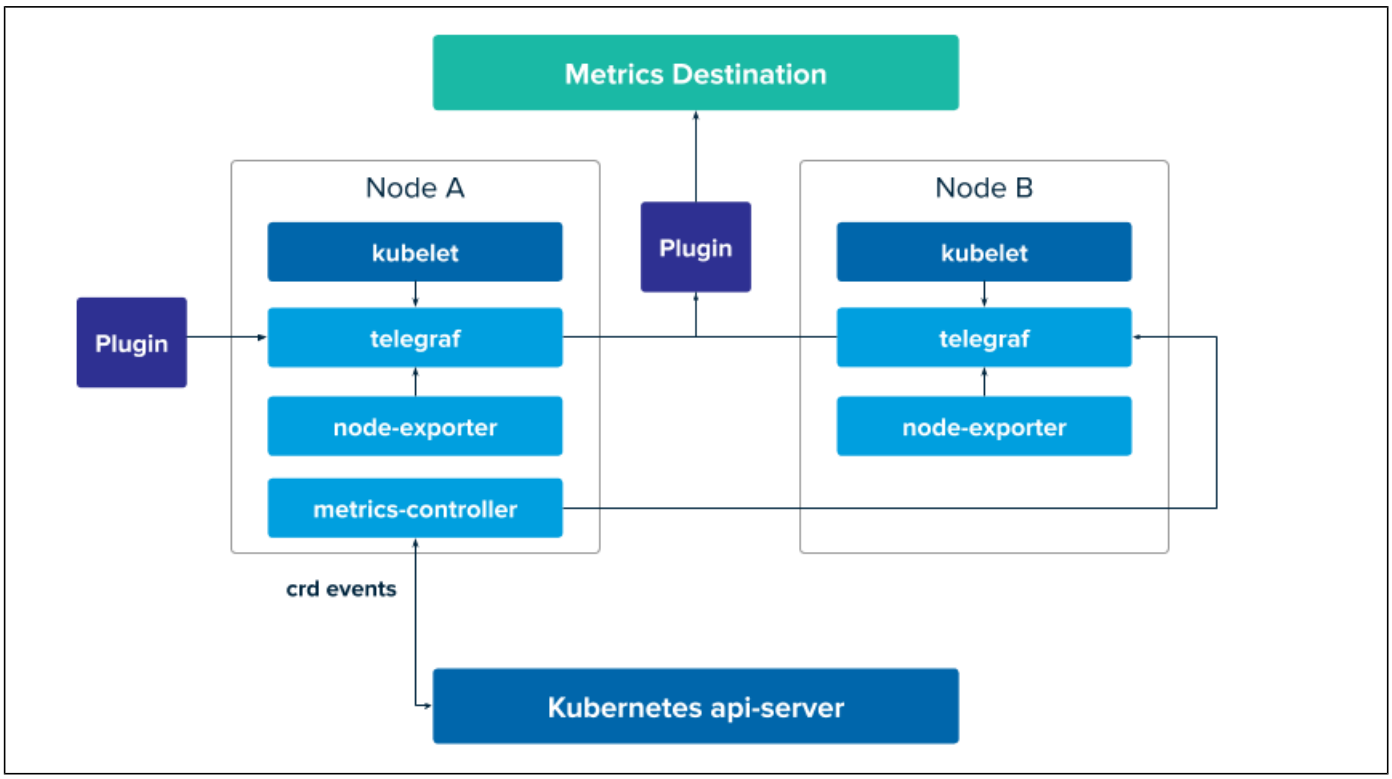
All aggregated log entries are marshaled to a common log destination.

Note: When sinks are added or removed, all of the Fluent Bit pods are refreshed with new sink information.

Metric Sink Architecture

The PKS metric sink aggregates workload metrics and forwards them to a common metrics destination.

The following diagram details PKS metric sink architecture:



A metric sink collects and writes metrics from a cluster to specified outputs using input and output plugins.

Workload metrics are monitored by a set of third-party plugins. The plugins forward the metrics to a Telegraf service pod.

A pair of kubelets monitors Kubernetes and forwards Kubernetes metrics to a pair of Telegraf service pods.

If Node Exporter is enabled on the worker nodes in the Enterprise PKS tile, a Node Exporter `DaemonSet` is included in all clusters. For more information about Node Exporter metrics, see the Node Exporter repository in GitHub.

To define the collected unstructured metrics, a metric-controller monitors Kubernetes for custom resource definitions and forwards those definitions to the Telegraf services.

The Telegraf services collect, process, and aggregate gathered metrics. All aggregated metrics are marshaled to an additional plugin for forwarding to a third-party application.

Note: When sinks are added or removed, all of the Telegraf pods are refreshed with new sink information.

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

Creating and Managing Sink Resources

In this topic

Page last updated:

This topic describes how to create and manage sink resources for a Kubernetes cluster provisioned with VMware Enterprise PKS (PKS), or for a namespace within a cluster.

Overview

Sinks collect logs and metrics about Kubernetes worker nodes in your PKS deployment and workloads that are running on them.

You can create two types of sinks:

- Log sinks
- Metric sinks

For more conceptual information about sinks, see Sink Architecture in Enterprise PKS.

Prerequisites

Before creating a sink resource:

1. Review Sink Types in *Sink Architecture in Enterprise PKS*.

2. Configure sink resources in the **Enterprise PKS** tile > **In-Cluster Monitoring**:

   - If you want to create a `ClusterLogSink` or `LogSink` resource, select the **Enable Log Sink Resources** checkbox.
   - If you want to create a `ClusterMetricSink` or `MetricSink` resource, select the **Enable Metric Sink Resources** checkbox.
   - If you want to use Node Exporter to send worker node metrics to metric sinks of kind `ClusterMetricSink` as described in Create a ClusterMetricSink Resource for Node Exporter Metricsbelow, select the **Enable node exporter on workers** checkbox.

     For more information about these configuration settings, see the PKS installation topic for your IaaS:

- Installing Enterprise PKS on vSphere
- Installing Enterprise PKS on vSphere with NSX-T Integration
- Installing Enterprise PKS on GCP
- Installing Enterprise PKS on AWS
- Installing Enterprise PKS on Azure

3. Install the Kubernetes CLI, `kubectl`. For installation instructions, see Installing the Kubernetes CLI.

Create Sinks

You can create log and metric sinks for clusters and namespaces.

- If you want to create a `ClusterLogSink` or `LogSink`, see ClusterLogSink and LogSink Resources below.
- If you want to create a `ClusterMetricSink` or `MetricSink`, see ClusterMetricSink and MetricSink Resources below.

ClusterLogSink and LogSink Resources

To create `ClusterLogSink` or `LogSink` resources, you can:

- Create a Syslog ClusterLogSink or LogSink Resource
- Create a Webhook ClusterLogSink or LogSink Resource
- Create a ClusterLogSink or LogSink Resource with a Fluent Bit Output Plugin

Note: Log sinks created in PKS do not support UDP connections.

Note: PKS requires a secure connection for log forwarding when using `ClusterLogSink` and `LogSink` resources of type `syslog` or `webhook`. To forward logs using an unsecured connection, see Unsecured ClusterLogSink and LogSink Log Forwarding below.

Create a Syslog ClusterLogSink or LogSink Resource

`ClusterLogSink` and `LogSink` resources of type `syslog` deliver logs using the TCP-based syslog protocol.

To define a syslog `ClusterLogSink` or `LogSink` resource, perform the following steps:

1. Create a YAML file that specifies your log destination in the following format:

```
apiVersion: pksapi.io/v1beta1
kind: YOUR-SINK-RESOURCE
metadata:
  name: YOUR-SINK
  namespace: YOUR-NAMESPACE
spec:
  type: syslog
  host: YOUR-LOG-DESTINATION
  port: YOUR-LOG-DESTINATION-PORT
  enable_tls: true
```

Where:

- `YOUR-SINK-RESOURCE` is the sink resource you want to create. This must be either `ClusterLogSink` or `LogSink`. For information about these sink resources, see Overview.
- `YOUR-SINK` is a name you choose for your sink.
- `YOUR-NAMESPACE` is the name of your namespace. Omit this line if creating `ClusterLogSink`.
- `YOUR-LOG-DESTINATION` is the URL or IP address of your log management service.
- `YOUR-LOG-DESTINATION-PORT` is the port number of your log management service.

> **Note:** `enable_tls` must be `true`.

2. Save the YAML file with an appropriate file name. For example, `my-cluster-log-sink.yml`.

3. Apply the `ClusterLogSink` or `LogSink` resource to your cluster by running the following command:

```
kubectl apply -f YOUR-SINK.yml
```

Where `YOUR-SINK.yml` is the name of your YAML file. For example:

```
$ kubectl apply -f my-cluster-log-sink.yml
```

Create a Webhook ClusterLogSink or LogSink Resource

`ClusterLogSink` and `LogSink` resources of type `webhook` batch logs into one-second units, wrap the resulting payload in JSON, and use the `POST` method to deliver the logs to the address of your log management service.

To define a webhook `ClusterLogSink` or `LogSink` resource, perform the following steps:

1. Create a YAML file that specifies your log destination in the following format:

```
apiVersion: pksapi.io/v1beta1
kind: YOUR-SINK-RESOURCE
metadata:
  name: YOUR-SINK
  namespace: YOUR-NAMESPACE
spec:
  type: webhook
  url: YOUR-LOG-DESTINATION
```

Where:

- `YOUR-SINK-RESOURCE` is the sink resource you want to create. This must be either `ClusterLogSink` or `LogSink`. For information about these sink resources, see Overview.
- `YOUR-SINK` is a name you choose for your sink.
- `YOUR-NAMESPACE` is the name of your namespace. Omit this line if creating `ClusterLogSink`.
- `YOUR-LOG-DESTINATION` is the URL or IP address of your log management service.

2. Save the YAML file with an appropriate filename. For example, `my-cluster-log-sink.yml`.

3. Apply the `ClusterLogSink` or `LogSink` resource to your cluster by running the following command:

```
kubectl apply -f YOUR-SINK.yml
```

Where `YOUR-SINK.yml` is the name of your YAML file. For example:

```
$ kubectl apply -f my-cluster-log-sink.yml
```

Create a ClusterLogSink or LogSink Resource with a Fluent Bit Output Plugin

`ClusterLogSink` and `LogSink` resources with a Fluent Bit output plugin deliver logs to the output plugin that you specify in your resource configuration.

To define a `ClusterLogSink` or `LogSink` resource with a Fluent Bit output plugin, perform the following steps:

1. Create a YAML file that specifies your log destination in the following format:

   ```
   apiVersion: pksapi.io/v1beta1
   kind: YOUR-SINK-RESOURCE
   metadata:
     name: YOUR-SINK
     namespace: YOUR-NAMESPACE
   spec:
     type: http
     output_properties:
       Host: example.com
       Format: json
       Port: 443
       tls: on
       tls.verify: off
   ```

   Where:

   - `YOUR-SINK-RESOURCE` is the sink resource you want to create. This must be either `ClusterLogSink` or `LogSink`. For information about these sink resources, see Overview.
   - `YOUR-SINK` is a name you choose for your log sink.
   - `YOUR-NAMESPACE` is the name of your namespace. Omit this line if creating `ClusterLogSink`.

   Note: This is a sample plugin configuration for `http`. For a full list of supported plugins, see the Fluent Bit documentation.

2. Save the YAML file with an appropriate filename. For example, `my-cluster-log-sink.yml`.

3. Apply the `ClusterLogSink` or `LogSink` resource to your cluster by running the following command:

   ```
   kubectl apply -f YOUR-SINK.yml
   ```

   Where `YOUR-SINK.yml` is the name of your YAML file. For example:

   ```
   $ kubectl apply -f my-cluster-log-sink.yml
   ```

Unsecured ClusterLogSink and LogSink Log Forwarding

By default, PKS requires a secure connection for log forwarding when using `ClusterLogSink` and `LogSink` resources of type `syslog` or `webhook`.

For debugging purposes on a local machine, you may want to temporarily forward logs using an unsecured connection. To do this, you must:

1. Disable sink forwarding validation by running the following command:

```
kubectl delete validatingwebhookconfigurations validator.pksapi.io
```

2. Set `enable_tls` to `false` in your log destination YAML file.

warning: Disabling secure log forwarding is not recommended.

ClusterMetricSink and MetricSink Resources

By default, a `ClusterMetricSink` resource collects metrics from a cluster using the Kubernetes Input Plugin and writes them to one or more outputs that you specify in your `ClusterMetricSink` configuration. Alternatively, you can use Node Exporter as your input plugin. To create a `ClusterMetricSink` resource, see:

- Create a ClusterMetricSink or MetricSink Resource. Follow these instructions if you want to use the default `ClusterMetricSink` configuration.
- Create a ClusterMetricSink Resource for Node Exporter Metrics. Follow these instructions if you want to use Node Exporter as your input plugin. For a list of Node Exporter metrics, see the Node Exporter GitHub repository.

A `MetricSink` resource collects metrics from a namespace within a cluster using `prometheus.io/scrape` annotations set to `true` and writes them to one or more outputs that you specify in your `MetricSink` configuration. To create a `MetricSink` resource, follow the instructions in Create a ClusterMetricSink or MetricSink Resource.

For a list of supported output plugins, see Output Plugins in the telegraf GitHub repository.

Create a ClusterMetricSink or MetricSink Resource

To define a `ClusterMetricSink` or `MetricSink` resource, perform the following steps:

1. Create a YAML file in the following format:

```
apiVersion: pksapi.io/v1beta1
kind: YOUR-SINK-RESOURCE
metadata:
  name: YOUR-SINK
  namespace: YOUR-NAMESPACE
spec:
  inputs:
  outputs:
  - type: YOUR-OUTPUT-PLUGIN
```

Where:

- `YOUR-SINK-RESOURCE` is the sink resource you want to create. This must be either `ClusterMetricSink` or `MetricSink`. For information about these sink resources, see Overview.
- `YOUR-SINK` is a name you choose for your sink.
- `YOUR-NAMESPACE` is the name of your namespace. Omit this line if creating `ClusterMetricSink`.
- `YOUR-OUTPUT-PLUGIN` is the name of the output plugin you want to use for your metrics.

Note: You can leave the `inputs` field blank. For `ClusterMetricSink`, this field is configured to include metrics from the kubelet by default. For `MetricSink`, the field includes all `prometheus.io/scrape` annotations set to `true` by default.

For example:

```
apiVersion: pksapi.io/v1beta1
kind: ClusterMetricSink
metadata:
  name: http
spec:
  inputs:
  outputs:
  - type: http
    url: https:example.com
    method: POST
    data_format: json
```

Create a ClusterMetricSink Resource for Node Exporter Metrics

To define a `ClusterMetricSink` resource for collecting Node Exporter metrics, perform the following steps:

1. Enable Node Exporter on your cluster workers by selecting the **Enable node exporter on workers** checkbox in the **Enterprise PKS** tile > **In-Cluster Monitoring**.

2. Create a YAML file in the following format:

```
apiVersion: pksapi.io/v1beta1
kind: ClusterMetricSink
metadata:
  name: YOUR-SINK
spec:
  inputs:
  - monitor_kubernetes_pods: true
    type: prometheus
  outputs:
  - type: YOUR-OUTPUT-PLUGIN
```

   Where:

   - `YOUR-SINK` is a name you choose for your sink.
   - `YOUR-OUTPUT-PLUGIN` is the name of the output plugin you want to use for your metrics.

   For example:

```
apiVersion: pksapi.io/v1beta1
kind: ClusterMetricSink
metadata:
  name: http
spec:
  inputs:
  - monitor_kubernetes_pods: true
    type: prometheus
  outputs:
  - type: http
    url: https:example.com
    method: POST
    data_format: json
```

3. Save the YAML file with an appropriate filename. For example, `my-cluster-metric-sink.yml`.

4. Apply the `ClusterMetricSink` resource to your cluster by running the following command:

```
kubectl apply -f YOUR-SINK.yml
```

Where `YOUR-SINK.yml` is the name of your YAML file. For example:

```
$ kubectl apply -f my-cluster-metric-sink.yml
```

Filter Sinks

The `LogSink` and `ClusterLogSink` resources allow users to set filters to include or exclude logs or events. For more information, see Monitoring Clusters with Log Sinks.

To filter log sinks, add a filter properties section to the YAML file that specifies your log destination in the following format:

```
apiVersion: pksapi.io/v1beta1
kind: YOUR-SINK-RESOURCE
metadata:
  name: YOUR-SINK
  namespace: YOUR-NAMESPACE
spec:
  type: syslog
  host: YOUR-LOG-DESTINATION
  port: YOUR-LOG-DESTINATION-PORT
  enable_tls: true
  filters:
   include-events: true
   include-logs: false
```

Where:

- `YOUR-SINK-RESOURCE` is the sink resource type that you created. This must be either `ClusterLogSink` or `LogSink`.
- `YOUR-SINK` is the name you chose for your sink.
- `YOUR-NAMESPACE` is the name of your namespace. Omit this line for `ClusterLogSink`.
- `YOUR-LOG-DESTINATION` is the URL or IP address of your log management service.
- `YOUR-LOG-DESTINATION-PORT` is the port number of your log management service.

The default values for these filter properties is **true**. If you do not specify filter properties, both logs and events are included.

List Sinks

To list sinks for clusters and namespaces, use the commands in the following sections.

ClusterLogSink and LogSink Resources

To list cluster log sinks, run the following command:

```
kubectl get clusterlogsinks
```

To list namespace log sinks, run the following command:

```
kubectl -n YOUR-NAMESPACE get logsinks
```

Where `YOUR-NAMESPACE` is the name of your namespace.

ClusterMetricSink and MetricSink Resources

To list cluster metric sinks, run the following command:

```
kubectl get clustermetricsinks
```

To list namespace metric sinks, run the following command:

```
kubectl -n YOUR-NAMESPACE get metricsinks
```

Where `YOUR-NAMESPACE` is the name of your namespace.

Delete Sinks

To delete sinks for clusters and namespaces, use the commands in the following sections.

ClusterLogSink and LogSink Resources

To delete a cluster log sink, run the following command:

```
kubectl delete clusterlogsink YOUR-SINK
```

Where `YOUR-SINK` is the name of your sink.

To delete a namespace log sink, run the following command:

```
kubectl -n YOUR-NAMESPACE delete logsink YOUR-SINK
```

Where:

- `YOUR-NAMESPACE` is the name of your namespace.
- `YOUR-SINK` is the name of your log sink.

ClusterMetricSink and MetricSink Resources

To delete a cluster metric sink, use the following command:

```
kubectl delete clustermetricsink YOUR-SINK
```

Where `YOUR-SINK` is the name of your sink.

To delete a namespace metric sink, use the following command:

```
kubectl -n YOUR-NAMESPACE delete metricsink YOUR-SINK
```

Where:

- `YOUR-NAMESPACE` is the name of your namespace.
- `YOUR-SINK` is the name of your metric sink.

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

Monitoring Clusters with Log Sinks

In this topic

Page last updated:

This topic describes the log sink resources you can use to monitor Kubernetes clusters provisioned by VMware Enterprise PKS and their workloads.

Overview

You can use the following sink resources to collect logs from your Kubernetes clusters.

Sink Resource	Sink Type	Description
`ClusterLogSink`	Log sink	Forwards logs from a cluster to a log destination. Logs are transported using one of the following: • The Syslog Protocol defined in RFC 5424 • WebHook • Fluent Bit output plugins
`LogSink`	Log sink	Forwards logs from a namespace within a cluster to a log destination. Logs are transported using one of the following: • The Syslog Protocol defined in RFC 5424 • WebHook • Fluent Bit output plugins

Log Sinks

`ClusterLogSink` and `LogSink` resources collect pod logs and events from the Kubernetes API in your Kubernetes clusters. For more information, see:

- Log Format
- Notable Kubernetes API Events

Log Format

In Enterprise PKS, you can create `ClusterLogSink` and `LogSink` resources of the following types:

- Syslog
- WebHook
- Fluent Bit output plugins

Your log format depends on the type of `ClusterLogSink` or `LogSink` you want to use. For example, if you use a `ClusterLogSink` or `LogSink` resource of type `syslog`, Enterprise PKS formats your logs as described in the sections below.

Syslog Format

All log entries collected by `ClusterLogSink` and `LogSink` resources of type `syslog` include a prefix in the following format:

```
APP-NAME/NAMESPACE/POD-ID
```

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.

Pod Logs

Pod logs are distinguished by the string `pod.log` in the `APP-NAME` field.

The following is a sample pod log entry:

```
36 <14>1 2018-11-26T18:51:41.647825+00:00 cluster-name
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:

- `cluster-name` is the human-readable cluster name used when creating the cluster.
- `pod.log` is the `APP-NAME`.
- `rocky-raccoon` is the `NAMESPACE`.
- `logspewer-6b58b6689d-dhddj` is the `POD-ID`.

Kubernetes API Events

Kubernetes API events are distinguished by the string `k8s.event` in the `APP-NAME` field.

The following is an example Kubernetes API event log entry:

```
Nov 14 16:01:49 cluster-name
k8s.event/rocky-raccoon/logspewer-6b58b6689d-j9n:
Successfully assigned rocky-raccoon/logspewer-6b58b6689d-j9nq7
to vm-38dfd896-bb21-43e4-67b0-9d2f339adaf1
```

Where:

- `cluster-name` is the human-readable cluster name used when creating the cluster.
- `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 Kubernetes scheduling problems in Enterprise 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 cluster-name 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 cluster-name 35.239.18.250 k8s.event/rocky-raccoon/logspewer-6b58b6689d/: Created pod: logspewer-6b58b6689d-sr96t`
	`Jan 25 09:14:55 cluster-name 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 cluster-name k8s.event/rocky-raccoon/logspewer-6b58b6689d-mkd: Killing container with id docker://logspewer:Need to kill Pod`
	`Jan 25 09:14:56 cluster-name k8s.event/rocky-raccoon/logspewer-6b58b6689d-sr9: Container image "oratos/logspewer:v0.1" already present on machine`
	`Jan 25 09:14:56 cluster-name k8s.event/rocky-raccoon/logspewer-6b58b6689d-sr9: Created container`
	`Jan 25 09:14:56 cluster-name 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 about log sinks, see:

- Creating and Managing Sink Resources
  Follow these instructions to create `ClusterLogSink` and `LogSink` resources, described in Overview above.
- Sink Architecture in Enterprise PKS
  See this topic for conceptual information about sinks.

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

Accessing Dashboard

In this topic

Page last updated:

This topic describes how to access Dashboard, a web-based Kubernetes UI, for your VMware Enterprise 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.
  ```

  Note: If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

- 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.

> **Note:** If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

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 kubeconfig token, paste your kubeconfig 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.

Viewing Usage Data

In this topic

Page last updated:

This topic describes how operators can view pod usage information from their VMware Enterprise PKS deployment. Operators can use this data to calculate billed usage, perform customer chargebacks, and generate usage reports.

The Enterprise 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 Enterprise PKS billing database. The agent pod collects usage data for the deployment and sends the data to the Enterprise PKS aggregator agent. The aggregator agent then stores the data in the Enterprise PKS database. You can access the Enterprise 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. Click the **Enterprise PKS** 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 API VM. For more information, see SSH into the PKS API VM in *Using BOSH Diagnostic Commands in Enterprise PKS*.

7. On the command line, log in to the billing database in one of the following ways:

   - Run the following command to connect by IP address from **inside the PKS API VM**:

     ```
     mysql -h 127.0.0.1 --port 13306 -u billing -p billing
     ```

   - Run the following command to connect using the MySQL socket file from **inside the PKS API VM**:

     ```
     mysql --socket=/var/vcap/sys/run/pxc-mysql/mysqld.sock -u billing -p billing
     ```

   - Run the following command to connect by IP address from **outside the PKS API VM**:

     ```
     mysql -h IP-ADDRESS --port 13306 -u billing -p billing
     ```

     Where `IP-ADDRESS` is the IP address that you located in a previous step.

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.

Backing up and Restoring Enterprise PKS

Page last updated:

This section describes how to back up and restore the VMware Enterprise PKS control plane and Enterprise PKS clusters. Enterprise PKS uses the Cloud Foundry BOSH Backup and Restore (BBR) framework to back up and restore the PKS control plane and clusters. For more information, see BOSH Backup and Restore.

BBR orchestrates triggering the backup or restore process on the BOSH deployment, and transfers the backup artifacts to and from the BOSH deployment.

BBR can back up the following components:

- BOSH Director
- Enterprise PKS control plane UAA MySQL database
- Enterprise PKS control plane PKS API MySQL database
- Enterprise PKS cluster data, from the clusters' ETCD databases

BBR cannot back up the following components:

- Harbor tile
- Persistent volumes attached to nodes
- Network resources. For example, load balancers to the cluster.

warning: BBR is not validated for backing up vSphere with NSX-T Objects.

For more information about installing and using BBR, see the following topics:

- Installing BOSH Backup and Restore
- Backing up Enterprise PKS
- Restore the BOSH Director
- Restore the Enterprise PKS Control Plane
- Restore Enterprise PKS Clusters

For information about troubleshooting BBR, see BBR Logging.

If you are using vSphere to run Enterprise PKS, also see:

- Overview of Backup and Restore options in vCenter Server 6.x (2149237) in the VMware documentation
- Backing Up and Restoring the NSX Manager in the VMware documentation

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

Installing BOSH Backup and Restore

In this topic

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 Enterprise PKS deployment.

For more information about using BOSH Backup and Restore, see:

- To perform a backup, see Backing up Enterprise PKS.
- To perform a restore of the BOSH Director, see Restore the BOSH Director.
- To perform a restore of the Enterprise PKS Control Plane, see Restore the Enterprise PKS Control Plane.
- To perform a restore of the Enterprise PKS Clusters. see Restore Enterprise 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 Enterprise PKS Release Notes. To download a BBR installation file, see BOSH Backup and Restore on the Pivotal Network.

Configure Your Jumpbox

Configure your jumpbox to meet the following requirements:

- Your jumpbox must be able to communicate with the network that contains your Enterprise 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 Enterprise PKS instances using port 22 by default.

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.

Backing Up PKS

In this topic

Page last updated:

This topic describes how to use BOSH Backup and Restore (BBR) to back up the VMware Enterprise PKS Control Plane and its cluster deployments.

Overview

The BOSH Director, Enterprise 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 Enterprise PKS Control Plane.
- To perform a restore of a cluster deployment, see Restore Enterprise PKS Clusters.

To view the BBR release notes, see the Cloud Foundry documentation, BOSH Backup and Restore Release Notes.

Recommendations

Pivotal recommends:

- Follow the full procedure documented in this topic when creating a backup. This ensures that you always have a consistent backup of Pivotal Operations Manager and Enterprise PKS to restore from.
- Back up frequently, especially before upgrading your Enterprise PKS deployment.
- For BOSH v270.0 and above (currently in Pivotal Platform 2.7), prune the BOSH blobstore by running `bosh clean-up --all` prior to running a backup of the BOSH director. This removes all unused resources, including packages compiled against older stemcell versions, which can result in a smaller, faster backup of the BOSH Director. For more information see the `clean-Up` command.

> **Note:** The command `bosh clean-up --all` is a destructive operation and can remove resources that are unused but needed. For example, if an On-Demand Service Broker such as Enterprise PKS is deployed **and** no service instances have been created, the releases needed to create a service instance will be categorized as unused and removed.

Supported Components

This section describes the components that are supported and not supported by BBR.

BBR can back up the following components:

- BOSH Director
- Enterprise PKS control plane UAA MySQL database
- Enterprise PKS control plane PKS API MySQL database
- Enterprise PKS cluster data, from the clusters' ETCD databases

BBR cannot back up the following components:

- Harbor tile
- Persistent volumes attached to nodes
- Network resources. For example, load balancers to the cluster.

warning: BBR is not validated for backing up vSphere with NSX-T Objects.

Preparing to Back Up

Before using BBR you must perform the following steps:

- Verify your BBR Version
- Retrieve the BBR SSH Credentials
- Retrieve the BOSH Director Credentials
- Retrieve the UAA Client Credentials
- Retrieve the BOSH Director Address
- Download the Root CA Certificate
- Retrieve 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 Enterprise PKS release.

1. For your current Enterprise PKS release's minimum version information, see the Enterprise 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 use BBR to perform a backup or restore, you will need to collect accounts and credentials to authenticate into your jumpbox, BOSH Director, and Ops Manager.

Retrieve 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
```

Retrieve 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 and record the value of the `password` 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 and record the value of the `password` field.

Retrieve 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 **Enterprise PKS** 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 Retrieve 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 Enterprise 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**.

Retrieve 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** > **Enterprise PKS** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
   - `USERNAME` is your user name.

   See Logging in to Enterprise PKS for more information about the `pks login` command.

   > **Note:** If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

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 Retrieve the BOSH Command Line Credentials.
    - `UUID` is the cluster UUID that you recorded in the previous step.

Back Up Enterprise PKS

To back up your Enterprise 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 JUMPBOX-USERNAME@JUMPBOX-ADDRESS
  ```

Where:

- `PATH-TO-KEY` is the local path to your private key for the jumpbox host.
- `JUMPBOX-USERNAME` is your jumpbox username.
- `JUMPBOX-ADDRESS` is the address of the 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-USERNAME@JUMPBOX-ADDRESS -i JUMPBOX-KEY-FILE -o ServerAliveInterval=60
     ```

     Where:

     - `SOCKS-PORT` is the local SOCKS port.
     - `JUMPBOX-USERNAME` is your jumpbox username.
     - `JUMPBOX-ADDRESS` is the address of the jumpbox.
     - `JUMPBOX-KEY-FILE` is the local SSH private key for accessing the 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-USERNAME@JUMPBOX-ADDRESS:SOCKS-PORT?private_key=JUMPBOX-KEY-FILE
     ```

     Where:

     - `JUMPBOX-USERNAME` is your jumpbox username.
     - `JUMPBOX-ADDRESS` is the address of the 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 Enterprise 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 Enterprise PKS BOSH Director

To back up BOSH Director you will validate your current configuration, then execute the `bbr` backup command.

Validate the Enterprise 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 Enterprise 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 Enterprise PKS Control Plane

To back up your Enterprise PKS Control Plane you will validate the Control Plane, then execute the `bbr` backup command.

Locate the Enterprise PKS Deployment Name

Locate and record your Enterprise 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 Enterprise 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 deployments | grep pivotal-container-se

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 Enterprise 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 Enterprise 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 Enterprise PKS BOSH deployment name that you located in the Locate the Enterprise 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 Enterprise 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 Enterprise PKS BOSH deployment name that you located in the Locate the Enterprise 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 Enterprise PKS 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.

Back Up vCenter, and NSX if Used (vSphere Only)

If your are running Enterprise PKS on vSphere with Flannel or NSX-T networking, you must back up your vCenter in addition to completing the BBR procedures above.

For Enterprise PKS deployments with NSX-T networking, you must also back up the NSX Manager.

To complete the backup of your Enterprise PKS environment running on vSphere:

1. Back up vCenter. See Overview of Backup and Restore options in vCenter Server 6.x (2149237) in the VMware documentation.

2. If you use NSX-T networking, back up the NSX Manager. See Backing Up and Restoring the NSX Manager in the VMware documentation.

After Backing Up Enterprise PKS

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 Enterprise PKS, test your backup artifact by following the procedures in:
    - Restore the Enterprise PKS BOSH Director
    - Restore the Enterprise PKS Control Plane
    - Restore Enterprise 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 Enterprise PKS BOSH deployment name that you located in the Locate the Enterprise 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

In this topic

Page last updated:

This topic describes how to use BOSH Backup and Restore (BBR) to restore the BOSH Director, and the VMware Enterprise 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 Enterprise PKS Ops Manager Installation settings, using your BBR backup artifacts.

Before restoring using BBR:

- Review the requirements listed in Compatibility of Restore below.
- Complete all of the steps documented in Preparing to Restore a Backup below.

Use BBR to restore the following:

- The BOSH Director plane, see Restore the BOSH Director below.

- The Enterprise PKS control plane, see Restore Enterprise PKS Control Plane below.
- The Enterprise PKS clusters, see Restore Enterprise PKS Clusters below.

If you are running Enterprise PKS on vSphere, you might also need to restore the following:

- vCenter, see Overview of Backup and Restore options in vCenter Server 6.x (2149237) in the VMware documentation.
- The NSX Manager, see Backing Up and Restoring the NSX Manager in the VMware documentation.

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
- Retrieve the BBR SSH Credentials
- Retrieve the BOSH Director Credentials
- Retrieve the UAA Client Credentials
- Retrieve the BOSH Director Address
- Download the Root CA Certificate
- Retrieve 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 Enterprise PKS release.

1. For your current Enterprise PKS release's minimum version information, see the Enterprise 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 use BBR to perform a backup or restore, you will need to collect accounts and credentials to authenticate into your jumpbox, BOSH Director, and Ops Manager.

Retrieve 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
```

Retrieve 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 and record the value of the `password` 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 and record the value of the `password` field.

Retrieve 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 **Enterprise PKS** 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 Retrieve 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 Enterprise 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**.

Retrieve 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** > **Enterprise PKS** > **PKS API** > **API Hostname (FQDN)**. For example, `api.pks.example.com`.
   - `USERNAME` is your user name.

   See Logging in to Enterprise PKS for more information about the `pks login` command.

> **Note:** If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

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 Retrieve 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, Enterprise 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 Enterprise 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 Enterprise PKS by following the installation instructions specific to your IaaS in Installing Enterprise 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 Enterprise 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 Enterprise 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 Enterprise PKS Control Plane

You must redeploy the Enterprise PKS tile before restoring the Enterprise PKS control plane. By redeploying the Enterprise PKS tile you create the VMs that constitute the control plane deployment.

To redeploy the Enterprise 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 Stemcell

Do either the following procedures to determine the stemcell that PKS uses:

- Review the Stemcell Library:

  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  -
...
```

Note: At most, the PKS tile can have two stemcells, where one stemcell is Linux and the other stemcell is Windows.

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 Enterprise 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 Enterprise 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 Enterprise PKS control plane perform the following steps:

1. Ensure the Enterprise 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 Enterprise PKS Clusters

After successfully restoring the Enterprise PKS Control Plane, perform the following steps to restore the Enterprise 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 section 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

In this topic

Page last updated:

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.

Enterprise PKS Security

Page last updated:

This section includes security topics for VMware Enterprise PKS.

See the following topic:

- Enterprise PKS Security Disclosure and Release Process

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

Enterprise PKS Security Disclosure and Release Process

In this topic

Page last updated:

This topic describes the processes for disclosing security issues and releasing related fixes for VMware Enterprise PKS, Kubernetes, Cloud Foundry Container Runtime (CFCR), VMware NSX, and VMware Harbor.

Security Issues in Enterprise PKS

Pivotal and VMware provide security coverage for Enterprise 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, Enterprise 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, Enterprise 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 Enterprise PKS

Page last updated:

This topic is intended to provide assistance when diagnosing and troubleshooting issues installing or using VMware Enterprise PKS.

See the following sections:

- General Troubleshooting
- Using BOSH Diagnostic Commands in Enterprise PKS
- Verifying Deployment Health
- Service Interruptions

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

General Troubleshooting

In this topic

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 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 **Enterprise PKS** 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 Using BOSH Diagnostic Commands in Enterprise PKSfor 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 Enterprise 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 Using BOSH Diagnostic Commands in Enterprise PKSfor 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.

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

Enterprise 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 Using BOSH Diagnostic Commands in Enterprise PKS.

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 Enterprise 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 Enterprise 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.

Explanation

The Kubernetes features and functions listed above are provided by the following Enterprise PKS add-ons:

- **Kubernetes Dashboard** `kubernetes-dashboard`
- **DNS Resolution**: `kube-dns`

To enable these add-ons, Ops Manager must run scripts after deploying Enterprise 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 **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. 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 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 about monitoring the **Upgrade all clusters errand** using the BOSH CLI, see Upgrade the PKS Tile in *Upgrading Enterprise PKS*.

Explanation

During the Enterprise 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. Kubernetes may be unable to unschedule the node if the `PodDisruptionBudget` object has been configured to permit zero 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 app. `PodDisruptionBudget` objects specify the amount of replicas, proportional to the total, that must be available in your app, regardless of downtime. Operators can configure `PodDisruptionBudget` objects for each app using their spec file.

Some apps deployed using Helm charts might have a default `PodDisruptionBudget` set. For more information on configuring `PodDisruptionBudget` objects using a spec file, see Specifying a PodDisruptionBudget in the Kubernetes documentation.

If `.spec.replicas` is configured correctly, you can also configure the default node drain behavior to prevent cluster upgrades from hanging or failing.

Solution

To resolve this issue, do one of the following:

- 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 Enterprise PKS, see Prepare to Upgrade in *Upgrading Enterprise PKS*.
- Configure the default node drain behavior by doing the following:

1. Navigate to **Ops Manager Installation** > **Enterprise PKS** > **Plans**.
2. Set the default node drain behavior by configuring the following fields:

Field	Instructions
Node Drain Timeout	Enter a timeout in minutes for the node to drain pods. You must enter a valid integer between `0` and `1440`. If you set this value to `0`, the node drain does not terminate.
Pod Shutdown Grace	Enter a timeout in seconds for the node to wait before it forces the pod to terminate. You must enter a valid integer between `-1` and `86400`. If you set this value to `-1`, the timeout is set to the default timeout specified by the pod.
Force node to drain even if it has running pods not managed by a ReplicationController, ReplicaSet, Job, DaemonSet or StatefulSet.	If you enable this configuration, the node still drains when pods are not managed by a ReplicationController, ReplicaSet, Job, DaemonSet or StatefulSet.
Force node to drain even if it has running DaemonSet-managed pods.	If you enable this configuration, the node still drains when pods are managed by a DeamonSet.
Force node to drain even if it has running running pods using emptyDir.	If you enable this configuration, the node still drains when pods are using a emptyDir volume.
Force node to drain even if pods are still running after timeout.	If you enable this configuration and then during the timeout pods fail to drain on the worker node, the node forces running pods to terminate and the upgrade or scale continues.

warning: If you select **Force node to drain even if pods are still running after timeout**, the node kills all running workloads on pods. Before enabling this configuration, set **Node Drain Timeout** to greater than `0`.

warning: If you deselect **Force node to drain even if it has running DaemonSet-managed pods** with **Enable Metric Sink Resources**, **Enable Log Sink Resources**, or **Enable Node Exporter** selected, the upgrade will fail as all options deploy a DaemonSet in the `pks-system` namespace.

3. Navigate to **Ops Manager Installation Dashboard** > **Review Pending Changes**, select **Upgrade all clusters errand**, and **Apply Changes**. The new behavior takes effect during the next upgrade, not immediately after applying your changes.

Note: You can also use the PKS CLI to configure node drain behavior. To configure the default node drain behavior with the PKS CLI, run `pks update-cluster` with an action flag. You can view the current node drain behavior with `pks cluster --details`. For more information, see Configure Node Drain Behavior in *Upgrade Preparation Checklist for Enterprise PKS v1.6*.

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.

   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 <cluster-name>
   ```

   Note: If your operator has configured Enterprise PKS to use a SAML identity provider, you must include an additional SSO flag to use the above command. For information about the SSO flags, see the section for the above command in PKS CLI. For information about configuring SAML, see Connecting Enterprise PKS to a SAML Identity Provider

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 Enterprise 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 Enterprise 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 fails with an error “Credentials were rejected, please try again.”

Explanation

You may experience this issue when a large number of pods are running continuously in your Enterprise PKS deployment. As a result, pod information increases the size of the Enterprise PKS database, and the persistent disk on the Pivotal Container Service VM runs out of space.

Solution

1. Check the total number of pods in your Enterprise 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 Enterprise 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 Pivotal Container Service VM as follows:

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

Error: Plan Not Found

Symptom

Plan not found error when an active plan is deactivated.

Explanation

You may receive the error “plan UUID not found” if, after creating a cluster using a plan (such as Plan 1), you then deactivate the plan (Plan 1) from the PKS Tile in Ops Manager and then **Save** and **Apply Changes** with the **Upgrade all clusters errand** selected.

Ops Manager does not have capability to check clusters that are using a particular plan. Only when user saves the plan, the deployment process will check whether a plan can be deactivated. The error message “plan is displayed in the Ops Manager logs.

Solution

1. Do not disable or deactivate a plan that is in use by or more clusters.
2. Run the command `pks cluster my-cluster --details` to view what plan the cluster is using.

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

Using BOSH Diagnostic Commands in Enterprise PKS

In this topic

Page last updated:

This topic describes how to access information about your VMware Enterprise PKS deployment by using the BOSH Command Line Interface (BOSH CLI).

Overview

BOSH diagnostic commands such as `bosh ssh` and `bosh vms` enable you to access information about your Enterprise PKS deployment. For example, you can access Enterprise PKS log files after SSHing into the PKS API or a Kubernetes cluster VM:

1. SSH into the PKS API VM or SSH into a Kubernetes Cluster VM
2. View Log Files

Log in to the BOSH Director VM

To set a BOSH alias for your Enterprise PKS environment and log in to the BOSH Director VM, follow the steps below:

1. Gather your credential and IP address information for the BOSH Director and SSH into the Ops Manager VM. For instructions, see Advanced Troubleshooting with the BOSH CLI.
2. To create a BOSH alias for your Enterprise 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
```

Alternatively, you can set the BOSH environment variables on the Ops Manager VM to authenticate with the BOSH Director VM. For more information, see Authenticate with the BOSH Director VM in *Advanced Troubleshooting with the BOSH CLI* in the Pivotal documentation.

SSH into the PKS API VM

To SSH into the PKS API VM using the BOSH CLI, follow the steps below:

1. Log in to the BOSH Director. For instructions, see Log in to the BOSH Director VM

2. To identify your PKS deployment name, run the following command:

   ```
   bosh -e ENVIRONMENT deployments
   ```

   Where `ENVIRONMENT` is your BOSH environment alias.

   For example:

   ```
   $ bosh -e pks deployments
   ```

   Your PKS deployment name begins with `pivotal-container-service` and includes a BOSH-generated identifier.

3. To identify your PKS API VM 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 API VM name begins with `pivotal-container-service` and includes a BOSH-generated identifier.

   Note: The PKS API VM identifier is different from the identifier in your PKS deployment name.

4. To SSH into the PKS API VM, run the following command:

   ```
   bosh -e ENVIRONMENT -d DEPLOYMENT ssh PKS-API-VM
   ```

   Where:

   - `ENVIRONMENT` is the BOSH environment alias.
   - `DEPLOYMENT` is your PKS deployment name.

- `PKS-API-VM` is your PKS API VM name.

For example:

```
$ bosh -e pks \
-d pivotal-container-service/a1b2c333d444e5f66a77 \
ssh pivotal-container-service/000a1111-222b-3333-4cc5-de66f7a8899b
```

SSH into a Kubernetes Cluster VM

Each Kubernetes cluster corresponds to a BOSH deployment. To SSH into a PKS-provisioned Kubernetes cluster VM using the BOSH CLI, follow the steps below:

1. Log in to the BOSH Director. For instructions, see Log in to the BOSH Director VM

2. To identify your Kubernetes cluster deployment name, run the following command:

   ```
   bosh -e ENVIRONMENT deployments
   ```

   Where `ENVIRONMENT` is your BOSH environment alias.

   For example:

   ```
   $ bosh -e pks deployments
   ```

   Kubernetes cluster deployment names begin with `service-instance` and include a BOSH-generated identifier.

3. To identify your Kubernetes cluster VM name, run the following command:

   ```
   bosh -e ENVIRONMENT -d DEPLOYMENT vms
   ```

   Where:

   - `ENVIRONMENT` is the BOSH environment alias.
   - `DEPLOYMENT` is your Kubernetes cluster deployment name.

   For example:

   ```
   $ bosh -e pks -d service-instance_ae681cd1-7ff4-4661-b12c-49a5b543f16f vms
   ```

   Each Kubernetes cluster VM name begins with `master` or `worker` and includes a BOSH-generated identifier.

4. To SSH into your Kubernetes cluster VM, run the following command:

   ```
   bosh -e ENVIRONMENT -d DEPLOYMENT ssh CLUSTER-VM
   ```

   Where:

   - `ENVIRONMENT` is the BOSH environment alias.
   - `DEPLOYMENT` is your Kubernetes cluster deployment name.
   - `CLUSTER-VM` is your Kubernetes cluster VM name, either `master/VM-ID` or `worker/VM-ID`.

   For example:

```
$ bosh -e pks -d service-instance_ae681cd1-7ff4-4661-b12c-49a5b543f16f ssh master/000a1111-222b-3333-4cc5-de66f7a8899b
```

View Log Files

Log files contain error messages and other information you can use to diagnose issues with your Enterprise PKS deployment. To access Enterprise PKS log files, SSH into the PKS API or a Kubernetes cluster VM and then follow the steps below:

1. To act as super user on your VM, run the following command:

   ```
   sudo su
   ```

2. Navigate to the `/var/vcap/sys/log` log file directory:

   ```
   cd /var/vcap/sys/log
   ```

3. Examine the contents of the `/var/vcap/sys/log` directory. For example, when diagnosing issues with a Kubernetes cluster VM, you may want to review the following log files:

   - On a master VM, examine the `kube-apiserver` subdirectory.
   - On a worker VM, examine the `kubelet` subdirectory.

Verifying Deployment Health

In this topic

Page last updated:

This topic describes how to verify the health of your VMware Enterprise PKS deployment.

Verify Kubernetes Node and Pod Health

Verify the health of your Kubernetes nodes and pods by following the steps below:

1. From the Ops Manager VM, run the following command:

   ```
   bosh -e ENVIRONMENT login
   ```

   Where `ENVIRONMENT` is the alias you set for your BOSH Director. For more information, see Using BOSH Diagnostic Commands in Enterprise PKS.

   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
   ```

Verify Kubernetes Cluster Health

Verify the health of your Kubernetes clusters by following the steps below:

1. From the Ops Manager VM, run the following command:

   ```
   bosh -e ENVIRONMENT login
   ```

   Where `ENVIRONMENT` is the alias you set for your BOSH Director. For more information, see Using BOSH Diagnostic Commands in Enterprise PKS.

   For example:

   ```
   $ bosh -e pks login
   ```

2. To get the deployment name of a target Kubernetes cluster, 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-7e64d53fc570503b5690
                                        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 cluster deployment name.

> **Note**: If you have deployed multiple Kubernetes clusters, determine the UUID using `pks clusters` and then match that UUID with the Kubernetes cluster deployment you are targeting.

3. With each cluster in a deployment, or any specific cluster, check the status of the cluster's VMs by running the following command:

```
bosh -d K8S-DEPLOYMENT vms
```

Where `K8S-DEPLOYMENT` is the name of your Kubernetes cluster deployment. Kubernetes cluster deployment names begin with `service-instance` and include a unique BOSH-generated identifier.
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
```

4. With each cluster in a deployment, or any specific cluster, check the status of the cluster's processes by running the following command:

```
bosh -d K8S-DEPLOYMENT instances --ps
```

Where `K8S-DEPLOYMENT` is the name of your Kubernetes cluster deployment. Kubernetes cluster deployment names begin with `service-instance` and include a unique BOSH-generated identifier. This command returns status information for the processes on each Kubernetes cluster VM, including each master and worker node.

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)

NSX Container Plugin (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.

To verify the `ncp` process is running, do the following:

1. Run `bosh instances` in your Enterprise PKS environment:

```
bosh -e ENVIRONMENT -d K8S-DEPLOYMENT instances --ps
```

Where:

- `ENVIRONMENT` is the alias you set for your BOSH Director.
- `K8S-DEPLOYMENT` is the name of your Kubernetes cluster deployment. Kubernetes cluster deployment names begin with `service-instance` and include a unique BOSH-generated identifier.

For example:

```
$ bosh -e pks -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      -       -
```

Alternatively:

1. SSH into your target Kubernetes master node VM:

   ```
   bosh -e ENVIRONMENT -d K8S-DEPLOYMENT ssh master/VM-ID
   ```

   Where:

   - `ENVIRONMENT` is the alias you set for your BOSH Director.
   - `K8S-DEPLOYMENT` is the name of your Kubernetes cluster deployment. Kubernetes cluster deployment names begin with `service-instance` and include a unique BOSH-generated identifier.
   - `VM-ID` is your Kubernetes master node VM ID. This is a unique BOSH-generated identifier.

   For example:

   ```
   $ bosh -e pks -d service-instance_8de000ff-a87a-4930-81ba-106d42c2471e ssh master/b6d3c263-1682-4c79-a9ab-35939127dedb
   ```

2. From the master node VM, run `monit summary`.

3. (Optional) To check if the `ncp` process on your target master node is active or on standby, run `/var/vcap/jobs/ncp/bin/nsxcli -c get ncp-master status`. This applies only to multi-master clusters.

For information about troubleshooting NCP, see NSX-T NCP troubleshooting and debug logging in the VMware Knowledge Base.

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

Service Interruptions

In this topic

Page last updated:

This topic describes events in the lifecycle of a Kubernetes cluster deployed by VMware Enterprise PKS that can cause temporary service interruptions.

Stemcell or Service Update

An operator performs a stemcell version update or Enterprise PKS version update.

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

An Enterprise 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 Enterprise PKS components offline.

Impact

The entire Enterprise 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 Restore Enterprise PKS Control Plane in *Restoring Enterprise PKS*.

PKS CLI

In this topic

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: 1.6.1

pks cancel-task

Cancel a task

Synopsis

Cancel a task

```
pks cancel-task <task-id> [flags]
```

Examples

```
pks cancel-task 0941fc83-b254-41a0-a505-14b04919e2cd
```


Options

```
-h, --help   help for cancel-task
```

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

```
    --details   Show details
-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 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 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

If OIDC is enabled and is not SSO, the password could also be set through environment variable: `PKS_USER_PASSWORD`

Use the `--sso` flag if PKS tile is configured with SAML `pks get-credentials <cluster-name> [flags]`

Examples

```
pks get-credentials my-cluster

pks get-credentials my-cluster --sso
```

Options

```
-h, --help                help for get-credentials
    --sso                 Prompt for a one-time passcode to do Single sign-on
    --sso-auto            Auto launch local browser to do Single sign-on
    --sso-passcode string Single sign-on with one-time passcode
```

pks get-kubeconfig

Allows you to get kubeconfig for your username

Synopsis

Run this command in order to get a kubeconfig file so you can access the cluster through kubectl. Typically your kubeconfig will need to be updated based on any new role bindings you have been granted.

Use the `--sso` flag if PKS tile is configured with SAML `pks get-kubeconfig <cluster-name> -u username -p password -a api [flags]`

Examples

```
pks get-kubeconfig my-cluster -u username -p password -a 192.168.1.1


pks get-kubeconfig my-cluster --sso -a 192.168.1.1
```

Options

```
-a, --api string            API
    --ca-cert string        Path to CA Cert for PKS API
-h, --help                  help for get-kubeconfig
-p, --password string       Password
-k, --skip-ssl-validation   Skip SSL Validation
    --sso                   Prompt for a one-time passcode to do Single sign-on
    --sso-auto              Auto launch local browser to do Single sign-on
    --sso-passcode string   Single sign-on with one-time passcode
-u, --username string       Username
```

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

Use the `--sso` flag if PKS tile is configured with SAML `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]

pks login -a <API> --sso [--ca-cert <PATH TO CERT> | -k]

pks login -a <API> --sso-auto [--ca-cert <PATH TO CERT> | -k]

pks login -a <API> --sso-passcode <sso-passcode> [--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)
    --sso                     Prompt for a one-time passcode to do Single sign-on
    --sso-auto                Auto launch local browser to do Single sign-on
    --sso-passcode string     Single sign-on with one-time passcode
-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 task

View a task

Synopsis

View a status and details of a task

```
pks task <task-id> [flags]
```

Examples

```
pks task 0941fc83-b254-41a0-a505-14b04919e2cd
```

Options

```
-h, --help   help for task
    --json   Return the PKS-API output as json
```

pks update-cluster

Updates the configuration of a specific kubernetes cluster

Synopsis

Update-cluster requires a target cluster name and at least 1 valid action flag (e.g. –num-nodes). Update-cluster will update the cluster settings based on the passed flag values and all updated values will persist through cluster upgrades.

```
pks update-cluster <cluster-name> [flags]
```

Examples

```
pks update-cluster my-cluster --num-nodes 5
```

Options

```
    --network-profile string                  Action flag, Network profile name
    --num-nodes int32                         Action flag, Number of worker nodes
    --kubelet-drain-timeout string            Action flag, The length of time in minutes for drain to wait before giving up.
    --kubelet-drain-grace-period string       Action flag, Period of time in seconds given to each pod to terminate gracefully.
    --kubelet-drain-force string              Action flag, Force drain even if there are pods not managed by a ReplicationController, ReplicaSet, Job, DaemonSet or StatefulSet.
    --kubelet-drain-ignore-daemonsets string  Action flag, Ignore DaemonSet managed pods during drain.
    --kubelet-drain-delete-local-data string  Action flag, Drain even if there are pods using emptyDir.
    --kubelet-drain-force-node string         Action flag, Forcefully terminate pods which fail to drain. Use it with caution.
    --non-interactive                         Don't ask for user input
    --json                                    Return the PKS-API output as json
    --wait                                    Wait for the operation to finish
-h, --help                                    help for update-cluster
```

pks upgrade-cluster

Upgrades the kubernetes cluster

Synopsis

Upgrades the kubernetes cluster listed. You need to provide a single cluster name

```
pks upgrade-cluster <cluster-one> [flags]
```

Examples

```
pks upgrade-cluster <one-cluster>
```

Options

```
 -h, --help              help for upgrade-cluster
     --json              Return the PKS-API output as json
     --non-interactive   Don't ask for user input
     --wait              Wait for the operation to finish
```

pks upgrade-clusters

Upgrades the kubernetes clusters

Synopsis

Upgrades the kubernetes clusters listed. Specify clusters or canaries with a comma separated list of names.

```
pks upgrade-clusters [flags]
```

Examples

```
pks upgrade-clusters --clusters <cluster-1>,<cluster-2>,<cluster-3> --canaries <cluster-3>,<cluster-4> --max-in-flight 2
```

Options

```
     --canaries string       Optional, list of clusters to be treated as canaries. Will upgrade sequentially before other clusters. Should be a comma separated list of names.
 -c, --clusters string       List of clusters to be upgraded. Should be a comma separated list of names.
 -h, --help                  help for upgrade-clusters
     --json                  Return the PKS-API output as json
     --max-in-flight int32   Optional, number of clusters to be upgraded in parallel (default 1)
     --non-interactive       Don't ask for user input
     --wait                  Wait for the operation to finish
```

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