Tanzu Kubernetes Grid Integrated Edition (TKGI) v1.8

Documentation

v1.8

Published: June 30, 2020

Note: The contents of this PDF may have fallen out of date. For current documentation, see https://docs.pivotal.io/pks/1-8
VMware Tanzu Kubernetes Grid Integrated Edition

Overview

Tanzu Kubernetes Grid Integrated Edition deploys Kubernetes to BOSH and Ops Manager, and uses the On-Demand Broker to dynamically instantiate, deploy, and manage highly-available Kubernetes clusters on-premises or on a public cloud.

After operators install TKGI, developers can use the TKGI Command Line Interface (TKGI CLI) to provision Kubernetes clusters, and run container-based workloads on the clusters with the Kubernetes CLI, kubectl.

Operators install TKGI as a tile on the Ops Manager Installation Dashboard, or from the TKGI Management Console on vSphere.

You can run TKGI standalone or alongside VMware Tanzu Application Service for VMs on Ops Manager.

What Tanzu Kubernetes Grid Integrated Edition Adds to Kubernetes

The following table details the features that Tanzu Kubernetes Grid Integrated Edition adds to the Kubernetes platform.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Included in K8s</th>
<th>Included in Tanzu Kubernetes Grid Integrated Edition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single tenant ingress</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Secure multi-tenant ingress</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Stateful sets of pods</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Multi-container pods</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Rolling upgrades to pods</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Rolling upgrades to cluster infrastructure</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Pod scaling and high availability</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Cluster provisioning and scaling</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Monitoring and recovery of cluster VMs and processes</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Features

Tanzu Kubernetes Grid Integrated Edition 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

Tanzu Kubernetes Grid Integrated Edition Prerequisites

For information about the resource requirements for installing Tanzu Kubernetes Grid Integrated Edition, see the topic that corresponds to your cloud provider:

- vSphere Prerequisites and Resource Requirements
- vSphere with NSX-T Version Requirements and Hardware Requirements for Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T
- GCP Prerequisites and Resource Requirements
- AWS Prerequisites and Resource Requirements
- Azure Prerequisites and Resource Requirements

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

In this topic
TKGI v1.8.0
   Product Snapshot
   Upgrade Path
   Features
   Bug Fixes
   Known Issues
TKGI Management Console 1.8.0
   Features
   Product Snapshot
   Upgrade Path
   Known Issues

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic contains release notes for Tanzu Kubernetes Grid Integrated Edition (TKGI) v1.8.

⚠️ warning: Before installing or upgrading to Tanzu Kubernetes Grid Integrated Edition v1.8, review the Breaking Changes below.

TKGI v1.8.0

Release Date: June 30, 2020

Product Snapshot

<table>
<thead>
<tr>
<th>Release</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>v1.8.0</td>
</tr>
<tr>
<td>Release date</td>
<td>June 30, 2020</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kubernetes</td>
<td>v1.17.5</td>
</tr>
<tr>
<td>Docker</td>
<td>v19.03.5</td>
</tr>
<tr>
<td>On-Demand Broker</td>
<td>v0.38.0</td>
</tr>
<tr>
<td>CoreDNS</td>
<td>v1.6.2</td>
</tr>
<tr>
<td>NCP</td>
<td>v3.0.1</td>
</tr>
<tr>
<td>UAA</td>
<td>v74.5.15</td>
</tr>
</tbody>
</table>
Percona XtraDB Cluster (PXC) | v0.22.0
---|---
Metrics Server | v0.3.6
etcd | v3.4.3
kubor-release | v1.8.0

### Compatibilities

<table>
<thead>
<tr>
<th>Ops Manager</th>
<th><strong>Versions</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AWS, Azure, GCP</strong>: See <a href="#">VMware Tanzu Network</a></td>
<td></td>
</tr>
<tr>
<td><strong>vSphere v7.0</strong>: Ops Manager v2.9.3+</td>
<td></td>
</tr>
<tr>
<td><strong>vSphere v6.7 or v6.5</strong>: Ops Manager v2.9.3+, v2.8.2+, v2.7.15+</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>vSphere*</th>
<th>See <a href="#">VMware Product Interoperability Matrices</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>NSX-T</td>
<td>v3.0†, v2.5.1, v2.5.0</td>
</tr>
<tr>
<td>Xenial stemcells‡</td>
<td>See <a href="#">VMware Tanzu Network</a></td>
</tr>
<tr>
<td>Windows stemcells</td>
<td><strong>vSphere v7.0</strong>: v2019.15</td>
</tr>
<tr>
<td></td>
<td><strong>vSphere v6.7 or v6.5</strong>: v2019.15 and later</td>
</tr>
<tr>
<td>Harbor</td>
<td>v2.0, v1.10.3</td>
</tr>
<tr>
<td>CNS for vSphere</td>
<td>v1.0.2</td>
</tr>
<tr>
<td>Backup and Restore SDK</td>
<td>v1.18.0</td>
</tr>
</tbody>
</table>

* Excluding VCF 4; see VCF 4 and Converged VDS v7 Not Supported in TKGI v1.8

† TKGI supports NSX-T v3.0 as a beta integration. Upgrading NSX-T to v3.0 is not recommended for production or large-scale TKGI environments. For more information about NSX-T v3.0 support, see NSX-T v3.0 Compatibility below.

‡ See TKGI v1.8 With NSX-T and NCP v3.0.1 Not Compatible With Xenial Stemcells 621.76 and Later

### Upgrade Path

The supported upgrade paths to Tanzu Kubernetes Grid Integrated Edition v1.8.0 are from Enterprise PKS v1.7.0 and later patches.

### Features

This section describes new features and changes in VMware Tanzu Kubernetes Grid Integrated Edition v1.8.0.

#### Enterprise PKS Renamed to Tanzu Kubernetes Grid Integrated Edition

Enterprise PKS has been renamed to Tanzu Kubernetes Grid Integrated Edition (TKGI).

**What has changed:**

- The Tanzu Kubernetes Grid Integrated Edition v1.8 tile uses the new name.
- Tanzu Kubernetes Grid Integrated Edition v1.8 includes two downloads of the CLI, the TKGI CLI and PKS CLI. See PKS CLI Renamed to TKGI CLI below.

**What has **not** changed:**

- Internal components continue to use the old name and its alternatives, such as `PKS`, `pks`, and `pivotal-container-service`. This includes, but is not limited to, BOSH names, UAA roles, and text strings containing the product name in TKGI components and...
TKGI-provisioned clusters.

If you intend to continue using the PKS CLI in TKGI v1.8, no action is required. However, future releases of TKGI will deprecate and remove the PKS CLI.

PKS CLI Renamed to TKGI CLI

To support the product name change, Tanzu Kubernetes Grid Integrated Edition v1.8 is distributed with a TKGI CLI in addition to a PKS CLI.

Both CLIs work identically and accept the same commands and arguments. To run a TKGI CLI command, substitute `tkgi` where you previously used `pks`. For more information, see TKGI CLI.

To download the TKGI CLI or the PKS CLI, see VMware Tanzu Network.

vSphere v7 Compatibility

TKGI v1.8 can run on vSphere v7.

NSX-T v3.0 Compatibility

On vSphere, TKGI can run with NSX-T v3.0 container networking.

**warning:** TKGI supports NSX-T v3.0 as a beta integration. Intermittent upgrade failures and scale problems may occur if you upgrade to NSX-T v3.0. Upgrading your NSX-T environment to v3.0 in a production or large-scale deployment is not recommended until a patch resolving these issues has been released.

TKGI Control Plane and API

- The TKGI API VM no longer stores a copy of the control plane database that the v1.7 upgrade migrated to the Database VM. This deletion frees internal memory in the TKGI API VM. As a result, users may notice improved control plane performance.
- The PKS 1.7.x Upgrade - MySQL Clone errand has been removed from the TKGI tile Errands pane.

Kubernetes Control Plane

- On Azure, TKGI supports disabling the creation of a default outbound SNAT rule for clusters. See Kubernetes Cloud Provider for how to disable the default SNAT rule.

TKGI Monitoring and Logging

- All TKGI components use TLS v1.2 with strong ciphers, including the metrics-server component. `sslscan` on a `metrics-server` over port `443` now reports only TLS v1.2+ ciphers.

Customer Experience Improvement Program (CEIP) and Telemetry

- The legacy Telemetry DB has been removed from the TKGI Database.

Component Updates

The following components have been updated:
- Bumps Kubernetes to v1.17.5.
- Bumps NCP to v3.0.1.
- Bumps UAA to v74.5.15.

Bug Fixes

TKGI v1.8.0 includes the following bug fixes:

- `tkgi tasks` returns valid output for all clusters.
- `tkgi upgrade-cluster` errand no longer times out when stopping `dockerd` processes.
- `tkgi get-credentials` works for clusters that have not been upgraded.
- `tkgi update-cluster` retains the `compute_profile` value when changing settings for clusters created with a Compute Profile.

Known Issues

TKGI v1.8.0 has the following known issues:

VCF 4 and Converged VDS v7 Not Supported in TKGI v1.8

For installations on vSphere v7 with NSX-T v3.0 integration, TKGI v1.8 supports only N-VDS for NSX-T traffic. It does not support:

- Converged Virtual Distributed Switch (C-VDS) v7, which lets you use the same VDS for both vSphere and NSX-T traffic
- VMware Cloud Foundation (VCF) v4.x, which uses only VDS mode with NSX-T v3.0

For more information, see Configure vSphere Networking for ESXi Hosts in Installing and Configuring NSX-T Data Center v3.0 for Tanzu Kubernetes Grid Integrated Edition.

TKGI v1.8 With NSX-T and NCP v3.0.1 Not Compatible With Xenial Stemcells 621.76 and Later

TKGI with NSX-T and NCP v3.0.1 is compatible with Linux Ubuntu Xenial stemcell v621.75, but not with stemcell versions v621.76 and later.

TKGI v1.8 (Windows) on vSphere Not Compatible with Ops Manager v2.9

TKGI v1.8 installations with Windows worker-based Kubernetes clusters on vSphere (Flannel) are not compatible with Ops Manager v2.9. If you do not intend to deploy and run Windows worker-based Kubernetes clusters, you can use Ops Manager v2.9 with TKGI v1.8.

For Ops Manager compatibility information, see VMware Tanzu Network.

Pinging Windows Workers Does Not Work

TKGI-provisioned Windows workers inherit a Kubernetes limitation that prevents outbound ICMP communication from workers. As a result, pinging Windows workers does not work.

For information about this limitation, see Limitations > Networking in the Windows in Kubernetes documentation.

TMC Integration Not Supported on GCP

TKGI on Google Cloud Platform (GCP) does not support Tanzu Mission Control integration, which is configured in the Tanzu

Copyright © 2020 VMware, Inc. All Rights Reserved.
Kubernetes Grid Integrated Edition tile > the Tanzu Mission Control (Experimental) pane.

If you intend to run TKGI v1.8 on GCP, skip this pane when configuring the Tanzu Kubernetes Grid Integrated Edition tile.

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 Knowledge Base.

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 TKGI, 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.

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.
Difficulty Changing Proxy for Windows Workers

You must configure a global proxy in the Tanzu Kubernetes Grid Integrated Edition tile > Networking pane before you create any Windows workers that use the proxy.

You cannot change the proxy configuration for Windows workers in an existing cluster.

Character Limitations in HTTP Proxy Password

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

TKGI Management Console 1.8.0

Release Date: June 30, 2020

Features

Tanzu Kubernetes Grid Integrated Edition Management Console v1.8.0 updates include:

- Support for vSphere 7
- Support for NSX-T 3.0
- Rebranding to Tanzu Kubernetes Grid Integrated Edition Management Console
- Specify FQDN for the Ops Manager VM during upgrade

Product Snapshot

Note: Tanzu Kubernetes Grid Integrated Edition Management Console provides an opinionated installation of TKGI. The supported versions may differ from or be more limited than what is generally supported by TKGI.

<table>
<thead>
<tr>
<th>Element</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>v1.8.0</td>
</tr>
<tr>
<td>Release date</td>
<td>June 30, 2020</td>
</tr>
<tr>
<td>Installed Tanzu Kubernetes Grid Integrated Edition version</td>
<td>v1.8.0</td>
</tr>
<tr>
<td>Installed Ops Manager version</td>
<td>v2.9.0</td>
</tr>
<tr>
<td>Installed Kubernetes version</td>
<td>v1.17.5</td>
</tr>
<tr>
<td>Compatible NSX-T versions</td>
<td>v3.0, v2.5.1, v2.5.0</td>
</tr>
<tr>
<td>Installed Harbor Registry version</td>
<td>v2.0, v1.10.3</td>
</tr>
<tr>
<td>Windows stemcells</td>
<td>v2019.20 and later</td>
</tr>
</tbody>
</table>

Upgrade Path

The supported upgrade path to Tanzu Kubernetes Grid Integrated Edition Management Console v1.8.0 is from Tanzu Kubernetes Grid Integrated Edition v1.7.0 and later.

Known Issues
The Tanzu Kubernetes Grid Integrated Edition Management Console v1.8.0 has the following known issues:

vSphere HA causes Management Console ovfenv Data Corruption

**Symptom**

If you enable vSphere HA on a cluster, if the TKGI Management Console appliance VM is running on a host in that cluster, and if the host reboots, vSphere HA recreates a new TKGI Management Console appliance VM on another host in the cluster. Due to an issue with vSphere HA, the ovfenv data for the newly created appliance VM is corrupted and the new appliance VM does not boot up with the correct network configuration.

**Workaround**

- In the vSphere Client, right-click the appliance VM and select **Power > Shut Down Guest OS**.
- Right-click the appliance again and select **Edit Settings**.
- Select **VM Options** and click **OK**.
- Verify under Recent Tasks that a **Reconfigure virtual machine** task has run on the appliance VM.
- Power on the appliance VM.

Base64 encoded file arguments are not decoded in Kubernetes profiles

**Symptom**

Some file arguments in Kubernetes profiles are base64 encoded. When the management console displays the Kubernetes profile, some file arguments are not decoded.

**Workaround**

Run `echo "$content" | base64 --decode`

Network profiles not immediately selectable

**Symptom**

If you create network profiles and then try to apply them in the Create Cluster page, the new profiles are not available for selection.

**Workaround**

Log out of the management console and log back in again.

Real-Time IP information not displayed for network profiles

**Symptom**

In the cluster summary page, only default IP pool, pod IP block, node IP block values are displayed, rather than the real-time values from the associated network profile.

**Workaround**
None

Please send any feedback you have to pks-feedback@pivotal.io.
Tanzu Kubernetes Grid Integrated Edition Concepts

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes VMware Tanzu Kubernetes Grid Integrated Edition concepts. See the following sections:

- Tanzu Kubernetes Grid Integrated Edition Architecture
- About Tanzu Kubernetes Grid Integrated Edition Upgrades
- TKGI API Authentication
- Load Balancers in Tanzu Kubernetes Grid Integrated Edition
- VM Sizing for Tanzu Kubernetes Grid Integrated Edition Clusters
- Telemetry
- Sink Architecture in Tanzu Kubernetes Grid Integrated Edition

Please send any feedback you have to pks-feedback@pivotal.io.
This topic describes how VMware Tanzu Kubernetes Grid Integrated Edition manages the deployment of Kubernetes clusters.

**Tanzu Kubernetes Grid Integrated Edition Overview**

An Tanzu Kubernetes Grid Integrated Edition environment consists of a TKGI Control Plane and one or more workload clusters. Tanzu Kubernetes Grid Integrated Edition administrators use the TKGI Control Plane to deploy and manage Kubernetes clusters. The workload clusters run the apps pushed by developers.

The following illustrates the interaction between Tanzu Kubernetes Grid Integrated Edition components:

---

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.
Administrators access the TKGI Control Plane through the TKGI Command Line Interface (TKGI CLI) installed on their local workstations.

Within the TKGI Control Plane the TKGI API and TKGI Broker use BOSH to execute the requested cluster management functions. For information about the TKGI Control Plane, see TKGI Control Plane Overview below. For instructions on installing the TKGI CLI, see Installing the TKGI CLI.

Kubernetes deploys and manages workloads on Kubernetes clusters. Administrators use the Kubernetes CLI, `kubectl`, to direct Kubernetes from their local workstations. For information about `kubectl`, see Overview of kubectl in the Kubernetes documentation.

TKGI Control Plane Overview


The control plane provides the following via the TKGI API:

- 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 TKGI Control Plane can upgrade all existing clusters using the Upgrade all clusters BOSH errand. For more information, see Upgrade Kubernetes Clusters in Upgrading Tanzu Kubernetes Grid Integrated Edition (Flannel Networking).

TKGI Control Plane is hosted on a pair of VMs:

• The TKGI API VM hosts cluster management services.
• The TKGI Database VM stores cluster management data.

TKGI API VM

The TKGI API VM hosts the following services:

• User Account and Authentication (UAA)
• TKGI API
• TKGI Broker
• Billing and Telemetry

The following sections describe UAA, TKGI API, and TKGI Broker services, the primary services hosted on the TKGI API VM.

UAA

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

UAA must be configured with the appropriate users and user permissions. For more information, see Managing Tanzu Kubernetes Grid Integrated Edition Users with UAA.

TKGI API

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

On AWS, GCP, and vSphere without NSX-T deployments the TKGI CLI communicates with the TKGI API within the control plane via the TKGI API Load Balancer. On vSphere with NSX-T deployments the TKGI API host is accessible via a DNAT rule. For information about enabling the TKGI API on vSphere with NSX-T, see the Share the TKGI API Endpoint section in Installing Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T Integration.

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

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

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

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

TKGI Database VM

The TKGI Database VM hosts MySQL, proxy, and other data-related services. These data-related functions persist TKGI Control Plane data for the following services:

- TKGI API
- UAA
- Billing
- Telemetry

Availability Zones

Tanzu Kubernetes Grid Integrated Edition uses Availability Zones (AZs) to provide high availability for Kubernetes cluster workers.

When an operator creates Plans for developers, they assign AZs to the Plans. Assigning multiple AZs to a Plan allows developers to provide high-availability for their worker clusters. When a cluster has more than one node, Ops Manager balances those nodes across the Availability Zones assigned to the cluster.

Public-cloud IaaSes such as AWS and Azure provide AZs as part of their service. In vSphere with NSX-T, you define and create AZs using vCenter clusters and resource pools. See Step 4: Create Availability Zones in Configuring BOSH Director with NSX-T for Tanzu Kubernetes Grid Integrated Edition for how to create AZs in NSX-T.

For instructions on selecting AZs for your Tanzu Kubernetes Grid Integrated Edition Plans, see Plans in Installing Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T.

For instructions on selecting the AZ for the Tanzu Kubernetes Grid Integrated Edition control plane, see Assign AZs and Networks in Installing Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T.

Windows Worker-Based Kubernetes Cluster (Beta) High Availability

Windows worker-based cluster (beta) Linux nodes can be configured in either standard or high availability modes.

- In standard mode, a single Master/etcd node and a single Linux worker manage a cluster’s Windows Kubernetes VMs.
- In high availability mode, multiple Master/etcd and Linux worker nodes manage a cluster’s Windows Kubernetes VMs.

The following illustrates the interaction between the Tanzu Kubernetes Grid Integrated Edition Management Plane and Windows worker-based Kubernetes clusters:
To configure Tanzu Kubernetes Grid Integrated Edition Windows worker-based clusters for high availability, set these fields in the Plan pane as described in Plans in Configuring Windows Worker-Based Kubernetes Clusters (Beta):

- **Enable HA Linux workers**
- **Master/ETCD Node Instances**
- **Worker Node Instances**

Please send any feedback you have to pks-feedback@pivotal.io.
About Tanzu Kubernetes Grid Integrated Edition Upgrades

In this topic

Overview
Deciding Between Full and Two-Phase Upgrade
What Happens During Full TKGI and TKGI Control Plane Upgrades
What Happens During Cluster Upgrades

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic provides conceptual information about Tanzu Kubernetes Grid Integrated Edition upgrades, including upgrading the TKGI control plane and TKGI-provisioned Kubernetes clusters.

For step-by-step instructions on upgrading Tanzu Kubernetes Grid Integrated Edition and TKGI-provisioned Kubernetes clusters, see:

- Upgrading Tanzu Kubernetes Grid Integrated Edition (Flannel Networking)
- Upgrading Tanzu Kubernetes Grid Integrated Edition (NSX-T Networking)
- Upgrading Clusters

Overview

An Tanzu Kubernetes Grid Integrated Edition upgrade modifies the version of Tanzu Kubernetes Grid Integrated Edition, for example, from v1.7.x to v1.8.0 or from v1.8.0 to v1.8.1.

By default, Tanzu Kubernetes Grid Integrated Edition is set to perform a full upgrade, which upgrades both the TKGI control plane and all TKGI-provisioned Kubernetes clusters.

However, you can choose to upgrade Tanzu Kubernetes Grid Integrated Edition in two phases by upgrading the TKGI control plane first and then upgrading your TKGI-provisioned Kubernetes clusters later.

Both the full upgrade and the TKGI control plane upgrade are performed through the Tanzu Kubernetes Grid Integrated Edition tile only. When upgrading TKGI-provisioned Kubernetes clusters, you can use either the Tanzu Kubernetes Grid Integrated Edition tile or the TKGI CLI. See the table below.

<table>
<thead>
<tr>
<th>Upgrade type</th>
<th>Upgrade method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TKGI Tile</td>
</tr>
<tr>
<td>Full TKGI upgrade</td>
<td>✔️</td>
</tr>
<tr>
<td>TKGI control plane only</td>
<td>✔️</td>
</tr>
<tr>
<td>Kubernetes clusters only</td>
<td>✔️</td>
</tr>
</tbody>
</table>

Typically, if you choose to upgrade TKGI-provisioned Kubernetes clusters only, you will upgrade them through the TKGI CLI.
Deciding Between Full and Two-Phase Upgrade

When deciding whether to perform the default full upgrade or to upgrade the TKGI control plane and TKGI-provisioned Kubernetes clusters separately, consider your organization needs.

For example, if your organization runs TKGI-provisioned Kubernetes clusters in both development and production environments and you want to upgrade only one environment first, you can achieve your goal by upgrading the TKGI control plane and TKGI-provisioned Kubernetes separately instead of performing a full upgrade.

Examples of other advantages of upgrading Tanzu Kubernetes Grid Integrated Edition in two phases include:

- Faster Tanzu Kubernetes Grid Integrated Edition tile upgrades. If you have a large number of clusters in your Tanzu Kubernetes Grid Integrated Edition deployment, performing a full upgrade can significantly increase the amount of time required to upgrade the Tanzu Kubernetes Grid Integrated Edition tile.
- More granular control over cluster upgrades. In addition to enabling you to upgrade subsets of clusters, the TKGI CLI supports upgrading each cluster individually.
- Not a monolithic upgrade. This helps isolate the root cause of an error when troubleshooting upgrades. For example, when a cluster-related upgrade error occurs during a full upgrade, the entire Tanzu Kubernetes Grid Integrated Edition tile upgrade may fail.

What Happens During Full TKGI and TKGI Control Plane Upgrades

You can perform full TKGI upgrades and TKGI control plane upgrades only through the Tanzu Kubernetes Grid Integrated Edition tile.

After you add a new Tanzu Kubernetes Grid Integrated Edition tile version to your staging area on the Ops Manager Installation Dashboard, Ops Manager automatically migrates your configuration settings into the new tile version.

For more information, see:

- Full TKGI Upgrades
- TKGI Control Plane Upgrades

Full TKGI Upgrades

During a full TKGI upgrade, the Tanzu Kubernetes Grid Integrated Edition tile does the following:

1. Upgrades the TKGI control plane, which includes the TKGI API and UAA servers and the TKGI database. This control plane upgrade causes temporary outages as described in Control Plane Outages below.

2. Upgrades TKGI-provisioned Kubernetes clusters.
   - Upgrading TKGI-provisioned Kubernetes clusters is controlled by the Upgrade all clusters errand in the Tanzu Kubernetes Grid Integrated Edition tile.
   - The cluster upgrade process recreates all clusters, which may cause cluster outages. For more information, see What Happens During Cluster Upgrades below.
TKGI Control Plane Upgrades

When upgrading the **TKGI control plane only**, the Tanzu Kubernetes Grid Integrated Edition tile follows the process described in Full TKGI Upgrades above, step 1. It does not upgrade TKGI-provisioned Kubernetes clusters, step 2.

Control Plane Outages

Upgrading the Tanzu Kubernetes Grid Integrated Edition control plane temporarily interrupts the following:

- Logging in to the TKGI CLI and using all `tkgi` commands
- Using the TKGI API to retrieve information about clusters
- Using the TKGI API to create and delete clusters
- Using the TKGI API to resize clusters

These outages do not affect the Kubernetes clusters themselves. During a TKGI control plane upgrade, you can still interact with clusters and their workloads using the Kubernetes Command Line Interface, `kubectl`.

For more information about the TKGI control plane, see TKGI Control Plane Overview in Tanzu Kubernetes Grid Integrated Edition Architecture.

Canary Instances

The Tanzu Kubernetes Grid Integrated Edition 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.


What Happens During Cluster Upgrades

Upgrading TKGI-provisioned Kubernetes clusters updates their Kubernetes version to the version included with the Tanzu Kubernetes Grid Integrated Edition tile. It also updates the TKGI version tagged in your clusters to the Tanzu Kubernetes Grid Integrated Edition tile version.

You can upgrade TKGI-provisioned Kubernetes clusters either through the Tanzu Kubernetes Grid Integrated Edition tile or the TKGI CLI. See the table below.

<table>
<thead>
<tr>
<th>This method</th>
<th>Upgrades</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Upgrade all clusters errand in the Tanzu Kubernetes Grid Integrated Edition tile &gt; Errands</td>
<td>All clusters. Clusters are upgraded serially.</td>
</tr>
<tr>
<td><code>tkgi upgrade-cluster</code></td>
<td>One cluster.</td>
</tr>
<tr>
<td><code>tkgi upgrade-clusters</code></td>
<td>Multiple clusters. Clusters are upgraded serially or in parallel.</td>
</tr>
</tbody>
</table>

During an upgrade of TKGI-provisioned clusters, Tanzu Kubernetes Grid Integrated Edition recreates your clusters. This includes the following stages for each cluster you upgrade:
1. Master nodes are recreated.

2. Worker nodes are recreated.

Depending on your cluster configuration, these recreations may cause Master Nodes Outage or Worker Nodes Outage as described below.

**Master Nodes Outage**

When Tanzu Kubernetes Grid Integrated Edition upgrades a single-master cluster, you cannot interact with your cluster, use `kubectl`, or push new workloads.

To avoid this loss of functionality, VMware recommends using multi-master clusters.

**Worker Nodes Outage**

When Tanzu Kubernetes Grid Integrated Edition upgrades a worker node, the node stops running containers. If your workloads run on a single node, they will experience downtime.

To avoid downtime for stateless workloads, VMware recommends using at least one worker node per availability zone (AZ). For stateful workloads, VMware recommends using a minimum of two worker nodes per AZ.

**Note:** When the *Upgrade all clusters errand* is enabled in the Tanzu Kubernetes Grid Integrated Edition tile, updating the 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 **Master Nodes Outage** and **Worker Nodes Outage** above and in *Maintaining Workload Uptime*.

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

In this topic

Authentication of TKGI API Requests
Routing to the TKGI API VM

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how the VMware Tanzu Kubernetes Grid Integrated Edition API works with User Account and Authentication (UAA) to manage authentication and authorization in your Tanzu Kubernetes Grid Integrated Edition deployment.

Authentication of TKGI API Requests

Before users can log in and use the TKGI CLI, you must configure TKGI API access with UAA. For more information, see Managing Tanzu Kubernetes Grid Integrated Edition Users with UAA and Logging in to Tanzu Kubernetes Grid Integrated Edition.

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 TKGI API using the TKGI CLI and grant cluster access to new or existing users.

When a user with cluster access logs in to the TKGI 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 TKGI CLI. When the user runs TKGI CLI commands, for example, `tkgi clusters`, the CLI sends the request to the TKGI API server and includes the user's UAA token.

The TKGI 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 TKGI API uses the cluster information from the TKGI broker to respond to the request. For example, if the user runs `tkgi clusters`, the CLI returns a list of the clusters that the user is authorized to manage.

Routing to the TKGI API VM

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

<table>
<thead>
<tr>
<th>Server</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>TKGI API</td>
<td>api.tkgi.example.com:9021</td>
</tr>
<tr>
<td>UAA</td>
<td>api.tkgi.example.com:8443</td>
</tr>
</tbody>
</table>

Refer to Ops Manager > Tanzu Kubernetes Grid Integrated Edition tile > TKGI API > API Hostname (FQDN) for your TKGI API domain.

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

Please send any feedback you have to pks-feedback@pivotal.io.
Load Balancers in Tanzu Kubernetes Grid Integrated Edition

In this topic

Load Balancers in Tanzu Kubernetes Grid Integrated Edition Deployments without NSX-T
  About the TKGI API Load Balancer
  About Kubernetes Cluster Load Balancers
  About Workload Load Balancers
Load Balancers in Tanzu Kubernetes Grid Integrated Edition Deployments on vSphere with NSX-T
  Resizing Load Balancers

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes the types of load balancers that are used in VMware Tanzu Kubernetes Grid Integrated Edition deployments. Load balancers differ by the type of deployment.

Load Balancers in Tanzu Kubernetes Grid Integrated Edition Deployments without NSX-T

For Tanzu Kubernetes Grid Integrated Edition deployments on GCP, AWS, or vSphere without NSX-T, you can configure load balancers for the following:

- **TKGI API**: Configuring this load balancer enables you to run TKGI Command Line Interface (TKGI 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 Tanzu Kubernetes Grid Integrated Edition 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 TKGI API Load Balancer

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

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

About Kubernetes Cluster Load Balancers

When you create an Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition Clusters
- Creating and Configuring an AWS Load Balancer for Tanzu Kubernetes Grid Integrated Edition Clusters
- Creating and Configuring an Azure Load Balancer for Tanzu Kubernetes Grid Integrated Edition Clusters

**About Workload Load Balancers**

To enable external access to your Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition deployment, traffic routing is controlled by the ingress resource rules you define. VMware recommends configuring Tanzu Kubernetes Grid Integrated Edition deployments with both a workload load balancer and an ingress controller.

The following diagram shows how the ingress routing can be used within your Tanzu Kubernetes Grid Integrated Edition deployment.
The load balancer on Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition Deployments on vSphere with NSX-T

Tanzu Kubernetes Grid Integrated Edition deployments on vSphere with NSX-T do not require a load balancer configured to access the TKGI API. They require only a DNAT rule configured so that the TKGI API host is accessible. For more information, see Share the Tanzu Kubernetes Grid Integrated Edition Endpoint in Installing Tanzu Kubernetes Grid Integrated Edition 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.
- **service:** 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.
Resizing Load Balancers

When a new Kubernetes cluster is provisioned using the TKGI 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 Tanzu Kubernetes Grid Integrated Edition Clusters

In this topic
Overview
Master Node VM Size
Worker Node VM Number and Size
   Example Worker Node Requirement Calculation
Customize Master and Worker Node VM Size and Type

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how VMware Tanzu Kubernetes Grid Integrated Edition recommends you approach the sizing of VMs for cluster components.

Overview

When you configure plans in the Tanzu Kubernetes Grid Integrated Edition tile, you provide VM sizes for the master and worker node VMs. For more information about configuring plans, see the Plans section of Installing Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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

<table>
<thead>
<tr>
<th>Number of Workers</th>
<th>CPU</th>
<th>RAM (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-5</td>
<td>1</td>
<td>3.75</td>
</tr>
<tr>
<td>6-10</td>
<td>2</td>
<td>7.5</td>
</tr>
<tr>
<td>11-100</td>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>101-250</td>
<td>8</td>
<td>30</td>
</tr>
<tr>
<td>251-500</td>
<td>16</td>
<td>60</td>
</tr>
<tr>
<td>500+</td>
<td>32</td>
<td>120</td>
</tr>
</tbody>
</table>

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 \([W]\) = \(\frac{p}{100}\)
- Minimum RAM per worker = \(m \times 100\)
- Minimum number of CPUs per worker = \(c \times 100\)

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 Tanzu Kubernetes Grid Integrated Edition-provisioned Kubernetes cluster is set by the Maximum number of workers on a cluster field in the Plans pane of the Tanzu Kubernetes Grid Integrated Edition 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
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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition installation topic for your IaaS. For example, Installing Tanzu Kubernetes Grid Integrated Edition 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

Overview

Participation Levels

Configure CEIP and Telemetry

System Components

Data Dictionary

Sample Reports

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes the VMware Customer Experience Improvement Program (CEIP) and the Telemetry Program used in the Tanzu Kubernetes Grid Integrated Edition tile.

Overview

The CEIP and Telemetry program allows VMware to collect data from customer installations to improve your Tanzu Kubernetes Grid Integrated Edition experience. Collecting data at scale enables us to identify patterns and alert you to warning signals in your Tanzu Kubernetes Grid Integrated Edition installation.

Participation Levels

You can configure Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition.
- **Enhanced**: This level enables VMware to warn you about security vulnerabilities and potential issues with your software configurations. For more information, see Benefits of the Enhanced Participation Level below.

Note: Tanzu Kubernetes Grid Integrated Edition does not collect any personally identifiable information (PII) at either participation level. For a list of the data Tanzu Kubernetes Grid Integrated Edition collects, see Data Dictionary.

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 Tanzu Kubernetes Grid Integrated Edition installation. See sample reports below for more details.

- **Access to your telemetry data**: This gives you access to configuration and usage data about your Tanzu Kubernetes Grid Integrated Edition installation. See sample reports below for more details.
Proactive support: This enables VMware to proactively warn you about unhealthy patterns.

Benchmarks: This is your usage relative to the rest of the Tanzu Kubernetes Grid Integrated Edition user base.

The table below compares the Standard and Enhanced participation levels.

<table>
<thead>
<tr>
<th>Benefit</th>
<th>Standard Level</th>
<th>Enhanced Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage data</td>
<td>Raw data</td>
<td>Reports and trend analysis</td>
</tr>
<tr>
<td>Access to your telemetry data</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Proactive support</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Benchmarks</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

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

Configure CEIP and Telemetry

**Video:** For information about configuring CEIP and Telemetry participation, see the [CEIP Opt-In Walkthrough video](https://www.youtube.com) on YouTube.

To configure CEIP and Telemetry, see the **CEIP and Telemetry** section of the installation topic for your IaaS:

- Installing Tanzu Kubernetes Grid Integrated Edition on vSphere
- Installing Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T
- Installing Tanzu Kubernetes Grid Integrated Edition on AWS
- Installing Tanzu Kubernetes Grid Integrated Edition on Azure
- Installing Tanzu Kubernetes Grid Integrated Edition on GCP

Proxy Communication

If you use a proxy server, the Tanzu Kubernetes Grid Integrated Edition proxy settings apply to outgoing telemetry data.

To configure Tanzu Kubernetes Grid Integrated Edition proxy settings for CEIP and Telemetry and other communications, see the following:

- For AWS, see [Using Proxies with Tanzu Kubernetes Grid Integrated Edition on AWS](#)
- For vSphere, see [Networking in Installing Tanzu Kubernetes Grid Integrated Edition on vSphere](#)
- For vSphere with NSX-T, see [Using Proxies with Tanzu Kubernetes Grid Integrated Edition on NSX-T](#)

System Components

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

- **Telemetry Server:** This component runs on the TKGI control plane. The server receives telemetry events from the TKGI 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 TKGI Telemetry collection and reporting, see the TKGI 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 TKGI, I need a way to monitor pod consumption across my TKGI 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 TKGI within my organization

2. **API heartbeats + Cluster heartbeats:** As an Operator of TKGI I need a way to see the version of TKGI each of my environments was running over time, so I can:
   - Keep track of all my TKGI environments and clusters
   - Identify environments and clusters in need of upgrading

3. **Cluster creation events:** As an Operator of TKGI I want to see how often cluster creation succeeds across my TKGI 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 TKGI 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 TKGI, 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 TKGI, I want to see which container images are in use across my TKGI installations so I can:
   - Conduct an audit of container images and identify prohibited or problematic images
   - Infer which workloads are running on TKGI, to inform my planning, resourcing, and outreach

Please send any feedback you have to pks-feedback@pivotal.io.
Installing Tanzu Kubernetes Grid Integrated Edition

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

**Page last updated:**

Tanzu Kubernetes Grid Integrated Edition Management Console (vSphere Only)

See the following documentation for the Management Console, which is the recommended method for installing Tanzu Kubernetes Grid Integrated Edition on vSphere:

- Install Tanzu Kubernetes Grid Integrated Edition on vSphere with the Management Console

For more information, see [When Should I Use Tanzu Kubernetes Grid Integrated Edition Management Console?](#).

Tanzu Kubernetes Grid Integrated Edition on Ops Manager

See the following documentation for how to manually install Tanzu Kubernetes Grid Integrated Edition, using Ops Manager, on Ops Manager:

- vSphere with Flannel
- vSphere with NSX-T
- Google Cloud Platform
- Amazon Web Services
- Azure

**Note:** Tanzu Kubernetes Grid Integrated Edition supports air-gapped deployments on vSphere with or without NSX-T integration.

Please send any feedback you have to pks-feedback@pivotal.io.
Installing Tanzu Kubernetes Grid Integrated Edition on vSphere

In this topic

Overview

When Should I Use Tanzu Kubernetes Grid Integrated Edition Management Console?

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes options for installing Tanzu Kubernetes Grid Integrated Edition on vSphere.

Overview

You can install Tanzu Kubernetes Grid Integrated Edition (TKGI) on vSphere in three ways, depending on whether you use the TKGI Management Console, and which container networking overlay you use:

- Install Tanzu Kubernetes Grid Integrated Edition on vSphere with the Management Console
- Install Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T Using Ops Manager
- Install Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T Using Ops Manager

Where possible, VMware recommends using the management console to install Tanzu Kubernetes Grid Integrated Edition on vSphere. For more information, see When Should I Use Tanzu Kubernetes Grid Integrated Edition Management Console? below.

When Should I Use Tanzu Kubernetes Grid Integrated Edition Management Console?

Tanzu Kubernetes Grid Integrated Edition Management Console greatly simplifies the process of deploying Tanzu Kubernetes Grid Integrated Edition, especially in less complex environments. However, if you require more flexibility 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 Tanzu Kubernetes Grid Integrated Edition.

Before using Tanzu Kubernetes Grid Integrated Edition Management Console to deploy Tanzu Kubernetes Grid Integrated Edition, consider the following factors:

- If you want to deploy Tanzu Kubernetes Grid Integrated Edition Management Console to a No-NAT topology with an NSX-T Data Center logical switch, you must perform a BYOT deployment.
- 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 Tanzu Kubernetes Grid Integrated Edition Management Console Cannot Retrieve Cluster Data in a Multi-Tier0 Topology in Troubleshooting Tanzu Kubernetes Grid Integrated Edition Management Console.
- Deployments to a No-NAT topology with a vSphere Standard Switch or a vSphere Distributed Switch are not supported in any case.
- Multi-Foundation deployments are not supported in any case.

How users deploy Tanzu Kubernetes Grid Integrated Edition often depends on whether they already have a custom installation of the
Ops Manager tool, which TKGI uses:

- Users on vSphere who do not have a custom Ops Manager installation may prefer the management console.
- Users who already have Ops Manager installed for other uses, for example to run VMware Tanzu Application Service for VMs, may prefer to install TKGI manually.

Please send any feedback you have to pks-feedback@pivotal.io.
Install Tanzu Kubernetes Grid Integrated Edition on vSphere with the Management Console

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

VMware Tanzu Kubernetes Grid Integrated Edition Management Console provides a unified installation experience for deploying VMware Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition to vSphere:

- Configures networking for Tanzu Kubernetes Grid Integrated Edition
- Deploys Ops Manager
- Generates and registers SSL certificates
- Deploys BOSH Director
- Deploys Tanzu Kubernetes Grid Integrated Edition
- Deploys Harbor Registry

Tanzu Kubernetes Grid Integrated Edition Management Console is easy to use. If you are experienced with installing Tanzu Kubernetes Grid Integrated Edition on vSphere, the help and the tool tips in the installer UI should be enough to complete the process. If you are new to Tanzu Kubernetes Grid Integrated Edition, refer to this documentation as needed to assist with the installation.

See the following topics:

- Prerequisites for Tanzu Kubernetes Grid Integrated Edition Management Console Deployment
- Deploy the Tanzu Kubernetes Grid Integrated Edition Management Console
- Deploy Tanzu Kubernetes Grid Integrated Edition from the Management Console

After you have deployed Tanzu Kubernetes Grid Integrated Edition on vSphere, you can use the management console to deploy Kubernetes clusters and manage their lifecycle, and monitor and manage the operation of your Tanzu Kubernetes Grid Integrated Edition deployment. For information about how to use the management console after deployment, see the following topics:

- Create and Manage Clusters in the Management Console
- Monitor and Manage Tanzu Kubernetes Grid Integrated Edition in the Management Console
- Troubleshooting Tanzu Kubernetes Grid Integrated Edition Management Console

Please send any feedback you have to pks-feedback@pivotal.io.
Prerequisites for Tanzu Kubernetes Grid Integrated Edition Management Console Deployment

In this topic

Network Configurations

💡 **Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

VMware Tanzu Kubernetes Grid Integrated Edition Management Console is provided as an OVA template that requires at a minimum the vSphere resources described in Virtual Infrastructure Prerequisites.

For more information, see When Should I Use Tanzu Kubernetes Grid Integrated Edition Management Console?

Network Configurations


- **Bring your own topology:** Deploy Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition to an existing NSX-T Data Center network that you have not fully set up, that Tanzu Kubernetes Grid Integrated Edition Management Console helps to configure for you. See Prerequisites for an Automated NAT Deployment to NSX-T Data Center

- **Flannel:** Deploy Tanzu Kubernetes Grid Integrated Edition to a Flannel network that Tanzu Kubernetes Grid Integrated Edition Management Console provisions for you. See Prerequisites for a Flannel Network

For the list of firewall ports that must be open for an Tanzu Kubernetes Grid Integrated Edition Management Console deployment, see Firewall Ports and Protocols Requirements for Tanzu Kubernetes Grid Integrated Edition Management Console

When your environment meets the prerequisites for vSphere and for your chosen type of networking, you can Deploy the Tanzu Kubernetes Grid Integrated Edition Management Console.

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

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

**Page last updated:**

The vSphere environment to which you deploy the management console 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 Tanzu Kubernetes Grid Integrated Edition.

**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 Tanzu Kubernetes Grid Integrated Edition from the management console:

- Management cluster for TKGI 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 Tanzu Kubernetes Grid Integrated Edition Management Console

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

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 Tanzu Kubernetes Grid Integrated Edition on vSphere, it 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 Tanzu Kubernetes Grid Integrated Edition, 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 a Tanzu Kubernetes Grid Integrated Edition Management Console deployment.

**Notes:** The Source Component is IP address of the Tanzu Kubernetes Grid Integrated Edition Management Console VM.

In a standard Tanzu Kubernetes Grid Integrated Edition deployment, it is assumed that Ops Manager and BOSH are already deployed before you deploy Tanzu Kubernetes Grid Integrated Edition. This is not the case with Tanzu Kubernetes Grid Integrated Edition deployments from the management console, in which you do not know the IP addresses in the deployment network that will be assigned to TKGI 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 VM to the entire deployment subnet.

<table>
<thead>
<tr>
<th>Source Component</th>
<th>Destination Component</th>
<th>Destination Protocol</th>
<th>Destination Port</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management Console VM</td>
<td>All System Components</td>
<td>TCP</td>
<td>22</td>
<td>ssh</td>
</tr>
<tr>
<td>Management Console VM</td>
<td>All System Components</td>
<td>TCP</td>
<td>80</td>
<td>http</td>
</tr>
<tr>
<td>Management Console VM</td>
<td>All System Components</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Management Console VM</td>
<td>Cloud Foundry BOSH Director</td>
<td>TCP</td>
<td>25555</td>
<td>bosh director rest api</td>
</tr>
<tr>
<td>Management Console VM</td>
<td>DNS validation for Ops Manager</td>
<td>TCP</td>
<td>53</td>
<td>netcat</td>
</tr>
<tr>
<td>Source Component</td>
<td>Destination Component</td>
<td>Destination Protocol</td>
<td>Destination Port</td>
<td>Service</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------------------------------------------------------</td>
<td>-----------------------</td>
<td>------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Management Console VM</td>
<td>Kubernetes Cluster API Server - LB VIP</td>
<td>TCP</td>
<td>8443</td>
<td>httpsca</td>
</tr>
<tr>
<td>Management Console VM</td>
<td>Pivotal Cloud Foundry Operations Manager</td>
<td>TCP</td>
<td>22</td>
<td>ssh</td>
</tr>
<tr>
<td>Management Console VM</td>
<td>Pivotal Cloud Foundry Operations Manager</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Management Console VM</td>
<td>TKGI Controller</td>
<td>TCP</td>
<td>9021</td>
<td>tkgi api server</td>
</tr>
<tr>
<td>Management Console VM</td>
<td>vCenter Server</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
</tbody>
</table>

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
- General Requirements
- NSX-T Data Center Configuration Requirements
- Proof-of-Concept Deployments

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

**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 Tanzu Kubernetes Grid Integrated Edition. For example, an NSX-T Data Center instance that you have used in a previous deployment of Tanzu Kubernetes Grid Integrated Edition. 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/active Tier-0 Router created.
- A logical switch on an NSX-T Virtual Distributed Switch (N-VDS) for use by the TKGI management plane is prepared. The switch must be either under the Tier-0 router, or under the Tier-1 router if the Tier-1 router is directly under the Tier-0 router.
- 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 Tanzu Kubernetes Grid Integrated Edition Management Console VM as the network for the management plane when you deploy Tanzu Kubernetes Grid Integrated Edition. Using the same network for the management console 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, Tanzu Kubernetes Grid Integrated Edition 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, Tanzu Kubernetes Grid Integrated Edition Management Console automatically retrieves the tier0 HA mode from
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 TKGI Management Plane
- Tier-1 Router configured and connected to the Tier-0 Router
- Routing for TKGI 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
General Requirements
Proof-of-Concept Deployments

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

An unprepared environment is an NSX-T Data Center instance that you have not already configured for use with Tanzu Kubernetes Grid Integrated Edition. Tanzu Kubernetes Grid Integrated Edition Management Console helps you to complete the configuration of an unprepared environment on vSphere, 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, Tanzu Kubernetes Grid Integrated Edition 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

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

---

**Page last updated:**

You can select the option for Tanzu Kubernetes Grid Integrated Edition Management Console to provision a Flannel container networking interface for you during Tanzu Kubernetes Grid Integrated Edition deployment on vSphere.

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 Tanzu Kubernetes Grid Integrated Edition
- 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 Tanzu Kubernetes Grid Integrated Edition Management Console

In this topic
Prerequisites
Step 1: Deploy the OVA Template
Step 2: Log In to Tanzu Kubernetes Grid Integrated Edition Management Console
Next Steps

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to deploy the VMware Tanzu Kubernetes Grid Integrated Edition Management Console from the OVA template.

If you have deployed a previous version of VMware Tanzu Kubernetes Grid Integrated Edition Management Console, you can use the management console to upgrade it to a newer version. For information about upgrading, see Upgrade Tanzu Kubernetes Grid Integrated Edition Management Console.

Prerequisites

- 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 Tanzu Kubernetes Grid Integrated Edition Management Console deployment. For information about the vCenter Server requirements, see Virtual Infrastructure Prerequisites.

Step 1: Deploy the OVA Template

To deploy the Tanzu Kubernetes Grid Integrated Edition Management Console 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 management console and to select the vSphere resources for it to use.
   - Accept or modify the management console VM name
   - Select the destination datacenter or folder
   - Select the destination cluster or resource pool for the management console VM
   - Accept the end user license agreements (EULA)
   - Select the disk format and destination datastore for the management console VM

3. On the Select Networks page, select a network port group to which to connect the management console VM.

⚠️ important: If you intend to deploy Tanzu Kubernetes Grid Integrated Edition in a bring your own topology NSX-T Data
4. On the **Customize template** page, expand **Appliance Configuration**.

   - Set the root password for the management console VM. Setting the root password for the VM 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 management console 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 management console VM certificate, that is used by all of the services that run in the management console VM 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 management console VM 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 management console VM. To use DHCP, leave these properties blank.

   - To set a static IP address on the management console VM, 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 management console VM, set the **FQDN** setting.
   - If necessary, update **Docker Container Network Subnet** and **Docker Container Network Gateway**.

   Services in the management console VM 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.


8. Click **Next** to review the settings that you have made.


Use the Recent Tasks panel at the bottom of the vSphere Client to check the status of the OVA import and deployment of the management console VM. The management console VM takes a few minutes to deploy.

If the management console VM fails to deploy, see Troubleshooting.

Step 2: Log In to Tanzu Kubernetes Grid Integrated Edition Management Console

When the OVA deployment has completed successfully, you can access the management console.

1. In the vSphere Client, right-click the management console VM and select **Power > Power On**.

2. When the management console VM has booted, go to the **Summary** tab for the VM and copy its IP address.

3. Enter the management console VM IP address in a browser.

4. At the VMware Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition Management Console to deploy or upgrade Tanzu Kubernetes Grid Integrated Edition instances, either by using the configuration wizard or by importing an existing YAML configuration file.

- **Deploy Tanzu Kubernetes Grid Integrated Edition from the management console**
- **Upgrade Tanzu Kubernetes Grid Integrated Edition Management Console**

Please send any feedback you have to pks-feedback@pivotal.io.
Deploy Tanzu Kubernetes Grid Integrated Edition from the Management Console

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

**Page last updated:**

You can deploy a new VMware Tanzu Kubernetes Grid Integrated Edition instance on vSphere either by using the VMware Tanzu Kubernetes Grid Integrated Edition Management Portal configuration wizard to guide you through the configuration process, or by importing an existing YAML configuration file into the YAML editor.

- **Deploy Tanzu Kubernetes Grid Integrated Edition by Using the Configuration Wizard**
- **Deploy Tanzu Kubernetes Grid Integrated Edition by Importing a YAML Configuration File**

If you deploy Tanzu Kubernetes Grid Integrated Edition with plans that use Windows worker nodes, further configuration is required. 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 after you deploy Tanzu Kubernetes Grid Integrated Edition.

Please send any feedback you have to pks-feedback@pivotal.io.
Deploy Tanzu Kubernetes Grid Integrated Edition by Using the Configuration Wizard

In this topic

Prerequisites

Step 0: Launch the Configuration Wizard

Step 1: Connect to vCenter Server

Step 2: Configure Networking
  - Configure an Automated NAT Deployment to NSX-T Data Center
  - Configure a Bring Your Own Topology Deployment to NSX-T Data Center
  - Configure a Flannel Network

Step 3: Configure Identity Management
  - Use a Local Database
  - Use an External LDAP Server
  - Use a SAML Identity Provider
  - Optionally Configure UAA and Custom Certificates

Step 4: Configure Availability Zones

Step 5: Configure Resources and Storage

Step 6: Configure Plans

Step 7: Configure Integrations
  - Configure a Connection to VMware Tanzu Mission Control
  - Configure a Connection to Wavefront
  - Configure a Connection to VMware vRealize Operations Management Pack for Container Monitoring
  - Configure a Connection to VMware vRealize Log Insight
  - Configure a Connection to Syslog

Step 8: Configure Harbor

Step 9: Configure CEIP and Telemetry

Step 10: Generate Configuration File and Deploy Tanzu Kubernetes Grid Integrated Edition

Next Steps

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to use the configuration wizard to deploy Tanzu Kubernetes Grid Integrated Edition on vSphere.

- For information about how to deploy Tanzu Kubernetes Grid Integrated Edition from a YAML, see Deploy Tanzu Kubernetes Grid Integrated Edition by Importing a YAML Configuration File.
- For information about how to upgrade an existing deployment to this version, see Upgrade Tanzu Kubernetes Grid Integrated Edition Management Console.

Prerequisites

- Deploy the Tanzu Kubernetes Grid Integrated Edition Management Console to vCenter Server.
• The vCenter Server instance must be correctly configured for Tanzu Kubernetes Grid Integrated Edition 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


Step 0: Launch the Configuration Wizard


2. Click Start Configuration.
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 Tanzu Kubernetes Grid Integrated Edition.
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 Tanzu Kubernetes Grid Integrated Edition from the drop-down menu.

---

**warning:** Ideally, do not deploy TGKI from the management console to a datacenter that also includes TKGI instances that you deployed manually. If deploying management console and manual instances of TKGI to the same datacenter cannot be avoided, make sure that the TKGI instances that you deployed manually do not use the folder names `BoshVMFolder: pks_vms`, `BoshTemplateFolder: pks_templates`, `BoshDiskPath: pks_disk`. If a manual installation uses these folder names, the VMs that they contain will be deleted when you delete a TKGI instance from the management console.

5. Click Next to configure networking.

**Step 2: Configure Networking**

• **Automated NAT deployment**: Deploy Tanzu Kubernetes Grid Integrated Edition to an existing NSX-T Data Center network that you have not fully set up, that Tanzu Kubernetes Grid Integrated Edition Management Console configures for you. See Configure an Automated NAT Deployment to NSX-T Data Center below for instructions.

• **Bring your own topology**: Deploy Tanzu Kubernetes Grid Integrated Edition 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 Center below for instructions.


---

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 Tanzu Kubernetes Grid Integrated Edition. You provide information about your NSX-T Data Center setup, and Tanzu Kubernetes Grid Integrated Edition 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.


- **Deployment CIDR**: Enter a CIDR range to use for Tanzu Kubernetes Grid Integrated Edition components, for example 10.192.182.1/22.
- **Deployment DNS**: Enter the IP address of the DNS server to use for deploying Tanzu Kubernetes Grid Integrated Edition 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. You can specify additional reserved IP ranges by editing the YAML configuration for your deployment before you deploy it in **Step 10: Generate Configuration File and Deploy Tanzu Kubernetes Grid Integrated Edition**.
- **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.

---

**Important**: If NSX-T Data Center uses custom certificates and you do not provide the CA certificate for NSX Manager, Tanzu Kubernetes Grid Integrated Edition 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. If you have manually deployed TKGI instances to the same datacenter as the one to which you are deploying this instance, you must select Manage certificates manually for NSX and enter the current NSX-T manager 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 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 Tanzu Kubernetes Grid Integrated Edition. 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 TKGI 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 Tanzu Kubernetes Grid Integrated Edition Management Console VM as the network for the management plane. Using the same network for the management console 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.
   - **Tier-0 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 TKGI 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. You can specify additional reserved IP ranges by editing the YAML configuration for your deployment before you deploy it in **Step 10: Generate Configuration File and Deploy Tanzu Kubernetes Grid Integrated Edition**.
6. Optionally disable **NAT Mode** to implement a routable (No-NAT) topology.

   Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition**.

7. If you left NAT mode enabled, optionally enable **Hybrid NAT Mode**.

   If you enable hybrid NAT mode, the Tanzu Kubernetes Grid Integrated Edition management plane runs on a routable subnet but the cluster node network uses a non-routable subnet.

8. Optionally enable **Manage certificates manually for NSX** if NSX Manager uses a custom CA certificate.

   **Important**: If NSX-T Data Center uses custom certificates and you do not provide the CA certificate for NSX Manager, Tanzu Kubernetes Grid Integrated Edition 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. If you have manually deployed TKGI instances to the same datacenter as the one to which you are deploying this instance, you must select **Manage certificates manually for NSX** and enter the current NSX-T manager CA certificate.

   Enter the contents of the CA certificate in the **NSX Manager CA Cert** text box:

   ```text
   -----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.

9. Optionally enable Disable SSL certificates verification to allow unsecured connections to NSX Manager.

10. Click Next to configure identity management.

For the next steps, see Configure Identity Management

Configure a Flannel Network

Provide networking information so that Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition.
- **Deployment Network CIDR**: Enter a CIDR range to use for Tanzu Kubernetes Grid Integrated Edition 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 10.192.182.1.
- **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. You can specify additional reserved IP ranges by editing the YAML configuration for your deployment before you deploy it in Step 10: Generate Configuration File and Deploy Tanzu Kubernetes Grid Integrated Edition.
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. You can specify additional reserved IP ranges by editing the YAML configuration for your deployment before you deploy it in Step 10: Generate Configuration File and Deploy Tanzu Kubernetes Grid Integrated Edition.
   - **NTP Server**: Enter the IP address of an NTP server.

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**


- 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 Tanzu Kubernetes Grid Integrated Edition deployment. After deployment, you can add users and groups to the database and assign roles to them in the Identity Management view of the Tanzu Kubernetes Grid Integrated Edition Management Console.

1. Select the **Local user database** radio button.
2. In the **TKGI API FQDN** text box, enter an address for the TKGI API Server VM, for example `api.tkgi.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 Tanzu Kubernetes Grid Integrated Edition.

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.

Server SSL Certificate: If you are using an LDAPS endpoint, paste the contents of the LDAP server certificate certificate into the text box.

5. Optionally click the Test LDAP Server button to test the connection that you have configured.

6. In the TKGI API FQDN text box, enter an address for the TKGI API Server VM, for example api.tkgi.example.com.

For the next steps, see Optionally Configure UAA and Custom Certificates.
Use a SAML Identity Provider

You can configure Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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 TKGI. 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 Tanzu Kubernetes Grid Integrated Edition, see **Grant Tanzu Kubernetes Grid Integrated Edition Access to an External LDAP Group**.

9. By default, all SAML authentication requests from Tanzu Kubernetes Grid Integrated Edition 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 **TKGI API FQDN** text box, enter an address for the TKGI API Server VM, for example api.tkgi.example.com.
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 Tanzu Kubernetes Grid Integrated Edition 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`. 

---

**Copyright © 2020 VMware, Inc. All Rights Reserved.**
2. Optionally select **Manage Certificates Manually for TKGI API** to generate and upload your own certificates for the TKGI 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 **TKGI API Certificate** text box:

   ```
   -----BEGIN CERTIFICATE-----
   tkgi_api_certificate_contents
   -----END CERTIFICATE-----
   ```

   Enter the contents of the certificate key in the **Private Key PEM** text box:

   ```
   -----BEGIN RSA PRIVATE KEY-----
   tkgi_api_private_key_contents
   -----END RSA 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 Tanzu Kubernetes Grid Integrated Edition 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.

**Note:** If you select a cluster as an availability zone, Tanzu Kubernetes Grid Integrated Edition Management Console sets the DRS VM-host affinity rule on that cluster to **MUST**. If you select a host group as an availability zone, Tanzu Kubernetes Grid Integrated Edition Management Console sets the DRS VM-host affinity rule on that group to **SHOULD**.

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 TKGI Management Plane. The management plane consists of the TKGI 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, Tanzu Kubernetes Grid Integrated Edition 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**.

![Save Availability Zone](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 VM on which the Tanzu Kubernetes Grid Integrated Edition API and other component services, such as UAA, run. Allocate resources according to the workloads that TKGI will run.


You must also designate the datastores to use for the different types of storage required by your Tanzu Kubernetes Grid Integrated Edition deployment.

- Ephemeral storage is used to contain the files for ephemeral VMs that Tanzu Kubernetes Grid Integrated Edition creates during
installation, upgrade, and operation. Ephemeral VMs are automatically created and deleted as needed.

- Permanent storage is used for permanent Tanzu Kubernetes Grid Integrated Edition 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, Tanzu Kubernetes Grid Integrated Edition 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 TKGI Database Persistent Disk Size, select the size of the persistent disk for the Tanzu Kubernetes Grid Integrated Edition MySQL database VM. Set the TKGI Database Persistent Disk Size according to the amount of data that you expect the cluster workload to store.

2. Use the TKGI Database VM Type drop-down menu to select from different combinations of CPU, RAM, and storage for the Tanzu Kubernetes Grid Integrated Edition MySQL database VM. Choose the configuration for the TKGI Database VM depending on the volume of database operations that it will run.

3. For TKGI API Persistent Disk Size, select the size of the persistent disk for the Tanzu Kubernetes Grid Integrated Edition API VM. Set the TKGI API 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

4. Use the TKGI API VM Type drop-down menu to select from different combinations of CPU, RAM, and storage for the Tanzu Kubernetes Grid Integrated Edition 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.

5. 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.
6. Optionally enable **Specify Permanent Storage** to designate different datastores for ephemeral and permanent data.

7. 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.

8. Under **Kubernetes Persistent Volume Storage**, select one datastore in which to store Kubernetes volumes, or use the search field to find datastores by name.
9. Click **Next** to configure plans.

**Step 6: Configure Plans**

A plan is a cluster configuration template that defines the set of resources for Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition:

- 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 Tanzu Kubernetes Grid Integrated Edition 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, by default one Linux worker node is deployed per Windows cluster. The Linux node provides cluster services to the Windows worker nodes. You can optionally make the cluster services Linux node highly available, in which case two Linux nodes are deployed.
- If you use Windows worker nodes, after you deploy Tanzu Kubernetes Grid Integrated Edition, 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 Tanzu Kubernetes Grid Integrated Edition with Windows worker nodes, see [Enable Plans with Windows Worker Nodes](#).

---

Tanzu Kubernetes Grid Integrated Edition 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 TKG 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.

Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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.

- **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`
  - `SecurityContextDeny` If you use Windows worker nodes, this option is not available.

- **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. If you use Windows worker nodes, optionally enable the **Enable HA Linux Workers** option to deploy two Linux worker nodes per Windows cluster instead of one. The Linux nodes provide cluster services to the Windows clusters.

4. Click **Save Plan** for each plan that you edit.

5. 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.
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 TKGI to connect to those services. You can also configure TKGI to forward logs to a Syslog server.

Configure a Connection to VMware Tanzu Mission Control


Tanzu Mission Control integration lets you monitor and manage Tanzu Kubernetes Grid Integrated Edition 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.

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 TKGI 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.
6. Click Save.

7. Configure integrations with other applications, or click Next to install Harbor.

Configure a Connection to Wavefront

By connecting your Tanzu Kubernetes Grid Integrated Edition 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 TKGI Integration and VMware TKGI 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 TKGI

6. In the HTTP Proxy for TKGI text box, enter the address of the proxy server to use when it is not possible for the Tanzu Kubernetes Grid Integrated Edition 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 TKGI
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


If you enable the option to integrate Tanzu Kubernetes Grid Integrated Edition with VMware vRealize Operations Management Pack for Container Monitoring, the management console creates a `cAdvisor` container in your Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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.

5. Click Save.

6. Configure integrations with other applications, or click Next to install Harbor.

**Note:** If you enable integration with vRealize Log Insight, Tanzu Kubernetes Grid Integrated Edition Management Console generates a unique vRealize Log Insight agent ID for the management console. You must provide this agent ID to vRealize Log Insight so that it can pull the appropriate logs from the management console VM. For information about how to obtain the agent ID, see Obtain the VMware vRealize Log Insight Agent ID for TKGI Management Console in Troubleshooting Tanzu Kubernetes Grid Integrated Edition Management Console.

Configure a Connection to Syslog

You can configure your Tanzu Kubernetes Grid Integrated Edition 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.
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 Tanzu Kubernetes Grid Integrated Edition.

2. In the **Harbor FQDN** text box, enter a name for the Harbor VM, for example **harbor.tkgi.example.com**. This is the address at which you access the Harbor administration UI and registry service. Before you set the host name, you must check for potential host name conflicts between TKGI and Harbor.
   - If the host name might resolve to an IP address that is not one that you want it to, clear the DNS entry manually to avoid conflicts in subsequent use.
   - If the host name can be resolved to an IP address that you have intentionally created beforehand, be aware that the IP address in the DNS entry might not be the same as the reachable IP address that TKGI Management Console uses, resulting in network issues. If you must use a pre-created DNS entry, after the TKGI deployment finishes, check the IP address that TKGI Management Console uses for Harbor and update the DNS entry accordingly.

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 Tanzu Kubernetes Grid Integrated Edition.
5. If your environment does not allow Harbor components to access the external network on which Tanzu Kubernetes Grid Integrated Edition 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.

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.

   ```plaintext
   -----BEGIN CERTIFICATE-----
   ssl_certificate_contents
   -----END CERTIFICATE-----
   ```

   Paste the contents of the certificate key in the **SSL Key PEM** text box.

   ```plaintext
   -----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.

   ```plaintext
   -----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.

   ```plaintext
   Container Registry Storage Configuration
   Specify the File Storage Used for Storing Container Images
   Remote NFS Server
   ```

   ```plaintext
   NFS Server Address
   10.192.188.55:/path/to/s
   ```

   - **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.

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.

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 Tanzu Kubernetes Grid Integrated Edition 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.
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 Tanzu Kubernetes Grid Integrated Edition installation.
- (Default) **Standard**: If you select this option, data is collected from your Tanzu Kubernetes Grid Integrated Edition installation to improve Tanzu Kubernetes Grid Integrated Edition. 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 Tanzu Kubernetes Grid Integrated Edition 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 TKGI installation in the **TKGI 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 **TKGI Installation Type** list.

6. Click **Save**.

**Note:** If you join the CEIP and Telemetry Program for Tanzu Kubernetes Grid Integrated Edition, open your firewall to allow outgoing access to [https://vcsa.vmware.com/ph](https://vcsa.vmware.com/ph) on port 443.

**Note:** Even if you select **None**, Tanzu Kubernetes Grid Integrated Edition-provisioned clusters send usage data to the TKGI control plane. However, this data is not sent to VMware or Pivotal and remains on your Tanzu Kubernetes Grid Integrated Edition installation.

**Step 10: Generate Configuration File and Deploy Tanzu Kubernetes Grid Integrated Edition**

When all of the sections of the wizard are green, you can generate a YAML configuration file and deploy Tanzu Kubernetes Grid Integrated Edition.

1. Click **Generate Configuration** to see the generated YAML file.
2. (Optional) 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. (Optional) Specify an FQDN address for the Ops Manager VM by editing the YAML directly in the YAML editor.

   ![warning](image)

   **warning:** You cannot change the Ops Manager FQDN of Tanzu Kubernetes Grid Integrated Edition once it has already deployed.

   To specify an FQDN address for the Ops Manager VM, update the YAML as follows:

   a. Locate the `opsman_fqdn:` entry in the YAML file.
   b. Update the `opsman_fqdn:` entry with the Ops Manager VM FQDN:

   ```yaml
   opsman_fqdn: "myopsman.example.com"
   ```
   c. Make sure that the FQDN is mapped to the following IP address:

      - For NSX-T deployments map it to the first address in the floating IP range.
      - For Flannel deployments, map it to the first address in the deployment network, excluding the gateway, deployment DNS, and reserved IP range.

      If you start the deployment and you have not mapped the FQDN to an IP address, the deployment fails with an error. If this happens, configure the mapping as above, return to the YAML editor, and start the deployment again.

4. (Optional) Edit the YAML directly in the YAML editor to specify additional reserved IP ranges in the deployment network or service network.

   No VMs will be deployed in the reserved ranges that you specify. To specify additional reserved IP ranges, update the YAML as follows:

   a. Locate the `additional_dep_reserved_ip_range:` and `additional_svc_reserved_ip_range:` entries in the YAML file.
   b. Update the `additional_dep_reserved_ip_range:` and `additional_svc_reserved_ip_range:` entries to specify reserved IP ranges in the deployment and service networks:

   ```yaml
   ```

   - Deployment network:
5. Click **Apply Configuration** then **Continue** to deploy Tanzu Kubernetes Grid Integrated Edition.

6. On the TKGI Configuration page, follow the progress of the deployment.

7. When the deployment has completed successfully, click **Continue** to monitor and manage your deployment.
Next Steps

You can now access the Tanzu Kubernetes Grid Integrated Edition control plane and begin deploying Kubernetes clusters. For information about how to deploy clusters directly from the management console, see Create and Manage Clusters in the Management Console.


**Important:** If you deployed Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition fails to deploy, see Troubleshooting.

Please send any feedback you have to pks-feedback@pivotal.io.
Deploy Tanzu Kubernetes Grid Integrated Edition by Importing a YAML Configuration File

In this topic
Prerequisites
Import a YAML Configuration File
Networking Options in the YAML File
Next Steps

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

Notes: You can only use the management console to deploy Tanzu Kubernetes Grid Integrated Edition from a YAML file if that YAML file was generated by Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition by using the configuration wizard, see Deploy Tanzu Kubernetes Grid Integrated Edition by Using the Configuration Wizard.

warning: Ideally, do not deploy TGKI from the management console to a datacenter that also includes TKGI instances that you deployed manually. If deploying management console and manual instances of TKGI to the same datacenter cannot be avoided, make sure that the TKGI instances that you deployed manually do not use the folder names BoshVMFolder: pks_vms, BoshTemplateFolder: pks_templates, BoshDiskPath: pks_disk. If a manual installation uses these folder names, the VMs that they contain will be deleted when you delete a TKGI instance from the management console.

br /> If NSX-T Data Center uses custom certificates and you do not provide the CA certificate for NSX Manager, Tanzu Kubernetes Grid Integrated Edition 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. If you have manually deployed TKGI instances to the same datacenter as the one to which you are deploying this instance, you must manage NSX-T certificates manually by specifying the nsx_ca_crt option.

For information about how to upgrade an existing deployment, see Upgrade Tanzu Kubernetes Grid Integrated Edition Management Console.

YAML Files and Passwords
When Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition Management Console to vCenter Server.**
- The vCenter Server instance must be correctly configured for Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition Management Console**
- You have an existing YAML configuration file that you exported during a previous Tanzu Kubernetes Grid Integrated Edition deployment from Tanzu Kubernetes Grid Integrated Edition 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**

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 Tanzu Kubernetes Grid Integrated Edition that you want to deploy.
If the YAML was generated by an instance of the 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition from this configuration file.

6. On the TKGI Configuration page, follow the progress of the deployment.

7. When the deployment has completed successfully, click **Continue** to monitor and manage your deployment.
Networking Options in the YAML File

The networking parameters for the three types of Tanzu Kubernetes Grid Integrated Edition 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.

<table>
<thead>
<tr>
<th>Unprepared NSX-T Data Center</th>
<th>Prepared NSX-T Data Center</th>
<th>Flannel</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSX</td>
<td>Success</td>
<td>Completed</td>
</tr>
<tr>
<td>Ops Manager</td>
<td>Success</td>
<td>Completed</td>
</tr>
<tr>
<td>BOSH</td>
<td>Success</td>
<td>Completed</td>
</tr>
<tr>
<td>PKS</td>
<td>Success</td>
<td>Completed</td>
</tr>
<tr>
<td>Harbor</td>
<td>Success</td>
<td>Completed</td>
</tr>
</tbody>
</table>
Next Steps

You can now access the Tanzu Kubernetes Grid Integrated Edition control plane and begin deploying Kubernetes clusters. For information about how to deploy clusters directly from the management console, see Create and Manage Clusters in the Management Console.


**Important:** If you deployed Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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

Prerequisites

Step 1: Create a Windows Server Stemcell
Step 2: Install the Windows Server Stemcell in Operations Manager
Step 3: Reconfigure Your Tanzu Kubernetes Grid Integrated Edition Deployment to Use the Stemcell

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

If you used Tanzu Kubernetes Grid Integrated Edition Management Console to deploy Tanzu Kubernetes Grid Integrated Edition on vSphere, and you created Plans that implement Windows worker nodes, you must use Operations Manager to provide BOSH with a vSphere stemcell for Windows Server. See the release notes for the correct version of vSphere stemcell for this release.

Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition from Tanzu Kubernetes Grid Integrated Edition Management Console, you can only install the stemcell after you have deployed Tanzu Kubernetes Grid Integrated Edition.

After you deploy Tanzu Kubernetes Grid Integrated Edition 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

- During Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition by Using the Configuration Wizard.
- If you did not create a plan that uses Windows worker nodes when you deployed Tanzu Kubernetes Grid Integrated Edition, or if you have upgraded Tanzu Kubernetes Grid Integrated Edition Management Console from a version that did not support Windows worker nodes, you can use Tanzu Kubernetes Grid Integrated Edition Management Console to reconfigure the plans of your existing deployment. For information about reconfiguring Tanzu Kubernetes Grid Integrated Edition in the management console, see Reconfigure Your Tanzu Kubernetes Grid Integrated Edition Deployment

Step 1: Create a Windows Server Stemcell
vSphere stemcells for Windows Server version 2019 are not available on the Tanzu Network. You must create Windows Server stemcells for vSphere by using Stembuild and your own Windows Server ISO.

Create a vSphere stemcell for Windows Server version 2019 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition with the Windows stemcell installed.


2. If you did not already create a plan that uses Windows worker nodes, or if you have upgraded this Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition instance.

6. When the deployment finishes, go to the TKG Integrated Edition 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.
Install Tanzu Kubernetes Grid Integrated Edition on vSphere with Flannel Using Ops Manager

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

**Page last updated:**

This topic lists the procedures to follow to install Tanzu Kubernetes Grid Integrated Edition on vSphere with Flannel networking manually, using Ops Manager.

**Note:** The recommended method for installing Tanzu Kubernetes Grid Integrated Edition on vSphere is to use the Tanzu Kubernetes Grid Integrated Edition Management Console. For information, see [Install on vSphere with the Management Console](#).

To install Tanzu Kubernetes Grid Integrated Edition on vSphere with Flannel networking 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 Tanzu Kubernetes Grid Integrated Edition on vSphere
- Configuring a TKGI API Load Balancer
- Setting Up Tanzu Kubernetes Grid Integrated Edition Admin Users on vSphere
- *(Optional)* Integrating VMware Harbor with Tanzu Kubernetes Grid Integrated Edition

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

Install the TKGI and Kubernetes CLIs


To install the CLIs, follow the instructions below:

- Installing the TKGI CLI
- Installing the Kubernetes CLI

Please send any feedback you have to pks-feedback@pivotal.io.
vSphere Prerequisites and Resource Requirements

In this topic
Prerequisites
vSphere Version Requirements
Resource Requirements
Network Communication Requirements

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes the prerequisites and resource requirements for installing VMware Tanzu Kubernetes Grid Integrated Edition on vSphere.

For prerequisites and resource requirements for installing Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T integration, see vSphere with NSX-T Version Requirements and Hardware Requirements for Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T.

Prerequisites

Before installing Tanzu Kubernetes Grid Integrated Edition:

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 Tanzu Kubernetes Grid Integrated Edition on vSphere version requirements, refer to the VMware Product Interoperability Matrices.

Resource Requirements

Installing Ops Manager and Tanzu Kubernetes Grid Integrated Edition requires the following virtual machines (VMs):

<table>
<thead>
<tr>
<th>VM</th>
<th>CPU</th>
<th>Memory (GB)</th>
<th>Ephemeral Disk (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOSH Director</td>
<td>2</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>Ops Manager</td>
<td>1</td>
<td>8</td>
<td>160</td>
</tr>
<tr>
<td>TKGI API</td>
<td>2</td>
<td>8</td>
<td>64</td>
</tr>
<tr>
<td>TKGI Database</td>
<td>2</td>
<td>8</td>
<td>64</td>
</tr>
</tbody>
</table>
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 TKGI Database VM as follows:

<table>
<thead>
<tr>
<th>Number of Pods</th>
<th>Persistent Disk Requirements (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000 pods</td>
<td>20</td>
</tr>
<tr>
<td>5,000 pods</td>
<td>100</td>
</tr>
<tr>
<td>10,000 pods</td>
<td>200</td>
</tr>
<tr>
<td>50,000 pods</td>
<td>1,000</td>
</tr>
</tbody>
</table>

Ephemeral VM Resources


To enable Tanzu Kubernetes Grid Integrated Edition to dynamically create the ephemeral VMs when needed, ensure that the following resources are available in your vSphere infrastructure before deploying Tanzu Kubernetes Grid Integrated Edition:

<table>
<thead>
<tr>
<th>Ephemeral VM</th>
<th>VM Count</th>
<th>CPU Cores</th>
<th>Memory (GB)</th>
<th>Ephemeral Disk (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOSH Compilation VMs</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>32</td>
</tr>
</tbody>
</table>

Kubernetes Cluster Resources

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

<table>
<thead>
<tr>
<th>VM</th>
<th>VM Count</th>
<th>CPU Cores</th>
<th>Memory (GB)</th>
<th>Ephemeral Disk (GB)</th>
<th>Persistent Disk (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>master</td>
<td>1 or 3</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>worker</td>
<td>1 or more</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>50</td>
</tr>
<tr>
<td>errand (ephemeral)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>none</td>
</tr>
</tbody>
</table>

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.
This topic describes the firewall ports and protocols requirements for using VMware Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition, VMware recommends that you disable security policies that filter traffic between the networks supporting the system. With Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition deployment.

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

### Tanzu Kubernetes Grid Integrated Edition Ports and Protocols

The following tables list ports and protocols required for network communications between Tanzu Kubernetes Grid Integrated Edition v1.5.0 and later, and vSphere 6.7 and later.

**Tanzu Kubernetes Grid Integrated Edition Users Ports and Protocols**
The following table lists ports and protocols used for network communication between Tanzu Kubernetes Grid Integrated Edition user interface components.

<table>
<thead>
<tr>
<th>Source Component</th>
<th>Destination Component</th>
<th>Destination Protocol</th>
<th>Destination Port</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin/Operator Console</td>
<td>All System Components</td>
<td>TCP</td>
<td>22</td>
<td>ssh</td>
</tr>
<tr>
<td>Admin/Operator Console</td>
<td>All System Components</td>
<td>TCP</td>
<td>80</td>
<td>http</td>
</tr>
<tr>
<td>Admin/Operator Console</td>
<td>All System Components</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Admin/Operator Console</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25555</td>
<td>bosh director rest api</td>
</tr>
<tr>
<td>Admin/Operator Console</td>
<td>Ops Manager</td>
<td>TCP</td>
<td>22</td>
<td>ssh</td>
</tr>
<tr>
<td>Admin/Operator Console</td>
<td>Ops Manager</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Admin/Operator Console</td>
<td>TKGI Controller</td>
<td>TCP</td>
<td>9021</td>
<td>tkgi api server</td>
</tr>
<tr>
<td>Admin/Operator Console</td>
<td>vCenter Server</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Admin/Operator Console</td>
<td>vCenter Server</td>
<td>TCP</td>
<td>5480</td>
<td>vami</td>
</tr>
<tr>
<td>Admin/Operator Console</td>
<td>vSphere ESXI Hosts Mgnt. vmknic</td>
<td>TCP</td>
<td>902</td>
<td>ideafarm-door</td>
</tr>
<tr>
<td>Admin/Operator and Developer Consoles</td>
<td>Harbor Private Image Registry</td>
<td>TCP</td>
<td>80</td>
<td>http</td>
</tr>
<tr>
<td>Admin/Operator and Developer Consoles</td>
<td>Harbor Private Image Registry</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Admin/Operator and Developer Consoles</td>
<td>Harbor Private Image Registry</td>
<td>TCP</td>
<td>4443</td>
<td>notary</td>
</tr>
<tr>
<td>Admin/Operator and Developer Consoles</td>
<td>Kubernetes App Load-Balancer Svc</td>
<td>TCP/UDP</td>
<td>Varies</td>
<td>varies with apps</td>
</tr>
<tr>
<td>Admin/Operator and Developer Consoles</td>
<td>Kubernetes Cluster API Server-LB VIP</td>
<td>TCP</td>
<td>8443</td>
<td>httpsca</td>
</tr>
<tr>
<td>Admin/Operator and Developer Consoles</td>
<td>Kubernetes Cluster Ingress Controller</td>
<td>TCP</td>
<td>80</td>
<td>http</td>
</tr>
<tr>
<td>Admin/Operator and Developer Consoles</td>
<td>Kubernetes Cluster Ingress Controller</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Admin/Operator and Developer Consoles</td>
<td>Kubernetes Cluster Worker Node</td>
<td>TCP/UDP</td>
<td>30000-32767</td>
<td>kubernetes nodeport</td>
</tr>
<tr>
<td>Admin/Operator and Developer Consoles</td>
<td>TKGI Controller</td>
<td>TCP</td>
<td>8443</td>
<td>httpsca</td>
</tr>
<tr>
<td>All User Consoles (Operator, Developer, Consumer)</td>
<td>Kubernetes App Load-Balancer Svc</td>
<td>TCP/UDP</td>
<td>Varies</td>
<td>varies with apps</td>
</tr>
<tr>
<td>All User Consoles (Operator, Developer, Consumer)</td>
<td>Kubernetes Cluster Ingress Controller</td>
<td>TCP</td>
<td>80</td>
<td>http</td>
</tr>
<tr>
<td>All User Consoles (Operator, Developer, Consumer)</td>
<td>Kubernetes Cluster Ingress Controller</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>All User Consoles (Operator, Developer, Consumer)</td>
<td>Kubernetes Cluster Worker Node</td>
<td>TCP/UDP</td>
<td>30000-32767</td>
<td>kubernetes nodeport</td>
</tr>
</tbody>
</table>
The following table lists ports and protocols used for network communication between core Tanzu Kubernetes Grid Integrated Edition components.

<table>
<thead>
<tr>
<th>Source Component</th>
<th>Destination Component</th>
<th>Destination Protocol</th>
<th>Destination Port</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>All System Components</td>
<td>Corporate Domain Name Server</td>
<td>TCP/UDP</td>
<td>53</td>
<td>dns</td>
</tr>
<tr>
<td>All System Components</td>
<td>Network Time Server</td>
<td>UDP</td>
<td>123</td>
<td>ntp</td>
</tr>
<tr>
<td>All System Components</td>
<td>vRealize LogInsight</td>
<td>TCP/UDP</td>
<td>514/1514</td>
<td>syslog/tls syslog</td>
</tr>
<tr>
<td>All System Control Plane Components</td>
<td>AD/LDAP Directory Server</td>
<td>TCP/UDP</td>
<td>389/636</td>
<td>ldap/ldaps</td>
</tr>
<tr>
<td>Ops Manager</td>
<td>Admin/Operator Console</td>
<td>TCP</td>
<td>22</td>
<td>ssh</td>
</tr>
<tr>
<td>Ops Manager</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>6868</td>
<td>bosh agent http</td>
</tr>
<tr>
<td>Ops Manager</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>8443</td>
<td>httpsca</td>
</tr>
<tr>
<td>Ops Manager</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>8844</td>
<td>credhub</td>
</tr>
<tr>
<td>Ops Manager</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25555</td>
<td>bosh director rest api</td>
</tr>
<tr>
<td>Ops Manager</td>
<td>Harbor Private Image Registry</td>
<td>TCP</td>
<td>22</td>
<td>ssh</td>
</tr>
<tr>
<td>Ops Manager</td>
<td>Kubernetes Cluster Master/ETcd Node</td>
<td>TCP</td>
<td>22</td>
<td>ssh</td>
</tr>
<tr>
<td>Ops Manager</td>
<td>Kubernetes Cluster Worker Node</td>
<td>TCP</td>
<td>22</td>
<td>ssh</td>
</tr>
<tr>
<td>Ops Manager</td>
<td>TKGI Controller</td>
<td>TCP</td>
<td>22</td>
<td>ssh</td>
</tr>
<tr>
<td>Ops Manager</td>
<td>vCenter Server</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Ops Manager</td>
<td>vSphere ESXI Hosts Mgmt. vmknic</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>BOSH Director</td>
<td>vCenter Server</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>BOSH Director</td>
<td>vSphere ESXI Hosts Mgmt. vmknic</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>BOSH Compilation Job VM</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>4222</td>
<td>bosh nats server</td>
</tr>
<tr>
<td>BOSH Compilation Job VM</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25250</td>
<td>bosh blobstore</td>
</tr>
<tr>
<td>BOSH Compilation Job VM</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25923</td>
<td>health monitor daemon</td>
</tr>
<tr>
<td>BOSH Compilation Job VM</td>
<td>Harbor Private Image Registry</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>BOSH Compilation Job VM</td>
<td>Harbor Private Image Registry</td>
<td>TCP</td>
<td>8853</td>
<td>bosh dns health</td>
</tr>
<tr>
<td>TKGI Controller</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>4222</td>
<td>bosh nats server</td>
</tr>
<tr>
<td>TKGI Controller</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>8443</td>
<td>httpsca</td>
</tr>
<tr>
<td>TKGI Controller</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25250</td>
<td>bosh blobstore</td>
</tr>
<tr>
<td>TKGI Controller</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25555</td>
<td>bosh director rest api</td>
</tr>
<tr>
<td>Source Component</td>
<td>Destination Component</td>
<td>Destination Protocol</td>
<td>Destination Port</td>
<td>Service</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>----------------------------------------</td>
<td>-----------------------</td>
<td>------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>TKGI Controller</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25923</td>
<td>health monitor daemon</td>
</tr>
<tr>
<td>TKGI Controller</td>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>TCP</td>
<td>8443</td>
<td>httpsca</td>
</tr>
<tr>
<td>TKGI Controller</td>
<td>TKGI Database VM</td>
<td>TCP</td>
<td>3306</td>
<td>tkgi db proxy</td>
</tr>
<tr>
<td>TKGI Controller</td>
<td>vCenter Server</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Harbor Private Image Registry</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>4222</td>
<td>bosh nats server</td>
</tr>
<tr>
<td>Harbor Private Image Registry</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25250</td>
<td>bosh blobstore</td>
</tr>
<tr>
<td>Harbor Private Image Registry</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25923</td>
<td>health monitor daemon</td>
</tr>
<tr>
<td>Harbor Private Image Registry</td>
<td>IP NAS Storage Array</td>
<td>TCP</td>
<td>111</td>
<td>nfs rpc portmapper</td>
</tr>
<tr>
<td>Harbor Private Image Registry</td>
<td>IP NAS Storage Array</td>
<td>TCP</td>
<td>2049</td>
<td>nfs</td>
</tr>
<tr>
<td>Harbor Private Image Registry</td>
<td>Public CVE Source Database</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>kube-system pod/telemetry-agent</td>
<td>TKGI Controller</td>
<td>TCP</td>
<td>24224</td>
<td>fluentd out_forward</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>4222</td>
<td>bosh nats server</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25250</td>
<td>bosh blobstore</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25923</td>
<td>health monitor daemon</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>TCP</td>
<td>2379</td>
<td>etcd client</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>TCP</td>
<td>2380</td>
<td>etcd server</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>TCP</td>
<td>8443</td>
<td>httpsca</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>TCP</td>
<td>8853</td>
<td>bosh dns health</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>Kubernetes Cluster Worker Node</td>
<td>TCP</td>
<td>4194</td>
<td>cadvisor</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>Kubernetes Cluster Worker Node</td>
<td>TCP</td>
<td>10250</td>
<td>kubelet api</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>Kubernetes Cluster Worker Node</td>
<td>TCP</td>
<td>31194</td>
<td>cadvisor</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>TKGI Controller</td>
<td>TCP</td>
<td>8443</td>
<td>httpsca</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>TKGI Controller</td>
<td>TCP</td>
<td>8853</td>
<td>bosh dns health</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>vCenter Server</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Source Component</td>
<td>Destination Component</td>
<td>Destination Protocol</td>
<td>Destination Port</td>
<td>Service</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>------------------------------------------------------</td>
<td>----------------------</td>
<td>------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>Kubernetes Cluster Worker Node</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>4222</td>
<td>bosh nats server</td>
</tr>
<tr>
<td>Kubernetes Cluster Worker Node</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25250</td>
<td>bosh blobstore</td>
</tr>
<tr>
<td>Kubernetes Cluster Worker Node</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25923</td>
<td>health monitor daemon</td>
</tr>
<tr>
<td>Kubernetes Cluster Worker Node</td>
<td>Harbor Private ImageRegistry</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Kubernetes Cluster Worker Node</td>
<td>Harbor Private ImageRegistry</td>
<td>TCP</td>
<td>8853</td>
<td>bosh dns health</td>
</tr>
<tr>
<td>Kubernetes Cluster Worker Node</td>
<td>IP NAS Storage Array</td>
<td>TCP</td>
<td>111</td>
<td>nfs rpc portmapper</td>
</tr>
<tr>
<td>Kubernetes Cluster Worker Node</td>
<td>IP NAS Storage Array</td>
<td>TCP</td>
<td>2049</td>
<td>nfs</td>
</tr>
<tr>
<td>Kubernetes Cluster Worker Node</td>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>TCP</td>
<td>8443</td>
<td>httpsca</td>
</tr>
<tr>
<td>Kubernetes Cluster Worker Node</td>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>TCP</td>
<td>8853</td>
<td>bosh dns health</td>
</tr>
<tr>
<td>Kubernetes Cluster Worker Node</td>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>TCP</td>
<td>10250</td>
<td>kubelet api</td>
</tr>
<tr>
<td>pks-system pod/cert-generator</td>
<td>TKGI Controller</td>
<td>TCP</td>
<td>24224</td>
<td>fluentd out_forward</td>
</tr>
<tr>
<td>pks-system pod/fluent-bit</td>
<td>TKGI Controller</td>
<td>TCP</td>
<td>24224</td>
<td>fluentd out_forward</td>
</tr>
</tbody>
</table>

### 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.

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

VMware Optional Integration Ports and Protocols

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

<table>
<thead>
<tr>
<th>Source Component</th>
<th>Destination Component</th>
<th>Destination Protocol</th>
<th>Destination Port</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin/Operator Console</td>
<td>vRealize Operations Manager</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>vRealize Operations Manager</td>
<td>Kubernetes Cluster API Server - LB VIP</td>
<td>TCP</td>
<td>8443</td>
<td>httpsca</td>
</tr>
<tr>
<td>vRealize Operations Manager</td>
<td>TKGI Controller</td>
<td>TCP</td>
<td>8443</td>
<td>httpsca</td>
</tr>
<tr>
<td>vRealize Operations Manager</td>
<td>Kubernetes Cluster API Server - LB VIP</td>
<td>TCP</td>
<td>8443</td>
<td>httpsca</td>
</tr>
<tr>
<td>Admin/Operator Console</td>
<td>vRealize LogInsight</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Kubernetes Cluster Ingress</td>
<td>vRealize LogInsight</td>
<td>TCP</td>
<td>9000</td>
<td>ingestion api</td>
</tr>
<tr>
<td>Controller</td>
<td>vRealize LogInsight</td>
<td>TCP</td>
<td>9000</td>
<td>ingestion api</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd</td>
<td>vRealize LogInsight</td>
<td>TCP</td>
<td>9543</td>
<td>ingestion api-tls</td>
</tr>
<tr>
<td>Node</td>
<td>vRealize LogInsight</td>
<td>TCP</td>
<td>9000</td>
<td>ingestion api</td>
</tr>
<tr>
<td>Kubernetes Cluster Worker Node</td>
<td>vRealize LogInsight</td>
<td>TCP</td>
<td>9000</td>
<td>ingestion api</td>
</tr>
<tr>
<td>Source Component</td>
<td>Destination Component</td>
<td>Destination Protocol</td>
<td>Destination Port</td>
<td>Service</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>----------------------------------------</td>
<td>----------------------</td>
<td>------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Kubernetes Cluster Worker Node</td>
<td>vRealize LogInsight</td>
<td>TCP</td>
<td>9543</td>
<td>ingestion api - tls</td>
</tr>
<tr>
<td>TKGI Controller</td>
<td>vRealize LogInsight</td>
<td>TCP</td>
<td>9000</td>
<td>ingestion api</td>
</tr>
<tr>
<td>Admin/Operator and Developer Consoles</td>
<td>Wavefront SaaS APM</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>kube-system pod/wavefront-proxy</td>
<td>Wavefront SaaS APM</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>kube-system pod/wavefront-proxy</td>
<td>Wavefront SaaS APM</td>
<td>TCP</td>
<td>8443</td>
<td>httpsca</td>
</tr>
<tr>
<td>pks-system pod/wavefront-collector</td>
<td>TKGI Controller</td>
<td>TCP</td>
<td>24224</td>
<td>fluentd out_forward</td>
</tr>
<tr>
<td>Admin/Operator Console</td>
<td>vRealize Network Insight Platform</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Admin/Operator Console</td>
<td>vRealize Network Insight Proxy</td>
<td>TCP</td>
<td>22</td>
<td>ssh</td>
</tr>
<tr>
<td>vRealize Network Insight Proxy</td>
<td>Kubernetes Cluster API Server - LB VIP</td>
<td>TCP</td>
<td>8443</td>
<td>httpsca</td>
</tr>
<tr>
<td>vRealize Network Insight Proxy</td>
<td>TKGI Controller</td>
<td>TCP</td>
<td>8443</td>
<td>httpsca</td>
</tr>
<tr>
<td>vRealize Network Insight Proxy</td>
<td>TKGI Controller</td>
<td>TCP</td>
<td>9021</td>
<td>tkgi api server</td>
</tr>
</tbody>
</table>

Please send any feedback you have to pks-feedback@pivotal.io.
Creating Dedicated Users and Roles for vSphere (Optional)

In this topic
Overview
Prerequisites
Create the Master Node User Account
Grant Storage Permissions
  Static Only Persistent Volume Provisioning
  Dynamic Persistent Volume Provisioning (with Storage Policy-Based Volume Placement)
  Dynamic Volume Provisioning (without Storage Policy-Based Volume Placement)
Create the BOSH/Ops Manager User Account
Grant Permissions to the BOSH/Ops Manager User Account
Configure DNS for the TKGI API
Next Installation Step

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to create dedicated users and roles for your vSphere environment before deploying VMware Tanzu Kubernetes Grid Integrated Edition.

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

Overview

Before you install Tanzu Kubernetes Grid Integrated Edition on vSphere without NSX-T integration, you can prepare your vSphere environment by creating the required user accounts and configuring DNS for the TKGI 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 TKGI 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 Tanzu Kubernetes Grid Integrated Edition 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.

VMware 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.](image)

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 TKGI API**

**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:

<table>
<thead>
<tr>
<th>Privilege (UI)</th>
<th>Privilege (API)</th>
</tr>
</thead>
</table>

**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 Tanzu Kubernetes Grid Integrated Edition 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:

<table>
<thead>
<tr>
<th>Privilege (UI)</th>
<th>Privilege (API)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual Machine &gt; Add existing disk</td>
<td>VirtualMachine.Config.AddExistingDisk</td>
</tr>
<tr>
<td>Virtual Machine &gt; Add or remove device</td>
<td>VirtualMachine.Config.AddRemoveDevice</td>
</tr>
<tr>
<td>Virtual Machine &gt; Remove disk</td>
<td>VirtualMachine.Config.RemoveDisk</td>
</tr>
</tbody>
</table>

   c. Select the Propagate to Child Objects checkbox.

2. (Optional) 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 privilege at the **Datastore** level using either the vCenter UI or API:

<table>
<thead>
<tr>
<th>Privilege (UI)</th>
<th>Privilege (API)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datastore &gt; Low level file operations</td>
<td>Datastore.FileManagement</td>
</tr>
</tbody>
</table>

   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:
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 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:

<table>
<thead>
<tr>
<th>Privilege (UI)</th>
<th>Privilege (API)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read-only</td>
<td>System.Anonymous</td>
</tr>
<tr>
<td></td>
<td>System.Read</td>
</tr>
<tr>
<td></td>
<td>System.View</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Privilege (UI)</th>
<th>Privilege (API)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual Machine &gt; Add existing disk</td>
<td>VirtualMachine.Config.AddExistingDisk</td>
</tr>
<tr>
<td>Virtual Machine &gt; Add or remove device</td>
<td>VirtualMachine.Config.AddRemoveDevice</td>
</tr>
<tr>
<td>Virtual Machine &gt; Remove disk</td>
<td>VirtualMachine.Config.RemoveDisk</td>
</tr>
</tbody>
</table>

   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:

<table>
<thead>
<tr>
<th>Privilege (UI)</th>
<th>Privilege (API)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datastore &gt; Allocate space</td>
<td>Datastore.AllocateSpace</td>
</tr>
<tr>
<td>Datastore &gt; Low level file operations</td>
<td>Datastore.FileManagement</td>
</tr>
</tbody>
</table>

   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:

<table>
<thead>
<tr>
<th>Privilege (UI)</th>
<th>Privilege (API)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read-only</td>
<td>System.Anonymous</td>
</tr>
<tr>
<td></td>
<td>System.Read</td>
</tr>
<tr>
<td></td>
<td>System.View</td>
</tr>
</tbody>
</table>

**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 TKGI 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 TKGI.

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 TKGI API

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

When you configure the Tanzu Kubernetes Grid Integrated Edition tile, enter this FQDN in the TKGI API pane.

After you deploy Tanzu Kubernetes Grid Integrated Edition, you map the IP address of the TKGI API to this FQDN. You can then use this FQDN to access the TKGI 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
Prerequisites
Install and Configure Ops Manager
Next Installation Step

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

This topic describes how to install and configure Ops Manager before deploying VMware Tanzu Kubernetes Grid Integrated Edition on vSphere.

Prerequisites
You use Ops Manager to install and configure Tanzu Kubernetes Grid Integrated Edition. 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

Each version of Tanzu Kubernetes Grid Integrated Edition is compatible with multiple versions of Ops Manager. To determine Ops Manager compatibility, see VMware Tanzu Network.

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

<table>
<thead>
<tr>
<th>Version</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ops Manager v2.7</td>
<td>1. Deploying Ops Manager on vSphere</td>
</tr>
<tr>
<td></td>
<td>2. Configuring BOSH Director on vSphere</td>
</tr>
<tr>
<td>Ops Manager v2.8</td>
<td>1. Deploying Ops Manager on vSphere</td>
</tr>
<tr>
<td></td>
<td>2. Configuring BOSH Director on vSphere</td>
</tr>
</tbody>
</table>

Next Installation Step

Please send any feedback you have to pks-feedback@pivotal.io.
Installing Tanzu Kubernetes Grid Integrated Edition on vSphere

In this topic

Prerequisites

Step 1: Install Tanzu Kubernetes Grid Integrated Edition
Step 2: Configure Tanzu Kubernetes Grid Integrated Edition
  Assign AZs and Networks
  TKGI API
  Plans
  Kubernetes Cloud Provider
  Networking
  UAA
  (Optional) Host Monitoring
  (Optional) In-Cluster Monitoring
  Tanzu Mission Control (Experimental)
  CEIP and Telemetry
  Errands
  Resource Config
Step 3: Apply Changes

Next Installation Step

Page last updated:

This topic describes how to install and configure VMware Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition:

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 Tanzu Kubernetes Grid Integrated Edition**

To install Tanzu Kubernetes Grid Integrated Edition, do the following:

1. Download the product file from [VMware Tanzu Network](https://tanzu.vmware.com/).

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 **Tanzu Kubernetes Grid Integrated Edition** in the left column, click the plus sign to add this product to your staging area.

**Step 2: Configure Tanzu Kubernetes Grid Integrated Edition**

Click the orange **Tanzu Kubernetes Grid Integrated Edition** tile to start the configuration process.

---

**warning**: When you configure the Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition fails.

---

Assign AZs and Networks

To configure the availability zones (AZs) and networks used by the Tanzu Kubernetes Grid Integrated Edition control plane:

1. Click **Assign AZs and Networks**.

2. Under **Place singleton jobs in**, select the AZ where you want to deploy the TKGI API and TKGI Database.

   - **Note:** You must specify the Balance other jobs in AZ, but the selection has no effect in the current version of Tanzu Kubernetes Grid Integrated Edition.

4. Under Network, select the infrastructure subnet that you created for Tanzu Kubernetes Grid Integrated Edition component VMs, such as the TKGI API and TKGI Database VMs.

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

6. Click Save.

TKGI API

Perform the following steps:

1. Click **TKGI API**.

2. Under **Certificate to secure the TKGI API**, provide a certificate and private key pair.
The certificate that you supply should cover the specific subdomain that routes to the TKGi 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 *.tkgi.EXAMPLE.com does not permit communication to *.api.tkgi.EXAMPLE.com. If the certificate does not contain the correct FQDN for the TKGi 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. Click **Generate RSA Certificate** for a new install or **Change** to update a previously-generated certificate.
b. Enter the domain for your API hostname. This must match the domain you configured under **TKGI API > API Hostname (FQDN)** in the Tanzu Kubernetes Grid Integrated Edition tile. It can be a standard FQDN or a wildcard domain.
c. Click **Generate**.
3. Under **API Hostname (FQDN)**, enter the FQDN that you registered to point to the TKGI API load balancer, such as `api.tkgi.example.com`. To retrieve the public IP address or FQDN of the TKGI 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 TKGI 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
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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition. If you select more than one AZ, Tanzu Kubernetes Grid Integrated Edition deploys the master VM in the first AZ and the worker VMs across the remaining AZs. If you are using multiple masters, Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition can deploy for each cluster. Enter any whole number in this field.

10. Under **Worker Node Instances**, specify the default number of Kubernetes worker nodes the TKGI CLI provisions for each cluster. The **Worker Node Instances** setting must be less than, or equal to, the **Maximum number of workers on a cluster** setting.

    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 TKGI 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 Tanzu Kubernetes Grid Integrated Edition 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. Tanzu Kubernetes Grid Integrated Edition 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
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.


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 Tanzu Kubernetes Grid Integrated Edition.

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.
Choose your iaas*

- GCP
- vSphere

vCenter Master Credentials *

user@example.com

***************

vCenter Host *

vcenter-example.com

Datacenter Name *

example-dc

Datastore Name *

example-ds

Stored VM Folder *

pks_vms

---

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 Tanzu Kubernetes Grid Integrated Edition*.

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. Tanzu Kubernetes Grid Integrated Edition 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**.

![Networking Configurations](image)

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.

Configure Tanzu Kubernetes Grid Integrated Edition to use your proxy and enable the following:

- TKGI API access to public Internet services and other internal services.
- Tanzu Kubernetes Grid Integrated Edition-deployed Kubernetes nodes access to public Internet services and other internal services.
- Tanzu Kubernetes Grid Integrated Edition Telemetry ability to forward Telemetry data to the CEIP and Telemetry program.

Note: This setting does not set the proxy for running Kubernetes workloads or pods.

4. 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 Tanzu Kubernetes Grid Integrated Edition communication.

The No Proxy list should include `127.0.0.1` and `localhost`.

Also include the following in the No Proxy list:

- Your Tanzu Kubernetes Grid Integrated Edition environment’s CIDRs, such as the service network CIDR where your Tanzu Kubernetes Grid Integrated Edition cluster is deployed, the deployment network CIDR, the node network IP block CIDR, and the pod network IP block CIDR.

- The FQDN of any registry, such as the Harbor API FQDN, or component communicating with Tanzu Kubernetes Grid Integrated Edition, using a hostname instead of an IP address.

- The IP addresses for your NSX Manager, vCenter Server, and all ESXi hosts, if you are upgrading and have an existing proxy configuration for reaching a Docker registry or other external services.

- 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,
exmaple3.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`, and your Tanzu Kubernetes Grid Integrated Edition FQDN are not proxied. This allows internal Tanzu Kubernetes Grid Integrated Edition communication.

Do not use the `_` character in the No Proxy field. Entering an underscore character in this field can cause upgrades to fail.

Because some jobs in the VMs accept `*` as a wildcard, while others only accept `.`, we recommend that you define a wildcard domain using both of them. For example, to denote `example.com` as a wildcard domain, add both `*.example.com` and `example.com` to the No Proxy property.

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

6. Click Save.

UAA

To configure the UAA server:
1. Click **UAA**.

2. Under **TKGI API Access Token Lifetime**, enter a time in seconds for the TKGI API access token lifetime. This field defaults to 600.

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

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

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

6. Under **Configure created clusters to use UAA as the OIDC provider**, select **Enabled** or **Disabled**. This is a global default setting for TKGI-provisioned clusters. For more information, see [OIDC Provider for Kubernetes Clusters](#).

To configure Tanzu Kubernetes Grid Integrated Edition to use UAA as the OIDC provider:

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

---

Note: VMware recommends using the default UAA token timeout values. By default, access tokens expire after ten minutes and refresh tokens expire after six hours.
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`. The default value is `oidc:`.

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, you 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`. The default value is `oidc:`.

⚠️ **warning:** VMware recommends adding OIDC prefixes to prevent users and groups from gaining unintended cluster privileges. If you change the above values for a pre-existing Tanzu Kubernetes Grid Integrated Edition 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, 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 Tanzu Kubernetes Grid Integrated Edition to an LDAP Server**.
- To use SAML for UAA, select **SAML Identity Provider** and continue to **Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider**.

**Optional** Host Monitoring

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

- To configure Syslog, 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 TKGI API, master node, and worker node VMs to a monitoring service, such as Wavefront or Datadog.

For more information about these components, see [Monitoring TKGI and TKGI-Provisioned Clusters](#).

### Configure PKS Monitoring Features on Host

<table>
<thead>
<tr>
<th>Feature</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Syslog for PKS?</td>
<td>No, Yes</td>
</tr>
<tr>
<td>Enable VMware vRealize Log Insight Integration?</td>
<td>No, Yes</td>
</tr>
<tr>
<td>Enable Telegraf Outputs?</td>
<td>No, Yes</td>
</tr>
</tbody>
</table>

To configure Syslog for all BOSH-deployed VMs in Tanzu Kubernetes Grid Integrated Edition:

1. Click [Host Monitoring](#).
2. Under **Enable Syslog for TKGI**, select **Yes**.
3. Under **Address**, enter the destination syslog endpoint.
4. Under **Port**, enter the destination syslog port.
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. VMware 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.]

1. By default, vRLI logging is disabled. To configure vRLI logging, under **Enable VMware vRealize Log Insight Integration?**, select **Yes** and then perform the following steps:

   ![Enable VMware vRealize Log Insight Integration?](image)

   2. Under **Host**, enter the IP address or FQDN of the vRLI host.

   3. **(Optional) Select the Enable SSL? checkbox to encrypt the logs being sent to vRLI using SSL.**
4. 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.

5. (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.

6. 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.

7. 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 Tanzu Kubernetes Grid Integrated Edition tile does not validate your vRLI configuration settings. To verify your setup, look for log entries in vRLI.

**Telegraf**

To configure Tanzu Kubernetes Grid Integrated Edition to use Telegraf for metric collection:

1. Create a configuration file for your monitoring service. For instructions, see [Create a Configuration File](#).

2. Return to the Tanzu Kubernetes Grid Integrated Edition tile > **Settings** > **Host Monitoring**.

3. Under **Enable Telegraf Outputs?**, select **Yes**.

4. Configure the Telegraf checkboxes as described in the table below. Components you enable in this step will be visible to TKGI admins only.

<table>
<thead>
<tr>
<th>Enable this checkbox...</th>
<th>...to send these metrics to your monitoring service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable node exporter on TKGI API</td>
<td>Node Exporter metrics from the TKGI API VM</td>
</tr>
<tr>
<td>Enable node exporter on master</td>
<td>Node Exporter metrics from Kubernetes master nodes</td>
</tr>
<tr>
<td>Include etcd metrics</td>
<td>etcd server and debugging metrics</td>
</tr>
<tr>
<td>Enable node exporter on worker</td>
<td>Node Exporter metrics from Kubernetes worker nodes</td>
</tr>
</tbody>
</table>
### Include Kubernetes Controller Manager metrics

Kubernetes controller manager metrics

- These metrics provide information about the state of each cluster.

### Include Kubernetes API Server metrics

Kubernetes API server metrics

### Include kubelet metrics

kubelet metrics for all workloads running in all your Kubernetes clusters

- If you enable **Include kubelet metrics**, be prepared for a high volume of metrics.

5. In **Set Up Telegraf Outputs**, replace the default value `[[outputs.discard]]` with the contents of the configuration file that you created above. See the following example for an HTTP output plugin:

```yaml
[[outputs.http]]
  url="https://example.com"
  method="POST"
  data_format="json"
[[processors.override]]
  director = "bosh-director-1"
```

6. 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition.

   1. In the Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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


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 TKGI, 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 TKGI-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, Tanzu Kubernetes Grid Integrated Edition 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, Tanzu Kubernetes Grid Integrated Edition 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 TKGI-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, Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition tile to integrate their Tanzu Kubernetes Grid Integrated Edition deployment with Tanzu Mission Control.

Tanzu Mission Control integration lets you monitor and manage Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition with Tanzu Mission Control:

1. Confirm that the TKGI API VM has internet access and can connect to cna.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.


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 Tanzu Kubernetes Grid Integrated Edition clusters in Tanzu Mission Control.

4. Click **Save**.

**warning**: After the Tanzu Kubernetes Grid Integrated Edition 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 Telemetry Program, do the following:

1. Click **CEIP and Telemetry**.

2. Review the information about the 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 ("TKGI") on a regular basis. Since TKGI 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 TKGI, you should not join below. You may join or leave CEIP and Telemetry for TKGI 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 TKGI, but is not identifiable to your organization. This participation tier allows us to identify your organization so we may provide proactive support and other benefits.

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 TKGI 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
- 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 Tanzu Kubernetes Grid Integrated Edition installation.
- **Standard:** If you select this option, data is collected from your Tanzu Kubernetes Grid Integrated Edition installation to improve Tanzu Kubernetes Grid Integrated Edition. 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 Tanzu Kubernetes Grid Integrated Edition 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 account number or 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 started as a Pivotal customer, you can find your Customer Number in your Order Confirmation email.

(Optional) Enter a descriptive name for your TKGI installation in the **TKGI 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 **TKGI Installation Type** list.

6. Click **Save**.

**Note:** If you join the CEIP and Telemetry Program for Tanzu Kubernetes Grid Integrated Edition, open your firewall to allow outgoing access to https://vcsa.vmware.com/ph on port 443.

**Note:** Even if you select None, Tanzu Kubernetes Grid Integrated Edition-provisioned clusters send usage data to the TKGI control plane. However, this data is not sent to VMware and remains on your Tanzu Kubernetes Grid Integrated Edition installation.

**Errands**

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

To configure which post-deploy and pre-delete errands run for Tanzu Kubernetes Grid Integrated Edition:

1. Make a selection in the dropdown next to each errand.
### Errands

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

#### Post-Deploy Errands

<table>
<thead>
<tr>
<th>Errand Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSX-T Validation errand</td>
<td>Validates NSX-T configuration</td>
</tr>
<tr>
<td>Run smoke tests</td>
<td>Run smoke tests to validate TKGI lifecycle operations</td>
</tr>
<tr>
<td>Upgrade all clusters errand</td>
<td>Upgrades all Kubernetes clusters provisioned by TKGI after the TKGI Tile upgrade is applied</td>
</tr>
<tr>
<td>Create pre-defined Wavefront alerts</td>
<td>Create pre-defined Wavefront alerts</td>
</tr>
</tbody>
</table>

#### Pre-Delete Errands

<table>
<thead>
<tr>
<th>Errand Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete all clusters errand</td>
<td>Deletes all clusters provisioned by TKGI when the TKGI tile is deleted</td>
</tr>
<tr>
<td>Delete pre-defined Wavefront alerts</td>
<td>Delete pre-defined Wavefront alerts</td>
</tr>
</tbody>
</table>

### Note:

We recommend that you use the default settings for all errands except for the **Run smoke tests** errand.

2. *(Optional)* Set the **Run smoke tests errand** to **On**.

   This errand uses the TKGI CLI to create a Kubernetes cluster and then delete it. If the creation or deletion fails, the errand fails and the installation of the Tanzu Kubernetes Grid Integrated Edition tile is aborted.

3. *(Optional)* To ensure that all of your cluster VMs are patched, configure the **Upgrade all clusters errand** to **On**.

   Updating the Tanzu Kubernetes Grid Integrated Edition tile with a new Linux stemcell and the **Upgrade all clusters errand** enabled triggers the rolling of every Linux VM in each Kubernetes cluster. Similarly, updating the Tanzu Kubernetes Grid Integrated Edition tile with a new Windows stemcell triggers the rolling of every Windows VM in your Kubernetes clusters.

   **warning:** To avoid workload downtime, use the resource configuration recommended in About Tanzu Kubernetes Grid Integrated Edition Upgrades and Maintaining Workload Uptime.

### Resource Config

To modify the resource configuration of Tanzu Kubernetes Grid Integrated Edition, follow the steps below:

1. Select **Resource Config**.
2. For each job, review the Automatic values in the following fields:

- **VM TYPE**: By default, the TKGI Database and TKGI API jobs are set to the same Automatic VM type. If you want to adjust this value, we recommend that you select the same VM type for both jobs.

  ![Note: The Automatic VM TYPE values match the recommended resource configuration for the TKGI API and TKGI Database jobs.]

- **PERSISTENT DISK TYPE**: By default, the TKGI Database and TKGI API jobs are set to the same persistent disk type. If you want to adjust this value, you can change the persistent disk type for each of the jobs independently. Using the same persistent disk type for both jobs is not required.

3. Under each job, leave **NSX-T CONFIGURATION** and **NSX-V CONFIGURATION** blank.

### 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 TKGI API load balancer, follow the instructions in **Configure TKGI API Load Balancer**.

---

Please send any feedback you have to pks-feedback@pivotal.io.
Configuring a TKGI API Load Balancer

Overview

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

To set up an external load balancer for the TKGI API, do the following after you install the Tanzu Kubernetes Grid Integrated Edition tile:

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

Prerequisites

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

- The TKGI API certificate that you provided in the [Tanzu Kubernetes Grid Integrated Edition tile > TKGI API > Certificate to secure the TKGI API](#).
- The TKGI API hostname that you entered in the [Tanzu Kubernetes Grid Integrated Edition tile > TKGI API > API Hostname (FQDN)](#).

Step 1: Retrieve the TKGI API Endpoint

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

To retrieve the TKGI API endpoint, do the following:

1. Navigate to the Ops Manager [Installation Dashboard](#).
2. Click the [Tanzu Kubernetes Grid Integrated Edition tile](#).
3. Click the **Status** tab and locate the **TKGI API** job. The IP address of the TKGI API job is the TKGI API endpoint.

**Step 2: Configure an External Load Balancer**

To set up an external load balancer for the TKGI API, configure the external load balancer to resolve to the domain name you entered in the **Tanzu Kubernetes Grid Integrated Edition** tile > **TKGI API** > **API Hostname (FQDN)** using the following information:

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

**Next Installation Step**

To set up Tanzu Kubernetes Grid Integrated Edition admin users who can create and manage Kubernetes clusters, follow the instructions in **Setting Up Tanzu Kubernetes Grid Integrated Edition Admin Users on vSphere**

Please send any feedback you have to pks-feedback@pivotal.io.
Setting Up Tanzu Kubernetes Grid Integrated Edition Admin Users on vSphere

In this topic
- Overview
- Prerequisites
- Step 1: Connect to the TKGI API VM
- Step 2: Log In as a UAA Admin
- Step 3: Assign Tanzu Kubernetes Grid Integrated Edition Cluster Scopes

Next Step

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

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

Overview

UAA is the identity management service for Tanzu Kubernetes Grid Integrated Edition. Tanzu Kubernetes Grid Integrated Edition includes a UAA server, which is hosted on the TKGI API 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 Tanzu Kubernetes Grid Integrated Edition, you must have one of the following:

- SSH access to the Ops Manager VM
- A machine that can connect to your TKGI API VM

Step 1: Connect to the TKGI API VM

You can connect to the TKGI API 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 TKGI API VM by logging in to the Ops Manager VM through SSH.

To SSH into the Ops Manager VM on vSphere, do the following:
1. Locate the credentials that were used to import the Ops Manager .ova or .ovf file into your virtualization system. You configured these credentials when you installed Ops Manager and used them to complete the **Prepare vSphere** steps in *Deploying Ops Manager on vSphere*.

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

   ```bash
   chmod 600 PRIVATE-KEY
   ```

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

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

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

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

   For example:

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

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

**Option 2: Connect through a Non-Ops Manager Machine**

To connect to the TKGI API VM and run UAA commands, do the following:

1. Install UAAC on your machine. For example:

   ```bash
   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 TKGI 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 **Tanzu Kubernetes Grid Integrated Edition** 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://TKGI-API:8443 --ca-cert CERTIFICATE-PATH
```

Where:
- **TKGI-API** is the domain name of your TKGI API server. You entered this domain name in the Tanzu Kubernetes Grid Integrated Edition tile > TKGI > API Hostname (FQDN).
- **CERTIFICATE-PATH** is the path to your Ops Manager root CA certificate. Provide this certificate to validate the TKGI 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.tkgi.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 Tanzu Kubernetes Grid Integrated Edition Cluster Scopes**


To create Tanzu Kubernetes Grid Integrated Edition 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 TKGI cluster scopes to an individual user, see Grant Tanzu Kubernetes Grid Integrated Edition Access to an Individual User. Follow this procedure if you selected Internal UAA when you configured UAA in the Tanzu Kubernetes Grid Integrated Edition tile. For more information, see Installing Tanzu Kubernetes Grid Integrated Edition on vSphere.
- To assign TKGI cluster scopes to an LDAP group, see Grant Tanzu Kubernetes Grid Integrated Edition Access to an External LDAP Group. Follow this procedure if you selected LDAP Server when you configured UAA in the Tanzu Kubernetes Grid Integrated Edition tile. For more information, see Installing Tanzu Kubernetes Grid Integrated Edition TKGI on vSphere.
- To assign TKGI cluster scopes to a SAML group, see Grant Tanzu Kubernetes Grid Integrated Edition Access to an External SAML Group. Follow this procedure if you selected SAML Identity Provider when you configured UAA in the Tanzu Kubernetes Grid Integrated Edition tile. For more information, see Installing Tanzu Kubernetes Grid Integrated Edition TKGI on vSphere.
- To assign TKGI cluster scopes to a client, see Grant Tanzu Kubernetes Grid Integrated Edition Access to a Client.
Next Step

After you create admin users in Tanzu Kubernetes Grid Integrated Edition, the admin users can create and manage Kubernetes clusters in Tanzu Kubernetes Grid Integrated Edition. For more information, see Managing Kubernetes Clusters and Workloads.

Please send any feedback you have to pks-feedback@pivotal.io.
Install Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T Using Ops Manager

In this topic

Step 1: Prepare to Install Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T
Step 2: Install and Configure NSX-T Data Center for Tanzu Kubernetes Grid Integrated Edition
Step 3: Create the Management Plane for Tanzu Kubernetes Grid Integrated Edition
Step 4: Create the Compute Plane for Tanzu Kubernetes Grid Integrated Edition
Step 5: Deploy Ops Manager for Tanzu Kubernetes Grid Integrated Edition with NSX-T
Step 6: Generate the NSX-T Management Cluster Root CA Certificate and Key
Step 7: Configure BOSH Director for vSphere with NSX-T
Step 8: Generate and Register the NSX-T Management Cluster Super User Principal Identity Certificate and Key
Step 9: Install Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T
Step 11: Install the TKGI and Kubectl CLIs
Step 12: Create Admin Users for Tanzu Kubernetes Grid Integrated Edition
Step 13: Verify the Installation of Tanzu Kubernetes Grid Integrated Edition
Step 14: Perform Desired Post-Installation Configurations
Step 15: Create Network Profiles to Customize Cluster Deployments

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic lists the procedures to follow to install Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T networking manually, using Ops Manager.

Note: The recommended method for installing Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T is to use the Tanzu Kubernetes Grid Integrated Edition Management Console. For information, see Install on vSphere with the Management Console.

Step 1: Prepare to Install Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T

In preparation for installing Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T Data Center, review all of the topics in the subsection Preparing to Install Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T Data Center

Step 2: Install and Configure NSX-T Data Center for Tanzu Kubernetes Grid Integrated Edition

NSX-T Data Center must be installed and configured before you install Tanzu Kubernetes Grid Integrated Edition.
For instructions, see one of the following:

- Installing and Configuring NSX-T Data Center v3.0 for Tanzu Kubernetes Grid Integrated Edition
- Installing and Configuring NSX-T Data Center v2.5 for Enterprise PKS in the v1.7 documentation.

**Step 3: Create the Management Plane for Tanzu Kubernetes Grid Integrated Edition**

Prepare the vSphere and NSX-T infrastructure for the Tanzu Kubernetes Grid Integrated Edition Management Plane where Ops Manager, BOSH Director, Tanzu Kubernetes Grid Integrated Edition components, and Harbor Registry are deployed. This includes creating a vSphere resource pool for Tanzu Kubernetes Grid Integrated Edition 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 Create Management Plane in Installing and Configuring NSX-T Data Center v3.0 for TKGI.

**Step 4: Create the Compute Plane for Tanzu Kubernetes Grid Integrated Edition**

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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition. The network also provides IP addresses for Kubernetes API access and Kubernetes exposed services.

For instructions, see Create IP Blocks and Pool for Compute Plane in Installing and Configuring NSX-T Data Center v3.0 for TKGI.

**Step 5: Deploy Ops Manager for Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition

**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 Generate and Register the NSX-T Management SSL Certificate and Private Key.

**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 Tanzu Kubernetes Grid Integrated Edition
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 Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T

At this point your NSX-T environment is prepared for Tanzu Kubernetes Grid Integrated Edition installation using the Tanzu Kubernetes Grid Integrated Edition tile in Ops Manager.

For instructions, see Installing Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T.


The VMware Harbor Registry is recommended for Tanzu Kubernetes Grid Integrated Edition. Install Harbor in the NSX Management Plane with other Tanzu Kubernetes Grid Integrated Edition components, such as the TKGI API and TKGI database, 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 TKGI management network. See Create DNAT Rule.

For instructions, see Installing VMware Harbor Registry.

Step 11: Install the TKGI and Kubectl CLIs

See Installing the TKGI CLI and Installing the Kubernetes CLI.

Step 12: Create Admin Users for Tanzu Kubernetes Grid Integrated Edition


Step 13: Verify the Installation of Tanzu Kubernetes Grid Integrated Edition

Create a Kubernetes cluster using the TKGI 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 Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T, refer to the following subsection for topics describing additional NSX-T configuration options: Advanced Configurations for Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T Data Center

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

**Page last updated:**

This topic lists the sections to follow when installing VMware Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T Data Center.

Preparing to Install Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T

In preparation for installing Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T, review the following documentation:

- Version Requirements for Tanzu Kubernetes Grid Integrated Edition
- Hardware Requirements for Tanzu Kubernetes Grid Integrated Edition
- Firewall Ports and Protocols Requirements
- NSX-T Deployment Topologies for Tanzu Kubernetes Grid Integrated Edition
- NSX-T Cluster Objects Created for Tanzu Kubernetes Grid Integrated Edition
- Network Planning Tanzu Kubernetes Grid Integrated Edition

Please send any feedback you have to pks-feedback@pivotal.io.
vSphere with NSX-T Version Requirements

In this topic
- vSphere Version Requirements
- NSX-T Integration Component Version Requirements

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes the version requirements for installing VMware Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T integration.

For prerequisites and resource requirements for installing Tanzu Kubernetes Grid Integrated Edition on vSphere without NSX-T integration, see vSphere Prerequisites and Resource Requirements.

For hardware and resource requirements for deploying Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T in production environments, see Hardware Requirements for Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T.

Tanzu Kubernetes Grid Integrated Edition 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 VMware Partner documentation.

vSphere Version Requirements

For Tanzu Kubernetes Grid Integrated Edition on vSphere version requirements, refer to the VMware Product Interoperability Matrices.

NSX-T Integration Component Version Requirements


Please send any feedback you have to pks-feedback@pivotal.io.
Hardware Requirements for Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T

In this topic

- vSphere Clusters for Tanzu Kubernetes Grid Integrated Edition
  - Tanzu Kubernetes Grid Integrated Edition Management Cluster
  - Tanzu Kubernetes Grid Integrated Edition Edge Cluster
  - Tanzu Kubernetes Grid Integrated Edition Compute Cluster
  - Tanzu Kubernetes Grid Integrated Edition Management Plane Placement Considerations
- Configuration Requirements for vSphere Clusters for Tanzu Kubernetes Grid Integrated Edition
- RPD for Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T
  - RPD for Tanzu Kubernetes Grid Integrated Edition with vSAN
  - RPD for Tanzu Kubernetes Grid Integrated Edition without vSAN
- MPD for Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T
  - MPD Topology
  - MPD Configuration Requirements
  - MPD Considerations
- VM Inventory and Sizes
  - Management Plane VMs and Sizes
  - NSX-T Edge Node VMs and Sizes
  - Kubernetes Cluster Nodes VMs and Sizes
- Hardware Requirements
  - RPD Hardware Requirements
  - MPD Hardware Requirements
  - Adding Hardware Capacity

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic provides hardware requirements for production deployments of VMware Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T.

vSphere Clusters for Tanzu Kubernetes Grid Integrated Edition

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

- Tanzu Kubernetes Grid Integrated Edition Management Cluster
- Tanzu Kubernetes Grid Integrated Edition Edge Cluster
- Tanzu Kubernetes Grid Integrated Edition Compute Cluster

For more information on creating vSphere clusters, see Creating Clusters in the vSphere documentation.
Tanzu Kubernetes Grid Integrated Edition Management Cluster

The Tanzu Kubernetes Grid Integrated Edition Management Cluster on vSphere comprises the following components:

- vCenter Server
- NSX-T Manager v2.5 or later (quantity 3)

For more information, see Installing and Configuring NSX-T Data Center v3.0 for TKGI.

Tanzu Kubernetes Grid Integrated Edition Edge Cluster

The Tanzu Kubernetes Grid Integrated Edition 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. Tanzu Kubernetes Grid Integrated Edition supports running Edge Node pairs in active/standby mode only.

For more information, see Installing and Configuring NSX-T Data Center v3.0 for TKGI.

Tanzu Kubernetes Grid Integrated Edition Compute Cluster

The Tanzu Kubernetes Grid Integrated Edition Compute Cluster on vSphere comprises the following components:

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

For more information, see Installing and Configuring NSX-T Data Center v3.0 for TKGI.

Tanzu Kubernetes Grid Integrated Edition Management Plane Placement Considerations

The Tanzu Kubernetes Grid Integrated Edition Management Plane comprises the following components:

- Ops Manager
- BOSH Director
- TKGI Control Plane
- VMware Harbor Registry


Configuration Requirements for vSphere Clusters for Tanzu Kubernetes Grid Integrated Edition

For each vSphere cluster defined for Tanzu Kubernetes Grid Integrated Edition, 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

RDP for Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T

The recommended production deployment (RPD) topology represents the VMware-recommended configuration to run production workloads in Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T.

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

RDP for Tanzu Kubernetes Grid Integrated Edition with vSAN

The RPD for Tanzu Kubernetes Grid Integrated Edition 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 cluster 1 (AZ1) with three ESXi hosts.
  - Compute cluster 2 (AZ2) with three ESXi hosts.
  - Compute cluster 3 (AZ3) with three ESXi hosts.
- Each Compute cluster runs one instance of an Tanzu Kubernetes Grid Integrated Edition-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 Tanzu Kubernetes Grid Integrated Edition, 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 Tanzu Kubernetes Grid Integrated Edition without vSAN**

The RPD for Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition-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 Tanzu Kubernetes Grid Integrated Edition components and Persistent Volumes (PVs) for Kubernetes pods.
All datastores can be collapsed to a 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 Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T

The minimum production deployment (MPD) topology represents the baseline requirements for running Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T.

**Note:** The MPD topology for Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition.

**MPD Topology**

The MPD topology for Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition deployment requires a minimum of three nodes, Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition, 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 Tanzu Kubernetes Grid Integrated Edition, keep in mind the following:

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

For more information, see [Create Management Plane](#) and [Create IP Blocks and Pool for Compute Plane](#) in *Installing and Configuring NSX-T Data Center v3.0 for TKGI*.

**VM Inventory and Sizes**

The following tables list the VMs and their sizes for deployments of Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T.

**Management Plane VMs and Sizes**

The following table lists the resource requirements for NSX-T infrastructure and Tanzu Kubernetes Grid Integrated Edition Management Plane VMs.

<table>
<thead>
<tr>
<th>VM</th>
<th>CPU Cores</th>
<th>Memory (GB)</th>
<th>Ephemeral Disk (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOSH Director</td>
<td>2</td>
<td>8</td>
<td>103</td>
</tr>
<tr>
<td>Harbor Registry</td>
<td>2</td>
<td>8</td>
<td>167</td>
</tr>
</tbody>
</table>

Copyright © 2020 VMware, Inc. All Rights Reserved.
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 TKGI Database VM as follows:

<table>
<thead>
<tr>
<th>Number of Pods</th>
<th>Persistent Disk Requirements (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000 pods</td>
<td>20</td>
</tr>
<tr>
<td>5,000 pods</td>
<td>100</td>
</tr>
<tr>
<td>10,000 pods</td>
<td>200</td>
</tr>
<tr>
<td>50,000 pods</td>
<td>1,000</td>
</tr>
</tbody>
</table>

NSX-T Edge Node VMs and Sizes

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

<table>
<thead>
<tr>
<th>VM</th>
<th>CPU Cores</th>
<th>Memory (GB)</th>
<th>Ephemeral Disk (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSX-T Edge Node 1</td>
<td>8</td>
<td>32</td>
<td>120</td>
</tr>
<tr>
<td>NSX-T Edge Node 2</td>
<td>8</td>
<td>32</td>
<td>120</td>
</tr>
<tr>
<td>TOTAL</td>
<td>16</td>
<td>64</td>
<td>0.25</td>
</tr>
</tbody>
</table>

* Intel CPUs only.

**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 Tanzu Kubernetes Grid Integrated Edition tile.

<table>
<thead>
<tr>
<th>VM</th>
<th>CPU Cores</th>
<th>Memory (GB)</th>
<th>Ephemeral Disk (GB)</th>
<th>Persistent Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master Nodes</td>
<td>1 to 16</td>
<td>1 to 64</td>
<td>8 to 256</td>
<td>1 GB to 32 TB</td>
</tr>
</tbody>
</table>
For illustrative purposes, the following table shows sizing information for two example Kubernetes clusters. Each cluster has three master nodes and five worker nodes.

<table>
<thead>
<tr>
<th>VM</th>
<th>VM Count</th>
<th>CPU Cores</th>
<th>Memory (GB)</th>
<th>Ephemeral Disk (GB)</th>
<th>Persistent Disk (TB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master Nodes</td>
<td>6</td>
<td>2</td>
<td>8</td>
<td>64</td>
<td>128</td>
</tr>
<tr>
<td>Worker Nodes</td>
<td>10</td>
<td>4</td>
<td>16</td>
<td>64</td>
<td>256</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>16</strong></td>
<td><strong>52</strong></td>
<td><strong>208</strong></td>
<td><strong>1.0</strong></td>
<td><strong>3.4</strong></td>
</tr>
</tbody>
</table>

**Hardware Requirements**

The following tables list the hardware requirements for RDP and MPD topologies for Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T.

**RPD Hardware Requirements**

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

<table>
<thead>
<tr>
<th>VM</th>
<th>VM Count</th>
<th>CPU Cores (with HT)</th>
<th>Memory (GB)</th>
<th>NICS</th>
<th>Shared Persistent Disk (TB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management/Edge Cluster</td>
<td>3</td>
<td>16</td>
<td>98</td>
<td>2x 10GbE</td>
<td>1.5</td>
</tr>
<tr>
<td>Compute cluster1 (AZ1)</td>
<td>3</td>
<td>6</td>
<td>48</td>
<td>2x 10GbE</td>
<td>192</td>
</tr>
<tr>
<td>Compute cluster2 (AZ2)</td>
<td>3</td>
<td>6</td>
<td>48</td>
<td>2x 10GbE</td>
<td>192</td>
</tr>
<tr>
<td>Compute cluster3 (AZ3)</td>
<td>3</td>
<td>6</td>
<td>48</td>
<td>2x 10GbE</td>
<td>192</td>
</tr>
</tbody>
</table>

**Note:** The **CPU Cores** values assume the use of hyper-threading (HT).

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

<table>
<thead>
<tr>
<th>VM</th>
<th>VM Count</th>
<th>CPU Cores (with HT)</th>
<th>Memory (GB)</th>
<th>NICS</th>
<th>Shared Persistent Disk (TB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management/Edge Cluster</td>
<td>3</td>
<td>16</td>
<td>98</td>
<td>2x 10GbE</td>
<td>1.5</td>
</tr>
<tr>
<td>Compute cluster1 (AZ1)</td>
<td>2</td>
<td>10</td>
<td>70</td>
<td>2x 10GbE</td>
<td>192</td>
</tr>
<tr>
<td>Compute cluster2 (AZ2)</td>
<td>2</td>
<td>10</td>
<td>70</td>
<td>2x 10GbE</td>
<td>192</td>
</tr>
<tr>
<td>Compute cluster3 (AZ3)</td>
<td>2</td>
<td>10</td>
<td>70</td>
<td>2x 10GbE</td>
<td>192</td>
</tr>
</tbody>
</table>

**Note:** The **CPU Cores** 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.

<table>
<thead>
<tr>
<th>VM</th>
<th>VM Count</th>
<th>CPU Cores (with HT)</th>
<th>Memory (GB)</th>
<th>NICS</th>
<th>Shared Persistent Disk (TB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worker Nodes</td>
<td>1 to 16</td>
<td>1 to 64</td>
<td>8 to 256</td>
<td>1 GB</td>
<td>to 32 TB</td>
</tr>
</tbody>
</table>

Copyright © 2020 VMware, Inc. All Rights Reserved.
| Collapsed Cluster | 3 | 32 | 236 | 4x 10GbE or 2x pNICs* | 5.9 |

*If necessary, you can use two pNICs. For more information, see [Fully Collapsed vSphere Cluster NSX-T Deployment](#) in the NSX-T Data Center documentation.

Adding Hardware Capacity

To add hardware capacity to your Tanzu Kubernetes Grid Integrated Edition 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 [Configure vSphere Networking for ESXi Hosts](#) in *Installing and Configuring NSX-T Data Center v3.0 for TKGI*.

Please send any feedback you have to pks-feedback@pivotal.io.
This topic describes the firewall ports and protocols requirements for using VMware Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition, VMware 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 Tanzu Kubernetes Grid Integrated Edition, 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 Tanzu Kubernetes Grid Integrated Edition deployment.

**Note**: To control which groups access deploying and scaling your organization’s Tanzu Kubernetes Grid Integrated Edition-deployed Kubernetes clusters, configure your firewall settings as described on the Operator –> TKGI API server lines below.

**Tanzu Kubernetes Grid Integrated Edition Ports and Protocols**

The following tables list ports and protocols required for network communications between Tanzu Kubernetes Grid Integrated Edition v1.5.0 and later, and vSphere 6.7 and NSX-T 2.4.0.1 and later.
Tanzu Kubernetes Grid Integrated Edition Users Ports and Protocols

The following table lists ports and protocols used for network communication between Tanzu Kubernetes Grid Integrated Edition user interface components.

<table>
<thead>
<tr>
<th>Source Component</th>
<th>Destination Component</th>
<th>Destination Protocol</th>
<th>Destination Port</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admin/Operator Console</td>
<td>All System Components</td>
<td>TCP</td>
<td>22</td>
<td>ssh</td>
</tr>
<tr>
<td>Admin/Operator Console</td>
<td>All System Components</td>
<td>TCP</td>
<td>80</td>
<td>http</td>
</tr>
<tr>
<td>Admin/Operator Console</td>
<td>All System Components</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Admin/Operator Console</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25555</td>
<td>bosh director rest api</td>
</tr>
<tr>
<td>Admin/Operator Console</td>
<td>NSX-T API VIP</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Admin/Operator Console</td>
<td>Ops Manager</td>
<td>TCP</td>
<td>22</td>
<td>ssh</td>
</tr>
<tr>
<td>Admin/Operator Console</td>
<td>Ops Manager</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Admin/Operator Console</td>
<td>TKGI Controller</td>
<td>TCP</td>
<td>9021</td>
<td>tkgi api server</td>
</tr>
<tr>
<td>Admin/Operator Console</td>
<td>vCenter Server</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Admin/Operator Console</td>
<td>vCenter Server</td>
<td>TCP</td>
<td>5480</td>
<td>vami</td>
</tr>
<tr>
<td>Admin/Operator Console</td>
<td>vSphere ESXI Hosts Mgmt. vmknic</td>
<td>TCP</td>
<td>902</td>
<td>ideafarm-door</td>
</tr>
<tr>
<td>Admin/Operator and Developer Consoles</td>
<td>Harbor Private Image Registry</td>
<td>TCP</td>
<td>80</td>
<td>http</td>
</tr>
<tr>
<td>Admin/Operator and Developer Consoles</td>
<td>Harbor Private Image Registry</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Admin/Operator and Developer Consoles</td>
<td>Harbor Private Image Registry</td>
<td>TCP</td>
<td>4443</td>
<td>notary</td>
</tr>
<tr>
<td>Admin/Operator and Developer Consoles</td>
<td>Kubernetes App Load-Balancer Svc</td>
<td>TCP/UDP</td>
<td>Varies</td>
<td>varies with apps</td>
</tr>
<tr>
<td>Admin/Operator and Developer Consoles</td>
<td>Kubernetes Cluster API Server-LB VIP</td>
<td>TCP</td>
<td>8443</td>
<td>httpsca</td>
</tr>
<tr>
<td>Admin/Operator and Developer Consoles</td>
<td>Kubernetes Cluster Ingress Controller</td>
<td>TCP</td>
<td>80</td>
<td>http</td>
</tr>
<tr>
<td>Admin/Operator and Developer Consoles</td>
<td>Kubernetes Cluster Ingress Controller</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Admin/Operator and Developer Consoles</td>
<td>Kubernetes Cluster Worker Node</td>
<td>TCP/UDP</td>
<td>30000-32767</td>
<td>kubernetes nodeport</td>
</tr>
<tr>
<td>Admin/Operator and Developer Consoles</td>
<td>TKGI Controller</td>
<td>TCP</td>
<td>8443</td>
<td>httpsca</td>
</tr>
<tr>
<td>All User Consoles (Operator, Developer, Consumer)</td>
<td>Kubernetes App Load-Balancer Svc</td>
<td>TCP/UDP</td>
<td>Varies</td>
<td>varies with apps</td>
</tr>
<tr>
<td>All User Consoles (Operator, Developer, Consumer)</td>
<td>Kubernetes Cluster Ingress Controller</td>
<td>TCP</td>
<td>80</td>
<td>http</td>
</tr>
<tr>
<td>All User Consoles (Operator, Developer, Consumer)</td>
<td>Kubernetes Cluster Ingress Controller</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
</tbody>
</table>
### Tanza Kubernetes Grid Integrated Edition Core Ports and Protocols

The following table lists ports and protocols used for network communication between core Tanzu Kubernetes Grid Integrated Edition components.

<table>
<thead>
<tr>
<th>Source Component</th>
<th>Destination Component</th>
<th>Destination Protocol</th>
<th>Destination Port</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>All User Consoles (Operator, Developer, Consumer)</td>
<td>Kubernetes Cluster Worker Node</td>
<td>TCP/UDP</td>
<td>30000-32767</td>
<td>kubernetes nodeport</td>
</tr>
</tbody>
</table>

**Note:** The `type:NodePort` service type is not supported for Tanzu Kubernetes Grid Integrated Edition deployments on vSphere with NSX-T. Only `type:LoadBalancer` and Services associated with Ingress rules are supported on vSphere with NSX-T.
<table>
<thead>
<tr>
<th>Source Component</th>
<th>Destination Component</th>
<th>Destination Protocol</th>
<th>Destination Port</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOSH Compilation Job VM</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>4222</td>
<td>bosh nats server</td>
</tr>
<tr>
<td>BOSH Compilation Job VM</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25250</td>
<td>bosh blobstore</td>
</tr>
<tr>
<td>BOSH Compilation Job VM</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25923</td>
<td>health monitor daemon</td>
</tr>
<tr>
<td>BOSH Compilation Job VM</td>
<td>Harbor Private Image Registry</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>BOSH Compilation Job VM</td>
<td>Harbor Private Image Registry</td>
<td>TCP</td>
<td>8853</td>
<td>bosh dns health</td>
</tr>
<tr>
<td>TKGI Controller</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>4222</td>
<td>bosh nats server</td>
</tr>
<tr>
<td>TKGI Controller</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>8443</td>
<td>httpsca</td>
</tr>
<tr>
<td>TKGI Controller</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25250</td>
<td>bosh blobstore</td>
</tr>
<tr>
<td>TKGI Controller</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25555</td>
<td>bosh director rest api</td>
</tr>
<tr>
<td>TKGI Controller</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25923</td>
<td>health monitor daemon</td>
</tr>
<tr>
<td>TKGI Controller</td>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>TCP</td>
<td>8443</td>
<td>httpsca</td>
</tr>
<tr>
<td>TKGI Controller</td>
<td>TKGI Database VM</td>
<td>TCP</td>
<td>3306</td>
<td>tkgi db proxy</td>
</tr>
<tr>
<td>TKGI Controller</td>
<td>NSX-T API VIP</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>TKGI Controller</td>
<td>vCenter Server</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Harbor Private Image Registry</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>4222</td>
<td>bosh nats server</td>
</tr>
<tr>
<td>Harbor Private Image Registry</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25250</td>
<td>bosh blobstore</td>
</tr>
<tr>
<td>Harbor Private Image Registry</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25923</td>
<td>health monitor daemon</td>
</tr>
<tr>
<td>Harbor Private Image Registry</td>
<td>IP NAS Storage Array</td>
<td>TCP</td>
<td>111</td>
<td>nfs rpc portmapper</td>
</tr>
<tr>
<td>Harbor Private Image Registry</td>
<td>IP NAS Storage Array</td>
<td>TCP</td>
<td>2049</td>
<td>nfs</td>
</tr>
<tr>
<td>Harbor Private Image Registry</td>
<td>Public CVE Source Database</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>kubectl-system pod/telemetry-agent</td>
<td>TKGI Controller</td>
<td>TCP</td>
<td>24224</td>
<td>fluentd out_forward</td>
</tr>
<tr>
<td>Kubernetes Cluster Ingress Controller</td>
<td>NSX-T API VIP</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>4222</td>
<td>bosh nats server</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25250</td>
<td>bosh blobstore</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25923</td>
<td>health monitor daemon</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>TCP</td>
<td>2379</td>
<td>etcd client</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>TCP</td>
<td>2380</td>
<td>etcd server</td>
</tr>
<tr>
<td>Source Component</td>
<td>Destination Component</td>
<td>Destination Protocol</td>
<td>Destination Port</td>
<td>Service</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>------------------------------------------------</td>
<td>----------------------</td>
<td>------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>TCP</td>
<td>8443</td>
<td>httpsca</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>TCP</td>
<td>8853</td>
<td>bosh dns health</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>Kubernetes Cluster Worker Node</td>
<td>TCP</td>
<td>4194</td>
<td>cadvisor</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>Kubernetes Cluster Worker Node</td>
<td>TCP</td>
<td>10250</td>
<td>kubelet api</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>Kubernetes Cluster Worker Node</td>
<td>TCP</td>
<td>31194</td>
<td>cadvisor</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>NSX-T API VIP</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>TKGI Controller</td>
<td>TCP</td>
<td>8443</td>
<td>httpsca</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>TKGI Controller</td>
<td>TCP</td>
<td>8853</td>
<td>bosh dns health</td>
</tr>
<tr>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>vCenter Server</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Kubernetes Cluster Worker Node</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>4222</td>
<td>bosh nats server</td>
</tr>
<tr>
<td>Kubernetes Cluster Worker Node</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25250</td>
<td>bosh blobstore</td>
</tr>
<tr>
<td>Kubernetes Cluster Worker Node</td>
<td>BOSH Director</td>
<td>TCP</td>
<td>25923</td>
<td>health monitor daemon</td>
</tr>
<tr>
<td>Kubernetes Cluster Worker Node</td>
<td>Harbor Private Image Registry</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>Kubernetes Cluster Worker Node</td>
<td>Harbor Private Image Registry</td>
<td>TCP</td>
<td>8853</td>
<td>bosh dns health</td>
</tr>
<tr>
<td>Kubernetes Cluster Worker Node</td>
<td>IP NAS Storage Array</td>
<td>TCP</td>
<td>111</td>
<td>nfs rpc portmapper</td>
</tr>
<tr>
<td>Kubernetes Cluster Worker Node</td>
<td>IP NAS Storage Array</td>
<td>TCP</td>
<td>2049</td>
<td>nfs</td>
</tr>
<tr>
<td>Kubernetes Cluster Worker Node</td>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>TCP</td>
<td>8443</td>
<td>httpsca</td>
</tr>
<tr>
<td>Kubernetes Cluster Worker Node</td>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>TCP</td>
<td>8853</td>
<td>bosh dns health</td>
</tr>
<tr>
<td>Kubernetes Cluster Worker Node</td>
<td>Kubernetes Cluster Master/Etcd Node</td>
<td>TCP</td>
<td>10250</td>
<td>kubelet api</td>
</tr>
<tr>
<td>pks-system pod/cert-generator</td>
<td>TKGI Controller</td>
<td>TCP</td>
<td>24224</td>
<td>fluentd out_forward</td>
</tr>
<tr>
<td>pks-system pod/fluent-bit</td>
<td>TKGI Controller</td>
<td>TCP</td>
<td>24224</td>
<td>fluentd out_forward</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Source Component</th>
<th>Destination Component</th>
<th>Destination Protocol</th>
<th>Destination Port</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>vCenter Server</td>
<td>NSX-T Manager/Controller Node</td>
<td>TCP</td>
<td>8080</td>
<td>http alt</td>
</tr>
<tr>
<td>vCenter Server</td>
<td>vSphere ESXi Hosts Mgmt. vmknic</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>vCenter Server</td>
<td>vSphere ESXi Hosts Mgmt. vmknic</td>
<td>TCP</td>
<td>8080</td>
<td>http alt</td>
</tr>
<tr>
<td>vCenter Server</td>
<td>vSphere ESXi Hosts Mgmt. vmknic</td>
<td>TCP</td>
<td>9080</td>
<td>io filter storage</td>
</tr>
<tr>
<td>vSphere ESXi Hosts Mgmt. vmknic</td>
<td>NSX-T Manager/Controller Node</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>vSphere ESXi Hosts Mgmt. vmknic</td>
<td>NSX-T Manager/Controller Node</td>
<td>TCP</td>
<td>1235</td>
<td>netcpa</td>
</tr>
<tr>
<td>vSphere ESXi Hosts Mgmt. vmknic</td>
<td>NSX-T Manager/Controller Node</td>
<td>TCP</td>
<td>5671</td>
<td>amqp traffic</td>
</tr>
<tr>
<td>vSphere ESXi Hosts Mgmt. vmknic</td>
<td>NSX-T Manager/Controller Node</td>
<td>TCP</td>
<td>8080</td>
<td>http alt</td>
</tr>
<tr>
<td>vSphere ESXi Hosts Mgmt. vmknic</td>
<td>vCenter Server</td>
<td>UDP</td>
<td>902</td>
<td>ideafarm-door</td>
</tr>
<tr>
<td>vSphere ESXi Hosts Mgmt. vmknic</td>
<td>vCenter Server</td>
<td>TCP</td>
<td>9084</td>
<td>update manager</td>
</tr>
<tr>
<td>vSphere ESXi Hosts Mgmt. vmknic</td>
<td>vSphere ESXi Hosts Mgmt. vmknic</td>
<td>TCP</td>
<td>8182</td>
<td>vSphere HA</td>
</tr>
<tr>
<td>vSphere ESXi Hosts Mgmt. vmknic</td>
<td>vSphere ESXi Hosts Mgmt. vmknic</td>
<td>UDP</td>
<td>8182</td>
<td>vSphere HA</td>
</tr>
<tr>
<td>vSphere ESXi Hosts vMotion vmknic</td>
<td>vSphere ESXi Hosts vMotion vmknic</td>
<td>TCP</td>
<td>8000</td>
<td>vmotion</td>
</tr>
<tr>
<td>vSphere ESXi Hosts IP Storage vmknic</td>
<td>IP NAS Storage Array</td>
<td>TCP</td>
<td>111</td>
<td>nfs rpc portmapper</td>
</tr>
<tr>
<td>vSphere ESXi Hosts IP Storage vmknic</td>
<td>IP NAS Storage Array</td>
<td>TCP</td>
<td>2049</td>
<td>nfs</td>
</tr>
<tr>
<td>vSphere ESXi Hosts IP Storage vmknic</td>
<td>IP NAS Storage Array</td>
<td>TCP</td>
<td>3260</td>
<td>iscsi</td>
</tr>
<tr>
<td>vSphere ESXi Hosts vSAN vmknic</td>
<td>vSphere ESXi Hosts vSAN vmknic</td>
<td>TCP</td>
<td>2233</td>
<td>vsan transport</td>
</tr>
<tr>
<td>vSphere ESXi Hosts vSAN vmknic</td>
<td>vSphere ESXi Hosts vSAN vmknic</td>
<td>UDP</td>
<td>12321</td>
<td>unicast agent</td>
</tr>
<tr>
<td>vSphere ESXi Hosts vSAN vmknic</td>
<td>vSphere ESXi Hosts vSAN vmknic</td>
<td>UDP</td>
<td>12345</td>
<td>vsan cluster svc</td>
</tr>
<tr>
<td>Source Component</td>
<td>Destination Component</td>
<td>Protocol</td>
<td>Port</td>
<td>Service</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------</td>
<td>----------</td>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>vSphere ESXI Hosts vsAN vmknic</td>
<td>vSphere ESXI Hosts vsAN vmknic</td>
<td>UDP</td>
<td>23451</td>
<td>vsan cluster svc</td>
</tr>
<tr>
<td>vSphere ESXI Hosts TEP vmknic</td>
<td>vSphere ESXI Hosts TEP vmknic</td>
<td>UDP</td>
<td>3784</td>
<td>bfd</td>
</tr>
<tr>
<td>vSphere ESXI Hosts TEP vmknic</td>
<td>vSphere ESXI Hosts TEP vmknic</td>
<td>UDP</td>
<td>3785</td>
<td>bfd</td>
</tr>
<tr>
<td>vSphere ESXI Hosts TEP vmknic</td>
<td>vSphere ESXI Hosts TEP vmknic</td>
<td>UDP</td>
<td>6081</td>
<td>geneve</td>
</tr>
<tr>
<td>vSphere ESXI Hosts TEP vmknic</td>
<td>NSX-T Edge TEP vNIC</td>
<td>UDP</td>
<td>3784</td>
<td>bfd</td>
</tr>
<tr>
<td>vSphere ESXI Hosts TEP vmknic</td>
<td>NSX-T Edge TEP vNIC</td>
<td>UDP</td>
<td>3785</td>
<td>bfd</td>
</tr>
<tr>
<td>vSphere ESXI Hosts TEP vmknic</td>
<td>NSX-T Edge TEP vNIC</td>
<td>UDP</td>
<td>6081</td>
<td>geneve</td>
</tr>
<tr>
<td>NSX-T Manager/Controller Node</td>
<td>NSX-T API VIP</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>NSX-T Manager/Controller Node</td>
<td>NSX-T Manager/Controller Node</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>NSX-T Manager/Controller Node</td>
<td>NSX-T Manager/Controller Node</td>
<td>TCP</td>
<td>5671</td>
<td>amqp traffic</td>
</tr>
<tr>
<td>NSX-T Manager/Controller Node</td>
<td>NSX-T Manager/Controller Node</td>
<td>TCP</td>
<td>8080</td>
<td>http alt</td>
</tr>
<tr>
<td>NSX-T Manager/Controller Node</td>
<td>NSX-T Manager/Controller Node</td>
<td>TCP</td>
<td>9000</td>
<td>loginsight ingestion api</td>
</tr>
<tr>
<td>NSX-T Manager/Controller Node</td>
<td>Traceroute Destination</td>
<td>UDP</td>
<td>33434-33523</td>
<td>traceroute</td>
</tr>
<tr>
<td>NSX-T Manager/Controller Node</td>
<td>vCenter Server</td>
<td>TCP</td>
<td>80</td>
<td>http</td>
</tr>
<tr>
<td>NSX-T Manager/Controller Node</td>
<td>vCenter Server</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>NSX-T Manager/Controller Node</td>
<td>vSphere ESXI Hosts Mgmt. vmknic</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>NSX-T Edge Management</td>
<td>NSX-T Edge Management</td>
<td>TCP</td>
<td>1167</td>
<td>DHCP backend</td>
</tr>
<tr>
<td>NSX-T Edge Management</td>
<td>NSX-T Edge Management</td>
<td>TCP</td>
<td>2480</td>
<td>Nestdb</td>
</tr>
<tr>
<td>NSX-T Edge Management</td>
<td>NSX-T Edge Management</td>
<td>UDP</td>
<td>3784</td>
<td>bfd</td>
</tr>
<tr>
<td>NSX-T Edge Management</td>
<td>NSX-T Edge Management</td>
<td>UDP</td>
<td>50263</td>
<td>high-availability</td>
</tr>
<tr>
<td>NSX-T Edge Management</td>
<td>NSX-T Manager/Controller Node</td>
<td>TCP</td>
<td>443</td>
<td>https</td>
</tr>
<tr>
<td>NSX-T Edge Management</td>
<td>NSX-T Manager/Controller Node</td>
<td>TCP</td>
<td>1235</td>
<td>netcpa</td>
</tr>
<tr>
<td>NSX-T Edge Management</td>
<td>NSX-T Manager/Controller Node</td>
<td>TCP</td>
<td>5671</td>
<td>amqp traffic</td>
</tr>
<tr>
<td>NSX-T Edge Management</td>
<td>NSX-T Manager/Controller Node</td>
<td>TCP</td>
<td>8080</td>
<td>http alt</td>
</tr>
<tr>
<td>NSX-T Edge Management</td>
<td>Traceroute Destination</td>
<td>UDP</td>
<td>33434-33523</td>
<td>traceroute</td>
</tr>
<tr>
<td>NSX-T Edge TEP vNIC</td>
<td>NSX-T Edge TEP vNIC</td>
<td>UDP</td>
<td>3784</td>
<td>bfd</td>
</tr>
<tr>
<td>NSX-T Edge TEP vNIC</td>
<td>NSX-T Edge TEP vNIC</td>
<td>UDP</td>
<td>3785</td>
<td>bfd</td>
</tr>
<tr>
<td>NSX-T Edge TEP vNIC</td>
<td>NSX-T Edge TEP vNIC</td>
<td>UDP</td>
<td>6081</td>
<td>geneve</td>
</tr>
<tr>
<td>NSX-T Edge TEP vNIC</td>
<td>NSX-T Edge TEP vNIC</td>
<td>UDP</td>
<td>50263</td>
<td>high-availability</td>
</tr>
<tr>
<td>NSX-T Edge TEP vNIC</td>
<td>vSphere ESXI Hosts TEP vmknic</td>
<td>UDP</td>
<td>3784</td>
<td>bfd</td>
</tr>
<tr>
<td>NSX-T Edge TEP vNIC</td>
<td>vSphere ESXI Hosts TEP vmknic</td>
<td>UDP</td>
<td>3785</td>
<td>bfd</td>
</tr>
<tr>
<td>NSX-T Edge TEP vNIC</td>
<td>NSX-T Edge TEP vNIC</td>
<td>UDP</td>
<td>6081</td>
<td>geneve</td>
</tr>
<tr>
<td>NSX-T Edge Tier-0 Uplink IP(s) / HA VIP</td>
<td>Physical Network Router</td>
<td>TCP</td>
<td>179</td>
<td>bgp</td>
</tr>
</tbody>
</table>
The following table lists ports and protocols used for network communication between optional VMware integrations.

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

Please send any feedback you have to pks-feedback@pivotal.io.
Creating NSX-T Objects for Tanzu Kubernetes Grid Integrated Edition

In this topic
Create the Nodes IP Block
Create the Pods IP Block
Create Floating IP Pool
Next Step

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

Installing VMware Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T requires the creation of NSX IP blocks for Kubernetes node and pod networks, as well as a Floating IP Pool from which you can assign routable IP addresses to cluster resources.

Create separate NSX-T IP Blocks for the node networks and the pod networks. The subnets for both nodes and pods should have a size of 256 (/16). For more information, see Plan IP Blocks and Reserved IP Blocks. For more information about NSX-T IP Blocks, see Advanced IP Address Management in the VMware NSX-T Data Center documentation.

- **NODE-IP-BLOCK** is used by Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition. 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 Tanzu Kubernetes Grid Integrated Edition tile.

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. For example:
   - **Name**: NODES-IP-BLOCK
3. Verify creation of the Nodes IP Block.

4. Record the UUID of the Nodes IP Block object. You use this UUID when you install Tanzu Kubernetes Grid Integrated Edition with NSX-T.
Create the Pods IP Block

1. In NSX Manager, go to Advanced Networking & Security > Networking > IPAM.

2. Add a new IP Block for Pods. For example:
   - **Name:** TKGI-PODS-IP-BLOCK
   - **CIDR:** 172.16.0.0/16

3. Verify creation of the Pods IP Block.
4. Record the UUID of the Pods IP Block object. You use this UUID when you install Tanzu Kubernetes Grid Integrated Edition with NSX-T.

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**: TKGI-FLOATING-IP-POOL
- **Gateway**: 10.40.14.254
- **CIDR**: 10.40.14.0/24

3. Verify creation of the Nodes IP Block.

Next Step

After you complete this procedure, follow the instructions in Installing Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T.

Please send any feedback you have to pks-feedback@pivotal.io.
NSX-T Deployment Topologies for Tanzu Kubernetes Grid Integrated Edition

In this topic

- NAT Topology
- No-NAT Topology
  - No-NAT with Virtual Switch (VSS/VDS) Topology
  - No-NAT with Logical Switch (NSX-T) Topology
- Hybrid Topology
- vSAN Stretched Cluster Topologies

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes supported topologies for deploying VMware Tanzu Kubernetes Grid Integrated Edition with NSX-T.

NAT Topology

The following figure shows a Network Address Translation (NAT) deployment:
This topology has the following characteristics:

- TKGI Management Plane (Ops Manager, BOSH Director, and Tanzu Kubernetes Grid Integrated Edition VMs such as the TKGI API and TKGI Database VMs) 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 TKGI 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:
TKGI Database VMs) components are using corporate routable IP addresses.

- Kubernetes cluster master and worker nodes are using corporate routable IP addresses.
- The TKGI Management 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:

- TKGI Management Plane (Ops Manager, BOSH Director, and Tanzu Kubernetes Grid Integrated Edition VMs such as the TKGI API and TKGI Database VMs) components are using corporate routable IP addresses.
- Kubernetes cluster master and worker nodes are using corporate routable IP addresses.
- The TKGI Management Plane is deployed inside of the NSX-T network. Both the TKGI Management Plane components (VMs) and the Kubernetes Nodes use corporate routable IP addresses.

Hybrid Topology

With a hybrid topology, the TKGI Management Network is on a routable subnet, while the Kubernetes Nodes Network uses a non-routable subnet (NAT mode is checked in the TKGI tile).
The following figure shows a hybrid topology deployment:

![Hybrid Topology Diagram](image)

**NSX-T Virtual Network**

**TKGI Management Plane**: Components using corporate routable IP addresses. Includes:
- **Ops Manager**
- **Bosh Director**
- **TKGI API**
- **TKGI DB**

**T0** provides routing for NAT.

**T1** MGMT K8s-2

**Cluster Network**: NAT

**K8s Master**

**K8s Worker**

**Non-Routable IP**

**vSAN Stretched Cluster Topologies**

A vSAN stretched cluster topology runs across two sites to support highly resilient workloads. vSAN stretched cluster topologies include:

- **Topology 1**: Dedicated vSphere clusters
- **Topology 2**: Fully collapsed vSphere clusters

For more information about vSAN stretched cluster topologies for Tanzu Kubernetes Grid Integrated Edition, see [Solution Guide for Enabling Highly Resilient Kubernetes Workloads Using vSAN Stretched Clusters](#).

Please send any feedback you have to pks-feedback@pivotal.io.
vSphere with NSX-T Cluster Objects

In this topic
- vSphere Virtual Machines
- NSX-T Logical Switches
- NSX-T Tier-1 Logical Routers
- NSX-T Load Balancers
- NSX-T DDI/IPAM
- NSX-T Tier-0 Logical Routers
- NSX-T Distributed Firewall (DFW) Rules

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

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


vSphere Virtual Machines

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

<table>
<thead>
<tr>
<th>Object Number</th>
<th>Object Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 or 3</td>
<td>Kubernetes master nodes. The number depends on the plan used to create the cluster.</td>
</tr>
<tr>
<td>1 or more</td>
<td>Kubernetes worker nodes. The number depends on the plan used to create the cluster, or the number specified during cluster creation.</td>
</tr>
</tbody>
</table>

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

NSX-T Logical Switches

When a new Kubernetes cluster is created, Tanzu Kubernetes Grid Integrated Edition creates the following NSX-T logical switches:

<table>
<thead>
<tr>
<th>Object Number</th>
<th>Object Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Logical switch for Kubernetes master and worker nodes.</td>
</tr>
</tbody>
</table>
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, Tanzu Kubernetes Grid Integrated Edition creates the following **NSX-T Tier-1 logical routers**:

<table>
<thead>
<tr>
<th>Object Number</th>
<th>Object Description</th>
</tr>
</thead>
</table>
| 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, Tanzu Kubernetes Grid Integrated Edition creates a single instance of a small **NSX-T load balancer**. This load balancer contains the objects listed in the following table:

<table>
<thead>
<tr>
<th>Object Number</th>
<th>Object Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Virtual Server (VS) to access Kubernetes control plane API on port 8443.</td>
</tr>
<tr>
<td>1</td>
<td>Server Pool containing the 3 Kubernetes master nodes.</td>
</tr>
<tr>
<td>1</td>
<td>VS for HTTP Ingress Controller.</td>
</tr>
<tr>
<td>1</td>
<td>VS for HTTPS Ingress Controller.</td>
</tr>
</tbody>
</table>

The IP address allocated to each VS is derived from the **Floating IP Pool** that was created for use with Tanzu Kubernetes Grid Integrated Edition. 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, Tanzu Kubernetes Grid Integrated Edition extracts and allocates the following NSX-T subnets from the **IP blocks** created in preparation for installing Tanzu Kubernetes Grid Integrated Edition with NSX-T:

<table>
<thead>
<tr>
<th>Object Number</th>
<th>Object Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A /24 subnet from the <strong>Nodes IP Block</strong> will be extracted and allocated for the Kubernetes master and worker nodes.</td>
</tr>
</tbody>
</table>
| 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, Tanzu Kubernetes Grid Integrated Edition defines the following **NSX-T NAT rules** on the Tier-0 logical router:
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.

(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, Tanzu Kubernetes Grid Integrated Edition defines the following NSX-T distributed firewall rules:

<table>
<thead>
<tr>
<th>Object Amount</th>
<th>Object Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DFW rule for <code>kube-dns</code>, applied to CoreDNS pod logical port: Source=Kubernetes worker node (hosting the DNS Pod); Destination=Any; Port: TCP/8080; Action: allow</td>
</tr>
<tr>
<td>1</td>
<td>DFW rule for Validator in <code>pks-system</code> namespace, applied to Validator pod logical port: Source=Kubernetes worker node (hosting the DNS Pod) IP Address; Destination=Any; Port: TCP/9000; Action: allow</td>
</tr>
<tr>
<td>1</td>
<td>For clusters with Kubernetes Dashboard installed, DFW rule for <code>kubernetes-dashboard</code>: Source=Kubernetes worker node (hosting the Dashboard Pod); Destination=Dashboard Pod IP; Port: TCP/8443; Action: allow</td>
</tr>
</tbody>
</table>

Please send any feedback you have to pks-feedback@pivotal.io.
Network Planning for Installing Tanzu Kubernetes Grid Integrated Edition with NSX-T

In this topic
Prerequisites
Understand Component Interactions
Plan Deployment Topology
Plan Network CIDRs
Plan IP Blocks
  Pods IP Block
  Nodes IP Block
  Reserved IP Blocks
Gather Other Required IP Addresses

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:
Before you install VMware Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition documentation:
- vSphere with NSX-T Version Requirements
- Hardware Requirements for Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T
- Firewall Ports and Protocols Requirements for vSphere with NSX-T
- Network Objects Created by NSX-T for Tanzu Kubernetes Grid Integrated Edition
Understand Component Interactions

Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition-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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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.
- **TKGI MANAGEMENT CIDR**: This small network is used to access Tanzu Kubernetes Grid Integrated Edition management components such as Ops Manager, BOSH Director, and Tanzu Kubernetes Grid Integrated Edition VMs as well as the Harbor Registry VM if deployed. For example, a CIDR of \(10.172.1.0/24\) 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 Tanzu Kubernetes Grid Integrated Edition management components.
- **TKGI LB CIDR**: This network provides your load balancing address space for each Kubernetes cluster created by Tanzu Kubernetes Grid Integrated Edition. 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 Tanzu Kubernetes Grid Integrated Edition or when the services are deployed. You enter this network in the Floating IP Pool ID field in the Networking pane of the Tanzu Kubernetes Grid Integrated Edition tile.

Plan IP Blocks

When you install Tanzu Kubernetes Grid Integrated Edition on NSX-T, you are required to specify the **Pods IP Block ID** and **Nodes IP Block ID** in the Networking pane of the Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition, 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 Tanzu Kubernetes Grid Integrated Edition.

**Note:** By default, **Pods IP Block** is a block of non-routable, private IP addresses. After you deploy Tanzu Kubernetes Grid Integrated Edition, 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
Nodes IP Block

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

---

### ip-block-pks-nodes-snat

<table>
<thead>
<tr>
<th>Overview</th>
<th>Subnets</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Summary</th>
<th>EDIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>ip-block-pks-nodes-snat</td>
</tr>
<tr>
<td>ID</td>
<td>ad51f33b-e7ae-45f5-81dd-fd481177f1dc</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>CIDR</td>
<td>172.15.0.0/16</td>
</tr>
<tr>
<td>Created</td>
<td>5/21/2018, 11:53:50 AM by admin</td>
</tr>
<tr>
<td>Last Updated</td>
<td>7/16/2018, 8:43:32 AM by pks-nsx-t-superuser</td>
</tr>
</tbody>
</table>

### Tags | MANAGE

#### 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 Tanzu Kubernetes Grid Integrated Edition Management Plane must not use the use 172.17.0.0/16 subnet. This restriction applies to all virtual machines (VMs) deployed during the Tanzu Kubernetes Grid Integrated Edition installation process, including Tanzu Kubernetes Grid Integrated Edition components, 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
If Tanzu Kubernetes Grid Integrated Edition 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

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 Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T, you will need to know the following:

- Subnet name where you will install Tanzu Kubernetes Grid Integrated Edition
- 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 v3.0 for Tanzu Kubernetes Grid Integrated Edition

In this topic

Prerequisites for Installing NSX-T Data Center v3.0 for Tanzu Kubernetes Grid Integrated Edition

Install the NSX-T Management Hosts
- Deploy NSX-T Manager 1
- Create a Compute Manager
- Deploy NSX-T Managers 2 and 3
- Configure the NSX-T Management VIP

Generate and Register the NSX-T Management SSL Certificate and Private Key
- Generate the SSL Certificate and Private Key
- Import the SSL Certificate and Private Key to the NSX-T Management Console
- Register the SSL Certificate and Private Key with the NSX API Server

Complete the NSX-T Management Plane Configuration
- Enable the NSX-T Manager Interface
- Add the NSX-T Manager License

Create IP Pools

Create Transport Zones
- Configure vSphere Networking for ESXi Hosts
  - Create vSwitch Port-Groups for Edge Nodes
  - Set vSwitch0 with MTU at 9000

Deploy NSX-T Edge Nodes
- Install Edge Node 1
- Install Edge Node 2

Create Uplink Profile for ESXi Transport Node

Deploy ESXi Host Transport Nodes Using N-VDS
- Verify TEP to TEP Connectivity

Create NSX-T Edge Cluster

Create Uplink Logical Switch

Create Tier-0 Router
- Configure and Test the Tier-0 Router
- Create IP Blocks and Pool for Compute Plane

Create Management Plane
- Create Tier-1 Router and Switch
- Create NAT Rules
- Configure the NSX-T Password Interval (Optional)
  - Update the NSX-T Manager Password and Password Interval
  - Update the Password for NSX Edge Nodes

Next Steps

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

This topic provides instructions for installing and configuring NSX-T Data Center v3.0 for use with VMware Tanzu Kubernetes Grid Integrated Edition on vSphere.

Prerequisites for Installing NSX-T Data Center v3.0 for Tanzu Kubernetes Grid Integrated Edition

To perform a new installation of NSX-T Data Center for Tanzu Kubernetes Grid Integrated Edition, complete the following steps in the order presented.

1. Read the Release Notes for the target TKGI version you are installing and verify NSX-T 3.0 support.

2. Read the topics in the Preparing to Install Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T Data Center section of the documentation.
Install the NSX-T Management Hosts

Create the NSX-T Management Plane by installing 3 NSX Managers and configuring a VIP address.

Deploy NSX-T Manager 1

Deploy the NSX-T Manager OVA in vSphere.

1. Using the vSphere Client, right-click the vCenter cluster and select Deploy OVF Template.

2. At the Select an OVF Template screen, browse to and select the NSX Unified Appliance OVA file.
3. At the Select a name and folder screen, select the target Datacenter object.
4. At the **Select a compute resource** screen, select the target vCenter cluster.

5. Review the details.
6. At the Configuration screen, select at least Medium for the configuration size.
7. At the Select storage screen, choose Thin Provision and the desired datastore.

8. For Network1, enter the VLAN management network, such as PG-MGMT-VLAN-1548.
9. Enter strong passwords for all user types.
10. Enter the hostname, such as `nsx-manager-1`.

11. Enter the rolename, such as `NSX Manager`.

12. Enter the Gateway IP address, such as `10.173.62.253`.

13. Enter a public IP address for the VM, such as `10.173.62.44`.

14. Enter the Netmask, such as `255.255.255.0`.

15. Enter the DNS server, such as `10.172.40.1`.

16. Enter the NTP server, such as `10.113.60.176`.

17. Enable the **Enable SSH** checkbox.

18. Enable the **Allow SSH root logins** checkbox.
19. Click **Finish**, and NSX-T Manager 1 starts deploying.
20. Monitor the deployment using the **Recent Tasks** pane.

![Recent Tasks screenshot]

21. When the deployment completes, select the VM and power it on.

22. Access the NSX-T Manager web console by navigating to the URL, such as: [https://10.173.62.44/](https://10.173.62.44/).
23. Log in and verify the installation. Note the system message that a “3 node cluster” is recommended.
A compute manager is required for NSX-T environments with multiple NSX-T Managers. We configure vCenter Server as the Compute Manager.

1. In the NSX Manager, navigate to **System > Appliances**.

2. Before you can add a second NSX-T manager, you first need to create a Compute Manager.

3. Select **Compute Managers**.

4. Click **Add**.

5. Enter a Name, such as **vCenter**.

6. Enter an IP address, such as **10.173.62.43**.

7. Enter the vCenter username, such as **administrator@vsphere.local**.

8. Set the **Enable Trust** toggle to **Yes**.
9. Click **Add**.

10. Click **Add** again at the thumbprint warning.
11. Verify that the Compute Manager is added and registered.

```
<table>
<thead>
<tr>
<th>Compute Manager</th>
<th>ID</th>
<th>FQDN or IP Address</th>
<th>Type</th>
<th>Registration Status</th>
<th>Version</th>
<th>Connection Status</th>
<th>Last Inventory Update</th>
<th>Alarms</th>
</tr>
</thead>
<tbody>
<tr>
<td>vCenter-demo</td>
<td>0x28, 241#</td>
<td>10.113.02.43</td>
<td>vCenter</td>
<td>Registered</td>
<td>7.0.0</td>
<td>Up</td>
<td>May 26, 2020 3:22:21</td>
<td></td>
</tr>
</tbody>
</table>
```

Deploy NSX-T Managers 2 and 3

1. Select Add NSX Appliance.
2. Enter a hostname, such as `nsx-manager-2`.

3. Enter the Management IP/netmask, such as `10.173.62.45/24`.

4. Enter the Gateway, such as `10.173.62.253`.

5. For the Node size, choose `medium`.
6. For the Compute Manager, select `vCenter`.

7. For the Compute Cluster, enter `MANAGEMENT-cluster`.

8. For the Datastore, select the datastore, such as `datastore2`.

9. For the Virtual Disk Format, select `thin provision`.

10. For the Network, select the VLAN management network, such as `PG-MGMT-VLAN-1548`. 
11. Select Enable SSH.

12. Select Enable root access.

13. Enter a strong password.
14. Click Install Appliance.

15. Verify that the NSX-T Manager 2 appliance is added.
16. Repeat the same operation for NSX-T Manager 3 that you performed for NSX-T Manager 2.

17. Use a different public IP address.

18. Verify that the NSX-T Manager 3 appliance is added.
Configure the NSX-T Management VIP

1. Click the Set Virtual IP button.

2. Enter a Virtual IP address, such as `10.173.62.47`.

Set Virtual IP

NSX-T Managers Cluster offer a built-in VIP for high-availability, but the usage of an external load balancer offers the following benefits: 1) Load spread across all NSX-T Managers; 2) NSX-T Managers can be in different subnets and 3) Faster failover.

Virtual IP Address* 10.173.62.47

3. Verify that the VIP is added.
Set Virtual IP

NSX-T Managers Cluster offer a built-in VIP for high-availability, but the usage of an external load balancer offers the following benefits: 1) Load spread across all NSX-T Managers; 2) NSX-T Managers can be in different subnets and 3) Faster failover.

Setting up the Virtual IP Address for the Management Cluster.

It might take few minutes to set a new IP address.

4. Access the NSX-T Management console using the VIP, such as: https://10.173.62.47/login.jsp.
Generate and Register the NSX-T Management SSL Certificate and Private Key

A SSL certificate is automatically created for each NSX-T Manager. You can verify this by SSHing to one of the NSX Manager nodes using the command `get certificate cluster`. For example:

```
nsx-manager-1 > get certificate cluster
```

The issuer, however, is `CN=nsx-manager-1`, which means the cluster certificate is linked to a particular NSX Manager, in this case NSX-T Manager 1. If you go to System > Certificates, you will see there is no certificate automatically created for NSX-T manager VIP.

You need to generate a new SSL certificate that uses the NSX-T Management VIP address so that the the cluster certificate contains `CN=VIP`. If you go to System > Certificates, you will see there is no certificate automatically created for NSX-T Manager VIP.

The following steps assume that you are working on a Linux host. If you are using a different OS, adjust the steps accordingly.

**Generate the SSL Certificate and Private Key**

1. Create a certificate signing request file named `nsx-cert.cnf` and populate it with the contents below. Replace the IP addresses for `commonName` and `DNS.1` with the VIP address for your NSX-T Management plane.

```
nsx-cert.cnf
[ 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 = 10.173.62.47   # REPLACE
[ req_ext ]
subjectAltName = @alt_names
[alt_names]
DNS.1 = 10.173.62.47   # REPLACE
```

2. Create the following environment variables. Replace the IP address with your VIP.

```
export NSX_MANAGER_IP_ADDRESS=10.173.62.47
export NSX_MANAGER_COMMONNAME=10.173.62.47
```

3. Use OpenSSL to generate the SSL certificate and private key.
openssl req -newkey rsa:2048 -nodes -out nsx.key -new -out nsx.crt -subj /CN=$NSX_MANAGER_COMMONNAME \
  -config<(cat ./nsx-cert.cnf \n  <(printf "[SAN]\nsubjectAltName=DNS:$NSX_MANAGER_COMMONNAME,IP:$NSX_MANAGER_IP_ADDRESS")) >sha256 -days 1000

4. Verify that you see the following:

   Generating a 2048 bit RSA private key
   ............+++
   ............+++
   writing new private key to 'nsx.key'

5. Verify certificate and key generation by running the `ls` command. You should see the following 3 files.

   nsx-cert.cnf  nsx.crt  nsx.key

6. Run the following command to verify the certificate and private key.

   openssl x509 -in nsx.crt -text -noout

   You should see that the common name (CN) and Subject Alternative Name are both the VIP address. For example:

   Subject: CN = 10.173.62.47
   Subject Alternative Name:
   DNS:10.173.62.47, IP Address:10.173.62.47

Import the SSL Certificate and Private Key to the NSX-T Management Console

1. Import certificate and private key to NSX-T by navigating to the System > Certificates page.

   2. Select import certificate.

   3. Enter a Name, such as `CERT-NSX-T-VIP`.

   4. Copy and paste the Certificate Contents from the `nsx.crt` file. If you copy the `nsx.crt` file to your local machine, you can import the certificate by clicking Browse and selecting it.

   5. Copy and paste the Private Key from the `nsx.key` file. If you copy the `nsx.key` file to your local machine, you can import the certificate by clicking Browse and selecting it.

   6. For the Service Certificate option, select No.
7. Click **Import**.

![Import Certificate](image)

- **Name**: CERT-NSX-T-VIP
- **Certificate Contents**: R4IA880Y
  ----END CERTIFICATE-----
- **Private Key**: /L3pZqtKG4EMWlyCZhsfkl=
  ----END PRIVATE KEY-----
- **Passphrase**: 
- **Description**: 
- **Service Certificate**: No

Turn Service Certificate on to use the certificate with services such as Load Balancer and VPN.

Turn Service Certificate off to use the certificate with NSX Manager appliance nodes.

8. Verify that you see the certificate in the list of Certificates.

![Certificates Table](image)
Register the SSL Certificate and Private Key with the NSX API Server

1. Retrieve the UDID of the certificate from the NSX-T Management Console > Certificates screen.

2. Copy this UUID to the clipboard, such as 170a6d52-5c61-4fef-a9e0-09c6229fe833.

3. Create the following environment variables. Replace the IP address with your VIP address.

   ```
   export NSX_MANAGER_IP_ADDRESS=10.173.62.47
   export CERTIFICATE_ID=170a6d52-5c61-4fef-a9e0-09c6229fe833
   ```

4. Post the certificate to the NSX-T Manager API.

   ```
   curl --insecure -u admin:'VMware1!VMware1!' -X POST 'https://$NSX_MANAGER_IP_ADDRESS/api/v1/cluster/api-certificate?action=set_cluster_certificate&certificate_id=$CERTIFICATE_ID' 
   {
     "certificate_id": "170a6d52-5c61-4fef-a9e0-09c6229fe833"
   }
   ```

5. Verify by SSHing to one of the NSX-T Management nodes and running the `get certificate cluster` command. The certificate that is returned should match the generated one.

   nsx-manager-1> get certificate cluster

Complete the NSX-T Management Plane Configuration

Enable the NSX-T Manager Interface

The Manager interface provides the advanced UI for networking and security that is used for configuring NSX-T object for TKGI. Do NOT use the Policy interface for TKGI objects.

1. In the NSX-T Manager console, navigate to System > User Interface Settings.
2. Select **Toggle Visibility**: Visible to all Users.
3. Select **Default Mode**: Manager.
4. Click Save.

5. Verify that you see the Manager option in the upper right of the console next to Policy.

Add the NSX-T Manager License

If you do not add the proper NSX-T license, you will receive an error when you try to deploy a Edge Node VM.

1. In the NSX-T Manager console, navigate to System > Licenses.

2. Add the NSX Data Center Advanced (CPU) license.

3. Verify that the license is added.

Create IP Pools

Create IP Pool for VTEP.

1. Select the Manager interface (upper right).

2. Go to Networking > IP Address Pool.
3. Select Add.

4. Enter a Name, such as "TEP-IP-POOL".

5. Enter an IP range, such as "192.23.213.1 - 192.23.213.10".

6. Enter a CIDR address, such as "192.23.213.0/24".

7. Click Add.

8. Verify that the pool is added.
Create Transport Zones

You need two transport zones, an overlay and a VLAN. By default NSX-T creates two transport zones for you: `nsx-overlay-transportzone` and `nsx-vlan-transportzone`. However, the default TZs cannot be used by TKGI. If you do use them, you will receive an `pks-nsx-t-obs-proxy` BOSH error when you try to deploy TKGI.

Therefore, create two new transport zones for use with TKGI:

- `tz-overlay` (switch name: `switch-overlay`), and
- `tz-vlan` (switch name: `switch-vlan`)

1. Go to `System > Fabric > Transport Zone`.

2. Click `Add`.

3. Enter a Name, such as `tz-overlay`.

4. Enter a switch name, such as `switch-overlay`.

5. For the Traffic Type, select `overlay`.

6. Click `Add`.
7. Verify that you see the newly created TZ named `tz-overlay` in the list.

8. Click Add.

9. Enter a name, such as `tz-vlan`.

10. Enter a switch name, such as `switch-vlan`.

11. For the Traffic Type, select VLAN.
12. Click Add.

13. Verify that you see the newly created TZ named tz-vlan in the list.

Configure vSphere Networking for ESXi Hosts

In this section, you configure the vSphere networking and port groups for ESXi hosts (the vSwitch). If you have created separate vSphere clusters for Management and Compute, perform this operation on each ESXi host in the Management cluster. If you have not created separate vSphere clusters, perform this operation on each ESXi host in the cluster.

The following instructions describe how to configure a vSphere Virtual Standard vSwitch (VSS). For production environments, it is recommended that you configure a Virtual Distributed vSwitch (VDS). You configure the VDS from the vCenter Networking tab and then add the ESXi hosts to the VDS. The configuration settings for the VDS are similar to the VSS configuration described below. For instructions on configuring the VDS, see Create a vSphere Distributed Switch in the vSphere 7 documentation.

TKGI v1.8 supports 1 VDS (or VSS) for vSphere traffic and 1 N-VDS for NSX-T v3.0 traffic (NSX-T transport nodes). TKGI v1.8 does not support Converged VDS v7.
which runs vSphere v7 traffic and NSX-T v3.0 traffic on the same VDS.

⚠️ warning: TKGI v1.8 does not support VMware Cloud Foundation (VCF) v4, which requires a converged VDS.

Create vSwitch Port-Groups for Edge Nodes

Create vSwitch Port-Groups for the Edge Nodes on the ESXi hosts in the MANAGEMENT-cluster.

For each ESXi host in the MANAGEMENT-cluster, create the following vSwitch Port Groups:

- **EDGE-VTEP-PG**: VLAN 3127
- **EDGE-UPLINK-PG**: VLAN trunk (All (4095))

1. Log in to the vCenter Server.

2. Select the ESXi host in the MANAGEMENT-cluster.

3. Select **Configure > Virtual switches**.

4. Select **Add Networking** (upper right).

5. Select the option **Virtual Machine Port Group for a Standard Switch** and click **Next**.
6. Select the existing standard switch named `vSwitch0` and click **Next**.
7. Enter a **Network Label**, such as `EDGE-VTEP-PG`.

8. Enter a **VLAN ID**, such as `3127`.
9. Click Finish.
10. Verify that you see the newly created port group.

11. Select **Add Networking** (upper right).

12. Select the option **Virtual Machine Port Group for a Standard Switch** and click **Next**.

13. Select the existing standard switch named **vSwitch0** and click **Next**.

14. Enter a **Network Label**, such as **EDGE-UPLINK-PG**.
15. For the VLAN ID, select \( [4095] \) from the drop-down.

16. Click **Finish**.

17. Verify that you see the newly created port group.

Set vSwitch0 with MTU at 9000
For each ESXi host in the MANAGEMENT-cluster, or each ESXi host in the vCenter cluster if you have not created separate Management and Compute clusters, you must enable the virtual switch with jumbo MTU, that is, set vSwitch0 with MTU=9000. If you do not do this, network overlay traffic will jam. The TEP interface for the NSX-T Edge Nodes must be connected to a port group that supports > 1600 bytes. The default is 1500.

1. Select the Virtual Switch on each ESXi host in the MANAGEMENT-cluster, or each host in the vCenter cluster.

2. Click Edit.

3. For the MTU (bytes) setting, enter 9000.

```
<table>
<thead>
<tr>
<th>Properties</th>
<th>Number of ports</th>
<th>Elastic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Traffic shaping</td>
<td>MTU (Bytes)</td>
<td>9000</td>
</tr>
<tr>
<td>Teaming and failover</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

4. Click OK to complete the operation.

### Deploy NSX-T Edge Nodes

In this section you deploy two NSX-T Edge Nodes.

NSX Edge Nodes provide the bridge between the virtual network environment implemented using NSX-T and the physical network. Edge Nodes for Tanzu Kubernetes Grid Integrated Edition run load balancers for TKGI API traffic, Kubernetes load balancer services, and ingress controllers. See Load Balancers in Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition:

- 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 TKGI API

The number of virtual servers that can be run depends on the size of the load balancer which depends on the size of the Edge Node. Tanzu Kubernetes Grid Integrated Edition 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.

For this installation, we use the Large VM form factor for the Edge Node. See VMware Configuration Maximums for more information.
Install Edge Node 1

Deploy the 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
   - **Host name/FQDN**: edge-node-1.lab.com
   - **Form Factor**: Large

5. Configure Credentials as follows:
   - **CLI User Name**: admin
   - **CLI Password**: Enter a strong password for the admin user that complies with the NSX-T requirements.
   - **Enable SSH Login**: Yes
   - **System Root Password**: Enter a strong password for the root user that complies with the NSX-T requirements.
Enable Root SSH Login: Yes
Audit Credentials: Enter an audit user name and password.

### Add Edge VM

1. Name and Description
2. Credentials
3. Configure Deployment
4. Configure Node Settings
5. Configure NSX

### Credentials

CLI credentials will be set on the NSX Edge VM. These credentials can be used to login to the read only command line interface of the appliance.

- **CLI User Name**: admin
- **CLI Password**: **********
- **CLI Confirm Password**: **********
- **Allow SSH Login**: Yes

### Root Credentials

- **System Root Password**: **********
- **System Root Confirm Password**: **********
- **Allow Root SSH Login**: Yes

### Audit Credentials

6. Configure the deployment as follows:

- **Compute Manager**: vCenter
- **Cluster**: MANAGEMENT-Cluster
- **Datastore**: Select the datastore

### Configure Deployment

- **Compute Manager**: vCenter-demo
- **Cluster**: MANAGEMENT-cluster
- **Resource Pool**
- **Host**
- **Datastore**: datastore2

Did not find expected? Try refresh to fetch latest datastores from System.
7. Configure the node settings as follows:
   - **IP Assignment**: Static
   - **Management IP**: 10.173.62.49/24, for example
   - **Default Gateway**: 10.173.62.253, for example
   - **Management Interface**: PG-MGMT-VLAN-1548, for example

8. Configure the first NSX switch for the Edge Node as follows:
   - **Edge Switch Name**: switch-overlay (be sure to use the exact switch name that was configured for tz-overlay)
   - **Transport Zone**: tz-overlay
   - **Uplink Profile**: nsx-edge-single-nic-uplink-profile
   - **IP Assignment**: Use IP Pool
   - **IP Pool**: TEP-IP-POOL
   - **Uplinks**: uplink-1 / EDGE-VTEP-PG
9. Configure the second NSX switch for the Edge Node as follows:

- Click **Add Switch** (at the top of the dialog)
- **Edge Switch Name**: `switch-vlan` (be sure to use the same switch name that was configured for `tz-vlan`)
- **Uplink Profile**: `nsx-edge-single-nic-uplink-profile`
- **Uplinks**: `uplink-1 / EDGE-UPLINK-PG`
10. Click **Finish** to complete the configuration. The installation begins.

11. 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.

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

13. Click the N-VDS link and verify that you see both switches.
14. In vCenter verify that the Edge Node is created.

Install Edge Node 2

Repeat the same operation for Edge Node 2, and for each additional NSX Edge Node pair you intend to use for Tanzu Kubernetes Grid Integrated Edition.

1. Install `nsx-edge-2` following the same procedure as `nsx-edge-1`.
   - **name:** `edge-node-2`
   - **hostname/FQDN:** `edge-node-2.lab.com`, for example
   - **Form Factor:** Large
   - **IP Assignment:** Static
- IP: 10.173.62.58/24, for example
- GW: 10.173.62.253, for example
- Management Interface: PG-MGMT-VLAN-1548
- Edge Switch 1:
  - Name: switch-overlay (use the same switch name that was configured for tz-overlay)
  - Transport Zone: tz-overlay
  - Uplink Profile: nsx-edge-single-nic-uplink-profile
  - IP Assignment: Use IP Pool
  - IP Pool: TEP-IP-POOL
  - Uplinks: uplink-1 / EDGE-VTEP-PG
- Edge Switch 2:
  - Name: switch-vlan (use the same switch name that was configured for tz-vlan)
  - Transport Zone: tz-vlan
  - Uplink Profile: nsx-edge-single-nic-uplink-profile
  - Uplinks: uplink-1 / EDGE-UPLINK-PG

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

<table>
<thead>
<tr>
<th>Host Transport Nodes</th>
<th>Edge Transport Nodes</th>
<th>Edge Clusters</th>
<th>ESXi Bridge Clusters</th>
<th>NCP Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ADD EDGE VM</strong></td>
<td><strong>EDIT</strong></td>
<td><strong>DELETE</strong></td>
<td><strong>+ ACTIONS +</strong></td>
<td>View All</td>
</tr>
<tr>
<td>Edge</td>
<td>ID</td>
<td>Deployment Type</td>
<td>Management IP</td>
<td>Host</td>
</tr>
<tr>
<td>edge-node-1</td>
<td>9099, b272</td>
<td>Virtual Machine</td>
<td>10.173.62.49</td>
<td>SMO</td>
</tr>
<tr>
<td>edge-node-2</td>
<td>8466, a877</td>
<td>Virtual Machine</td>
<td>10.173.62.58</td>
<td>SMO</td>
</tr>
</tbody>
</table>

Create Uplink Profile for ESXi Transport Node

To configure the TEP, we used the default profile named nsx-default-uplink-hostswitch-profile. However, because the TEP is on VLAN 3127, you must modify the uplink profile for the ESXi Transport Node (TN). NSX-T does not allow you to edit settings the default uplink profile, so we create a new one.

1. Go to System > Fabric > Profiles > Uplink Profiles.

2. Click Add.

3. Configure the New Uplink Profile as follows:
   - **Name**: nsx-esxi-uplink-hostswitch-profile
   - **Teaming Policy**: Failover Order
   - **Active Uplinks**: uplink-1
   - **Transport vLAN**: 3127
4. Click **Add**.

5. Verify that the **Uplink Profile** is created.

---

**New Uplink Profile**

- **Name**: nsx-esxi-uplink-hostswitch-profile
- **Description**: 
- **LAGs**: No LAGs found
- **Teamings**: [Default Teaming] Failover Order uplink-1
- **Transport VLAN**: 3127

---

**Deploy ESXi Host Transport Nodes Using N-VDS**
Deploy each ESXi host in the COMPUTE-cluster as an ESXi host transport node (TN) in NSX-T. If you have not created a separate COMPUTE-cluster for ESXi hosts, deploy each ESXi host in the vSphere cluster as a host transport node in NSX-T.

1. Go to System > Fabric > Nodes > Host Transport Nodes.

2. Expand the Compute Manager and select the ESXi host in the COMPUTE-cluster, or each ESXi host in the vSphere cluster.

3. Click Configure NSX.

4. In the Host Details tab, enter a name, such as `10.172.210.57`.

5. In the Configure NSX tab, configure the transport node as follows:
   - **Type**: N-VDS (do not select the VDS option; Converged VDS is not supported by TKG)
   - **Name**: switch-overlay (you must use the same switch name that was configured for `tz-overlay` transport zone)
   - **Transport Zone**: `tz-overlay`
   - **NIOC Profile**: `nsx-default-nioc-hostswitch-profile`
   - **Uplink Profile**: `nsx-esxi-uplink-hostswitch-profile`
   - **LLDP Profile**: LLDP (Send Packet Disabled)
   - **IP Assignment**: Use IP Pool
   - **IP Pool**: TEP-IP-POOL
6. Click Finish.

7. Verify that the host TN is configured.

Verify TEP to TEP Connectivity

To avoid any overlay communication in the future due to MTU issue, test TEP to TEP connectivity and verify that it is working.
1. SSH to edge-node-1 and get the local TEP IP address, such as 192.23.213.1. Use the command `get_vteps` to get the IP.

2. SSH to edge-node-2 and get the local TEP IP address, such as 192.23.213.2. Use the command `get_vteps` to get the IP.

3. SSH to the ESXi host and get the TEP IP address, such as 192.23.213.3. Use the command `esxcfg-vmknic -l` to get the IP. The interface will be `vmk10` and the NetStack will be `vxlan`.

4. From each ESXi transport node, test the connections to each NSX-T Edge Node, for example:

   a. Test the connection from NSX-T edge node 1 and edge node 2 to ESXi TN:

   ```
   # vmkping +netstack=vxlan 192.23.213.1 -d -s 1572 -I vmk10: OK
   # vmkping +netstack=vxlan 192.23.213.2 -d -s 1572 -I vmk10: OK
   ```

   b. Test the connection from NSX-T edge node 1 to NSX-T edge node 2:

   ```
   > vrf 0
   > ping 192.23.213.1 size 1572 dfbit enable: OK
   > vrf 0
   > ping 192.23.213.2 size 1572 dfbit enable: OK
   ```

Create NSX-T Edge Cluster

1. Go to **System > Fabric > Nodes > Edge Clusters**.

2. Click **Add**.

   - Enter a name, such as `edge-cluster-1`.
   - Add members, including `edge-node-1` and `edge-node-2`.
3. Click **Add**.

4. Verify.
Create Uplink Logical Switch

Create an uplink Logical Switch to be used for the Tier-0 Router.

1. At upper-right, select the Manager tab.

2. Go to Networking > Logical Switches.

3. Click Add.

4. Configure the new logical switch as follows:

   - Name: LS-T0-uplink
   - Transport Zone: tz-vlan
   - VLAN: 1548
5. Click Add.

6. Verify.
Create Tier-0 Router

1. Select **Networking** from the **Manager** tab.

2. Select **Tier-0 Logical Router**.

3. Click **Add**.

4. Configure the new Tier-0 Router as follows:
   - **Name**: T0-router
   - **Edge Cluster**: edge-cluster-1
   - **HA mode**: Active-Standby
   - **Failover mode**: Non-Preemptive
New Tier-0 Router

Tier-0 Router  Advanced

Name*  T0-router

Description

Edge Cluster  edge-cluster-1

High Availability Mode  
- Active-Active
- Active-Standby

Failover Mode  
- Preemptive
- Non-Preemptive

OR Create a New Edge Cluster

CANCEL  ADD

5. Click Save and verify.

6. Select the T0 router.
7. Select **Configuration > Router Ports**.

8. Click **Add**.

9. Configure a new router port as follows:
   - **Name**: T0-uplink-1
   - **Type**: uplink
   - **Transport Node**: edge-node-1
   - **Logical Switch**: LS-T0-uplink
   - **Logical Switch Port**: Attach to a new switch port
   - **Subnet**: 10.173.62.50 / 24
10. Click **Add** and verify.

11. Select the T0 router.

12. Select **Configuration > Router Ports**.

13. Add a second uplink by creating a second router port for edge-node-2:
   - **Name**: T0-uplink-1
   - **Type**: uplink
   - **Transport Node**: edge-node-2
   - **Logical Switch**: LS-T0-uplink
   - **Logical Switch Port**: Attach to a new switch port
   - **Subnet**: 10.173.62.51 / 24
Once completed, verify that you have two connected router ports.

**Configure and Test the Tier-0 Router**

Create an HA VIP for the T0 router, and a default route for the T0 router. Then test the T0 router.

1. Select the Tier-0 Router you created.

2. Select **Configuration > HA VIP**.

3. Click **Add**.

4. Configure the HA VIP as follows:
   - **VIP address**: 10.173.62.52/24, for example
   - **Uplink ports**: T0-uplink-1 and T0-uplink-2

5. Click **Add** and verify.
6. Select **Routing > Static Routes**.

7. Click **Add**.
   - **Network**: 0.0.0.0/0
   - **Next Hop**: 10.173.62.253

8. Click **Add** and verify.
9. Verify the Tier 0 router by making sure the T0 uplinks and HA VIP are reachable from your laptop.

For example:

```
> ping 10.173.62.50
PING 10.173.62.50 (10.173.62.50): 56 data bytes
Request timeout for icmp_seq 0
64 bytes from 10.173.62.50: icmp_seq=0 ttl=58 time=71.74 ms
64 bytes from 10.173.62.50: icmp_seq=1 ttl=58 time=1074.67 ms

> ping 10.173.62.51
PING 10.173.62.51 (10.173.62.51): 56 data bytes
Request timeout for icmp_seq 0
64 bytes from 10.173.62.51: icmp_seq=0 ttl=58 time=1156.63 ms
64 bytes from 10.173.62.51: icmp_seq=1 ttl=58 time=151.41 ms

> ping 10.173.62.52
PING 10.173.62.52 (10.173.62.52): 56 data bytes
64 bytes from 10.173.62.52: icmp_seq=0 ttl=58 time=6.86 ms
64 bytes from 10.173.62.52: icmp_seq=1 ttl=58 time=7.78 ms
```

Create IP Blocks and Pool for Compute Plane

TKGI requires a Floating IP Pool for NSX-T load balancer assignment and the following 2 IP blocks for Kubernetes pods and nodes:

- PKS-POD-IP-BLOCK: 172.16.0.0/16
- PKS-NODE-IP-BLOCK: 172.23.0.0/16

1. In the Manager interface, go to Networking > IP Address Pools > IP Block.

2. Click Add.

3. Configure the Pod IP Block as follows:
   - **Name**: PKS-POD-IP-BLOCK
   - **CIDR**: 172.16.0.0/16
4. Click **Add** and verify.

5. Repeat same operation for the Node IP Block.
   - **Name**: PKS-NODE-IP-BLOCK
   - **CIDR**: 172.23.0.0/16

6. Click **Add** and verify.

7. Select **IP Pools** tab.
8. Click **Add**.

9. Configure the IP pool as follows:
   - **Name**: PKS-FLOATING-IP-POOL
   - **IP ranges**: 10.173.62.111 - 10.173.62.150
   - **CIDR**: 10.173.62.0/24

```
Add New IP Pool

Name *
PKS-FLOATING-IP-POOL

Description

Subnets

<table>
<thead>
<tr>
<th>IP Ranges *</th>
<th>Gateway</th>
<th>CIDR *</th>
<th>DNS Servers</th>
<th>DNS Suffix</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.173.62.111 - 10.173.62.150</td>
<td>10.173.62.0/24</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

10. Click **Add** and verify.
Create Management Plane

Networking for the TKGI Management Plane consists of a Tier-1 Router and Switch with NAT Rules for the Management Plane VMs.

Create Tier-1 Router and Switch

Create Tier-1 Logical Switch and Router for TKGI Management Plane VMs. Complete the configuration by enabling Route Advertisement on the T1 router.

1. In the NSX Management console, navigate to Networking > Logical Switches.

2. Click Add.

3. Create the LS for TKGI Management plane VMs:
   - Name: LS-PKS-MGMT
   - Transport Zone: tz-overlay
4. Click **Add** and verify creation of the T1 logical switch.

5. Go to Networking > Tier-1 Logical Router.

---

**Add New Logical Switch**

**General**

Name*  
LS-PKS-MGMT

Description

Transport Zone*  
tz-overlay

Uplink Teaming Policy Name*  
[Use Default]

Admin Status  
Up

Replication Mode  
- Hierarchical Two-Tier replication
- Head End replication

**VLAN**

Only VLAN Trunk Spec is allowed (eg: 1, 5, 10-12, 31-35).

---

**Table:**

<table>
<thead>
<tr>
<th>Switches</th>
<th>Ports</th>
<th>Switching Profiles</th>
<th>Edge Bridge Profiles</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ADD</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logical Switch</td>
<td>ID</td>
<td>Admin Status</td>
<td>Logical Ports</td>
</tr>
<tr>
<td>LS-PKS-MGMT</td>
<td>fe6a:efbb</td>
<td>Up</td>
<td>Overlay</td>
</tr>
<tr>
<td>LS-T1-uplink</td>
<td>f70b:36f8</td>
<td>Up</td>
<td>4</td>
</tr>
</tbody>
</table>

---

Copyright © 2020 VMware, Inc. All Rights Reserved.
6. Click **Add**.

7. Configure the Tier-1 logical router as follows:
   - **Name**: T1-PKS-MGMT
   - **To router**: T0-router
   - **Edge Cluster**: edge-cluster-1
   - **Edge Cluster Members**: edge-node-1 and edge-node-2
New Tier-1 Router

Name*  T1-PKS-MGMT

Description

Tier-O Router  T0-router

Edge Cluster  edge-cluster-1

StandBy Relocation  Off  Disable

Failover Mode  ○ Preemptive  ○ Non-Preemptive

Edge Cluster Members  edge-node-1  edge-node-2

8. Click Add and verify.

9. Select the T1 router and go to Configuration > Router port.
10. Click Add.

11. Configure the T1 router port as follows:
   - **Name**: T1-PKS-MGMT-port
   - **Logical Switch**: LS-PKS-MGMT
   - **Subnet**: 10.1.1.1/24

12. Click Add and verify.
13. Select **Routing** tab.

14. Click **Edit** and configure route advertisement as follows:
   - **Status**: Enabled
   - **Advertise All Connected Routes**: Yes
15. Click Save and verify.
Create NAT Rules

You need to create the following NAT rules on the Tier-0 router for the TKGI Management Plane VMs.

- **DNAT:** 10.173.62.220 (for example) to access Ops Manager
- **DNAT:** 10.173.62.221 (for example) to access Harbor
- **SNAT:** 10.173.62.222 (for example) for all TKGI management plane VM traffic destined to the outside world

1. In the NSX Management console, navigate to **Networking > NAT**.

2. In the Logical Router field, select the T0-router you defined for TKGI.

3. Click **Add**.

4. Configure the Ops Manager DNAT rule as follows:
   - **Priority:** 1000
   - **Action:** DNAT
   - **Protocol:** Any
   - **Destination IP:** 10.173.62.220, for example
   - **Translated IP:** 10.1.1.2, for example
**New NAT Rule**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>1000</td>
</tr>
<tr>
<td>Action</td>
<td>DNAT</td>
</tr>
<tr>
<td>Protocol</td>
<td>Any Protocol</td>
</tr>
<tr>
<td>Source IP</td>
<td></td>
</tr>
<tr>
<td>Destination IP</td>
<td>10.173.62.220</td>
</tr>
<tr>
<td>Translated IP</td>
<td>10.1.1.2</td>
</tr>
<tr>
<td>Translated Ports</td>
<td></td>
</tr>
<tr>
<td>Applied To</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>Enabled</td>
</tr>
<tr>
<td>Logging</td>
<td>Disabled</td>
</tr>
<tr>
<td>Firewall Bypass</td>
<td></td>
</tr>
</tbody>
</table>

5. Click **Add** and verify.
6. Add a second DNAT rule for Harbor by repeating the same operation.
   - **Priority**: 1000
   - **Action**: DNAT
   - **Protocol**: Any Protocol
   - **Destination IP**: 10.173.62.221, for example
   - **Translated IP**: 10.1.1.6, for example

7. Verify the creation of the DNAT rules.

8. Create the SNAT rule for the management plane traffic as follows:
   - **Priority**: 9024
   - **Action**: SNAT
   - **Protocol**: Any Protocol
   - **Source IP**: 10.1.1.0/24, for example
   - **Translated IP**: 10.173.62.222, for example
### New NAT Rule

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>9024</td>
</tr>
<tr>
<td>Action*</td>
<td>SNAT</td>
</tr>
<tr>
<td>Protocol</td>
<td>Any Protocol</td>
</tr>
<tr>
<td>Source IP</td>
<td>10.1.1.0/24</td>
</tr>
<tr>
<td>Destination IP</td>
<td></td>
</tr>
<tr>
<td>Translated IP*</td>
<td>10.173.62.222</td>
</tr>
<tr>
<td>Applied To</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>Enabled</td>
</tr>
<tr>
<td>Logging</td>
<td>Disabled</td>
</tr>
<tr>
<td>Firewall Bypass</td>
<td></td>
</tr>
</tbody>
</table>

9. Verify the creation of the SNAT rule.
Configure the NSX-T Password Interval (Optional)

The default NSX-T password expiration interval is 90 days. After this period, the NSX-T passwords will expire on all NSX-T Manager Nodes and all NSX-T Edge Nodes. To avoid this, you can extend or remove the password expiration interval, or change the password if needed.

**Note:** For existing Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition Tile](#) for more information.

**Update the NSX-T Manager Password and Password Interval**

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 into 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`. For example:

```bash
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.

**Retrieve the Password Expiration Interval**

To retrieve the password expiration interval, use the following command:

```bash
get user USERNAME password-expiration
```

For example:

```
NSX CLI (Manager, Policy, Controller 3.0.0.0.0.0.0.0.0.15946739). Press ? for cost or enter: help
nsx-mgr-1> 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:

```bash
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-mgr-1> clear user admin password-expiration
nsx-mgr-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.

*Note: Unlike the NSX-T Manager nodes, you must update the password or password interval on each 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 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 CLI (Edge 3.0.0.0.0.15946012). Press ? for command list or enter: help
nsx-edge-2> get user admin password-expiration
Password expires 90 days after last change. Current password will expire in 7 days.
nsx-edge-2> clear user admin password-expiration
nsx-edge-2> get user admin password-expiration
Password expiration not configured for this user
```

Next Steps

Once you have completed the installation of NSX-T v3.0, return to the TKGI installation workflow and proceed with the next phase of the process. See Install Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T Using Ops Manager.

Please send any feedback you have to pks-feedback@pivotal.io.
Deploying Ops Manager with NSX-T for Tanzu Kubernetes Grid Integrated Edition

In this topic

Prerequisites

Step 1: Generate SSH Key Pair

Step 2: Deploy Ops Manager for Tanzu Kubernetes Grid Integrated Edition

Network Selection for vSphere v6.5

Step 3: Configure Ops Manager for Tanzu Kubernetes Grid Integrated Edition

Next Step

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic provides instructions for deploying Ops Manager on VMware vSphere with NSX-T integration for use with VMware Tanzu Kubernetes Grid Integrated Edition.

Prerequisites

Before deploying Ops Manager with NSX-T for Tanzu Kubernetes Grid Integrated Edition, you must have completed the following tasks:

- Preparing to Install Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T Data Center
- Depending on your NSX-T version:
  - NSX-T v3.0: Installing and Configuring NSX-T Data Center v3.0 for Tanzu Kubernetes Grid Integrated Edition
  - NSX-T v2.5: See the v2.7 documentation
- Installing and Configuring NSX-T Data Center v2.5 for Enterprise PKS
- Creating the Tanzu Kubernetes Grid Integrated Edition Management Plane
- Create Tanzu Kubernetes Grid Integrated Edition Compute Plane

In addition, review the supported Ops Manager versions for Tanzu Kubernetes Grid Integrated Edition. See VMware Tanzu Network.

Review the known issues for your version of Ops Manager. See one of the following:

- Ops Manager v2.7 Release Notes
- Ops Manager v2.8 Release Notes

Step 1: Generate SSH Key Pair

You cannot deploy Ops Manager 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, 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/bynbeHOD4Ptes ... x87/dUuqvVISAJ8yGu2hJobx9PPStFERtUsfx5x+WIu9XIkrl5tzxgH9hBDsOS9cVUYJ7kKUUf1yyro6ocHyu6TWHJHSJLt8Z2FULxMPpqdn+8Xw==
```

Step 2: Deploy Ops Manager for Tanzu Kubernetes Grid Integrated Edition


2. Before starting, refer to the known issues in the Ops Manager Release v2.7 Release Notes or the Ops Manager Release v2.8 Release Notes.

3. Download the Ops Manager for vSphere installation file from the VMware Tanzu Network.
   a. Open a browser to the Ops Manager download page on the VMware Tanzu Network.
   b. Use the dropdown menu to select the supported Ops Manager release.
   c. Select the Ops Manager for vSphere download option. This downloads the 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.


7. At the Select an OVF template screen:
   a. Click Browse.
   b. Select the Ops Manager OVA file you downloaded and click Open.
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 Tanzu Kubernetes Grid Integrated Edition 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**.
12. At the Select Networks screen:

- If you are using vSphere 6.7, select the Tanzu Kubernetes Grid Integrated Edition Management T1 Logical Switch that you defined when Create Management Plane in Installing and Configuring NSX-T Data Center v3.0 for TKGI, and click Next.

13. At the Customize template screen, enter the following information, and click Next.

<table>
<thead>
<tr>
<th>Information</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Address</td>
<td>The IP address of the Ops Manager network interface, for example 10.0.0.2 (assuming non-routable NAT-mode).</td>
</tr>
<tr>
<td>Netmask</td>
<td>The network mask for Ops Manager, for example, 255.255.255.0.</td>
</tr>
<tr>
<td>Default Gateway</td>
<td>The default gateway for Ops Manager to use, for example 10.0.0.1 (assuming non-routable NAT-mode).</td>
</tr>
<tr>
<td>DNS</td>
<td>One or more DNS servers for the Ops Manager VM to use, for example 10.14.7.1.</td>
</tr>
<tr>
<td>NTP Servers</td>
<td>The IP address of one or more NTP servers for Ops Manager, for example 10.113.60.176.</td>
</tr>
<tr>
<td>Public SSH Key</td>
<td>(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 Step 1: Generate SSH Key Pair.</td>
</tr>
<tr>
<td>Custom hostname</td>
<td>The hostname for the Ops Manager VM, for example ops-manager.</td>
</tr>
</tbody>
</table>
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 TKGI Management Plane, for example LS-MGMT-TKGI.

Step 3: Configure Ops Manager for Tanzu Kubernetes Grid Integrated Edition

The first time you start Ops Manager, you are required to select an authentication system. These instructions use Internal Authentication. See Set Up Ops Manager in the Ops Manager 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 Management Plane in Installing and Configuring NSX-T Data Center v3.0 for TKGI 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 Ops Manager 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.

![Image](image)

*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.

![Image](image)

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.

![Image](image)

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.

*Note:* It is normal to experience a brief delay before the interface is accessible while the web server and VM start up.
Next Step

After you complete this procedure, follow the instructions in Generate and Register the NSX-T Management SSL Certificate and Private Key in Installing and Configuring NSX-T Data Center v3.0 for TKGI.

Please send any feedback you have to pks-feedback@pivotal.io.
Configuring BOSH Director with NSX-T for Tanzu Kubernetes Grid Integrated Edition

In this topic
Prerequisites
How Ops Manager Accesses NSX Manager
Step 1: Log in to Ops Manager
Step 2: Configure vCenter for Tanzu Kubernetes Grid Integrated Edition
Step 3: Configure BOSH Director
Step 4: Create Availability Zones
Step 5: Create Networks
Step 6: Assign AZs and Networks
Step 7: Configure Security
Step 8: Configure BOSH DNS
Step 9: Configure Logging
Step 10: Configure Resources
Step 11: (Optional) Add Custom VM Extensions
Step 12: Deploy BOSH
Step 12: Update Network Availability Zones
Next Step

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to configure BOSH Director for vSphere with NSX-T integration for VMware Tanzu Kubernetes Grid Integrated Edition.

Prerequisites

Before you begin this procedure, ensure that you have successfully completed all preceding steps for installing Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T, including:

- Preparing to Install Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T Data Center
- Hardware Requirements for Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T
- Depending on your NSX-T version:
  - NSX-T v3.0: Installing and Configuring NSX-T Data Center v3.0 for Tanzu Kubernetes Grid Integrated Edition
  - NSX-T v2.5: See the v1.7 documentation:
    - Installing and Configuring NSX-T Data Center v2.5 for Enterprise PKS
    - Creating the Tanzu Kubernetes Grid Integrated Edition Management Plane
    - Create Tanzu Kubernetes Grid Integrated Edition Compute Plane
Deploying Ops Manager with NSX-T for Tanzu Kubernetes Grid Integrated Edition

Deploying Ops Manager with NSX-T for Tanzu Kubernetes Grid Integrated Edition

Generate and Register the NSX-T Management SSL Certificate and Private Key in Installing and Configuring NSX-T Data Center v3.0 for TKGI

---

How Ops Manager Accesses NSX Manager

To create, delete, and modify NSX-T networking resources, Ops Manager tiles and APIs use a VMware NSX Manager account with the Enterprise Administrator role and permissions.

Users configure Ops Manager to authenticate to NSX Manager for different purposes in different tiles:

- **Tanzu Kubernetes Grid Integrated Edition tile**
  The Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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 Ops Manager component VMs, such as VMware Tanzu Application Service for VMs routers.

  To configure the BOSH Director for vSphere tile’s authentication to NSX Manager, see Configure vCenter for Tanzu Kubernetes Grid Integrated Edition, 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 Tanzu Kubernetes Grid Integrated Edition.

2. Click the BOSH Director for vSphere tile.
Step 2: Configure vCenter for Tanzu Kubernetes Grid Integrated Edition

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. Tanzu Kubernetes Grid Integrated Edition 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**.

   ![NSX Networking](image)

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.
   - **Template Folder**: The vSphere datacenter folder where Ops Manager places VMs.
   - **Disk path Folder**: The vSphere datastore folder where Ops Manager creates attached disk images. You must not nest this folder.
Step 3: Configure BOSH Director

1. Select **Director Config**.

6. Click **Save**.
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 VMs deployed by the BOSH Director** 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 Tanzu Kubernetes Grid Integrated Edition.

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

8. For typical Tanzu Kubernetes Grid Integrated Edition 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 Ops Manager 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 Tanzu Kubernetes Grid Integrated Edition 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 Ops Manager documentation for details.
9. Click **Save**.

---

**Step 4: Create Availability Zones**

On vSphere with NSX-T, operators define and create Availability Zones (AZs) using vCenter clusters and resource pools. Plans defined in the TKGI tile then use these AZs to enable high availability for TKGI clusters.

The Tanzu Kubernetes Grid Integrated Edition control plane also runs in one of the AZs.

For more information on AZs in TKGI, see Availability Zones in Tanzu Kubernetes Grid Integrated Edition Architecture.

To create Availability Zones in the BOSH Director tile:

1. Select **Create Availability Zones**.
2. Use the following steps to create one or more Availability Zones for Tanzu Kubernetes Grid Integrated Edition to use:

   a. Click **Add** and create the Tanzu Kubernetes Grid Integrated Edition Management AZ.
   b. Enter a unique **Name** for the Availability Zone, such as `AZ-MGMT`.
   c. Select the IaaS configuration (vSphere/vCenter).
   d. Enter the name of an existing vCenter **Cluster** to use as an Availability Zone, such as `COMP-Cluster-1`.
   e. Enter the name of the Tanzu Kubernetes Grid Integrated Edition Management **Resource Pool** in the vCenter cluster that you specified above, such as `RP-MGMT-TKGI`. The jobs running in this Availability Zone share the CPU and memory resources defined by the pool.
   f. Click **Add Cluster** and create at least one Tanzu Kubernetes Grid Integrated Edition Compute AZ.
   g. Specify the **Cluster** and the **Resource Pool**, such as `RP-TKGI-AZ`. Alternatively, specify the **Cluster** and the **Host Group**. See Using vSphere Host Group for more information.
   h. (Optional) If you are using a host group with vSAN stretched clusters, set the **VM-Host Affinity Rule** dropdown to **Should**. This setting maintains high availability by letting TKGI restart VMs in another host group if their AZ fails. TKGI ignores this setting if the vSAN cluster has no host group configured.

   For more information, see Ability to Set the VM-Host Affinity Rule to “Should” for Clusters in vSphere in the Ops Manager v2.9 Release Notes.

   i. Add additional clusters as necessary. Click the trash icon to delete a cluster. The first cluster cannot be deleted.
3. Click **Save**.
Step 5: Create Networks

1. Select **Create Networks**.
2. Select **Enable ICMP checks** to enable ICMP on your networks. Ops Manager uses ICMP checks to confirm that components within your network are reachable.

3. Click **Add Network**.

4. Create the following network:

   ![Network Configuration](image)

   **Note**: NSX-T automatically creates the service network to be used by the master and worker nodes (VMs) for Kubernetes clusters managed by Tanzu Kubernetes Grid Integrated Edition. 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 Create Management Plane in *Installing and Configuring NSX-T Data Center v3.0 for TKGI*. Reserve any IP addresses from the subnet that are already in use, such as the IP for Ops Manager and subnet gateway.

<table>
<thead>
<tr>
<th>Field</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>NET-MGMT-TKGI</td>
</tr>
<tr>
<td>vSphere Network Name</td>
<td>LS-MGMT-TKGI</td>
</tr>
<tr>
<td>CIDR</td>
<td>10.0.0.0/24</td>
</tr>
<tr>
<td>Reserved IP Ranges</td>
<td>10.0.0.1-10.0.0.2</td>
</tr>
<tr>
<td>DNS</td>
<td>10.20.20.1</td>
</tr>
<tr>
<td>Gateway</td>
<td>10.0.0.1</td>
</tr>
</tbody>
</table>

5. Select the **AZ-MGMT** Availability Zone to use with the **NET-MGMT-TKGI** network.
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 Tanzu Kubernetes Grid Integrated Edition, 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 Tanzu Kubernetes Grid Integrated Edition Management Plane network. Select the **NST-MGMT-TKGI** 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 Tanzu Kubernetes Grid Integrated Edition](https://docs.vmware.com/en/tanzu-kubernetes-grid-integrated-edition/2.4/tpoc/tpoc-tech/zh-cn/security.html) 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.

ıldığı süreye kadar devam eden belirli bir işlem yoksa, işlem başarısız olur. 

**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.
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**.
   
   ![Installation Dashboard](image)

2. Click **Review Pending Changes**.
3. Click **Apply Changes**.

4. Confirm changes applied successfully.

Step 12: Update Network Availability Zones

After BOSH is successfully deployed, update the network you defined above (NET-MGMT-TKGI) to include each of the Compute AZs that you defined. This ensures that both the Management AZ and the Compute AZs appear in the Tanzu Kubernetes Grid Integrated Edition tile for the Plans.

1. Return to the BOSH tile and click Create Networks.
2. Edit the network (NET-MGMT-TKGI) 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 Tanzu Kubernetes Grid Integrated Edition.

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
Prerequisites
How Ops Manager Accesses NSX Manager
Options for Generating the Certificate and Key
  Option A: Generate and Register the Certificate and Key Using Scripts
  Option B: Generate and Register the Certificate and Key Using the Tanzu Kubernetes Grid Integrated Edition Tile
Next Step

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

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 Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T.

The NSX Manager superuser for TKGI 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 Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T, including the following:

- Preparing to Install Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T Data Center
- Hardware Requirements for Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T
- Depending on your NSX-T version:
  - NSX-T v3.0: Installing and Configuring NSX-T Data Center v3.0 for Tanzu Kubernetes Grid Integrated Edition
  - NSX-T v2.5: See the v1.7 documentation:
    - Installing and Configuring NSX-T Data Center v2.5 for Enterprise PKS
    - Creating the Tanzu Kubernetes Grid Integrated Edition Management Plane
    - Create Tanzu Kubernetes Grid Integrated Edition Compute Plane
- Deploying Ops Manager with NSX-T for Tanzu Kubernetes Grid Integrated Edition
- Generate and Register the NSX-T Management SSL Certificate and Private Key in Installing and Configuring NSX-T Data Center v3.0 for TKGI
- Configuring BOSH Director with NSX-T for Tanzu Kubernetes Grid Integrated Edition

How Ops Manager Accesses NSX Manager
To create, delete, and modify NSX-T networking resources, Ops Manager tiles and APIs use a VMware NSX Manager account with the Enterprise Administrator role and permissions.

Users configure Ops Manager to authenticate to NSX Manager for different purposes in different tiles:

- **Tanzu Kubernetes Grid Integrated Edition tile**
  The Tanzu Kubernetes Grid Integrated Edition 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 **Tanzu Kubernetes Grid Integrated Edition** 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 Ops Manager component VMs, such as VMware Tanzu Application Service for VMs routers.

  To configure the **BOSH Director for vSphere** tile’s authentication to NSX Manager, see Configure vCenter for Tanzu Kubernetes Grid Integrated Edition in Configuring BOSH Director with NSX-T for Tanzu Kubernetes Grid Integrated Edition.

About the NSX Manager Super User Principal Identity

The TKGI API accesses the NSX Manager through an Enterprise Administrator account. This superuser account lets TKGI use NSX-T to create, delete, and modify networking resources for Kubernetes cluster nodes.

When you configure Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition tile.

See the **NSX Manager Super User Principal Identity Certificate** field in the following screenshot:

![NSX Manager Super User Principal Identity Certificate](image)

View a larger version of this image.

For more information, see the Networking section of Installing Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition tile. For more information, see Option B: Generate and Register the Certificate and Key Using the Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition tile.

**Note**: The Linux VM must have OpenSSL installed and have network access to the NSX Manager. For example, you can use the TKGI client VM where you install the TKGI 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:

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 \
  -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
') \
  -sha256 \
  -days 730

cert_request=$(cat <<END
{
  "display_name": "$PI_NAME",
  "pem_encoded": "$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.](image)

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:

```bash
#!/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**.

---

**Option B: Generate and Register the Certificate and Key Using the Tanzu Kubernetes Grid Integrated Edition Tile**

**Step 1: Generate the Certificate and Key**

To generate the certificate and key automatically in the **Networking** pane in the Tanzu Kubernetes Grid Integrated Edition tile, follow the steps below:


2. Click **Generate RSA Certificate** and provide a wildcard domain. For example, `*.nsx.tkgi.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 TKGI client VM where you install the TKGI 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`.

```bash
export NSX_MANAGER_IP="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:

   ```bash
cert_request=$(cat <<END
   {
   "display_name": "$PI_NAME",
   "pem_encoded": "$\(awk '{printf "%s\n", $0}'} $NSX_SUPERUSER_CERT_FILE"
   }
END)

```

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:

   ```bash
   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 Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T.

Please send any feedback you have to pks-feedback@pivotal.io.
Installing Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T

In this topic
Prerequisites
Step 1: Install Tanzu Kubernetes Grid Integrated Edition
Step 2: Configure Tanzu Kubernetes Grid Integrated Edition
  Assign AZs and Networks
  TKGI API
  Plans
  Kubernetes Cloud Provider
  Networking
  UAA
  (Optional) Host Monitoring
  (Optional) In-Cluster Monitoring
  Tanzu Mission Control (Experimental)
  CEIP and Telemetry
  Errands
  Resource Config
Step 3: Apply Changes
Step 4: Install the TKGI and Kubernetes CLIs
Step 5: Verify NAT Rules
Step 6: Configure Authentication for Tanzu Kubernetes Grid Integrated Edition
Next Steps

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to install and configure VMware Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T integration.

Prerequisites
Before you begin this procedure, ensure that you have successfully completed all preceding steps for installing Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T, including:

- Preparing to Install Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T Data Center
- Depending on your NSX-T version:
  - **NSX-T v3.0**: Installing and Configuring NSX-T Data Center v3.0 for Tanzu Kubernetes Grid Integrated Edition
  - **NSX-T v2.5**: See the v1.7 documentation:
    - Installing and Configuring NSX-T Data Center v2.5 for Enterprise PKS
    - Creating the Tanzu Kubernetes Grid Integrated Edition Management Plane
Step 1: Install Tanzu Kubernetes Grid Integrated Edition

To install Tanzu Kubernetes Grid Integrated Edition, do the following:

1. Download the product file from VMware Tanzu 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 Tanzu Kubernetes Grid Integrated Edition in the left column, click the plus sign to add this product to your staging area.

Step 2: Configure Tanzu Kubernetes Grid Integrated Edition

Click the orange Tanzu Kubernetes Grid Integrated Edition tile to start the configuration process.

**Note:** Configuration of NSX-T or Flannel cannot be changed after initial installation and configuration of Tanzu Kubernetes Grid Integrated Edition.

**Warning:** When you configure the Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition fails.

Assign AZs and Networks
To configure the availability zones (AZs) and networks used by the Tanzu Kubernetes Grid Integrated Edition 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 TKGI API and TKGI Database VMs.


   ![Assign AZs and Networks](image)

   **Note:** You must specify the **Balance other jobs in** AZ, but the selection has no effect in the current version of Tanzu Kubernetes Grid Integrated Edition.

4. Under **Network**, select the TKGI Management Network linked to the `ls-tkgi-mgmt` NSX-T logical switch you created in the [Create Networks Page](#) step of Configuring BOSH Director with NSX-T for Tanzu Kubernetes Grid Integrated Edition. This provides network placement for Tanzu Kubernetes Grid Integrated Edition component VMs, such as the TKGI API and TKGI Database VMs.

5. Under **Service Network**, your selection depends on whether you are installing a new Tanzu Kubernetes Grid Integrated Edition deployment or upgrading from a previous version of Tanzu Kubernetes Grid Integrated Edition.

   - If you are deploying Tanzu Kubernetes Grid Integrated Edition with NSX-T for the first time, select the TKGI Management Network that you specified in the **Network** field. You do not need to create or define a service network because Tanzu Kubernetes Grid Integrated Edition creates the service network for you during the installation process.
   - If you are upgrading from a previous version of Tanzu Kubernetes Grid Integrated Edition, then select the **Service Network** linked to the `ls-tkgi-service` NSX-T logical switch that Tanzu Kubernetes Grid Integrated Edition created for you during installation. The service network provides network placement for existing on-demand Kubernetes cluster service instances that were created by the Tanzu Kubernetes Grid Integrated Edition broker.

6. Click **Save**.

TKGI API

Perform the following steps:
1. Click **TKGI API**.

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

The certificate that you supply should cover the specific subdomain that routes to the TKGI 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 *.tkgi.EXAMPLE.com does not permit communication to *.api.tkgi.EXAMPLE.com. If the certificate does not contain the correct FQDN for the TKGI 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. Click **Generate RSA Certificate** for a new install or **Change** to update a previously-generated certificate.

b. Enter the domain for your API hostname. This must match the domain you configured under **TKGI API > API Hostname (FQDN)** in the Tanzu Kubernetes Grid Integrated Edition tile. It can be a standard FQDN or a wildcard domain.

c. Click **Generate**.
3. Under **API Hostname (FQDN)**, enter the FQDN that you registered to point to the TKGI API load balancer, such as `api.tkgi.example.com`. To retrieve the public IP address or FQDN of the TKGI 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 TKGI 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...
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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition. If you select more than one AZ, Tanzu Kubernetes Grid Integrated Edition deploys the master VM in the first AZ and the worker VMs across the remaining AZs. If you are using multiple masters, Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition can deploy for each cluster. Enter any whole number in this field.

10. Under Worker Node Instances, specify the default number of Kubernetes worker nodes the TKGI CLI provisions for each cluster. The Worker Node Instances setting must be less than, or equal to, the Maximum number of workers on a cluster setting.

   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 TKGI 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 Tanzu Kubernetes Grid Integrated Edition Clusters.

Note: If you install Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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. Tanzu Kubernetes Grid Integrated Edition 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.

   ![Allow Privileged checkbox](image)

   **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.


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.

   ![Node Drain Timeout](image)
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 Tanzu Kubernetes Grid Integrated Edition.

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.
Choose your IaaS*

- GCP
- vSphere

**vCenter Master Credentials** *

- user@example.com
- 

**vCenter Host** *

- vcenter-example.com

**Datacenter Name** *

- example-dc

**Datastore Name** *

- example-ds

**Stored VM Folder** *

- pks_vms

---

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 Tanzu Kubernetes Grid Integrated Edition.

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. Tanzu Kubernetes Grid Integrated Edition 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 Generate and Register the NSX-T Management SSL Certificate and Private Key. Use this certificate and key to connect to the NSX Manager.
   d. The Disable SSL certificate verification checkbox is not selected by default. In order to disable TLS verification, select the checkbox. You may want to disable TLS verification if you did not enter a CA certificate, or if your CA certificate is self-
signed.

**Note:** The NSX Manager CA Cert field and the Disable SSL certificate verification option are intended to be mutually exclusive. If you disable SSL certificate verification, leave the CA certificate field blank. If you enter a certificate in the NSX Manager CA Cert field, do not disable SSL certificate verification. If you populate the certificate field and disable certificate validation, insecure mode takes precedence.

e. If you are using a NAT deployment topology, leave the NAT mode checkbox selected. If you are using a No-NAT topology, clear this checkbox. For more information, see Deployment Topologies.

f. Enter the following IP Block settings:

```
| Pods IP Block ID | 0eb628ea-3032-4a97-b17b-a63be7baa7fb |
| Nodes IP Block ID | 2250dc43-63c8-4bb8-b8cf-c6e12c6f7de |
| T0 Router ID | 1df76028b-fafe-4430-5033-504f9a0d1f7d |
| Floating IP Pool ID | 7e35e93-e2fa-49ca-b065-4eabcc3cb7de |
| Nodes DNS | 192.168.115.1 |
| vSphere Cluster Names | kubo-az-1,kubo-az-2,kubo-az-3 |
| Kubernetes Service Network CIDR Range | 10.100.200.0/24 |
```

View a larger version of this image.

- **Pods IP Block ID:** Enter the UUID of the IP block to be used for Kubernetes pods. Tanzu Kubernetes Grid Integrated Edition 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. Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition.

For T0 Router ID, enter the t0-tkgi T0 router UUID. Locate this value in the NSX-T UI router overview.

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. Tanzu Kubernetes Grid Integrated Edition 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.

For Nodes DNS, enter one or more Domain Name Servers used by the Kubernetes nodes.

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`.

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 TKGI API server. See Using Proxies with Tanzu Kubernetes Grid Integrated Edition on NSX-T for instructions on how to enable a proxy.

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

5. Click Save.

UAA

To configure the UAA server:

1. Click UAA.

2. Under TKGI API Access Token Lifetime, enter a time in seconds for the TKGI API access token lifetime. This field defaults to 600.
Under **TKGI API Refresh Token Lifetime**, enter a time in seconds for the TKGI API refresh token lifetime. This field defaults to **21600**.

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

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

---

**Note:** VMware recommends using the default UAA token timeout values. By default, access tokens expire after ten minutes and refresh tokens expire after six hours.

6. Under **Configure created clusters to use UAA as the OIDC provider**, select **Enabled** or **Disabled**. This is a global default setting for TKGI-provisioned clusters. For more information, see [OIDC Provider for Kubernetes Clusters](#).

To configure Tanzu Kubernetes Grid Integrated Edition to use UAA as the OIDC provider:

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

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`. The default value is `oidc:`.

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, you can enter claims besides `user_name`, like `email` or `name`.

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`. The default value is `oidc:`.

**warning:** VMware recommends adding OIDC prefixes to prevent users and groups from gaining unintended cluster privileges. If you change the above values for a pre-existing Tanzu Kubernetes Grid Integrated Edition 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, 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 Tanzu Kubernetes Grid Integrated Edition to an LDAP Server.
   - To use SAML for UAA, select **SAML Identity Provider** and continue to Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider.

(Optional) Host Monitoring

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

- To configure Syslog, 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 TKGI API, master node, and worker node VMs to a monitoring service, such as Wavefront or Datadog.

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

Syslog

To configure Syslog for all BOSH-deployed VMs in Tanzu Kubernetes Grid Integrated Edition:

1. Click Host Monitoring.

2. Under Enable Syslog for TKGI select Yes.

3. Under Address, enter the destination syslog endpoint.

4. Under Port, enter the destination syslog port.


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. VMware recommends that you
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](#).

1. By default, vRLI logging is disabled. To configure vRLI logging, under **Enable VMware vRealize Log Insight Integration?**, select **Yes** and then perform the following steps:

   ![Enable VMware vRealize Log Insight Integration](image)

   2. Under **Host**, enter the IP address or FQDN of the vRLI host.
   3. **(Optional) Select the Enable SSL? checkbox to encrypt the logs being sent to vRLI using SSL.**
4. 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.

5. (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.

6. 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.

7. 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 Tanzu Kubernetes Grid Integrated Edition tile does not validate your vRLI configuration settings. To verify your setup, look for log entries in vRLI.

Telegraf

To configure Tanzu Kubernetes Grid Integrated Edition to use Telegraf for metric collection:

1. Create a configuration file for your monitoring service. For instructions, see Create a Configuration File.


3. Under Enable Telegraf Outputs?, select Yes.

4. Configure the Telegraf checkboxes as described in the table below. Components you enable in this step will be visible to TKGI admins only.

<table>
<thead>
<tr>
<th>Enable this checkbox…</th>
<th>…to send these metrics to your monitoring service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable node exporter on TKGI API</td>
<td>Node Exporter metrics from the TKGI API VM</td>
</tr>
<tr>
<td>Enable node exporter on master</td>
<td>Node Exporter metrics from Kubernetes master nodes</td>
</tr>
<tr>
<td>Include etcd metrics</td>
<td>etcd server and debugging metrics</td>
</tr>
<tr>
<td>Enable node exporter on worker</td>
<td>Node Exporter metrics from Kubernetes worker nodes</td>
</tr>
<tr>
<td>Include Kubernetes Controller Manager metrics</td>
<td>Kubernetes controller manager metrics</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>• These metrics provide information about the state of each cluster.</td>
<td></td>
</tr>
<tr>
<td>Include Kubernetes API Server metrics</td>
<td>Kubernetes API server metrics</td>
</tr>
<tr>
<td>Include kubelet metrics</td>
<td>kubelet metrics for all workloads running in all your Kubernetes clusters</td>
</tr>
<tr>
<td>• If you enable Include kubelet metrics, be prepared for a high volume of metrics.</td>
<td></td>
</tr>
</tbody>
</table>

5. In **Set Up Telegraf Outputs**, replace the default value `[[outputs.discard]]` with the contents of the configuration file that you created above. See the following example for an HTTP output plugin:

```yaml
[[outputs.http]]
  url="https://example.com"
  method="POST"
  data_format="json"
[[processors.override]]
  [processors.override.tags]
    director = "bosh-director-1"
```

6. 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 Tanzu Kubernetes Grid Integrated Edition tile, select In-Cluster Monitoring.
2. Under **Wavefront Integration**, select Yes.

3. Under **Wavefront URL**, enter the URL of your Wavefront subscription. For example:

   ```plaintext
   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:

   ```plaintext
   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:

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

6. Click Save.

To create alerts, you must enable errands in Tanzu Kubernetes Grid Integrated Edition.

1. In the Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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


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 TKGI, 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 TKGI-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, Tanzu Kubernetes Grid Integrated Edition 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, Tanzu Kubernetes Grid Integrated Edition 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 TKGI-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, Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition tile to integrate their Tanzu Kubernetes Grid Integrated Edition deployment with Tanzu Mission Control.

Tanzu Mission Control integration lets you monitor and manage Tanzu Kubernetes Grid Integrated Edition 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](https://www.vmware.com/support/tanzu-mission-control/index.html) home page.

To integrate Tanzu Kubernetes Grid Integrated Edition with Tanzu Mission Control:

1. Confirm that the TKGI API VM has internet access and can connect to `cna.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 **Tanzu Kubernetes Grid Integrated Edition** tile > the **Tanzu Mission Control (Experimental)** pane and select **Yes** under **Tanzu Mission Control Integration**.

![Tanzu Mission Control Integration](image)

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 Tanzu Kubernetes Grid Integrated Edition clusters in Tanzu Mission Control.

4. Click **Save**.

**warning**: After the Tanzu Kubernetes Grid Integrated Edition 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 Telemetry Program, do the following:

1. Click **CEIP and Telemetry**.

2. Review the information about the CEIP and Telemetry.
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 Tanzu Kubernetes Grid Integrated Edition installation.
- *(Default) Standard*: If you select this option, data is collected from your Tanzu Kubernetes Grid Integrated Edition installation to improve Tanzu Kubernetes Grid Integrated Edition. 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 Tanzu Kubernetes Grid Integrated Edition 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 account number or 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 started as a Pivotal customer, you can find your Customer Number in your Order Confirmation email.
- (Optional) Enter a descriptive name for your TKGI installation in the **TKGI 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 **TKGI Installation Type** list.

6. Click **Save**.

   - **Note:** If you join the CEIP and Telemetry Program for Tanzu Kubernetes Grid Integrated Edition, open your firewall to allow outgoing access to https://vcsa.vmware.com/ph on port 443.

   - **Note:** Even if you select **None**, Tanzu Kubernetes Grid Integrated Edition-provisioned clusters send usage data to the TKGI control plane. However, this data is not sent to VMware and remains on your Tanzu Kubernetes Grid Integrated Edition installation.

**Errands**

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

To configure which post-deploy and pre-delete errands run for Tanzu Kubernetes Grid Integrated Edition:

1. Make a selection in the dropdown next to each errand.
### Errands

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

**Post-Deploy Errands**

<table>
<thead>
<tr>
<th>Errand</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSX-T Validation errand</td>
<td>Validates NSX-T configuration</td>
</tr>
<tr>
<td>Run smoke tests</td>
<td>Run smoke tests to validate TKGI lifecycle operations</td>
</tr>
<tr>
<td>Upgrade all clusters errand</td>
<td>Upgrades all Kubernetes clusters provisioned by TKGI after the TKGI Tile upgrade is applied</td>
</tr>
<tr>
<td>Create pre-defined Wavefront alerts errand</td>
<td>Create pre-defined Wavefront alerts</td>
</tr>
</tbody>
</table>

**Pre-Delete Errands**

<table>
<thead>
<tr>
<th>Errand</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete all clusters errand</td>
<td>Deletes all clusters provisioned by TKGI when the TKGI Tile is deleted</td>
</tr>
<tr>
<td>Delete pre-defined Wavefront alerts errand</td>
<td>Delete pre-defined Wavefront alerts errand</td>
</tr>
</tbody>
</table>

---

**Note:** We recommend that you use the default settings for all errands except for the NSX-T validation and Run smoke tests errands.

---

2. **(Optional)** Set the **NSX-T validation** errand to **On**.

   This errand verifies the NSX-T objects.

3. **(Optional)** Set the **Run smoke tests** errand to **On**.

   This errand uses the TKGI CLI to create a Kubernetes cluster and then delete it. If the creation or deletion fails, the errand fails and the installation of the Tanzu Kubernetes Grid Integrated Edition tile is aborted.

4. **(Optional)** To ensure that all of your cluster VMs are patched, configure the **Upgrade all clusters errand** errand to **On**.

   Updating the Tanzu Kubernetes Grid Integrated Edition tile with a new Linux stemcell and the Upgrade all clusters errand enabled triggers the rolling of every Linux VM in each Kubernetes cluster. Similarly, updating the Tanzu Kubernetes Grid Integrated Edition tile with a new Windows stemcell triggers the rolling of every Windows VM in your Kubernetes clusters.

**warning:** To avoid workload downtime, use the resource configuration recommended in About Tanzu Kubernetes Grid Integrated Edition Upgrades and Maintaining Workload Uptime.
Resource Config

To modify the resource configuration of Tanzu Kubernetes Grid Integrated Edition, follow the steps below:

1. Select **Resource Config**.

2. For each job, review the **Automatic** values in the following fields:
   - **VM TYPE**: By default, the **TKGI Database** and **TKGI API** jobs are set to the same **Automatic** VM type. If you want to adjust this value, we recommend that you select the same VM type for both jobs.

     ![Note: The Automatic VM TYPE values match the recommended resource configuration for the TKGI API and TKGI Database jobs.]

   - **PERSISTENT DISK TYPE**: By default, the **TKGI Database** and **TKGI API** jobs are set to the same persistent disk type. If you want to adjust this value, you can change the persistent disk type for each of the jobs independently. Using the same persistent disk type for both jobs is not required.

3. Under each job, leave **NSX-T CONFIGURATION** and **NSX-V CONFIGURATION** blank.

Step 3: Apply Changes

After configuring the Tanzu Kubernetes Grid Integrated Edition 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 TKGI and Kubernetes CLIs

The TKGI CLI and the Kubernetes CLI help you interact with your Tanzu Kubernetes Grid Integrated Edition-provisioned Kubernetes clusters and Kubernetes workloads. To install the CLIs, follow the instructions below:

- **Installing the TKGI 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 Tanzu Kubernetes Grid Integrated Edition Management Plane. See [Create Management Plane](#) in *Installing and Configuring NSX-T Data Center v3.0 for TKGI* for details.

In addition, for NAT and no-NAT modes, verify that you created the required NAT rule for Kubernetes master nodes to access NSX Manager. For details, see [Create IP Blocks and Pool for Compute Plane](#) in *Installing and Configuring NSX-T Data Center v3.0 for TKGI*.

If you want your developers to be able to access the TKGI CLI from their external workstations, create a DNAT rule that maps a routable IP address to the TKGI API VM. This must be done after Tanzu Kubernetes Grid Integrated Edition is successfully deployed and it has an IP address. See [Create Management Plane](#) in *Installing and Configuring NSX-T Data Center v3.0 for TKGI* for details.
Step 6: Configure Authentication for Tanzu Kubernetes Grid Integrated Edition


Next Steps

After installing Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T integration, complete the following tasks:

- Integrate VMware Harbor with Tanzu Kubernetes Grid Integrated Edition to store and manage container images. For more information, see Integrating VMware Harbor Registry with Tanzu Kubernetes Grid Integrated Edition.
- Setting Up Tanzu Kubernetes Grid Integrated Edition Admin Users on vSphere
- Creating an Tanzu Kubernetes Grid Integrated Edition Cluster

Please send any feedback you have to pks-feedback@pivotal.io.
Advanced Configurations for Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T Data Center

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic lists the sections to follow when installing VMware Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T Data Center.

Post-Installation NSX-T Configurations

After you have installed Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T, refer to the following sections for additional NSX-T configuration options:

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

Please send any feedback you have to pks-feedback@pivotal.io.
Using Proxies with Tanzu Kubernetes Grid Integrated Edition on NSX-T

In this topic

Overview
Enable TKGI API and Kubernetes Proxy
Enable Ops Manager and BOSH Proxy

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to use proxies with VMware Tanzu Kubernetes Grid Integrated Edition with NSX-T.

Overview

If your environment includes HTTP proxies, you can configure Tanzu Kubernetes Grid Integrated Edition with NSX-T to use these proxies so that Tanzu Kubernetes Grid Integrated Edition-deployed Kubernetes master and worker nodes access public Internet services and other internal services through a proxy.

In addition, Tanzu Kubernetes Grid Integrated Edition proxy settings apply to the TKGI API instance. When an Tanzu Kubernetes Grid Integrated Edition operator creates a Kubernetes cluster, the TKGI API 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 Tanzu Kubernetes Grid Integrated Edition components use the same proxy service.

The following diagram illustrates the network architecture:
Enable TKGI API and Kubernetes Proxy

To configure a global HTTP proxy for all outgoing HTTP/HTTPS traffic from the Kubernetes cluster nodes and the TKGI API server, perform the following steps:

1. Navigate to Ops Manager and log in.
2. Click the Tanzu Kubernetes Grid Integrated Edition 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.

Configure Tanzu Kubernetes Grid Integrated Edition to use your proxy and enable the following:

- TKGI API access to public Internet services and other internal services.
- Tanzu Kubernetes Grid Integrated Edition-deployed Kubernetes nodes access to public Internet services and other internal services.
- Tanzu Kubernetes Grid Integrated Edition Telemetry ability to forward Telemetry data to the CEIP and Telemetry program.

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

5. 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 Tanzu Kubernetes Grid Integrated Edition communication.

The No Proxy list should include 127.0.0.1 and localhost.

Also include the following in the No Proxy list:

- Your Tanzu Kubernetes Grid Integrated Edition environment’s CIDRs, such as the service network CIDR where your Tanzu Kubernetes Grid Integrated Edition cluster is deployed, the deployment network CIDR, the node network IP block CIDR, and the pod network IP block CIDR.

- The FQDN of any registry, such as the Harbor API FQDN, or component communicating with Tanzu Kubernetes Grid Integrated Edition, using a hostname instead of an IP address.

- The IP addresses for your NSX Manager, vCenter Server, and all ESXi hosts, if you are upgrading and have an existing proxy configuration for reaching a Docker registry or other external services.

- 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, and your Tanzu Kubernetes Grid Integrated Edition FQDN are not proxied. This allows internal Tanzu Kubernetes Grid Integrated Edition 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.


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. Log in to Ops Manager.

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, Ops Manager IP, TKGI API VM IP, and the TKGI Database VM IP.

   `127.0.0.1,localhost,BOSH-DIRECTOR-IP,TKGI-API-IP,OPS-MANAGER-IP,TKGI-DATABASE-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

About Multi-T0 Router for Tenant Isolation
Prerequisites
Base Configuration
  Step 1: Plan and Provision Additional NSX Edge Nodes for Each Multi-T0 Router
  Step 2: Configure Inter-T0 Logical Switch
  Step 3: Configure a New Uplink Interface on the Shared Tier-0 Router
  Step 4: Provision Tier-0 Router for Each Tenant
  Step 5: Create Two Uplink Interfaces on Each Tenant Tier-0 Router
  Step 6: Verify the Status of the Shared and Tenant Tier-0 Routers
  Step 7: Configure Static Routes
  Step 8: Considerations for NAT Topology on Shared Tier-0
  Step 9: Considerations for NAT Topology on Tenant Tier-0
  Step 10: Configure BGP on Each Tenant Tier-0 Router
  Step 11: Configure BGP on the Shared Tier-0 Router
  Step 12: Test the Base Configuration
Security Configuration
  Secure Inter-Tenant Communications
  Secure Intra-Tenant Communications

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to create multiple NSX-T Tier-0 (T0) logical routers for use with VMware Tanzu Kubernetes Grid Integrated Edition multi-tenant environments.

About Multi-T0 Router for Tenant Isolation

Tanzu Kubernetes Grid Integrated Edition 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 TKGI 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 TKGI 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:

\[ 2 + (TENANTS \times 2) \]

Where \( TENANTS \) is the number of tenants you want to isolate.

For example, if you want to isolate three tenants, use the following calculation:

\[ 2 + (3 \times 2) = 8 \text{ NSX Edge Nodes} \]

To isolate ten tenants, use the following calculation:

\[ 2 + (10 \times 2) = 22 \text{ NSX Edge Nodes} \]

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 Data Center v3.0 for TKGI.

**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 50.0.0.28.

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 Data Center v3.0 for TKGI.

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 Installing and Configuring NSX-T Data Center v3.0 for TKGI.

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 Tier-0 Router in Installing and Configuring NSX-T Data Center v3.0 for TKGI.

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 Tier-0 Router in Installing and Configuring NSX-T Data Center v3.0 for TKGI. 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.

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.

**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 Tier-0 Router in Installing and Configuring NSX-T Data Center v3.0 for TKGI.

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.
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`:

```
<table>
<thead>
<tr>
<th>Network</th>
<th>ID</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.0/0</td>
<td>0eaa...9e7c</td>
<td>90.0.0.1</td>
</tr>
<tr>
<td>192.168.201.0/24</td>
<td>0a95...b030</td>
<td>90.0.0.1</td>
</tr>
</tbody>
</table>
```

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 Tanzu Kubernetes Grid Integrated Edition](#).

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 Tanzu Kubernetes Grid Integrated Edition 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 TKGI](#).

**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 TKGI management components residing on the Shared Tier-0 router must bypass NAT rules. This is required because TKGI-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 TKGI management components such as the TKGI 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 TKGI Management network where TKGI, 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 TKGI 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.

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 Deploy NSX-T Edge Nodes in Installing and Configuring NSX-T Data Center v3.0 for TKGI.

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`.
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.
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 TKGI management network where TKGI, 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 TKGI management components (TKGI, 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 TKGI 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.

---

### Edit IP Prefix List - shared-prefix-list

<table>
<thead>
<tr>
<th>Network</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.0.0.0/24</td>
<td>Permit</td>
</tr>
<tr>
<td>192.168.201.0/24</td>
<td>Permit</td>
</tr>
<tr>
<td>0.0.0.0/0</td>
<td>Deny</td>
</tr>
</tbody>
</table>

---

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 TKGI 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 TKGI 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 TKGI management components. In the following IP Set example, `30.0.0.0/24` is the CIDR block for the TKGI Management network.
Lastly, define an IP Set for the Inter-T0 CIDR created during the base configuration.

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:
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

---

Note: All firewall rules are applied to the Inter-T0-Uplink interface.
- **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 TKGI 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 TKGI management plane components
- **Service:** Any
- **Action:** Allow
- Apply the rule to the Inter-T0-Uplink interface.
- Save the firewall rule.

### TKGI Firewall Rule

This firewall rule allows TKGI management plane components to talk to Kubernetes nodes.

- **Name:** TKGI-to-Node-Network
- **Direction:** ingress
- **Source:** IP Set defined for the TKGI 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 (DFW) 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.

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, tkgi-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 TKGI. 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 Tanzu Kubernetes Grid Integrated Edition 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-TKGI-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 tkgi/clusters.
      iii. For Scope > Equals, enter tkgi/floating_ip.
      iv. For the second criterion, select Logical switch.
      v. For Scope > Equals, enter ncp/cluster.
After you configure the All-TKGI-Clusters NSGroup, the Membership Criteria tab looks as follows:

---

**Edit NSGroup - All-PKS-Clusters**

<table>
<thead>
<tr>
<th>General</th>
<th>Membership Criteria</th>
<th>Members</th>
</tr>
</thead>
</table>

**Maximum Criteria: 5**

<table>
<thead>
<tr>
<th>Logical Switch</th>
<th>Tag</th>
<th>Equals</th>
<th>Scope</th>
<th>Equals</th>
<th>pks/clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>Tag</td>
<td>Equals</td>
<td>Scope</td>
<td>Equals</td>
<td>pks/floating_ip</td>
</tr>
<tr>
<td>Logical Switch</td>
<td>Tag</td>
<td>Equals</td>
<td>Scope</td>
<td>Equals</td>
<td>ncp/cluster</td>
</tr>
</tbody>
</table>

Note: The `tkgi/clusters`, `tkgi/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-TKGI-Clusters NSGroup, the Membership Criteria tab looks as follows:

---

**All-PKS-Clusters**

<table>
<thead>
<tr>
<th>Overview</th>
<th>Membership Criteria</th>
<th>Members</th>
<th>Applications</th>
<th>Related</th>
</tr>
</thead>
</table>

**Membership Criteria**

1. Logical Switch:  
   - Scope Equals pks/clusters  
   - Scope Equals pks/floating_ip

2. Logical Switch: Scope Equals ncp/cluster

---

**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 `tkgi cluster YOUR-CLUSTER-NAME` command. For more information about the TKGI CLI, see TKGI 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 `tkgi-cluster-YOUR-CLUSTER-UUID`.
   
   iii. For **Scope > Equals**, enter `tkgi/cluster`.
   
   iv. For **Scope > Equals**, enter `tkgi/floating_ip`. For this scope, leave the **Tag** field empty as shown in the image below.

![Edit NSGroup - ClusterA-nodes](image)

After you configure the NSGroup for cluster nodes, the **Membership Criteria** tab looks as follows:

![ClusterA-nodes](image)

**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 `tkgi-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:

![NSGroup - ClusterA-pods](image)

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 `tkgi-cluster-YOUR-CLUSTER-UUID`.
      iii. For **Scope > Equals**, enter `ncp/cluster`.
      iv. For the second criterion, select **Logical Switch**.
      v. For **Tag > Equals**, enter `tkgi-cluster-YOUR-CLUSTER-UUID`.
      vi. For **Scope > Equals**, enter `tkgi/cluster`.

![Edit NSGroup - ClusterA-nodes-pods](image)
After you configure the NSGroup for cluster nodes and pods, the **Membership Criteria** tab looks as follows:

<table>
<thead>
<tr>
<th>Membership Criteria</th>
<th>EDIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Logical Port</td>
<td>Tag Equals pks-cluster-8de000ff-a87a-4930-81ba-106d42c2471e&lt;br&gt;Scope Equals ncp/cluster</td>
</tr>
<tr>
<td>2. Logical Switch</td>
<td>Tag Equals pks-cluster-8de000ff-a87a-4930-81ba-106d42c2471e&lt;br&gt;Scope Equals pks/cluster</td>
</tr>
</tbody>
</table>

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-TKGI-Clusters** NSGroup.

4. For **Destination**, select the **All-TKGI-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 the **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:

<table>
<thead>
<tr>
<th></th>
<th>Name</th>
<th>Source</th>
<th>Destination</th>
<th>Service</th>
<th>Applied To</th>
<th>Log</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>node-to-node</td>
<td>nodes-to-pods</td>
<td>Any</td>
<td></td>
<td></td>
<td>Allow</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>intra-cluster</td>
<td>intra-cluster</td>
<td>Any</td>
<td></td>
<td></td>
<td>Allow</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>inter-cluster</td>
<td>inter-cluster</td>
<td>Any</td>
<td></td>
<td></td>
<td>Allow</td>
</tr>
</tbody>
</table>

Please send any feedback you have to pks-feedback@pivotal.io.
Implementing a Multi-Foundation Tanzu Kubernetes Grid Integrated Edition Deployment

In this topic

About Multi-Foundation Tanzu Kubernetes Grid Integrated Edition

Requirements

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to deploy multiple instances of Tanzu Kubernetes Grid Integrated Edition (TKGI) on vSphere with NSX-T infrastructure.

About Multi-Foundation Tanzu Kubernetes Grid Integrated Edition


As shown in the diagram, with a multi-foundation Tanzu Kubernetes Grid Integrated Edition topology, each TKGI 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, TKGI management components are deployed to a dedicated network, for example, 10.0.0.0/24 for TKGI Foundation A; 20.0.0.0/24 for TKGI Foundation B. When Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition instance.
Requirements

To implement a multi-foundation Tanzu Kubernetes Grid Integrated Edition topology, adhere to the following requirements:

- One Tier-0 router for each Tanzu Kubernetes Grid Integrated Edition 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.

- Tanzu Kubernetes Grid Integrated Edition instances can be deployed in NAT and no-NAT mode. If more than one Tanzu Kubernetes Grid Integrated Edition 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 TKGI instance.

The image below shows three Tanzu Kubernetes Grid Integrated Edition installations across three Tier-0 foundations. Key considerations to keep in mind with a multi-foundation Tanzu Kubernetes Grid Integrated Edition 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.
Installing Tanzu Kubernetes Grid Integrated Edition on Google Cloud Platform (GCP)

This topic lists the procedures to follow to install VMware Tanzu Kubernetes Grid Integrated Edition on Google Cloud Platform (GCP).

Install Tanzu Kubernetes Grid Integrated Edition on GCP

To install Tanzu Kubernetes Grid Integrated Edition on GCP, follow the instructions below:

- Prerequisites and Resource Requirements
- Installing and Configuring Ops Manager on GCP
- Creating Service Accounts in GCP for Tanzu Kubernetes Grid Integrated Edition
- Creating a GCP Load Balancer for the TKGI API
- Installing Tanzu Kubernetes Grid Integrated Edition on GCP
- Setting Up Tanzu Kubernetes Grid Integrated Edition Admin Users on GCP

Install the TKGI and Kubernetes CLIs


To install the CLIs, follow the instructions below:

- Installing the TKGI CLI
- Installing the Kubernetes CLI

Please send any feedback you have to pks-feedback@pivotal.io.
GCP Prerequisites and Resource Requirements

In this topic
Prerequisites
Resource Requirements

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes the prerequisites and resource requirements for installing VMware Tanzu Kubernetes Grid Integrated Edition on Google Cloud Platform (GCP).

Prerequisites

You can install Tanzu Kubernetes Grid Integrated Edition on GCP manually or by using Terraform.

- Prerequisites for Installing Tanzu Kubernetes Grid Integrated Edition Manually
- Prerequisites for Installing Tanzu Kubernetes Grid Integrated Edition Using Terraform

Installing Tanzu Kubernetes Grid Integrated Edition Manually

If you are installing Tanzu Kubernetes Grid Integrated Edition manually, do the following before deploying Tanzu Kubernetes Grid Integrated Edition:

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 Tanzu Kubernetes Grid Integrated Edition

   Note: Perform this step after you install and configure Ops Manager.

4. Create a load balancer to access the TKGI API from outside the network and run tkgi commands from your local workstation. To create a load balancer in GCP, follow the instructions in Creating a GCP Load Balancer for the TKGI API

   Note: Perform this step before you install Tanzu Kubernetes Grid Integrated Edition. After you install Tanzu Kubernetes Grid Integrated Edition, 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 Tanzu Kubernetes Grid Integrated Edition Using Terraform

Copyright © 2020 VMware, Inc. All Rights Reserved. 375 1.8
If you are installing Tanzu Kubernetes Grid Integrated Edition using Terraform, do the following before deploying Tanzu Kubernetes Grid Integrated Edition:

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 Tanzu Kubernetes Grid Integrated Edition requires the following virtual machines (VMs):

<table>
<thead>
<tr>
<th>VM</th>
<th>CPU</th>
<th>Memory (GB)</th>
<th>Ephemeral Disk (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOSH Director</td>
<td>2</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>Ops Manager</td>
<td>1</td>
<td>8</td>
<td>160</td>
</tr>
<tr>
<td>TKGI API</td>
<td>2</td>
<td>8</td>
<td>64</td>
</tr>
<tr>
<td>TKGI Database</td>
<td>2</td>
<td>8</td>
<td>64</td>
</tr>
</tbody>
</table>

### 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 TKGI Database VM as follows:

<table>
<thead>
<tr>
<th>Number of Pods</th>
<th>Persistent Disk Requirements (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000 pods</td>
<td>20</td>
</tr>
<tr>
<td>5,000 pods</td>
<td>100</td>
</tr>
<tr>
<td>10,000 pods</td>
<td>200</td>
</tr>
<tr>
<td>50,000 pods</td>
<td>1,000</td>
</tr>
</tbody>
</table>

### Kubernetes Cluster Resources

Each Kubernetes cluster provisioned through Tanzu Kubernetes Grid Integrated Edition deploys the VMs listed below.

If you deploy more than one Kubernetes cluster, you must scale your allocated resources appropriately.

<table>
<thead>
<tr>
<th>VM</th>
<th>VM Count</th>
<th>CPU Cores</th>
<th>Memory (GB)</th>
<th>Ephemeral Disk (GB)</th>
<th>Persistent Disk (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>master</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>worker</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>32</td>
<td>50</td>
</tr>
</tbody>
</table>

Please send any feedback you have to pks-feedback@pivotal.io.
Installing and Configuring Ops Manager on GCP

In this topic
Prerequisites
Install and Configure Ops Manager
Next Installation Step

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to install and configure Ops Manager before deploying VMware Tanzu Kubernetes Grid Integrated Edition on Google Cloud Platform (GCP).

Prerequisites

You use Ops Manager to install and configure Tanzu Kubernetes Grid Integrated Edition. Before you install Ops Manager, review GCP Prerequisites and Resource Requirements.

Install and Configure Ops Manager

Each version of Tanzu Kubernetes Grid Integrated Edition is compatible with multiple versions of Ops Manager. To determine Ops Manager compatibility, see VMware Tanzu Network.

To install and configure Ops Manager, follow either the manual or Terraform instructions in the table below:

<table>
<thead>
<tr>
<th>Version</th>
<th>Manual Instructions</th>
<th>Terraform Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ops Manager v2.7</td>
<td>1. Preparing to Deploy Ops Manager on GCP Manually</td>
<td>1. Deploying Ops Manager on GCP Using Terraform</td>
</tr>
<tr>
<td></td>
<td>2. Deploying Ops Manager on GCP Manually</td>
<td>2. Configuring BOSH Director on GCP Using Terraform</td>
</tr>
<tr>
<td></td>
<td>3. Configuring BOSH Director on GCP Manually</td>
<td></td>
</tr>
<tr>
<td>Ops Manager v2.8</td>
<td>1. Preparing to Deploy Ops Manager on GCP Manually</td>
<td>1. Deploying Ops Manager on GCP Using Terraform</td>
</tr>
<tr>
<td></td>
<td>2. Deploying BOSH and Ops Manager to GCP Manually</td>
<td>2. Configuring BOSH Director on GCP Using Terraform</td>
</tr>
<tr>
<td></td>
<td>3. Configuring BOSH Director on GCP Manually</td>
<td></td>
</tr>
</tbody>
</table>
Next Installation Step

- If you installed Ops Manager manually, proceed to Creating Service Accounts in GCP for Tanzu Kubernetes Grid Integrated Edition.
- If you installed Ops Manager using Terraform, proceed to Installing Tanzu Kubernetes Grid Integrated Edition on GCP.

Please send any feedback you have to pks-feedback@pivotal.io.
Creating Service Accounts in GCP for Tanzu Kubernetes Grid Integrated Edition

In this topic
Create the Master Node Service Account
Create the Worker Node Service Account
Next Installation Step

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes the steps required to create service accounts for VMware Tanzu Kubernetes Grid Integrated Edition 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. VMware 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 TKGI API](#).

Please send any feedback you have to pks-feedback@pivotal.io.
Creating a GCP Load Balancer for the TKGI API

In this topic
Overview
Create a Load Balancer
Create a Firewall Rule
Create a DNS Entry
Install Tanzu Kubernetes Grid Integrated Edition
Create a Network Tag for the Firewall Rule

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to create a load balancer for the TKGI API using Google Cloud Platform (GCP).

Overview

Before you install VMware Tanzu Kubernetes Grid Integrated Edition, you must configure an external TCP load balancer to access the TKGI 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 Tanzu Kubernetes Grid Integrated Edition installation.

To create a GCP load balancer for the TKGI API, do the following:

1. Create a Load Balancer
2. Create a Firewall Rule
3. Create a DNS Entry
4. Install Tanzu Kubernetes Grid Integrated Edition
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. VMware recommends naming your load balancer `tkgi-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 Tanzu Kubernetes Grid Integrated Edition 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 TKGI API endpoint.
     1. Enter a name for your reserved IP address. For example, `tkgi-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 TKGI API VM using the UAA endpoint on port 8443 and the TKGI 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 `tkgi-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 TKGI 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 **tkgi-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 TKGI 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.tkgi` in this field to use `api.tkgi-example.com` as your TKGI API load balancer 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 Tanzu Kubernetes Grid Integrated Edition

Follow the instructions in Installing Tanzu Kubernetes Grid Integrated Edition on GCP to deploy Tanzu Kubernetes Grid Integrated Edition. After you finish installing Tanzu Kubernetes Grid Integrated Edition, continue to the Create a Network Tag for the Firewall Rule section below to complete the TKGI API load balancer configuration.
Create a Network Tag for the Firewall Rule

To apply the firewall rule to the VM or VMs hosting the TKGi API, the VM must have the `tkgi-api` tag in GCP. Do the following:

1. From the GCP console, navigate to **Compute Engine > VM instances**.

2. Locate your TKGi API VM, or VMs. 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 `tkgi-api` tag. Add the tag if it does not appear in the field.

6. Repeat the preceding steps for your other VMs with the `pivotal-container-service` job label and apply the `tkgi-api` tag to each.

7. Scroll to the bottom of the screen and click **Save**.

Please send any feedback you have to pks-feedback@pivotal.io.
Installing Tanzu Kubernetes Grid Integrated Edition on 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 Tanzu Kubernetes Grid Integrated Edition:

1. Navigate to Ops Manager.
2. Open the **Director Config** pane.
3. Select the **Enable Post Deploy Scripts** checkbox.

---

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to install and configure VMware Tanzu Kubernetes Grid Integrated Edition on Google Cloud Platform (GCP).
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 Tanzu Kubernetes Grid Integrated Edition**

To install Tanzu Kubernetes Grid Integrated Edition, do the following:

1. Download the product file from VMware Tanzu 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 **Tanzu Kubernetes Grid Integrated Edition** in the left column, click the plus sign to add this product to your staging area.

**Step 2: Configure Tanzu Kubernetes Grid Integrated Edition**

Click the orange **Tanzu Kubernetes Grid Integrated Edition** tile to start the configuration process.

**warning:** When you configure the Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition fails.

**Assign AZs and Networks**

To configure the availability zones (AZs) and networks used by the Tanzu Kubernetes Grid Integrated Edition control plane:

1. Click **Assign AZs and Networks**.

2. Under **Place singleton jobs in**, select the AZ where you want to deploy the TKGI API and TKGI Database.

   **Note:** You must specify the **Balance other jobs in** AZ, but the selection has no effect in the current version of Tanzu Kubernetes Grid Integrated Edition.

4. Under **Network**, select the infrastructure subnet that you created for Tanzu Kubernetes Grid Integrated Edition component VMs, such as the TKGI API and TKGI Database VMs.

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

6. Click **Save**.

**TKGI API**

Perform the following steps:

1. Click **TKGI API**.

2. Under **Certificate to secure the TKGI API**, provide a certificate and private key pair.
The certificate that you supply should cover the specific subdomain that routes to the TKGi API VM with TLS termination on the ingress.

<table>
<thead>
<tr>
<th>PKS API Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certificate to secure the PKS API *</td>
</tr>
<tr>
<td>Certificate PEM</td>
</tr>
<tr>
<td>Private Key PEM</td>
</tr>
<tr>
<td>Generate RSA Certificate</td>
</tr>
<tr>
<td>API Hostname (FQDN) *</td>
</tr>
<tr>
<td>pks.api.example.com</td>
</tr>
<tr>
<td>Worker VM Max in Flight *</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

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. Click **Generate RSA Certificate** for a new install or **Change** to update a previously-generated certificate.
- b. Enter the domain for your API hostname. This must match the domain you configured under **TKGI API > API Hostname (FQDN)** in the Tanzu Kubernetes Grid Integrated Edition tile. It can be a standard FQDN or a wildcard domain.
- c. Click **Generate**.

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

---

Copyright © 2020 VMware, Inc. All Rights Reserved.
3. Under **API Hostname (FQDN)**, enter the FQDN that you registered to point to the TKGI API load balancer, such as `api.tkgi.example.com`. To retrieve the public IP address or FQDN of the TKGI 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 TKGI 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. Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition. If you select more than one AZ, Tanzu Kubernetes Grid Integrated Edition deploys the master VM in the first AZ and the worker VMs across the remaining AZs. If you are using multiple masters, Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition can deploy for each cluster. Enter any whole number in this field.

10. Under **Worker Node Instances**, specify the default number of Kubernetes worker nodes the TKGI CLI provisions for each cluster. The Worker Node Instances setting must be less than, or equal to, the Maximum number of workers on a cluster setting.

    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 TKGI 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 Tanzu Kubernetes Grid Integrated Edition 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. Tanzu Kubernetes Grid Integrated Edition 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
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.

![Image](https://via.placeholder.com/150)

**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**. For more information see [Using Admission Control Plugins for Tanzu Kubernetes Grid Integrated Edition Clusters](#).

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.

![Image](https://via.placeholder.com/150)

21. (Optional) Under **Pod Shutdown Grace Period (seconds)**, enter a timeout in seconds for the node to wait before it forces

Copyright © 2020 VMware, Inc. All Rights Reserved.
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 Tanzu Kubernetes Grid Integrated Edition manually**: You configured the master node service account in *Create the Master Node Service Account* in *Creating Service Accounts in GCP for Tanzu Kubernetes Grid Integrated Edition*.
   - **If you are installing Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition manually**: You configured the worker node service account in *Create the Worker Node Service Account* in *Creating Service Accounts in GCP for Tanzu Kubernetes Grid Integrated Edition*.
   - **If you are installing Tanzu Kubernetes Grid Integrated Edition 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**.

![Networking Configurations](image)

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:

1. Click **UAA**.

2. Under **TKGI API Access Token Lifetime**, enter a time in seconds for the TKGI API access token lifetime. This field defaults to 600.
3. Under **TKGI API Refresh Token Lifetime**, enter a time in seconds for the TKGI API refresh token lifetime. This field defaults to 21600.

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

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

   **Note:** VMware recommends using the default UAA token timeout values. By default, access tokens expire after ten minutes and refresh tokens expire after six hours.

6. Under **Configure created clusters to use UAA as the OIDC provider**, select **Enabled** or **Disabled**. This is a global default setting for TKGI-provisioned clusters. For more information, see [OIDC Provider for Kubernetes Clusters](#).

To configure Tanzu Kubernetes Grid Integrated Edition to use UAA as the OIDC provider:

   a. Under **Configure created clusters to use UAA as the OIDC provider**, select **Enabled**.
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`. The default value is `oidc:`.

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, you 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`. The default value is `oidc:`.

⚠️ **warning:** VMware recommends adding OIDC prefixes to prevent users and groups from gaining unintended cluster privileges. If you change the above values for a pre-existing Tanzu Kubernetes Grid Integrated Edition 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, 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 Tanzu Kubernetes Grid Integrated Edition to an LDAP Server.
- To use SAML for UAA, select **SAML Identity Provider** and continue to Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider.

*(Optional) Host Monitoring*

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

- To configure Syslog, 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 TKGI API, master node, and worker node VMs to a monitoring service, such as Wavefront or Datadog.

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

Configure PKS Monitoring Features on Host

- Enable Syslog for PKS?*  
  - No 
  - Yes
- Enable VMware vRealize Log Insight Integration?*  
  - No 
  - Yes
- Enable Telegraf Outputs?*  
  - No 
  - Yes

Save

Syslog

To configure Syslog for all BOSH-deployed VMs in Tanzu Kubernetes Grid Integrated Edition:

1. Click Host Monitoring.

2. Under Enable Syslog for TKGI select Yes.

3. Under Address, enter the destination syslog endpoint.

4. Under Port, enter the destination syslog port.


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. VMware 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,
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 Tanzu Kubernetes Grid Integrated Edition to use Telegraf for metric collection:

1. Create a configuration file for your monitoring service. For instructions, see [Create a Configuration File](#).

2. Return to the Tanzu Kubernetes Grid Integrated Edition tile > **Settings** > **Host Monitoring**.

3. Under **Enable Telegraf Outputs?**, select **Yes**.

4. Configure the Telegraf checkboxes as described in the table below.

   Components you enable in this step will be visible to TKGI admins only.

<table>
<thead>
<tr>
<th>Enable this checkbox…</th>
<th>…to send these metrics to your monitoring service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable node exporter on TKGI API</td>
<td>Node Exporter metrics from the TKGI API VM</td>
</tr>
<tr>
<td>Enable node exporter on master</td>
<td>Node Exporter metrics from Kubernetes master nodes</td>
</tr>
<tr>
<td>Include etcd metrics</td>
<td>etcd server and debugging metrics</td>
</tr>
<tr>
<td>Enable node exporter on worker</td>
<td>Node Exporter metrics from Kubernetes worker nodes</td>
</tr>
</tbody>
</table>
   | Include Kubernetes Controller Manager metrics | Kubernetes controller manager metrics  
   | | • These metrics provide information about the state of each cluster. |
   | Include Kubernetes API Server metrics | Kubernetes API server metrics |
   | Include kubelet metrics | kubelet metrics for all workloads running in all your Kubernetes clusters  
   | | • If you enable include kubelet metrics, be prepared for a high volume of metrics. |

5. In **Set Up Telegraf Outputs**, replace the default value `[[outputs.discard]]` with the contents of the configuration file that you created above. See the following example for an HTTP output plugin:
6. 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.
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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition.

1. In the Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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](https://github.com/cadvisor). For general information about Kubernetes cluster monitoring, see [Tools for Monitoring Resources in the Kubernetes documentation](https://kubernetes.io/docs/tasks/tools/#monitoring).

**Metric Sink Resources**

You can configure TKGI-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](https://kubernetes.io/docs/tasks/tools/#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, Tanzu Kubernetes Grid Integrated Edition 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, Tanzu Kubernetes Grid Integrated Edition 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](https://kubernetes.io/docs/tasks/tools/#creating-and-managing-sink-resources).

3. Click **Save**.

**Log Sink Resources**
You can configure TKGI-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, Tanzu Kubernetes Grid Integrated Edition 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)

Tanzu Kubernetes Grid Integrated Edition does not support Tanzu Mission Control integration on GCP. Skip this configuration pane.

CEIP and Telemetry

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

1. Click **CEIP and Telemetry**.

2. Review the information about the 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 ("TKGI") on a regular basis.

Since TKGI 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 TKGI, you should not join below. You may join or leave CEIP and Telemetry for TKGI 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 TKGI, 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.

**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 TKGI 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
- Trust and Assurance Center
- Pivotal Telemetry
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 Tanzu Kubernetes Grid Integrated Edition installation. *(Default)*
- **Standard**: If you select this option, data is collected from your Tanzu Kubernetes Grid Integrated Edition installation to improve Tanzu Kubernetes Grid Integrated Edition. 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 Tanzu Kubernetes Grid Integrated Edition 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 account number or 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 started as a Pivotal customer, you can find your Customer Number in your Order Confirmation email.
- (Optional) Enter a descriptive name for your TKGI installation in the TKGI 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 TKGI Installation Type list.

6. Click Save.

**Note:** If you join the CEIP and Telemetry Program for Tanzu Kubernetes Grid Integrated Edition, open your firewall to allow outgoing access to [https://vcsa.vmware.com/ph](https://vcsa.vmware.com/ph) on port 443.

**Note:** Even if you select None, Tanzu Kubernetes Grid Integrated Edition-provisioned clusters send usage data to the TKGI control plane. However, this data is not sent to VMware and remains on your Tanzu Kubernetes Grid Integrated Edition installation.
Errands are scripts that run at designated points during an installation.

To configure which post-deploy and pre-delete errands run for Tanzu Kubernetes Grid Integrated Edition:

1. Make a selection in the dropdown next to each errand.

```
Errands

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

Post-Deploy Errands

- NSX-T Validation errand
  - Default (OFF) [ ]
  - Validates NSX-T configuration
- Run smoke tests
  - On [ ]
  - Run smoke tests to validate TKGI lifecycle operations
- Upgrade all clusters errand
  - Default (On) [ ]
  - Upgrades all Kubernetes clusters provisioned by TKGI after the TKGI Tile upgrade is applied
- Create pre-defined Wavefront alerts errand
  - Default (OFF) [ ]
  - Create pre-defined Wavefront alerts

Pre-Delete Errands

- Delete all clusters errand
  - Default (On) [ ]
  - Deletes all clusters provisioned by TKGI when the TKGI tile is deleted
- Delete pre-defined Wavefront alerts errand
  - Default (OFF) [ ]
  - Delete pre-defined Wavefront alerts errand

Save
```

💡 **Note:** We recommend that you use the default settings for all errands except for the **Run smoke tests** errand.

2. (Optional) Set the **Run smoke tests** errand to **On**.

   This errand uses the TKGI CLI to create a Kubernetes cluster and then delete it. If the creation or deletion fails, the errand fails and the installation of the Tanzu Kubernetes Grid Integrated Edition tile is aborted.

3. (Optional) To ensure that all of your cluster VMs are patched, configure the **Upgrade all clusters errand** errand to **On**.

   Updating the Tanzu Kubernetes Grid Integrated Edition tile with a new Linux stemcell and the **Upgrade all clusters errand** enabled triggers the rolling of every Linux VM in each Kubernetes cluster. Similarly, updating the Tanzu Kubernetes Grid Integrated Edition tile with a new Windows stemcell triggers the rolling of every Windows VM in your Kubernetes clusters.

⚠️ **warning:** To avoid workload downtime, use the resource configuration recommended in About Tanzu Kubernetes Grid Integrated Edition Upgrades and Maintaining Workload Uptime.
To modify the resource configuration of Tanzu Kubernetes Grid Integrated Edition and specify your TKGI API load balancer, follow the steps below:

1. Select **Resource Config**.

2. For each job, review the **Automatic** values in the following fields:
   - **VM TYPE**: By default, the TKGI Database and TKGI API jobs are set to the same **Automatic** VM type. If you want to adjust this value, we recommend that you select the same VM type for both jobs.
     
     ![Note] The **Automatic VM TYPE** values match the recommended resource configuration for the TKGI API and TKGI Database jobs.

   - **PERSISTENT DISK TYPE**: By default, the TKGI Database and TKGI API jobs are set to the same persistent disk type. If you want to adjust this value, you can change the persistent disk type for each of the jobs independently. Using the same persistent disk type for both jobs is not required.

3. For the **TKGI Database** job:
   - Leave the **LOAD BALancers** field blank.
   - (Optional) If you do not use a NAT instance, select **INTERNET CONNECTED**. This allows component instances direct access to the internet.

4. For the **TKGI API** job:
   - Enter the name of your TKGI API load balancer in the **LOAD BALancers** field, prefixed with **tcp**. For example, 
     
     ```
     tcp:TKGI-API-LB
     ```
     
     Replace `TKGI-API-LB` with the name of your TKGI API load balancer. You can find the name of your TKGI API load balancer by doing one of the following:
     - **If you are installing Tanzu Kubernetes Grid Integrated Edition manually**: The name of your TKGI API load balancer is the name you configured in the Create a Load Balancer section of Creating a GCP Load Balancer for the TKGI API.
     - **If you are installing Tanzu Kubernetes Grid Integrated Edition using Terraform**: The name of your TKGI 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 TKGI API VM an IP address. BOSH uses the name you provide in the **LOAD BALancers** field to locate your load balancer and then connect the load balancer to the TKGI API VM using its new IP address.

   - (Optional) If you do not use a NAT instance, select **INTERNET CONNECTED**. This allows 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 TKGI API Endpoint

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

To retrieve the TKGI API endpoint, do the following:

1. Navigate to the Ops Manager Installation Dashboard.

2. Click the Tanzu Kubernetes Grid Integrated Edition tile.

3. Click the Status tab and locate the TKGI API job. The IP address of the TKGI API job is the TKGI API endpoint.

Step 5: Configure External Load Balancer

If you are installing Tanzu Kubernetes Grid Integrated Edition manually, follow the procedure in the Create a Network Tag for the Firewall Rule section of Creating a GCP Load Balancer for the TKGI API.

Step 6: Install the TKGI and Kubernetes CLIs

The TKGI CLI and the Kubernetes CLI help you interact with your Tanzu Kubernetes Grid Integrated Edition-provisioned Kubernetes clusters and Kubernetes workloads. To install the CLIs, follow the instructions below:

- Installing the TKGI CLI
- Installing the Kubernetes CLI

Step 7: Configure Authentication for Tanzu Kubernetes Grid Integrated Edition

Follow the procedures in Setting Up Tanzu Kubernetes Grid Integrated Edition Admin Users on GCP.

Next Steps

After installing Tanzu Kubernetes Grid Integrated Edition on GCP, you may want to do one or more of the following:

- Create a load balancer for your Tanzu Kubernetes Grid Integrated Edition clusters. For more information, see Creating and Configuring a GCP Load Balancer for Tanzu Kubernetes Grid Integrated Edition Clusters.
- Create your first Tanzu Kubernetes Grid Integrated Edition cluster. For more information, see Creating Clusters.

Please send any feedback you have to pks-feedback@pivotal.io.
Setting Up Tanzu Kubernetes Grid Integrated Edition Admin Users on GCP

In this topic
- Overview
- Prerequisites
  - Step 1: Connect to the TKGI API VM
  - Step 2: Log In as a UAA Admin
  - Step 3: Assign Tanzu Kubernetes Grid Integrated Edition Cluster Scopes
- Next Step

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Overview

UAA is the identity management service for Tanzu Kubernetes Grid Integrated Edition. Tanzu Kubernetes Grid Integrated Edition includes a UAA server, which is hosted on the TKGI API 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 Tanzu Kubernetes Grid Integrated Edition, you must have one of the following:

- SSH access to the Ops Manager VM
- A machine that can connect to your TKGI API VM

Step 1: Connect to the TKGI API VM

You can connect to the TKGI API 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 TKGI API 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:

   ```bash
   $ 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 TKGI API VM and run UAA commands, do the following:

1. Install UAAC on your machine. For example:

   ```bash
   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 TKGI 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 **Tanzu Kubernetes Grid Integrated Edition** 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:

   ```bash
   uaac target https://TKGI-API:8443 --ca-cert CERTIFICATE-PATH
   ```
Where:

- **TKGI-API** is the domain name of your TKGI API server. You entered this domain name in the **Tanzu Kubernetes Grid Integrated Edition** tile > **TKGI API** > **API Hostname (FQDN)**.
- **CERTIFICATE-PATH** is the path to your Ops Manager root CA certificate. Provide this certificate to validate the TKGI 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:

```bash
$ uaac target api.tkgi.example.com:8443 --ca-cert /var/tempest/workspaces/default/root_ca_certificate
```

**Note:** If you receive a `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:

```bash
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 Tanzu Kubernetes Grid Integrated Edition Cluster Scopes


To create Tanzu Kubernetes Grid Integrated Edition 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 TKGI cluster scopes to an individual user**, see Grant Tanzu Kubernetes Grid Integrated Edition Access to an Individual User. Follow this procedure if you selected **Internal UAA** when you configured **UAA** in the Tanzu Kubernetes Grid Integrated Edition tile. For more information, see Installing Tanzu Kubernetes Grid Integrated Edition on GCP.

- **To assign TKGI cluster scopes to an LDAP group**, see Grant Tanzu Kubernetes Grid Integrated Edition Access to an External LDAP Group. Follow this procedure if you selected **LDAP Server** when you configured **UAA** in the Tanzu Kubernetes Grid Integrated Edition tile. For more information, see Installing Tanzu Kubernetes Grid Integrated Edition Tkgi on GCP.

- **To assign TKGI cluster scopes to a SAML group**, see Grant Tanzu Kubernetes Grid Integrated Edition Access to an External SAML Group. Follow this procedure if you selected **SAML Identity Provider** when you configured **UAA** in the Tanzu Kubernetes Grid Integrated Edition tile. For more information, see Installing Tanzu Kubernetes Grid Integrated Edition Tkgi on GCP.

- **To assign TKGI cluster scopes to a client**, see Grant Tanzu Kubernetes Grid Integrated Edition Access to a Client

### Next Step

After you create admin users in Tanzu Kubernetes Grid Integrated Edition, the admin users can create and manage Kubernetes clusters in Tanzu Kubernetes Grid Integrated Edition. For more information, see Managing Kubernetes Clusters and Workloads.
Please send any feedback you have to pks-feedback@pivotal.io.
Installing Tanzu Kubernetes Grid Integrated Edition on Amazon Web Services (AWS)

This topic lists the procedures to follow to install VMware Tanzu Kubernetes Grid Integrated Edition on Amazon Web Services (AWS).

Install Tanzu Kubernetes Grid Integrated Edition on AWS

To install Tanzu Kubernetes Grid Integrated Edition on AWS, follow the instructions below:

- **AWS Prerequisites and Resource Requirements**
- **Installing and Configuring Ops Manager on AWS**
- **Installing Tanzu Kubernetes Grid Integrated Edition on AWS**
- **Setting Up Tanzu Kubernetes Grid Integrated Edition Admin Users on AWS**

Install the TKGI and Kubernetes CLIs


To install the CLIs, follow the instructions below:

- **Installing the TKGI CLI**
- **Installing the Kubernetes CLI**

Please send any feedback you have to pks-feedback@pivotal.io.
AWS Prerequisites and Resource Requirements

In this topic
Prerequisites
Resource Requirements

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes the prerequisites and resource requirements for installing VMware Tanzu Kubernetes Grid Integrated Edition on Amazon Web Services (AWS).

Prerequisites

Before installing Tanzu Kubernetes Grid Integrated Edition:

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 Tanzu Kubernetes Grid Integrated Edition requires the following virtual machines (VMs):

<table>
<thead>
<tr>
<th>VM</th>
<th>VM Type</th>
<th>Default VM Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOSH Director</td>
<td>m4.large</td>
<td>1</td>
</tr>
<tr>
<td>TKG API</td>
<td>m4.large</td>
<td>1</td>
</tr>
<tr>
<td>TKG Database</td>
<td>m4.large</td>
<td>1</td>
</tr>
</tbody>
</table>

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 TKG Database VM as follows:

<table>
<thead>
<tr>
<th>Number of Pods</th>
<th>Persistent Disk Requirements (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000 pods</td>
<td>20</td>
</tr>
<tr>
<td>5,000 pods</td>
<td>100</td>
</tr>
<tr>
<td>10,000 pods</td>
<td>200</td>
</tr>
<tr>
<td>50,000 pods</td>
<td>1,000</td>
</tr>
</tbody>
</table>
Kubernetes Cluster Resources

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

<table>
<thead>
<tr>
<th>VM</th>
<th>VM Count</th>
<th>CPU Cores</th>
<th>Memory (GB)</th>
<th>Ephemeral Disk (GB)</th>
<th>Persistent Disk (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>master</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>worker</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>32</td>
<td>50</td>
</tr>
</tbody>
</table>

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

You use Ops Manager to install and configure Tanzu Kubernetes Grid Integrated Edition. Before you install Ops Manager, review AWS Prerequisites and Resource Requirements.

Install and Configure Ops Manager

Each version of Tanzu Kubernetes Grid Integrated Edition is compatible with multiple versions of Ops Manager. To determine Ops Manager compatibility, see VMware Tanzu Network.

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

<table>
<thead>
<tr>
<th>Version</th>
<th>Instructions</th>
</tr>
</thead>
</table>
| Ops Manager v2.7  | 1. Deploying Ops Manager on AWS Using Terraform  
|                   | 2. Configuring BOSH Director on AWS Using Terraform                         |
| Ops Manager v2.8  | 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 Tanzu Kubernetes Grid Integrated Edition, follow the instructions in Installing Tanzu Kubernetes Grid Integrated Edition on AWS.
Please send any feedback you have to pks-feedback@pivotal.io.
Installing Tanzu Kubernetes Grid Integrated Edition on AWS

In this topic
Prerequisites
Step 1: Install Tanzu Kubernetes Grid Integrated Edition
Step 2: Configure Tanzu Kubernetes Grid Integrated Edition
   Assign AZs and Networks
   TKGI API
   Plans
   Kubernetes Cloud Provider
   Networking
   UAA
   (Optional) Host Monitoring
   (Optional) In-Cluster Monitoring
   Tanzu Mission Control (Experimental)
   CEIP and Telemetry
   Errands
   Resource Config
Step 3: Apply Changes
Step 4: Retrieve the TKGI API Endpoint
Step 5: Install the TKGI and Kubernetes CLIs
Step 6: Configure Authentication for Tanzu Kubernetes Grid Integrated Edition
Next Steps

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:
This topic describes how to install and configure VMware Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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 VMware Tanzu documentation.

If you use an instance of Ops Manager that you configured previously to install other runtimes, perform the following steps before you install Tanzu Kubernetes Grid Integrated Edition:

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 Tanzu Kubernetes Grid Integrated Edition**

To install Tanzu Kubernetes Grid Integrated Edition, do the following:

1. Download the product file from [VMware Tanzu 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 **Tanzu Kubernetes Grid Integrated Edition** in the left column, click the plus sign to add this product to your staging area.

**Step 2: Configure Tanzu Kubernetes Grid Integrated Edition**

Click the orange **Tanzu Kubernetes Grid Integrated Edition** tile to start the configuration process.

**warning**: When you configure the Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition fails.

**Assign AZs and Networks**

To configure the availability zones (AZs) and networks used by the Tanzu Kubernetes Grid Integrated Edition control plane:

1. Click **Assign AZs and Networks**.
2. Under **Place singleton jobs in**, select the AZ where you want to deploy the TKGI API and TKGI Database.


   ![Balance other jobs in](image)

   **Note:** You must specify the **Balance other jobs in AZ**, but the selection has no effect in the current version of Tanzu Kubernetes Grid Integrated Edition.

4. Under **Network**, select the infrastructure subnet that you created for Tanzu Kubernetes Grid Integrated Edition component VMs, such as the TKGI API and TKGI Database VMs.

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

6. Click **Save**.

**TKGI API**

Perform the following steps:

1. Click **TKGI API**.

2. Under **Certificate to secure the TKGI API**, provide a certificate and private key pair.
The certificate that you supply should cover the specific subdomain that routes to the TKG I API VM with TLS termination on the ingress.

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. Click **Generate RSA Certificate** for a new install or **Change** to update a previously-generated certificate.

b. Enter the domain for your API hostname. This must match the domain you configured under **TKGI > API Hostname (FQDN)** in the Tanzu Kubernetes Grid Integrated Edition tile. It can be a standard FQDN or a wildcard domain.

c. Click **Generate**.

⚠️ **warning:** TLS certificates generated for wildcard DNS records only work for a single domain level. For example, a certificate generated for *.tkgi.EXAMPLE.com does not permit communication to *.api.tkgi.EXAMPLE.com. If the certificate does not contain the correct FQDN for the TKG I API, calls to the API will fail.
3. Under **API Hostname (FQDN)**, enter the FQDN that you registered to point to the TKGI API load balancer, such as `api.tkgi.example.com`. To retrieve the public IP address or FQDN of the TKGI 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 TKGI 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
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 Tanzu Kubernetes Grid Integrated Edition Clusters.

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

7. Under Master/ETCD Availability Zones, select one or more AZs for the Kubernetes clusters deployed by Tanzu Kubernetes Grid Integrated Edition. If you select more than one AZ, Tanzu Kubernetes Grid Integrated Edition deploys the master VM in the first AZ and the worker VMs across the remaining AZs. If you are using multiple masters, Tanzu Kubernetes Grid Integrated Edition deploys the master and worker VMs across the AZs in round-robin fashion.

8. Under Maximum number of workers on a cluster, set the maximum number of Kubernetes worker node VMs that Tanzu Kubernetes Grid Integrated Edition can deploy for each cluster. Enter any whole number in this field.

9. Under Worker Node Instances, specify the default number of Kubernetes worker nodes the TKGI CLI provisions for each cluster. The Worker Node Instances setting must be less than, or equal to, the Maximum number of workers on a cluster setting.

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 TKGI 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 Tanzu Kubernetes Grid Integrated Edition 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. Tanzu Kubernetes Grid Integrated Edition 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
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.


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**.

![Networking Configurations](image)

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 TKGI API server. See Using Proxies with Tanzu Kubernetes Grid Integrated Edition on AWS for instructions 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:

1. Click **UAA**.

2. Under **TKGI API Access Token Lifetime**, enter a time in seconds for the TKGI API access token lifetime. This field defaults to 600.

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

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

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

6. Under **Configure created clusters to use UAA as the OIDC provider**, select **Enabled** or **Disabled**. This is a global default setting for TKGI-provisioned clusters. For more information, see **OIDC Provider for Kubernetes Clusters**.

To configure Tanzu Kubernetes Grid Integrated Edition to use UAA as the OIDC provider:

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`. The default value is `oidc:`.

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, you 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`. The default value is `oidc:`.

⚠️ **warning**: VMware recommends adding OIDC prefixes to prevent users and groups from gaining unintended cluster privileges. If you change the above values for a pre-existing Tanzu Kubernetes Grid Integrated Edition 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, 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 Tanzu Kubernetes Grid Integrated Edition to an LDAP Server.
- To use SAML for UAA, select **SAML Identity Provider** and continue to Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider.

(Optional) Host Monitoring

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

- To configure Syslog, 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 TKGI API, master node, and worker node VMs to a monitoring service, such as Wavefront or Datadog.

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

### Configure PKS Monitoring Features on Host

Enable Syslog for PKS?
- No
- Yes

Enable VMware vRealize Log Insight Integration?
- No
- Yes

Enable Telegraf Outputs?
- No
- Yes

Save

### Syslog

To configure Syslog for all BOSH-deployed VMs in Tanzu Kubernetes Grid Integrated Edition:

1. Click Host Monitoring.
2. Under Enable Syslog for TKGI select Yes.
3. Under Address, enter the destination syslog endpoint.
4. Under Port, enter the destination syslog port.
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. VMware 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,
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 Tanzu Kubernetes Grid Integrated Edition to use Telegraf for metric collection:

1. Create a configuration file for your monitoring service. For instructions, see **Create a Configuration File**.

2. Return to the Tanzu Kubernetes Grid Integrated Edition tile > **Settings** > **Host Monitoring**.

3. Under **Enable Telegraf Outputs?**, select **Yes**.

4. Configure the Telegraf checkboxes as described in the table below. Components you enable in this step will be visible to TKGI admins only.

<table>
<thead>
<tr>
<th>Enable this checkbox…</th>
<th>…to send these metrics to your monitoring service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable node exporter on TKGI API</td>
<td>Node Exporter metrics from the TKGI API VM</td>
</tr>
<tr>
<td>Enable node exporter on master</td>
<td>Node Exporter metrics from Kubernetes master nodes</td>
</tr>
<tr>
<td>Include etcd metrics</td>
<td>etcd server and debugging metrics</td>
</tr>
<tr>
<td>Enable node exporter on worker</td>
<td>Node Exporter metrics from Kubernetes worker nodes</td>
</tr>
<tr>
<td>Include Kubernetes Controller Manager metrics</td>
<td>Kubernetes controller manager metrics</td>
</tr>
<tr>
<td>Include Kubernetes API Server metrics</td>
<td>Kubernetes API server metrics</td>
</tr>
<tr>
<td>Include kubelet metrics</td>
<td>kubelet metrics for all workloads running in all your Kubernetes clusters</td>
</tr>
<tr>
<td></td>
<td>- If you enable <strong>Include kubelet metrics</strong>, be prepared for a high volume of metrics.</td>
</tr>
</tbody>
</table>

5. In **Set Up Telegraf Outputs**, replace the default value `[[outputs.discard]]` with the contents of the configuration file that you created above. See the following example for an HTTP output plugin:
6. 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.
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 Tanzu Kubernetes Grid Integrated Edition 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**.

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.

Copyright © 2020 VMware, Inc. All Rights Reserved.
To create alerts, you must enable errands in Tanzu Kubernetes Grid Integrated Edition.

1. In the Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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](https://github.com/cadvisor/cadvisor) in the cAdvisor GitHub repository. For general information about Kubernetes cluster monitoring, see [Tools for Monitoring Resources](https://kubernetes.io) in the Kubernetes documentation.

**Metric Sink Resources**

You can configure TKGI-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](https://kubernetes.io).

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, Tanzu Kubernetes Grid Integrated Edition deploys Telegraf as a [DaemonSet](https://kubernetes.io), 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](https://kubernetes.io), select **Enable node exporter on workers**. If you enable this checkbox, Tanzu Kubernetes Grid Integrated Edition deploys Node Exporter as a [DaemonSet](https://kubernetes.io), 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](https://kubernetes.io) for Node Exporter metrics, see [Create a ClusterMetricSink Resource for Node Exporter Metrics in Creating and Managing Sink Resources](https://kubernetes.io).

3. Click **Save**.

**Log Sink Resources**

Copyright © 2020 VMware, Inc. All Rights Reserved.
You can configure TKGI-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, Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition tile to integrate their Tanzu Kubernetes Grid Integrated Edition deployment with Tanzu Mission Control.

Tanzu Mission Control integration lets you monitor and manage Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition with Tanzu Mission Control:

1. Confirm that the TKGI API VM has internet access and can connect to `cna.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 **Tanzu Kubernetes Grid Integrated Edition** 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 Tanzu Kubernetes Grid Integrated Edition clusters in Tanzu Mission Control.

4. Click **Save**.
To configure VMware’s Customer Experience Improvement Program (CEIP) and the Telemetry Program, do the following:

1. Click **CEIP and Telemetry**.

2. Review the information about the CEIP and Telemetry.

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 Tanzu Kubernetes Grid Integrated Edition installation.
   - **(Default) Standard**: If you select this option, data is collected from your Tanzu Kubernetes Grid Integrated Edition installation to improve Tanzu Kubernetes Grid Integrated Edition. This participation level is anonymous and does not permit

---

**warning**: After the Tanzu Kubernetes Grid Integrated Edition 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**

**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 ("TKGI") on a regular basis.

Since TKGI 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 TKGI, you should not join below. You may join or leave CEIP and Telemetry for TKGI 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 TKGI, 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.

**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 TKGI 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
- Trust and Assurance Center
- Pivotal Telemetry

---

View a larger version of this image.
the CEIP and Telemetry to identify your organization.

- **Enhanced**: If you select this option, data is collected from your Tanzu Kubernetes Grid Integrated Edition 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 account number or 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 started as a Pivotal customer, you can find your Customer Number in your Order Confirmation email.
- (Optional) Enter a descriptive name for your TKGI installation in the **TKGI 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 **TKGI Installation Type** list.

6. Click **Save**.

**Note**: If you join the CEIP and Telemetry Program for Tanzu Kubernetes Grid Integrated Edition, open your firewall to allow outgoing access to https://vcsa.vmware.com/plus on port 443.

**Note**: Even if you select **None**, Tanzu Kubernetes Grid Integrated Edition-provisioned clusters send usage data to the TKGI control plane. However, this data is not sent to VMware and remains on your Tanzu Kubernetes Grid Integrated Edition installation.

Errands

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

To configure which post-deploy and pre-delete errands run for Tanzu Kubernetes Grid Integrated Edition:

1. Make a selection in the dropdown next to each errand.
2. (Optional) Set the **Run smoke tests** errand to **On**.

This errand uses the TKGI CLI to create a Kubernetes cluster and then delete it. If the creation or deletion fails, the errand fails and the installation of the Tanzu Kubernetes Grid Integrated Edition tile is aborted.

3. (Optional) To ensure that all of your cluster VMs are patched, configure the **Upgrade all clusters errand** errand to **On**.

Updating the Tanzu Kubernetes Grid Integrated Edition tile with a new Linux stemcell and the **Upgrade all clusters errand** enabled triggers the rolling of every Linux VM in each Kubernetes cluster. Similarly, updating the Tanzu Kubernetes Grid Integrated Edition tile with a new Windows stemcell triggers the rolling of every Windows VM in your Kubernetes clusters.

**warning:** To avoid workload downtime, use the resource configuration recommended in About Tanzu Kubernetes Grid Integrated Edition Upgrades and Maintaining Workload Uptime.

### Resource Config

To modify the resource configuration of Tanzu Kubernetes Grid Integrated Edition and specify your TKGI API load balancer, follow the steps below:
1. Select Resource Config.

2. For each job, review the Automatic values in the following fields:
   - **VM TYPE**: By default, the TKGI Database and TKGI API jobs are set to the same Automatic VM type. If you want to adjust this value, we recommend that you select the same VM type for both jobs.
     
     ![Note: The Automatic VM TYPE values match the recommended resource configuration for the TKGI API and TKGI Database jobs.](image)
   
   - **PERSISTENT DISK TYPE**: By default, the TKGI Database and TKGI API jobs are set to the same persistent disk type. If you want to adjust this value, you can change the persistent disk type for each of the jobs independently. Using the same persistent disk type for both jobs is not required.

3. For the **TKGI Database** job:
   - Leave the LOAD BALANCERS field blank.
   - (Optional) If you do not use a NAT instance, select INTERNET CONNECTED. This allows component instances direct access to the internet.

4. For the **TKGI API** job:
   - In the LOAD BALANCERS field, enter all values of pks_api_target_groups from the Terraform output, prefixed with alb:. For example, alb:ENV-pks-tg-9021, alb:ENV-pks-tg-8443. Replace ENV with 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 TKGI API VM an IP address. BOSH uses the name you provide in the LOAD BALANCERS field to locate your load balancer and then connect the load balancer to the TKGI API VM using its new IP address.](image)
   
   - (Optional) If you do not use a NAT instance, select INTERNET CONNECTED. This allows 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 TKGI API Endpoint**

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

To retrieve the TKGI API endpoint, do the following:

1. Navigate to the Ops Manager Installation Dashboard.

2. Click the Tanzu Kubernetes Grid Integrated Edition tile.
3. Click the **Status** tab and locate the **TKGI API** job. The IP address of the TKGI API job is the TKGI API endpoint.

### Step 5: Install the TKGI and Kubernetes CLIs

The TKGI CLI and the Kubernetes CLI help you interact with your Tanzu Kubernetes Grid Integrated Edition-provisioned Kubernetes clusters and Kubernetes workloads. To install the CLIs, follow the instructions below:

- **Installing the TKGI CLI**
- **Installing the Kubernetes CLI**

### Step 6: Configure Authentication for Tanzu Kubernetes Grid Integrated Edition

Follow the procedures in [Setting Up Tanzu Kubernetes Grid Integrated Edition Admin Users on AWS](#).

### Next Steps

After installing Tanzu Kubernetes Grid Integrated Edition on AWS, you might want to do one or more of the following:

- Create a load balancer for your Tanzu Kubernetes Grid Integrated Edition clusters. For more information, see [Creating and Configuring an AWS Load Balancer for Tanzu Kubernetes Grid Integrated Edition Clusters](#).
- Create your first Tanzu Kubernetes Grid Integrated Edition cluster. For more information, see [Creating Clusters](#).

Please send any feedback you have to pks-feedback@pivotal.io.
Using Proxies with Tanzu Kubernetes Grid Integrated Edition on AWS

Overview

If your environment includes HTTP proxies, you can configure Tanzu Kubernetes Grid Integrated Edition on AWS to use these proxies so that Tanzu Kubernetes Grid Integrated Edition-deployed Kubernetes master and worker nodes access public Internet services and other internal services through a proxy.

In addition, Tanzu Kubernetes Grid Integrated Edition proxy settings apply to the TKGI API instance. When an Tanzu Kubernetes Grid Integrated Edition operator creates a Kubernetes cluster, the TKGI API 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 Tanzu Kubernetes Grid Integrated Edition components use the same proxy service.

The following diagram illustrates the network architecture:
Enable TKGI API and Kubernetes Proxy

To configure a global HTTP proxy for all outgoing HTTP/HTTPS traffic from the Kubernetes cluster nodes and the TKGI API server, perform the following steps:

1. Navigate to Ops Manager and log in.

2. Click the **Tanzu Kubernetes Grid Integrated Edition** tile.

3. Click **Networking**.

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.

Configure Tanzu Kubernetes Grid Integrated Edition to use your proxy and enable the following:

- TKGI API access to public Internet services and other internal services.
- Tanzu Kubernetes Grid Integrated Edition-deployed Kubernetes nodes access to public Internet services and other internal services.
- Tanzu Kubernetes Grid Integrated Edition Telemetry ability to forward Telemetry data to the CEIP and Telemetry program.

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

5. 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 Tanzu Kubernetes Grid Integrated Edition communication.

The **No Proxy** list should include 127.0.0.1 and localhost.

Also include the following in the **No Proxy** list:

- Your Tanzu Kubernetes Grid Integrated Edition environment’s CIDRs, such as the service network CIDR where your Tanzu Kubernetes Grid Integrated Edition cluster is deployed, the deployment network CIDR, the node network IP block CIDR, and the pod network IP block CIDR.

- The FQDN of any registry, such as the Harbor API FQDN, or component communicating with Tanzu Kubernetes Grid Integrated Edition, using a hostname instead of an IP address.

- 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 Tanzu Kubernetes Grid Integrated Edition FQDN are not proxied. This allows internal Tanzu Kubernetes Grid Integrated Edition 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.

6. To save your changes to the TKGI tile, click **Save**.


---

**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. Log in to Ops Manager.
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, Ops Manager IP, TKGI API VN IP, and the TKGI Database VM IP.

   - `127.0.0.1`, `localhost`, `BOSH-DIRECTOR-IP`, `TKGI-API-IP`, `OPS-MANAGER-IP`, `TKGI-DATABASE-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 Tanzu Kubernetes Grid Integrated Edition Admin Users on AWS

In this topic

Overview
Prerequisites
Step 1: Connect to the TKGI API VM
Step 2: Log In as a UAA Admin
Step 3: Assign Tanzu Kubernetes Grid Integrated Edition Cluster Scopes
Next Step

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

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

Overview

UAA is the identity management service for Tanzu Kubernetes Grid Integrated Edition. Tanzu Kubernetes Grid Integrated Edition includes a UAA server, which is hosted on the TKGI API 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 Tanzu Kubernetes Grid Integrated Edition, you must have one of the following:

- SSH access to the Ops Manager VM
- A machine that can connect to your TKGI API VM

Step 1: Connect to the TKGI API VM

You can connect to the TKGI API 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 TKGI API 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:
In the AWS console, click the Ops Manager VM.
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:

   ```bash
   $ chmod 600 ops_mgr.pem
   ```

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

   ```bash
   ssh -i ops_mgr.pem ubuntu@OPS-MANAGER-FQDN
   ```

   Where OPS-MANAGER-FQDN is the FQDN of Ops Manager. For example:

   ```bash
   $ 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 TKGI API VM and run UAA commands, do the following:

1. Install UAAC on your machine. For example:

   ```bash
   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 TKGI 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 Tanzu Kubernetes Grid Integrated Edition 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:
Where:

- **TKGI-API** is the domain name of your TKGI API server. You entered this domain name in the **Tanzu Kubernetes Grid Integrated Edition** tile > **TKGI API > API Hostname (FQDN)**.
- **CERTIFICATE-PATH** is the path to your Ops Manager root CA certificate. Provide this certificate to validate the TKGI 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.tkgi.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 Tanzu Kubernetes Grid Integrated Edition Cluster Scopes**


To create Tanzu Kubernetes Grid Integrated Edition 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 TKGI cluster scopes to an individual user, see [Grant Tanzu Kubernetes Grid Integrated Edition Access to an Individual User](#). Follow this procedure if you selected **Internal UAA** when you configured **UAA** in the Tanzu Kubernetes Grid Integrated Edition tile. For more information, see [Installing Tanzu Kubernetes Grid Integrated Edition on AWS](#)
- To assign TKGI cluster scopes to an LDAP group, see [Grant Tanzu Kubernetes Grid Integrated Edition Access to an External LDAP Group](#). Follow this procedure if you selected **LDAP Server** when you configured **UAA** in the Tanzu Kubernetes Grid Integrated Edition tile. For more information, see [Installing Tanzu Kubernetes Grid Integrated Edition TKGI on AWS](#)
- To assign TKGI cluster scopes to a SAML group, see [Grant Tanzu Kubernetes Grid Integrated Edition Access to an External SAML Group](#). Follow this procedure if you selected **SAML Identity Provider** when you configured **UAA** in the Tanzu Kubernetes Grid Integrated Edition tile. For more information, see [Installing Tanzu Kubernetes Grid Integrated Edition TKGI on AWS](#)
- To assign TKGI cluster scopes to a client, see [Grant Tanzu Kubernetes Grid Integrated Edition Access to a Client](#)

**Next Step**
After you create admin users in Tanzu Kubernetes Grid Integrated Edition, the admin users can create and manage Kubernetes clusters in Tanzu Kubernetes Grid Integrated Edition. For more information, see Managing Kubernetes Clusters and Workloads.

Please send any feedback you have to pks-feedback@pivotal.io.
Installing Tanzu Kubernetes Grid Integrated Edition on Microsoft Azure

This topic lists the procedures to follow to install VMware Tanzu Kubernetes Grid Integrated Edition on Microsoft Azure.

Install Tanzu Kubernetes Grid Integrated Edition on Azure

To install Tanzu Kubernetes Grid Integrated Edition on Azure, follow the instructions below:

- Azure Prerequisites and Resource Requirements
- Installing and Configuring Ops Manager on Azure
- Creating Managed Identities in Azure for Tanzu Kubernetes Grid Integrated Edition
- Installing Tanzu Kubernetes Grid Integrated Edition on Azure
- Configuring an Azure Load Balancer for the TKGI API
- Setting Up Tanzu Kubernetes Grid Integrated Edition Admin Users on Azure

Install the TKGI and Kubernetes CLIs


To install the CLIs, follow the instructions below:

- Installing the TKGI CLI
- Installing the Kubernetes CLI

Please send any feedback you have to pks-feedback@pivotal.io.
Azure Prerequisites and Resource Requirements

In this topic
Prerequisites
Subscription Requirements
Resource Requirements

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes the prerequisites and resource requirements for installing VMware Tanzu Kubernetes Grid Integrated Edition on Microsoft Azure.

Prerequisites

Before installing Tanzu Kubernetes Grid Integrated Edition:

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 Tanzu Kubernetes Grid Integrated Edition and Kubernetes services to run correctly, you must have at least a standard subscription tier.

Resource Requirements

Installing Ops Manager and Tanzu Kubernetes Grid Integrated Edition requires the following virtual machines (VMs):

<table>
<thead>
<tr>
<th>VM</th>
<th>CPU</th>
<th>Memory (GB)</th>
<th>Ephemeral Disk (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOSH Director</td>
<td>2</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>Ops Manager</td>
<td>1</td>
<td>8</td>
<td>120</td>
</tr>
<tr>
<td>TKGI API</td>
<td>2</td>
<td>8</td>
<td>64</td>
</tr>
<tr>
<td>TKGI Database</td>
<td>2</td>
<td>8</td>
<td>64</td>
</tr>
</tbody>
</table>

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 TKGI Database VM as follows:
### Number of Pods vs Persistent Disk Requirements (GB)

<table>
<thead>
<tr>
<th>Number of Pods</th>
<th>Persistent Disk Requirements (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000 pods</td>
<td>20</td>
</tr>
<tr>
<td>5,000 pods</td>
<td>100</td>
</tr>
<tr>
<td>10,000 pods</td>
<td>200</td>
</tr>
<tr>
<td>50,000 pods</td>
<td>1,000</td>
</tr>
</tbody>
</table>

**Kubernetes Cluster Resources**

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

<table>
<thead>
<tr>
<th>VM</th>
<th>VM Count</th>
<th>CPU Cores</th>
<th>Memory (GB)</th>
<th>Ephemeral Disk (GB)</th>
<th>Persistent Disk (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>master</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>worker</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>32</td>
<td>50</td>
</tr>
</tbody>
</table>

Please send any feedback you have to pks-feedback@pivotal.io.
Installing and Configuring Ops Manager on Azure

In this topic
Prerequisites
Install and Configure Ops Manager
Next Installation Step

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to install and configure Ops Manager before deploying VMware Tanzu Kubernetes Grid Integrated Edition on Azure.

Prerequisites

You use Ops Manager to install and configure Tanzu Kubernetes Grid Integrated Edition. Before you install Ops Manager, review Azure Prerequisites and Resource Requirements.

Install and Configure Ops Manager

Each version of Tanzu Kubernetes Grid Integrated Edition is compatible with multiple versions of Ops Manager. To determine Ops Manager compatibility, see VMware Tanzu Network.

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

<table>
<thead>
<tr>
<th>Version</th>
<th>Instructions</th>
</tr>
</thead>
</table>
| 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 |
| Ops Manager v2.8 | 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


Please send any feedback you have to pks-feedback@pivotal.io.
Creating Managed Identities in Azure for Tanzu Kubernetes Grid Integrated Edition

In this topic
Retrieve Your Subscription ID and Resource Group
Create the Master Node Managed Identity
Create the Worker Node Managed Identity
Next Installation Step

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to create managed identities for VMware Tanzu Kubernetes Grid Integrated Edition 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. VMware 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 Tanzu Kubernetes Grid Integrated Edition resource group.

You entered your subscription ID into the terra.tfvars file in Step 1: Download and Edit the Terraform Variables File in Deploying Ops Manager on Azure Using Terraform.

The name of your Tanzu Kubernetes Grid Integrated Edition resource group is exported from Terraform as the output pcf_resource_group_name.

To retrieve your subscription ID and the name of your Tanzu Kubernetes Grid Integrated Edition resource group, you must have access to the output from when you ran terraform apply to create resources for the Tanzu Kubernetes Grid Integrated Edition deployment in Create Azure Resources with Terraform in Deploying Ops Manager to Azure Using Terraform in the Ops Manager 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 Tanzu Kubernetes Grid Integrated Edition resource group. For more information about custom roles in Azure, see Custom Roles in Azure in the Azure documentation.

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Copyright © 2020 VMware, Inc. All Rights Reserved.
2. Save your template as `tkgi_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 `tkgi_master_role.json`:

   ```
   az role definition create --role-definition tkgi_master_role.json
   ```

5. Create a managed identity by running the following command:

   ```
   az identity create -g RESOURCE_GROUP -n tkgi-master
   ```

   Where `RESOURCE_GROUP` is the name of your Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition resource group by performing the following steps:

   a. Navigate to the Azure Portal and log in.
   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 TKGI master.
      iii. For Select, select the tkgi-master identity created above.
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 Tanzu Kubernetes Grid Integrated Edition resource group:

   ```json
   {
     "Name": "TKGI worker",
     "IsCustom": true,
     "Description": "Permissions for TKGI worker",
     "Actions": [
       "Microsoft.Compute/virtualMachines/read",
       "Microsoft.Storage/storageAccounts/*"
     ],
     "NotActions": [
     ],
     "DataActions": [
     ],
     "NotDataActions": [
     ],
     "AssignableScopes": [
       "subscriptions/SUBSCRIPTION-ID/resourceGroups/RESOURCE-GROUP"
     ]
   }
   ```

2. Save your template as `tkgi_worker_role.json`.

3. Create the role in Azure by running the following command from the directory with `tkgi_worker_role.json`:

   ```
   az role definition create --role-definition tkgi_worker_role.json
   ```

4. Create a managed identity by running the following command:

   ```
   az identity create -g RESOURCE_GROUP -n tkgi-worker
   ```

   Where `RESOURCE_GROUP` is the name of your Tanzu Kubernetes Grid Integrated Edition resource group.

5. Assign managed identity access to the Tanzu Kubernetes Grid Integrated Edition resource group by performing the following steps:
   a. Navigate to the Azure Portal and log in.
   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:

---

**Note:** The **TKGI 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**.

---

Copyright © 2020 VMware, Inc. All Rights Reserved.
For **Assign access to**, select **User Assigned Managed Identity**.

ii. For **Role**, select **TKGI worker**.

iii. For **Select**, select the **tkgi-worker** identity created above.

**Note:** The **TKGI 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


Please send any feedback you have to pks-feedback@pivotal.io.
Installing Tanzu Kubernetes Grid Integrated Edition on Azure

In this topic
Prerequisites
Step 1: Install Tanzu Kubernetes Grid Integrated Edition
Step 2: Configure Tanzu Kubernetes Grid Integrated Edition
  Assign Networks
  TKGI API
  Plans
  Kubernetes Cloud Provider
  Networking
  UAA
  (Optional) Host Monitoring
  (Optional) In-Cluster Monitoring
  Tanzu Mission Control (Experimental)
  CEIP and Telemetry
  Errands
  Resource Config
Step 3: Apply Changes
Step 4: Retrieve the TKGI API Endpoint
Step 5: Configure an Azure Load Balancer for the TKGI API
Step 6: Install the TKGI and Kubernetes CLIs
Step 7: Configure Authentication for Tanzu Kubernetes Grid Integrated Edition

Next Steps

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to install and configure VMware Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition:

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 Tanzu Kubernetes Grid Integrated Edition**

To install Tanzu Kubernetes Grid Integrated Edition, do the following:

1. Download the product file from [VMware Tanzu Network](https://tanzu.vmware.com).


3. Click **Import a Product** to upload the product file.

4. Under **Tanzu Kubernetes Grid Integrated Edition** in the left column, click the plus sign to add this product to your staging area.

**Step 2: Configure Tanzu Kubernetes Grid Integrated Edition**

Click the orange **Tanzu Kubernetes Grid Integrated Edition** tile to start the configuration process.

![Tanzu Kubernetes Grid Integrated Edition](image)

**warning**: When you configure the Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition fails.

**Assign Networks**

To configure the networks used by the Tanzu Kubernetes Grid Integrated Edition control plane:

1. Click **Assign Networks**.
2. Under **Network**, select the infrastructure subnet that you created for Tanzu Kubernetes Grid Integrated Edition component VMs, such as the TKGI API and TKGI Database VMs. For example, `infrastructure`.

3. Under **Service Network**, select the services subnet that you created for Kubernetes cluster VMs. For example, `services`.

4. Click **Save**.

**TKGI API**

Perform the following steps:

1. Click **TKGI API**.

2. Under **Certificate to secure the TKGI API**, provide a certificate and private key pair.
The certificate that you supply should cover the specific subdomain that routes to the TKGI 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 *.tkgi.EXAMPLE.com does not permit communication to *.api.tkgi.EXAMPLE.com. If the certificate does not contain the correct FQDN for the TKGI 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. Click **Generate RSA Certificate** for a new install or **Change** to update a previously-generated certificate.
b. Enter the domain for your API hostname. This must match the domain you configured under **TKGI API > API Hostname (FQDN)** in the Tanzu Kubernetes Grid Integrated Edition tile. It can be a standard FQDN or a wildcard domain.
c. Click **Generate**.
3. **Under API Hostname (FQDN)**, enter the FQDN that you registered to point to the TKGI API load balancer, such as `api.tkgi.example.com`. To retrieve the public IP address or FQDN of the TKGI 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 TKGI 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...
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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition. If you select more than one AZ, Tanzu Kubernetes Grid Integrated Edition deploys the master VM in the first AZ and the worker VMs across the remaining AZs. If you are using multiple masters, Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition can deploy for each cluster. Enter any whole number in this field.

10. Under Worker Node Instances, specify the default number of Kubernetes worker nodes the TKGI CLI provisions for each cluster. The Worker Node Instances setting must be less than, or equal to, the Maximum number of workers on a cluster setting.

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 TKGI 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 Tanzu Kubernetes Grid Integrated Edition 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. Tanzu Kubernetes Grid Integrated Edition 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.

![Checkbox for Allow Privileged]

**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.


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.

![Input field for Node Drain Timeout(mins) with 0 as default value]

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.
Kubernetes Cloud Provider Configuration

Kubernetes has the concept of a Cloud Provider, which is a module which provides an interface for managing TCP Load Balancers, Worker Nodes (Instances) and Networking Routes. Provide Kubernetes information about your cloud here.

Choose your IaaS*

- GCP
- vSphere
- AWS
- Azure

Azure Cloud Name*

[Input field]

Subscription ID *

[Input field]

Tenant ID *

[Input field]

Location *

[Input field]

Resource Group *

[Input field]

Virtual Network *

[Input field]

Virtual Network Resource Group *

[Input field]

Default Security Group *

[Input field]

Primary Availability Set *

[Input field]

Master Managed Identity *

[Input field]

Worker Managed Identity *

[Input field]

[ Checkbox ] Disable Outbound SNAT

[Submit button] 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:** Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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 VMware Tanzu documentation. You can also find the name of the availability set by logging in to the Azure console.


14. Select **Disable Outbound SNAT** to disable the default outbound SNAT rule for Azure.

15. 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:

1. Click **UAA**.

2. Under **TKGI API Access Token Lifetime**, enter a time in seconds for the TKGI API access token lifetime. This field defaults to 600.
3. Under **TKGI API Refresh Token Lifetime**, enter a time in seconds for the TKGI API refresh token lifetime. This field defaults to **21600**.

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

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

Note: VMware recommends using the default UAA token timeout values. By default, access tokens expire after ten minutes and refresh tokens expire after six hours.

6. Under **Configure created clusters to use UAA as the OIDC provider**, select **Enabled** or **Disabled**. This is a global default setting for TKGI-provisioned clusters. For more information, see **OIDC Provider for Kubernetes Clusters**.

To configure Tanzu Kubernetes Grid Integrated Edition to use UAA as the OIDC provider:

a. Under **Configure created clusters to use UAA as the OIDC provider**, select **Enabled**.
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`. The default value is `oidc:`.

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, you 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`. The default value is `oidc:`.

**warning:** VMware recommends adding OIDC prefixes to prevent users and groups from gaining unintended cluster privileges. If you change the above values for a pre-existing Tanzu Kubernetes Grid Integrated Edition 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, 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 Tanzu Kubernetes Grid Integrated Edition to an LDAP Server.
- To use SAML for UAA, select **SAML Identity Provider** and continue to Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider.

(Optional) Host Monitoring

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

- To configure Syslog, 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 TKGI API, master node, and worker node VMs to a monitoring service, such as Wavefront or Datadog.

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

### Configure PKS Monitoring Features on Host

<table>
<thead>
<tr>
<th>Feature</th>
<th>Option 1</th>
<th>Option 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Syslog for PKS?*</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Enable VMware vRealize Log Insight Integration?*</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Enable Telegraf Outputs?*</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Syslog**

To configure Syslog for all BOSH-deployed VMs in Tanzu Kubernetes Grid Integrated Edition:

1. Click **Host Monitoring**.

2. Under **Enable Syslog for TKGI**, select **Yes**.

3. Under **Address**, enter the destination syslog endpoint.

4. Under **Port**, enter the destination syslog port.


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. VMware 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,
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 Tanzu Kubernetes Grid Integrated Edition to use Telegraf for metric collection:

1. Create a configuration file for your monitoring service. For instructions, see Create a Configuration File.

2. Return to the Tanzu Kubernetes Grid Integrated Edition tile > **Settings** > **Host Monitoring**.

3. Under **Enable Telegraf Outputs?**, select **Yes**.

4. Configure the Telegraf checkboxes as described in the table below. Components you enable in this step will be visible to TKGI admins only.

<table>
<thead>
<tr>
<th>Enable this checkbox…</th>
<th>…to send these metrics to your monitoring service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable node exporter on TKGI API</td>
<td>Node Exporter metrics from the TKGI API VM</td>
</tr>
<tr>
<td>Enable node exporter on master</td>
<td>Node Exporter metrics from Kubernetes master nodes</td>
</tr>
<tr>
<td>Include etcd metrics</td>
<td>etcd server and debugging metrics</td>
</tr>
<tr>
<td>Enable node exporter on worker</td>
<td>Node Exporter metrics from Kubernetes worker nodes</td>
</tr>
<tr>
<td>Include Kubernetes Controller Manager metrics</td>
<td>Kubernetes controller manager metrics</td>
</tr>
<tr>
<td></td>
<td>• These metrics provide information about the state of each cluster.</td>
</tr>
<tr>
<td>Include Kubernetes API Server metrics</td>
<td>Kubernetes API server metrics</td>
</tr>
<tr>
<td>Include kubelet metrics</td>
<td>kubelet metrics for all workloads running in all your Kubernetes clusters</td>
</tr>
<tr>
<td></td>
<td>• If you enable <strong>Include kubelet metrics</strong>, be prepared for a high volume of metrics.</td>
</tr>
</tbody>
</table>

5. In **Set Up Telegraf Outputs**, replace the default value ```[outputs.discard]``` with the contents of the configuration file that you created above. See the following example for an HTTP output plugin:
6. 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.
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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition.

1. In the Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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](https://github.com/cadvisor) in the cAdvisor GitHub repository. For general information about Kubernetes cluster monitoring, see [Tools for Monitoring Resources](https://kubernetes.io/docs/tasks/administer-cluster/monitor-cluster-resources/) in the Kubernetes documentation.

---

### Metric Sink Resources

You can configure TKGI-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](https://kubernetes.io/docs/tasks/administer-cluster/monitor-cluster-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, Tanzu Kubernetes Grid Integrated Edition 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, Tanzu Kubernetes Grid Integrated Edition 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](https://kubernetes.io/docs/tasks/administer-cluster/monitor-cluster-resources/) in Creating and Managing Sink Resources.

3. Click **Save**.

---

### Log Sink Resources

---

Copyright © 2020 VMware, Inc. All Rights Reserved.
You can configure TKGI-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, Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition tile to integrate their Tanzu Kubernetes Grid Integrated Edition deployment with Tanzu Mission Control.

Tanzu Mission Control integration lets you monitor and manage Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition with Tanzu Mission Control:

1. Confirm that the TKGI API VM has internet access and can connect to cna.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.

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 Tanzu Kubernetes Grid Integrated Edition clusters in Tanzu Mission Control.

4. Click Save.
CEIP and Telemetry

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

1. Click **CEIP and Telemetry**.

2. Review the information about the CEIP and Telemetry.

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 Tanzu Kubernetes Grid Integrated Edition installation.
   - (Default) **Standard**: If you select this option, data is collected from your Tanzu Kubernetes Grid Integrated Edition installation to improve Tanzu Kubernetes Grid Integrated Edition. 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 Tanzu Kubernetes Grid Integrated Edition 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 account number or 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 started as a Pivotal customer, you can find your Customer Number in your Order Confirmation email.
- (Optional) Enter a descriptive name for your TKGI installation in the TKGI 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 TKGI Installation Typelist.

6. Click **Save**.

**Note**: If you join the CEIP and Telemetry Program for Tanzu Kubernetes Grid Integrated Edition, open your firewall to allow outgoing access to https://vcsa.vmware.com/plain on port 443.

**Note**: Even if you select None, Tanzu Kubernetes Grid Integrated Edition-provisioned clusters send usage data to the TKGI control plane. However, this data is not sent to VMware and remains on your Tanzu Kubernetes Grid Integrated Edition installation.

Errands

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

To configure which post-deploy and pre-delete errands run for Tanzu Kubernetes Grid Integrated Edition:

1. Make a selection in the dropdown next to each errand.
2. (Optional) Set the Run smoke tests errand to On.

This errand uses the TKGI CLI to create a Kubernetes cluster and then delete it. If the creation or deletion fails, the errand fails and the installation of the Tanzu Kubernetes Grid Integrated Edition tile is aborted.

3. (Optional) To ensure that all of your cluster VMs are patched, configure the Upgrade all clusters errand errand to On.

Updating the Tanzu Kubernetes Grid Integrated Edition tile with a new Linux stemcell and the Upgrade all clusters errand enabled triggers the rolling of every Linux VM in each Kubernetes cluster. Similarly, updating the Tanzu Kubernetes Grid Integrated Edition tile with a new Windows stemcell triggers the rolling of every Windows VM in your Kubernetes clusters.

⚠️ warning: To avoid workload downtime, use the resource configuration recommended in About Tanzu Kubernetes Grid Integrated Edition Upgrades and Maintaining Workload Uptime.

Resource Config

To modify the resource configuration of Tanzu Kubernetes Grid Integrated Edition and specify your TKGI API load balancer, follow the steps below:
1. Select Resource Config.

2. For each job, review the Automatic values in the following fields:

   - **VM TYPE:** By default, the TKGI Database and TKGI API jobs are set to the same Automatic VM type. If you want to adjust this value, we recommend that you select the same VM type for both jobs.
     
     Note: The Automatic VM TYPE values match the recommended resource configuration for the TKGI API and TKGI Database jobs.

   - **PERSISTENT DISK TYPE:** By default, the TKGI Database and TKGI API jobs are set to the same persistent disk type. If you want to adjust this value, you can change the persistent disk type for each of the jobs independently. Using the same persistent disk type for both jobs is not required.

3. For the TKGI Database job:

   - Leave the LOAD BALANCERS field blank.
   - (Optional) If you do not use a NAT instance, select INTERNET CONNECTED. This allows component instances direct access to the internet.

4. For the TKGI API job:

   - Enter the name of your TKGI API load balancer in the LOAD BALANCERS field. The name of your TKGI API load balancer is `YOUR-ENVIRONMENT-NAME-pks-lb`. Replace `YOUR-ENVIRONMENT-NAME` with the environment name that you configured during Step 1: Download Templates and Edit Variables File in Deploying Ops Manager on Azure Using Terraform. If needed, you can find your environment name in your `terraform.tfstate` file.
     
     Note: After you click Apply Changes for the first time, BOSH assigns the TKGI API VM an IP address. BOSH uses the name you provide in the LOAD BALANCERS field to locate your load balancer and then connect the load balancer to the TKGI API VM using its new IP address.

   - (Optional) If you do not use a NAT instance, select INTERNET CONNECTED. This allows 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 TKGI API Endpoint**

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

To retrieve the TKGI API endpoint, do the following:

1. Navigate to the Ops Manager Installation Dashboard.
2. Click the **Tanzu Kubernetes Grid Integrated Edition** tile.

3. Click the **Status** tab and locate the **TKGI API** job. The IP address of the TKGI API job is the TKGI API endpoint.

**Step 5: Configure an Azure Load Balancer for the TKGI API**

Follow the procedures in Configuring an Azure Load Balancer for the TKGI API to configure an Azure load balancer for the TKGI API.

**Step 6: Install the TKGI and Kubernetes CLIs**

The TKGI CLI and the Kubernetes CLI help you interact with your Tanzu Kubernetes Grid Integrated Edition-provisioned Kubernetes clusters and Kubernetes workloads. To install the CLIs, follow the instructions below:

- Installing the TKGI CLI
- Installing the Kubernetes CLI

**Step 7: Configure Authentication for Tanzu Kubernetes Grid Integrated Edition**


**Next Steps**

After installing Tanzu Kubernetes Grid Integrated Edition on Azure, you may want to do one or more of the following:

- Create a load balancer for your Tanzu Kubernetes Grid Integrated Edition clusters. For more information, see Creating and Configuring an Azure Load Balancer for Tanzu Kubernetes Grid Integrated Edition Clusters.
- Create your first Tanzu Kubernetes Grid Integrated Edition 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 TKGI API

In this topic
Overview
Identify Your TKGI API VM
Configure a Load Balancer Backend Pool
Create Health Probe
Create Load Balancing Rule
Create Inbound Security Rule
Verify Hostname Resolution
Next Step

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to create a load balancer for the VMware Tanzu Kubernetes Grid Integrated Edition 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.

Overview

To configure your TKGI API Load Balancer on Azure, complete the following:

- Identify Your TKGI API VM
- Configure a Load Balancer Backend Pool
- Create Health Probe
- Create Load Balancing Rule
- Create Inbound Security Rule
- Verify Hostname Resolution


Identify Your TKGI API VM

Before configuring your Azure Backend Pool you must know which of your VMs is the TKGI API VM.

To find the name of your TKGI API VM, complete either of the following procedures:

- Use the Azure Dashboard:
1. Open the Azure Dashboard.
2. In the Azure Dashboard, locate the VM tagged with `instance_group:pivotal-container-service`. This is your TKGI API VM.
3. Note the machine name and IP address for the listed TKGI API VM.

**Use BOSH:**

1. On the command line, run `bosh vms`.
2. Locate the VM tagged with `instance_group:pivotal-container-service`. This is your TKGI API VM.
3. Note the machine name and IP address for each listed TKGI API VM.

### Configure a Load Balancer Backend Pool

An Azure backend pool is a logical grouping of instances that receive similar traffic. On Azure, you must configure a load balancer backend pool to route your TKGI API traffic to your TKGI API VM.

**Tip:** You must reconfigure your TKGI API load balancer backend pool whenever you modify your TKGI API VM.

1. To open the backend pool configuration page for your TKGI API load balancer, do the following:
   
   a. From the Azure Dashboard, select *All services* from the left-hand menu.
   b. Select *All Resources* to open the *Load Balancers* service.
   c. In the *Settings* menu, select *Backend Pools*.
   d. On the *Backend Pools* page, select the backend pool for your TKGI API load balancer.

2. In the *Virtual machines* section, complete the following for the VM you identified while performing the steps in *Identify Your TKGI API VM*, above:

   a. **Virtual machine**: Select the VM ID for your TKGI API VM.
   b. **IP address**: Select the IP address corresponding to the VM specified in the *Virtual machine* column.
3. Click **OK**.

For information about Azure backend pools, see Backend pools in the Azure documentation. For more information about configuring your backend pool, see Remove or add VMs from the backend pool in the Azure documentation.

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 the TKGI API is deployed. If you deployed Tanzu Kubernetes Grid Integrated Edition 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 **Tanzu Kubernetes Grid Integrated Edition** tile.

3. Select **TKGI 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 TKGI API, complete the Tanzu Kubernetes Grid Integrated Edition installation by returning to the Install the TKGI and Kubernetes CLIs step of Installing Tanzu Kubernetes Grid Integrated Edition on Azure.

Please send any feedback you have to pks-feedback@pivotal.io.
Setting Up Tanzu Kubernetes Grid Integrated Edition Admin Users on Azure

In this topic
Overview
Prerequisites
Step 1: Connect to the TKGI API VM
Step 2: Log In as a UAA Admin
Step 3: Assign Tanzu Kubernetes Grid Integrated Edition Cluster Scopes

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Overview

UAA is the identity management service for Tanzu Kubernetes Grid Integrated Edition. Tanzu Kubernetes Grid Integrated Edition includes a UAA server, which is hosted on the TKGI API 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 Tanzu Kubernetes Grid Integrated Edition, you must have one of the following:

- SSH access to the Ops Manager VM
- A machine that can connect to your TKGI API VM

Step 1: Connect to the TKGI API VM

You can connect to the TKGI API 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 TKGI API 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:

   ```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:

   ```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:
   ```command
   $ 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 TKGI API VM and run UAA commands, do the following:

1. Install UAAC on your machine. For example:

   ```command
   gem install cf-uacc
   ```

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 TKGI 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 Tanzu Kubernetes Grid Integrated Edition 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://TKGI-API:8443 --ca-cert CERTIFICATE-PATH
```

Where:
- **TKGI-API** is the domain name of your TKGI API server. You entered this domain name in the **Tanzu Kubernetes Grid Integrated Edition** tile > **TKGI API** > **API Hostname (FQDN)**.
- **CERTIFICATE-PATH** is the path to your Ops Manager root CA certificate. Provide this certificate to validate the TKGI 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 tkgi.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 Tanzu Kubernetes Grid Integrated Edition Cluster Scopes


To create Tanzu Kubernetes Grid Integrated Edition 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 TKGI cluster scopes to an individual user, see [Grant Tanzu Kubernetes Grid Integrated Edition Access to an Individual User](#). Follow this procedure if you selected **Internal UAA** when you configured **UAA** in the Tanzu Kubernetes Grid Integrated Edition tile. For more information, see [Installing Tanzu Kubernetes Grid Integrated Edition on Azure](#).
- To assign TKGI cluster scopes to an LDAP group, see [Grant Tanzu Kubernetes Grid Integrated Edition Access to an External LDAP Group](#). Follow this procedure if you selected **LDAP Server** when you configured **UAA** in the Tanzu Kubernetes Grid Integrated Edition tile. For more information, see [Installing Tanzu Kubernetes Grid Integrated Edition TKGI on Azure](#).
- To assign TKGI cluster scopes to a SAML group, see [Grant Tanzu Kubernetes Grid Integrated Edition Access to an External SAML Group](#). Follow this procedure if you selected **SAML Identity Provider** when you configured **UAA** in the Tanzu Kubernetes Grid Integrated Edition tile. For more information, see [Installing Tanzu Kubernetes Grid Integrated Edition TKGI on Azure](#).
To assign TKG! cluster scopes to a client, see Grant Tanzu Kubernetes Grid Integrated Edition Access to a Client.

**Next Step**

After you create admin users in Tanzu Kubernetes Grid Integrated Edition, the admin users can create and manage Kubernetes clusters in Tanzu Kubernetes Grid Integrated Edition. For more information, see Managing Kubernetes Clusters and Workloads.

Please send any feedback you have to pks-feedback@pivotal.io.
Installing the TKGI CLI

In this topic
Install the TKGI CLI
Mac OS X
Linux
Windows

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to install the VMware Tanzu Kubernetes Grid Integrated Edition Command Line Interface (TKGI CLI).

For information on the TKGI CLI, see TKGI CLI.

Install the TKGI CLI

To install the TKGI CLI, follow the procedures for your operating system to download the TKGI CLI from VMware Tanzu Network. Binaries are only provided for 64-bit architectures.

Mac OS X

1. Navigate to VMware Tanzu Network and log in.
3. Select your desired release version from the Releases dropdown.
4. Click TKGI CLI.
5. Click TKGI CLI - Mac to download the Mac OS X binary.
6. Rename the downloaded binary file to tkgi.
7. On the command line, run the following command to make the TKGI CLI binary executable:
   ```sh
cshmod +x tkgi
   ```
8. Move the binary file into your PATH.
9. Run tkgi --version to verify the version of your TKGI CLI installed locally.

Linux

1. Navigate to VMware Tanzu Network and log in.
2. Click **Tanzu Kubernetes Grid Integrated Edition**.

3. Select your desired release version from the **Releases** dropdown.

4. Click **TKGI CLI**.

5. Click **TKGI CLI - Linux** to download the Linux binary.

6. Rename the downloaded binary file to `.tkgi`.

7. On the command line, run the following command to make the TKGI CLI binary executable:

   ```
   $ chmod +x tkgi
   ```

8. Move the binary file into your `PATH`.

9. Run `tkgi --version` to verify the version of your TKGI CLI installed locally.

---

**Windows**

1. Navigate to [VMware Tanzu Network](https://tanzu.vmware.com) and log in.

2. Click **Tanzu Kubernetes Grid Integrated Edition**.

3. Select your desired release version from the **Releases** dropdown.

4. Click **TKGI CLI**.

5. Click **TKGI CLI - Windows** to download the Windows executable file.

6. Rename the downloaded binary file to `.tkgi.exe`.

7. Move the binary file into your `PATH`.

8. Run `tkgi --version` to verify the version of your TKGI CLI installed locally.

---

Please send any feedback you have to `pks-feedback@pivotal.io`.
Installing the Kubernetes CLI

In this topic

Mac OS X
Linux
Windows

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

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 VMware Tanzu Network. Binaries are only provided for 64-bit architectures.

Mac OS X

1. Navigate to VMware Tanzu Network and log in.
2. Click VMware Tanzu Kubernetes Grid Integrated Edition
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 VMware Tanzu Network and log in.
2. Click VMware Tanzu Kubernetes Grid Integrated Edition
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 VMware Tanzu Network and log in.


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

Overview
Prerequisites
Configure a Windows Worker-Based Kubernetes Cluster
Plan
Networking
Upload the Windows Server Stemcell
Create a Windows Worker-Based Cluster
Prepare a Windows Pause Image for an Air-Gapped Environment

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes configuring Windows worker-based Kubernetes clusters in VMware Tanzu Kubernetes Grid Integrated Edition.

Overview

In Tanzu Kubernetes Grid Integrated Edition 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.
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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition:
- Your vSphere environment meets the vSphere Prerequisites and Resource Requirements.
- Tanzu Kubernetes Grid Integrated Edition 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.

- Tanzu Kubernetes Grid Integrated Edition has been configured as described in Installing Tanzu Kubernetes Grid Integrated Edition on vSphere.
- You must have a vSphere stemcell for Windows Server version 2019. For vSphere stemcell version requirements, see Product Snapshot in Release Notes.

**Note:** Windows stemcells for vSphere are not available on VMware Tanzu 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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 TKGI CLI.

5. Select **Enable HA Linux workers** to enable high availability Linux worker clusters. A high availability Linux worker cluster consists of three Linux worker nodes.
• Windows workers are mediated by one or three Linux workers.
• For an illustration of how Linux workers connect Windows workers to their master node, see Windows Worker-Based Kubernetes Cluster High Availability.

6. 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. Tanzu Kubernetes Grid Integrated Edition does not support changing the number of master/etcd nodes for plans with existing clusters.

7. 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 Tanzu Kubernetes Grid Integrated Edition Clusters](#).

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

9. Under **Master/ETCD Availability Zones**, select one or more AZs for the Kubernetes clusters deployed by Tanzu Kubernetes Grid Integrated Edition. If you select more than one AZ, Tanzu Kubernetes Grid Integrated Edition deploys the master VM in the first AZ and the worker VMs across the remaining AZs. If you are using multiple masters, Tanzu Kubernetes Grid Integrated Edition deploys the master and worker VMs across the AZs in round-robin fashion.

10. Under **Maximum number of workers on a cluster**, set the maximum number of Kubernetes worker node VMs that Tanzu Kubernetes Grid Integrated Edition can deploy for each cluster. Enter any whole number in this field.
11. Under **Worker Node Instances**, specify the default number of Kubernetes worker nodes the TKGI CLI provisions for each cluster. The **Worker Node Instances** setting must be less than, or equal to, the **Maximum number of workers on a cluster** setting.

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.

12. 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 Tanzu Kubernetes Grid Integrated Edition 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.

13. Under **Worker Availability Zones**, select one or more AZs for the Kubernetes worker nodes. Tanzu Kubernetes Grid Integrated Edition 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
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.


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.

17. 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.

18. (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
19. (Optional) Enable or disable one or more admission controller plugins: PodSecurityPolicy, and SecurityContextDeny. For more information see Using Admission Control Plugins for Tanzu Kubernetes Grid Integrated Edition Clusters Windows in Kubernetes does not support the DenyEscalatingExec Admission Plugin feature. See API in the Kubernetes documentation for additional information.

20. 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.

Configure Tanzu Kubernetes Grid Integrated Edition to use your proxy and enable the following:

- TKGI API access to public Internet services and other internal services.
- Tanzu Kubernetes Grid Integrated Edition-deployed Kubernetes nodes access to public Internet services and other internal services.
- Tanzu Kubernetes Grid Integrated Edition Telemetry ability to forward Telemetry data to the CEIP and Telemetry program.

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

4. 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 Tanzu Kubernetes Grid Integrated Edition communication.

The No Proxy list should include 127.0.0.1 and localhost.

Also include the following in the No Proxy list:

- Your Tanzu Kubernetes Grid Integrated Edition environment’s CIDRs, such as the service network CIDR where your Tanzu Kubernetes Grid Integrated Edition cluster is deployed, the deployment network CIDR, the node network IP block CIDR, and the pod network IP block CIDR.

- The FQDN of any registry, such as the Harbor API FQDN, or component communicating with Tanzu Kubernetes Grid Integrated Edition, using a hostname instead of an IP address.

- 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 Tanzu Kubernetes Grid Integrated Edition FQDN are not proxied. This allows internal Tanzu Kubernetes Grid Integrated Edition communication.

Do not use the _ character in the No Proxy field. Entering an underscore character in this field can cause upgrades to fail.

Because some jobs in the VMs accept * as a wildcard, while others only accept ., we recommend that you define a wildcard domain using both of them. For example, to denote example.com as a wildcard domain, add both *.example.com and example.com to the No Proxy property.

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

6. Click Save.

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 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 Tanzu Kubernetes Grid Integrated Edition environment.

To prepare a Windows pause image for an air-gapped environment, perform the following:

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:

   ```bash
docker pull mcr.microsoft.com/k8s/core/pause:1.3.0
   ```

6. To tag the Windows container image, run the following command:

   ```bash
docker tag mcr.microsoft.com/k8s/core/pause:1.3.0 REGISTRY-ROOT/windows/pause:1.3.0
   ```

   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:

   ```bash
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:1.3.0`.

To configure Tanzu Kubernetes Grid Integrated Edition to fetch your accessible Windows container image when deploying Windows pods, perform the following:

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:

   ```text
   my.private.registry/windows/pause:1.3.0
   ```
4. Click **Save**.

Please send any feedback you have to pks-feedback@pivotal.io.
Upgrading Tanzu Kubernetes Grid Integrated Edition

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This section describes how to upgrade VMware Tanzu Kubernetes Grid Integrated Edition. See the following topics:

- Upgrade Tanzu Kubernetes Grid Integrated Edition Management Console
- Upgrading Tanzu Kubernetes Grid Integrated Edition with Ops Manager

Please send any feedback you have to pks-feedback@pivotal.io.
Upgrade Tanzu Kubernetes Grid Integrated Edition Management Console

In this topic

Prerequisites

Step 1: Deploy the New OVA Template
Step 2: Log In to the New Version of Tanzu Kubernetes Grid Integrated Edition Management Console
Step 3: Migrate the Configuration from the Old Appliance to the New Version

Next Steps

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

To upgrade an existing installation of Tanzu Kubernetes Grid Integrated Edition Management Console, you download and deploy a new version of the Tanzu Kubernetes Grid Integrated Edition Management Console VM. You then use the management console UI of the new version to migrate the configuration of the old installation to the new one.

You can only use the management console to upgrade an Tanzu Kubernetes Grid Integrated Edition installation that was deployed from a previous version of the management console. You cannot use the console to upgrade an instance of Tanzu Kubernetes Grid Integrated Edition that you installed manually.

Prerequisites

- You have deployed and configured an older version of Tanzu Kubernetes Grid Integrated Edition Management Console.
- 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 Tanzu Kubernetes Grid Integrated Edition Management Console with a static IP address, and you want the new version to retain the same IP address after the upgrade, reconfigure the old Management Console VM to use a temporary IP address before you start the upgrade procedure:

  1. Shut down the previous version of the management console VM by selecting **Shut Down Guest OS**.

    **warning:** Do not select **Power Off**.

  2. Access the vApp options for the management console VM.

    - vSphere Client 6.5: Right-click the management console VM, and select **Edit Settings** and select **vApp Options**.
    - vSphere Client 6.7: Select the management console VM, select the **Configure** tab > **vApp Options** and scroll to the Properties section.

  3. Set a temporary IP address on the management console 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**.

Copyright © 2020 VMware, Inc. All Rights Reserved.
4. Save your changes and power the management console VM back on.

Step 1: Deploy the New OVA Template

Follow the instructions in Deploy the Tanzu Kubernetes Grid Integrated Edition Management Console to deploy and power on the new version of the management console VM 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 management console, configure the network settings of the new Management Console to use the same static IP address as previously.

If you used custom certificates when you deployed the previous version of the management console, you must use the same certificates when you deploy the new version of the management console. 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 Tanzu Kubernetes Grid Integrated Edition 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 Management Console VM and select Power > Power On.

2. When the new Management Console 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 Management Console VM in a browser.

4. At the VMware Tanzu Kubernetes Grid Integrated Edition 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

Tanzu Kubernetes Grid Integrated Edition 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.

2. Enter the IP address of the old version of the Tanzu Kubernetes Grid Integrated Edition Management Console VM in the **Endpoint** text box.

3. Enter the username and password for the old version of the management console VM and click **Connect**.

4. Under Resources, verify the list of components that will be upgraded and click **Next**.
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 Tanzu Kubernetes Grid Integrated Edition by Using the Configuration Wizard. For information about new parameters that have been added, see the release notes for the version to which you are upgrading.

6. When all of the sections of the configuration wizard are green, click Generate Configuration to see the generated YAML file.

7. (Optional) Specify an FQDN address for the Ops Manager VM by editing the YAML directly in the YAML editor.

   \[
   \text{warning: You cannot change the Ops Manager FQDN of Tanzu Kubernetes Grid Integrated Edition once it has already deployed.}
   \]

To specify an FQDN address for the Ops Manager VM, update the YAML as follows:

   a. Locate the `opsman_fqdn:` entry in the YAML file.
   b. Update the `opsman_fqdn:` entry with the Ops Manager VM FQDN:

   \[
   \text{opsman_fqdn: "myopsman.example.com"}
   \]

   c. Make sure that the FQDN is mapped to the following IP address:

      - For NSX-T deployments map it to the first address in the floating IP range.
      - For Flannel deployments, map it to the first address in the deployment network, excluding the gateway, deployment DNS, and reserved IP range.

      If you start the upgrade and you have not mapped the FQDN to an IP address, the deployment fails with an error. If this happens, configure the mapping as above, return to the YAML editor, and start the upgrade again.

8. 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`.

9. Click Apply Configuration then Continue to upgrade Tanzu Kubernetes Grid Integrated Edition.


Next Steps

You can now access the upgraded Tanzu Kubernetes Grid Integrated Edition control plane and continue deploying Kubernetes clusters. Any new clusters that you deploy from the upgraded Tanzu Kubernetes Grid Integrated Edition control plane will use the new version of Kubernetes.
For information about how you can use Tanzu Kubernetes Grid Integrated Edition Management Console to monitor and manage your upgraded deployment, see Monitor and Manage Tanzu Kubernetes Grid Integrated Edition in the Management Console.

You can decommission the old version of Tanzu Kubernetes Grid Integrated Edition by deleting the previous version of the Tanzu Kubernetes Grid Integrated Edition Management Console VM from the vSphere inventory.

Please send any feedback you have to pks-feedback@pivotal.io.
Upgrading Tanzu Kubernetes Grid Integrated Edition with Ops Manager

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

**Page last updated:**

This section describes how to upgrade the VMware Tanzu Kubernetes Grid Integrated Edition tile. See the following topics:

- Upgrade Preparation Checklist for Tanzu Kubernetes Grid Integrated Edition v1.8
- Upgrade Order for Tanzu Kubernetes Grid Integrated Edition Environments on vSphere
- Upgrading Tanzu Kubernetes Grid Integrated Edition (Flannel Networking)
- Upgrading Tanzu Kubernetes Grid Integrated Edition (NSX-T Networking)
- Maintaining Workload Uptime
- Configuring the Upgrade Pipeline


Please send any feedback you have to pks-feedback@pivotal.io.
Upgrade Preparation Checklist for Tanzu Kubernetes Grid Integrated Edition v1.8

In this topic

Back Up Your Tanzu Kubernetes Grid Integrated Edition Deployment
Review What Happens During Tanzu Kubernetes Grid Integrated Edition Upgrades
Review Changes in Tanzu Kubernetes Grid Integrated Edition v1.8
Determine Upgrade Order (vSphere Only)
Set User Expectations and Restrict Cluster Access
Upgrade All Clusters
Verify Your Clusters Support Upgrading
Verify Health of Kubernetes Environment
Verify NSX-T Configuration (vSphere with NSX-T Only)
Clean Up or Fix Failed Kubernetes Clusters
Verify Kubernetes Clusters Have Unique External Hostnames
Verify TKGI Proxy Configuration
Check PodDisruptionBudget Value
(Optional) Configure Node Drain Behavior
  Configure with the Tanzu Kubernetes Grid Integrated Edition Tile
  Configure with the TKGI CLI

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic serves as a checklist for preparing to upgrade VMware Tanzu Kubernetes Grid Integrated Edition from v1.7 to v1.8.

This topic lists 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, continue to Upgrading Tanzu Kubernetes Grid Integrated Edition (Flannel Networking) or Upgrading Tanzu Kubernetes Grid Integrated Edition (NSX-T Networking).

Back Up Your Tanzu Kubernetes Grid Integrated Edition Deployment


Review What Happens During Tanzu Kubernetes Grid Integrated Edition Upgrades

If you have not already done so, review About Tanzu Kubernetes Grid Integrated Edition Upgrades

Plan your upgrade based on your workload capacity and uptime requirements.
Review Changes in Tanzu Kubernetes Grid Integrated Edition v1.8


Determine Upgrade Order (vSphere Only)

To determine the upgrade order for your Tanzu Kubernetes Grid Integrated Edition environment, review Upgrade Order for Tanzu Kubernetes Grid Integrated Edition Environments on vSphere.

Set User Expectations and Restrict Cluster Access

Coordinate the Tanzu Kubernetes Grid Integrated Edition 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 TKGI or use the TKGI CLI and other TKGI 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 AllClusters

Tanzu Kubernetes Grid Integrated Edition v1.8 does not support clusters running versions of TKGI earlier than v1.7.

Before you upgrade from Tanzu Kubernetes Grid Integrated Edition v1.7 to v1.8, you must upgrade all of your TKGI-provisioned clusters to v1.7.

To upgrade TKGI-provisioned clusters:

1. Check the version of your clusters:

   tkgi clusters

2. If one or more of your clusters are running a version of TKGI earlier than v1.7, upgrade the clusters. For instructions, 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 Tanzu Kubernetes Grid Integrated Edition 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 `tkgi 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition.

### Clean Up or Fix Failed Kubernetes Clusters

Clean up or fix any previous failed attempts to create TKGI clusters with the TKGI Command Line Interface (TKGI CLI) by performing the following steps:

1. View your deployed clusters by running the following command:

   ```
tkgi 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 `tkgi clusters`.

For more information about troubleshooting and fixing failed clusters, see the 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 TKGI CLI. For more information, see Logging in to Tanzu Kubernetes Grid Integrated Edition. You must log in with an account that has the UAA scope of `pks.clusters.admin`. For more information about UAA scopes, see Managing Tanzu Kubernetes Grid Integrated Edition Users with UAA.

2. View your deployed TKGI clusters by running the following command:

   `tkgi clusters`

3. For each deployed cluster, run `tkgi cluster CLUSTER-NAME` to view the details of the cluster. For example:

   `$ tkgi cluster my-cluster`

Examine the output to verify that the Kubernetes Master Host is unique for each cluster.

Verify TKGI Proxy Configuration

Verify your current TKGI proxy configuration by performing the following steps:

1. Check whether an existing proxy is enabled:
   a. Log in to Ops Manager.
   b. Click the VMware Tanzu Kubernetes Grid Integrated Edition 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 Support:
   - `localhost`
   - Hostnames containing dashes, such as `my-host.mydomain.com`

Check PodDisruptionBudget Value

Tanzu Kubernetes Grid Integrated Edition 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:

1. Run the following `kubectl` command to verify the PodDisruptionBudget as the cluster administrator:

   ```bash
   kubectl get poddisruptionbudgets --all-namespaces
   ```

2. Examine the output to verify that no app displays 0 in the MAX UNAVAILABLE column.

(Optional) Configure Node Drain Behavior

During the Tanzu Kubernetes Grid Integrated Edition 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 configure default node drain behavior in Tanzu Kubernetes Grid Integrated Edition tile or with the TKGI CLI.

The new default behavior takes effect during the next upgrade, not immediately after configuring the behavior.

Configure with the Tanzu Kubernetes Grid Integrated Edition Tile

To configure node drain behavior in the Tanzu Kubernetes Grid Integrated Edition tile, see Worker Node Hangs Indefinitely in Troubleshooting.

Configure with the TKGI CLI

To configure default node drain behavior with the TKGI CLI:

1. View the current node drain behavior by running the following command:

   ```bash
   tkgi cluster CLUSTER-NAME --details
   ```

   Where `CLUSTER-NAME` is the name of your cluster.

   For example:
2. Configure the default node drain behavior by running the following command:

```
tkgi 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:

```
s tkgi 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 `tkgi cluster my-cluster` to monitor the state of your cluster

For a list of the available action flags for setting node drain behavior, see `tkgi update-cluster` in **TKGI CLI**.

Please send any feedback you have to pks-feedback@pivotal.io.
Upgrade Order for Tanzu Kubernetes Grid Integrated Edition Environments on vSphere

In this topic

Overview

TKGI on vSphere with NSX-T Networking
  Scenario 1: Upgrading to TKGI v1.8
  Scenario 2: Upgrading to TKGI v1.8 and NSX-T v3.0
  Scenario 3: Upgrading to TKGI v1.8, NSX-T v3.0, and vSphere v7.0

TKGI on vSphere with Flannel Networking
  Scenario 1: Upgrading to TKGI v1.8
  Scenario 2: Upgrading to TKGI v1.8 and vSphere v7.0

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic provides upgrade scenarios for Tanzu Kubernetes Grid Integrated Edition (TKGI) environments that are upgraded from v1.7 to v1.8 on vSphere.

Overview

When you upgrade TKGI on vSphere, you might also upgrade vSphere and, if you are using it, NSX-T.

TKGI, NSX-T, and vSphere upgrades depend on each other. Some combinations also require upgrading Ops Manager or TKGI-provisioned Kubernetes clusters.

For any combination of upgrades that you perform, you must follow the upgrade order described in this topic.

- If your environment is on vSphere with NSX-T networking, see TKGI on vSphere with NSX-T Networking below.
- If your environment is on vSphere with Flannel networking, see TKGI on vSphere with Flannel Networking below.

TKGI on vSphere with NSX-T Networking

When upgrading a TKGI environment on vSphere with NSX-T networking, you can choose to upgrade any of the following:

- TKGI only, optionally including Kubernetes clusters
- TKGI, Kubernetes clusters, and NSX-T
- TKGI, Kubernetes clusters, NSX-T, and vSphere

For more information, see below:

<table>
<thead>
<tr>
<th>To upgrade these components…</th>
<th>Use this order…</th>
<th>For more information, see…</th>
</tr>
</thead>
</table>

Note:
As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.
### Scenario 1: Upgrading to TKGI v1.8

In this upgrade scenario, you upgrade Tanzu Kubernetes Grid Integrated Edition from v1.7 to v1.8 and do not upgrade your NSX-T or vSphere infrastructure.

The upgrade scenario includes the following steps:

1. Upgrade Ops Manager to v2.9.3 or later, v2.8.2 or later, or v2.7.15 or later. These are the recommended Ops Manager versions for Tanzu Kubernetes Grid Integrated Edition v1.8.0. To verify Ops Manager compatibility with other v1.8 versions, see [VMware Tanzu Network](https://www.vmware.com/products/tanzu-network.html).


3. (Recommended) Upgrade all Kubernetes clusters to Tanzu Kubernetes Grid Integrated Edition v1.8. This upgrades the NCP version of your clusters to 3.0.1.

See the table below for version information and instructions for this upgrade scenario:

<table>
<thead>
<tr>
<th>Component</th>
<th>Pre-upgrade version</th>
<th>Post-upgrade version</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TKGI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Upgrade Ops Manager if necessary.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Upgrade TKGI.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. (Recommended) Upgrade Kubernetes clusters.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Warning:** TKGI supports NSX-T v3.0 as a beta integration. Intermittent upgrade failures and scale problems may occur if you upgrade to NSX-T v3.0. Upgrading your NSX-T environment to v3.0 in a production or large-scale deployment is not recommended until a patch resolving these issues has been released.

For a list of NSX-T and vSphere versions compatible with Tanzu Kubernetes Grid Integrated Edition v1.8, see:

- [VMware Product Interoperability Matrices](https://www.vmware.com/products/tanzu-network.html)
Scenario 2: Upgrading to TKGI v1.8 and NSX-T v3.0

⚠️ **warning:** TKGI supports NSX-T v3.0 as a beta integration. Intermittent upgrade failures and scale problems may occur if you upgrade to NSX-T v3.0. Upgrading your NSX-T environment to v3.0 in a production or large-scale deployment is not recommended until a patch resolving these issues has been released.

In this upgrade scenario, you upgrade Tanzu Kubernetes Grid Integrated Edition from v1.7 to v1.8 and NSX-T from v2.5.0 or v2.5.1 to v3.0.

The upgrade scenario includes the following steps:

1. Upgrade Ops Manager to v2.9.3 or later. These are the recommended Ops Manager versions for Tanzu Kubernetes Grid Integrated Edition v1.8.0. To verify Ops Manager compatibility with other v1.8 versions, see [VMware Tanzu Network](https://www.vmware.com/support/tanzu-network.html).


3. Upgrade all Kubernetes clusters to Tanzu Kubernetes Grid Integrated Edition v1.8. This upgrades the NCP version of your clusters to 3.0.1.

4. Upgrade NSX-T from v2.5.0 or v2.5.1 to v3.0.

See the table below for version information and instructions for this upgrade scenario:

<table>
<thead>
<tr>
<th>Component</th>
<th>Pre-upgrade version</th>
<th>Post-upgrade version</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TKGI</td>
<td>1.7</td>
<td>1.8</td>
<td>See Upgrading Tanzu Kubernetes Grid Integrated Edition (NSX-T Networking).</td>
</tr>
<tr>
<td>Ops Manager</td>
<td>2.7, 2.8, or 2.9</td>
<td>2.7, 2.8, or 2.9</td>
<td>n/a</td>
</tr>
<tr>
<td>NSX-T*</td>
<td>2.5.0 or 2.5.1</td>
<td>3.0</td>
<td>See Upgrade NSX-T Data Center v2.5 to v3.0 in Upgrading Tanzu Kubernetes Grid Integrated Edition (NSX-T Networking).</td>
</tr>
<tr>
<td>NCP</td>
<td>2.5.1</td>
<td>3.0.1</td>
<td>n/a</td>
</tr>
<tr>
<td>vSphere</td>
<td>6.5 or 6.7</td>
<td>6.5 or 6.7</td>
<td>n/a</td>
</tr>
</tbody>
</table>

* This scenario assumes that your Enterprise PKS v1.7 environment is running with NSX-T v2.5. Tanzu Kubernetes Grid Integrated Edition v1.8 does not support NSX-T v2.4. See Upgrade NSX-T Data Center v2.4.3 to v2.5 in the Enterprise PKS v1.7 documentation.

Copyright © 2020 VMware, Inc. All Rights Reserved.
Scenario 3: Upgrading to TKGI v1.8, NSX-T v3.0, and vSphere v7.0

⚠️ warning: TKGI supports NSX-T v3.0 as a beta integration. Intermittent upgrade failures and scale problems may occur if you upgrade to NSX-T v3.0. Upgrading your NSX-T environment to v3.0 in a production or large-scale deployment is not recommended until a patch resolving these issues has been released.

In this upgrade scenario, you upgrade:

- Tanzu Kubernetes Grid Integrated Edition from v1.7 to v1.8
- NSX-T from v2.5.0 or v2.5.1 to v3.0
- vSphere from v6.5 or v6.7 to v7.0

The upgrade scenario includes the following steps:

1. Upgrade Ops Manager to v2.9.3 or later. These are the recommended Ops Manager versions for Tanzu Kubernetes Grid Integrated Edition v1.8.0. To verify Ops Manager compatibility with other v1.8 versions, see VMware Tanzu Network.


3. Upgrade all Kubernetes clusters to Tanzu Kubernetes Grid Integrated Edition v1.8. This upgrades the NCP version of your clusters to 3.0.1.

4. Upgrade NSX-T from v2.5.0 or v2.5.1 to v3.0.

5. Upgrade vSphere from v6.5 or v6.7 to v7.0.

See the table below for version information and instructions for this upgrade scenario:

<table>
<thead>
<tr>
<th>Component</th>
<th>Pre-upgrade version</th>
<th>Post-upgrade version</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TKGI</td>
<td>1.7</td>
<td>1.8</td>
<td>See Upgrading Tanzu Kubernetes Grid Integrated Edition (NSX-T Networking).</td>
</tr>
<tr>
<td>Ops Manager</td>
<td>2.7, 2.8, or 2.9</td>
<td>2.9</td>
<td>n/a</td>
</tr>
<tr>
<td>NSX-T *</td>
<td>2.5.0 or 2.5.1</td>
<td>3.0</td>
<td>See Upgrade NSX-T Data Center v2.5 to v3.0 in Upgrading Tanzu Kubernetes Grid Integrated Edition (NSX-T Networking).</td>
</tr>
<tr>
<td>NCP</td>
<td>2.5.1</td>
<td>3.0.1</td>
<td>n/a</td>
</tr>
<tr>
<td>vSphere</td>
<td>6.5 or 6.7</td>
<td>7.0</td>
<td>See the vSphere documentation:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1. Upgrading the vCenter Server Appliance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Upgrading ESXi Hosts</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Also, see Overview of the vSphere Upgrade Process</td>
</tr>
</tbody>
</table>

* This scenario assumes that your Enterprise PKS v1.7 environment is running with NSX-T v2.5. Tanzu Kubernetes Grid Integrated Edition v1.8 does not support NSX-T v2.4. See Upgrade NSX-T Data Center v2.4.3 to v2.5 in the Enterprise PKS v1.7 documentation.
TKGI on vSphere with Flannel Networking

When upgrading a Tanzu Kubernetes Grid Integrated Edition environment on vSphere with Flannel networking, you can choose to upgrade any of the following:

- TKGI only, optionally including Kubernetes clusters
- TKGI, Kubernetes clusters, and vSphere

For more information, see below:

<table>
<thead>
<tr>
<th>To upgrade these components…</th>
<th>Use this order…</th>
<th>For more information, see…</th>
</tr>
</thead>
</table>
| TKGI                         | 1. Upgrade Ops Manager if necessary.  
                            2. Upgrade TKGI.  
                            3. (Recommended) Upgrade Kubernetes clusters. | Upgrading to TKGI v1.8 |
| TKGI and vSphere             | 1. Upgrade Ops Manager if necessary.  
                            2. Upgrade TKGI.  
                            3. Upgrade Kubernetes clusters.  
                            4. Upgrade vSphere. | Upgrading to TKGI v1.8 and vSphere v7.0 |

For a list of vSphere versions compatible with Tanzu Kubernetes Grid Integrated Edition v1.8, see VMware Product Interoperability Matrices.

Scenario 1: Upgrading to TKGI v1.8

In this upgrade scenario, you upgrade Tanzu Kubernetes Grid Integrated Edition from v1.7 to v1.8 and do not upgrade your vSphere infrastructure.

The upgrade scenario includes the following steps:

1. Upgrade Ops Manager to v2.9.3 or later, v2.8.2 or later, or v2.7.15 or later. These are the recommended Ops Manager versions for Tanzu Kubernetes Grid Integrated Edition v1.8.0. To verify Ops Manager compatibility with other v1.8 versions, see VMware Tanzu Network.


3. (Recommended) Upgrade all Kubernetes clusters to Tanzu Kubernetes Grid Integrated Edition v1.8.

See the table below for version information and instructions for this upgrade scenario:

<table>
<thead>
<tr>
<th>Component</th>
<th>Pre-upgrade version</th>
<th>Post-upgrade version</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TKGI</td>
<td>1.7</td>
<td>1.8</td>
<td>See Upgrading Tanzu Kubernetes Grid Integrated Edition (Flannel Networking).</td>
</tr>
<tr>
<td>Ops Manager</td>
<td>2.7, 2.8, or 2.9</td>
<td>2.7, 2.8, or 2.9</td>
<td>n/a</td>
</tr>
<tr>
<td>vSphere</td>
<td>6.5 or 6.7</td>
<td>6.5 or 6.7</td>
<td>n/a</td>
</tr>
</tbody>
</table>
Scenario 2: Upgrading to TKGI v1.8 and vSphere v7.0

In this upgrade scenario, you upgrade Tanzu Kubernetes Grid Integrated Edition from v1.7 to v1.8 and vSphere from v6.5 or v6.7 to v7.0.

The upgrade scenario includes the following steps:

1. Upgrade Ops Manager to v2.9.3 or later. These are the recommended Ops Manager versions for Tanzu Kubernetes Grid Integrated Edition v1.8.0. To verify Ops Manager compatibility with other v1.8 versions, see [VMware Tanzu Network](https://www.vmware.com).  


4. Upgrade vSphere from v6.5 or v6.7 to v7.0.

See the table below for version information and instructions for this upgrade scenario:

<table>
<thead>
<tr>
<th>Component</th>
<th>Pre-upgrade version</th>
<th>Post-upgrade version</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TKGI</td>
<td>1.7</td>
<td>1.8</td>
<td>See <a href="https://www.vmware.com">Upgrading Tanzu Kubernetes Grid Integrated Edition (Flannel Networking)</a>.</td>
</tr>
<tr>
<td>Ops Manager</td>
<td>2.7, 2.8, or 2.9</td>
<td>2.9</td>
<td>n/a</td>
</tr>
</tbody>
</table>
| vSphere   | 6.5 or 6.7          | 7.0                  | See the vSphere documentation:  
1. [Upgrading the vCenter Server Appliance](https://www.vmware.com)  
2. [Upgrading ESXi Hosts](https://www.vmware.com)  
Also, see [Overview of the vSphere Upgrade Process](https://www.vmware.com). |

Please send any feedback you have to pks-feedback@pivotal.io.
Upgrading Tanzu Kubernetes Grid Integrated Edition (Flannel Networking)

### In this topic

- Overview
- Prepare to Upgrade
- Perform the Upgrade
  - Upgrade Ops Manager
  - Download and Import Tanzu Kubernetes Grid Integrated Edition v1.8
  - Download and Import Stemcells
  - Verify Errand Configuration
  - Verify Other Configurations
  - Apply Changes to the Tanzu Kubernetes Grid Integrated Edition Tile
- After the Upgrade
  - Update the TKGI and Kubernetes CLIs
  - Verify the Upgrade

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

**Page last updated:**

This topic explains how to upgrade VMware Tanzu Kubernetes Grid Integrated Edition from v1.7 to v1.8 on vSphere with Flannel networking, Google Cloud Platform (GCP), Amazon Web Services (AWS), and Azure.

For instructions on upgrading Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T networking, see [Upgrading Tanzu Kubernetes Grid Integrated Edition (NSX-T Networking)](#).

**warning:** Do not manually upgrade your Kubernetes version. Tanzu Kubernetes Grid Integrated Edition includes the compatible Kubernetes version.

### Overview

Before you upgrade, follow the procedures in **Prepare to Upgrade** below to plan and prepare your upgrade.

After you complete the preparation steps, continue to the procedures in **Perform the Upgrade** below. These steps guide you through the process of upgrading Ops Manager and the Tanzu Kubernetes Grid Integrated Edition tile, importing an updated stemcell, and applying the changes to your deployment.

After you complete the upgrade, follow the procedures in **After the Upgrade** below to verify that your upgraded Tanzu Kubernetes Grid Integrated Edition deployment is running properly.

### Prepare to Upgrade

If you have not already, complete all of the steps in [Upgrade Preparation Checklist for Tanzu Kubernetes Grid Integrated Edition](#).
Perform the Upgrade

This section describes the steps required to upgrade to Tanzu Kubernetes Grid Integrated Edition v1.8:

1. Upgrade Ops Manager
2. Download and Import Tanzu Kubernetes Grid Integrated Edition v1.8
3. Download and Import Stemcells
4. Verify Errand Configuration
5. Verify Other Configurations
6. Apply Changes to the Tanzu Kubernetes Grid Integrated Edition Tile

Upgrade Ops Manager

Each version of Tanzu Kubernetes Grid Integrated Edition is compatible with multiple versions of Ops Manager. To determine Ops Manager compatibility and, if necessary, upgrade Ops Manager:

1. See VMware Tanzu Network to determine if your Ops Manager version is compatible with Tanzu Kubernetes Grid Integrated Edition v1.8.
2. If your Ops Manager version is not compatible with Tanzu Kubernetes Grid Integrated Edition v1.8, follow the steps below.
3. Upgrade Ops Manager. For instructions, see Import Installation to Ops Manager v2.9 VM.
4. Verify that the Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition control plane in the Tanzu Kubernetes Grid Integrated Edition tile > Status tab. Review the load and resource usage data for the TKGI API and TKGI Database VMs. If any levels are at capacity, scale up the VMs.

Download and Import Tanzu Kubernetes Grid Integrated Edition v1.8

When you upgrade Tanzu Kubernetes Grid Integrated Edition, your configuration settings typically migrate to the new version automatically. To download and import a Tanzu Kubernetes Grid Integrated Edition version:

1. Download the desired version of the product from VMware Tanzu Network.
2. Navigate to the Ops Manager Installation Dashboard and click Import a Product to upload the product file.
3. Under the Import a Product button, click + next to Tanzu Kubernetes Grid Integrated Edition. This adds the tile to your staging area.

Download and Import Stemcells
Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition v1.8, the Tanzu Kubernetes Grid Integrated Edition tile displays the message Missing stemcell. To download and import a new Xenial stemcell, follow the steps below:


2. In the Stemcell Library, locate the Tanzu Kubernetes Grid Integrated Edition tile and note the required stemcell version.

3. Navigate to the Stemcells (Ubuntu Xenial) page on VMware Tanzu 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 VMware Tanzu Network.


7. Verify that Ops Manager successfully applied the stemcell. The stemcell version you imported and applied appears in the Staged column for Tanzu Kubernetes Grid Integrated Edition.

8. Return to the Installation Dashboard.

Verify Errand Configuration

To verify your Errands pane is correctly configured, do the following:


2. Under Post-Deploy Errands:
   - Review the Upgrade all clusters errand:
     - If you want to upgrade the Tanzu Kubernetes Grid Integrated Edition tile and all your existing Kubernetes clusters simultaneously, confirm that Upgrade all clusters errand is set to Default (On). The errand upgrades all clusters. Upgrading Tanzu Kubernetes Grid Integrated Edition-provisioned Kubernetes clusters can temporarily interrupt the
service as described in Service Interruptions.

- If you want to upgrade the Tanzu Kubernetes Grid Integrated Edition tile only and then upgrade your existing Kubernetes clusters separately, disable Upgrade all clusters errand. For more information, see Upgrading Clusters.

⚠️ warning: Disabling the Upgrade all clusters errand causes the TKGI version tagged in your Kubernetes clusters to fall behind the Tanzu Kubernetes Grid Integrated Edition tile version. If you disable the Upgrade all clusters errand when upgrading the Tanzu Kubernetes Grid Integrated Edition tile, you must upgrade all your Kubernetes clusters before the next Tanzu Kubernetes Grid Integrated Edition upgrade.

- Configure the Run smoke tests errand:
  - Set the Run smoke tests errand to On. The errand uses the Tanzu Kubernetes Grid Integrated Edition Command Line Interface (TKGI CLI) to create a Kubernetes cluster and then delete it. If the creation or deletion fails, the errand fails and the installation of the Tanzu Kubernetes Grid Integrated Edition tile is aborted.

3. Click Save.

Verify Other Configurations

To confirm your other Tanzu Kubernetes Grid Integrated Edition tile panes are correctly configured, do the following:

1. Review the Assign AZs and Networks pane.

   ⚠️ Note: When you upgrade Tanzu Kubernetes Grid Integrated Edition, you must place singleton jobs in the AZ you selected when you first installed the Tanzu Kubernetes Grid Integrated Edition tile. You cannot move singleton jobs to another AZ.

2. Review the other configuration panes.

3. Make changes where necessary.

⚠️ warning: Do not change the number of master/etcd nodes for any plan that was used to create currently-running clusters. Tanzu Kubernetes Grid Integrated Edition does not support changing the number of master/etcd nodes for plans with existing clusters.

4. Click Save on any panes where you make changes.

Apply Changes to the Tanzu Kubernetes Grid Integrated Edition Tile

To complete the upgrade of the Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition deployment. For more information, see Using BOSH Diagnostic Commands in Tanzu Kubernetes Grid Integrated Edition.

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 Tanzu Kubernetes Grid Integrated Edition v1.8, complete the following verifications and upgrades:

- Update the TKGI and Kubernetes CLIs
- Verify the Upgrade

Update the TKGI and Kubernetes CLIs

Update the TKGI and Kubernetes CLIs on any local machine where you run commands that interact with your upgraded version of Tanzu Kubernetes Grid Integrated Edition.

To update your CLIs, download and re-install the TKGI and Kubernetes CLI distributions that are provided with Tanzu Kubernetes Grid Integrated Edition on VMware Tanzu Network.

For more information about installing the CLIs, see the following topics:

- Installing the TKGI CLI
- Installing the Kubernetes CLI

Verify the Upgrade

After you apply changes to the Tanzu Kubernetes Grid Integrated Edition tile and the upgrade is complete, do the following:

1. Verify that your Kubernetes environment is healthy. To verify the health of your Kubernetes environment, see Verifying Deployment Health.
   For any cluster upgrade that fails, you can use the BOSH ID of the upgrade task for debugging. To retrieve the BOSH task ID, see Retrieve Cluster Upgrade Task ID in Verifying Deployment Health.

2. Verify that the Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition control plane in the Tanzu Kubernetes Grid Integrated Edition tile > Status tab. Review the load and resource usage data for the TKGI API and TKGI Database VMs. If any levels are at capacity, scale up the VMs.

Please send any feedback you have to pks-feedback@pivotal.io.
Upgrading Tanzu Kubernetes Grid Integrated Edition (NSX-T Networking)

In this topic
Overview
Prerequisites
Prepare to Upgrade
Perform the Upgrade
  Upgrade Ops Manager
  Download and Import TKGI v1.8
  Download and Import the Stemcell if Needed
  Upgrade the TKGI Tile
After the Upgrade
  Update the TKGI and Kubernetes CLIs
  Upgrade Kubernetes Clusters if Needed
  Verify TKGI Upgrade
(Optional) Upgrade NSX-T Data Center v2.5 to v3.0
(Optional) Upgrade to vSphere 7
Troubleshoot the Upgrade

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic explains how to upgrade Tanzu Kubernetes Grid Integrated Edition (TKGI) from v1.7 to v1.8 on vSphere with NSX-T networking.

For instructions on upgrading TKGI with Flannel networking, see Upgrading Tanzu Kubernetes Grid Integrated Edition (Flannel Networking).

warning: Do not manually upgrade your Kubernetes version. TKGI includes the compatible Kubernetes version.

Overview

Before you upgrade, follow the procedures in Prepare to Upgrade below to plan and prepare your upgrade.

After you complete the preparation steps, continue to the procedures in Perform the Upgrade below. These steps guide you through the process of upgrading Ops Manager and the TKGI tile, importing an updated stemcell, and applying the changes to your deployment.

After you complete the upgrade, follow the procedures in After the Upgrade below to verify that your upgraded TKGI deployment is running properly and to optionally upgrade NSX-T and vSphere.
Prerequisites

TKGI v1.8 does not support NSX-T v2.4. You must upgrade to NSX-T v2.5 before upgrading to TKGI v1.8.

To see a list of NSX-T 2.5 versions compatible with TKGI v1.8, consult Product Snapshot in Release Notes for TKGI v1.8.

Prepare to Upgrade

If you have not already, complete all of the steps in Upgrade Preparation Checklist for Tanzu Kubernetes Grid Integrated Edition v1.8.

Perform the Upgrade

This section describes the steps required to upgrade to TKGI v1.8:

1. **Upgrade Ops Manager**
2. **Download and Import TKGI v1.8**
3. **Download and Import the Stemcell if Needed**
4. **Upgrade the TKGI Tile**

Upgrade Ops Manager

Each version of TKGI is compatible with multiple versions of Ops Manager. See VMware Tanzu Network to determine if your Ops Manager version is compatible with TKGI v1.8.

To upgrade Ops Manager from v2.8.5 or later to v2.9.3 or later:

1. **Log in to Ops Manager.**
2. **Click your username in the top right corner and navigate to Settings > Export Installation Settings.**
3. **Click Export Installation Settings.**
   - Ops Manager exports an encrypted archive of your current installation configuration.
   - Later, you import this configuration into your upgraded Ops Manager.
4. Log in to vCenter Server using the vSphere Client.

5. Shut down the Ops Manager VM.

6. Deploy the upgraded Ops Manager VM by following the first two steps of Deploying Ops Manager with NSX-T for TKGI
   a. Step 1: Generate SSH Key Pair
   b. Step 2: Deploy Ops Manager for Tanzu Kubernetes Grid Integrated Edition

7. Using a browser, navigate to the newly-deployed Ops Manager web interface.

8. On the welcome page, select **Import Existing Installation**.
9. Browse to and select the installation configuration archive you exported.

10. Log in to Ops Manager

11. Click **Apply Changes**.

12. Verify that the BOSH Director for vSphere tile shows the target, updated version.
Download and Import TKGI v1.8

When you upgrade TKGI, your configuration settings typically migrate to the new version automatically. To download and import a TKGI version:

1. Download the target version of the product from VMware Tanzu Network.

2. Import the target version of the TKGI tile to the Ops Manager Installation Dashboard.

3. Click Review Pending Changes.
4. Expand the **Errands** dropdown and enable or disable **Upgrade all clusters errand**
   - See [Deciding Between Full and Two-Phase Upgrade](#) to decide whether to upgrade TKGI-provisioned Kubernetes clusters along with TKGI, or upgrade them later.
   - VMware recommends that you upgrade Kubernetes clusters along with TKGI if possible.
   - Enable the **Upgrade all clusters errand** to upgrade clusters along with TKGI.

   **warning:** Disabling the **Upgrade all clusters errand** causes the TKGI version tagged in your Kubernetes clusters to fall behind the TKGI tile version. If you disable the **Upgrade all clusters errand** when upgrading the TKGI tile, you must upgrade all your Kubernetes clusters before the next TKGI upgrade.

   **Note:** If you are upgrading TKGI on NSX-T v2.5, you must select the **Upgrade all clusters errand** or plan to upgrade all clusters individually before upgrading to NSX-T v3.0.

5. Set the **Run smoke tests** errand to **On**. The errand uses the TKGI CLI to create a Kubernetes cluster and then delete it. If the creation or deletion fails, the errand fails and the installation of the TKGI tile is aborted.

Download and Import the Stemcell if Needed

TKGI requires a Xenial stemcell.

If Ops Manager does not have the Xenial stemcell required for TKGI v1.8, the TKGI tile displays the message **Missing stemcell**. To download and import a new Xenial stemcell, follow the steps below:

1. On the TKGI tile, click the **Missing stemcell** link.

2. In the **Stemcell Library**, locate the **TKGI** tile and note the required stemcell version.

3. Navigate to the **Stemcells (Ubuntu Xenial)** page on VMware Tanzu 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 VMware Tanzu Network.

6. Select the TKGI 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 TKGI.

8. Return to the **Installation Dashboard**.

**Upgrade the TKGI Tile**

To complete the upgrade of the TKGI 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**.

![Review Pending Changes](image)

4. (Optional) If you enabled the **Upgrade all clusters errand**, you can use the BOSH CLI to monitor its progress:
   
a. Log in to the BOSH Director by running `bosh -e MY-ENVIRONMENT log-in` from a VM that can access your TKGI deployment. For more information, see [Using BOSH Diagnostic Commands in Tanzu Kubernetes Grid Integrated Edition](#).
   
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.

5. Verify that the TKGI tile is updated to the target version.
After the Upgrade

After you complete the upgrade to TKGI v1.8, complete the following verifications and upgrades:

- Update the TKGI and Kubernetes CLIs
- Upgrade Kubernetes Clusters if Needed
- Verify TKGI Upgrade
- (Optional) Upgrade NSX-T Data Center v2.5 to v3.0
- (Optional) Upgrade to vSphere 7

Update the TKGI and Kubernetes CLIs

Update the TKGI and Kubernetes CLIs on any local machine where you run commands that interact with your upgraded version of TKGI.

To update your CLIs, download and re-install the TKGI and Kubernetes CLI distributions that are provided with TKGI on VMware Tanzu Network.

For more information about installing the CLIs, see the following topics:

- Installing the TKGI CLI
- Installing the Kubernetes CLI

Upgrade Kubernetes Clusters if Needed

If you upgraded TKGI with the Upgrade all clusters errand disabled, the next step is to upgrade the Kubernetes clusters individually using the TKGI CLI.
1. Log in to the TKGI environment using the TKGI CLI.

2. Run the command `tkgi clusters` to list all Kubernetes clusters with their current versions and status:

```
<table>
<thead>
<tr>
<th>PKS Version</th>
<th>Name</th>
<th>k8s Version</th>
<th>Plan Name</th>
<th>UUID</th>
<th>Status</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.7.0</td>
<td>pks-cluster-1-small</td>
<td>1.16.7</td>
<td>small</td>
<td>0bea03c8-aaf7-48c8-b249-814c0bc407b9</td>
<td>succeeded</td>
<td>CREATE</td>
</tr>
<tr>
<td>1.7.0</td>
<td>pks-cluster-2-medium</td>
<td>1.16.7</td>
<td>medium</td>
<td>5d9f4501-70cb-460b-9d78-0afbe074ceb8c</td>
<td>succeeded</td>
<td>CREATE</td>
</tr>
<tr>
<td>1.7.0</td>
<td>pks-cluster-3-large</td>
<td>1.16.7</td>
<td>large</td>
<td>b448117a-bb6f-49de-bc9b-452588bd44ef</td>
<td>succeeded</td>
<td>CREATE</td>
</tr>
</tbody>
</table>
```

3. Update each cluster one-by-one using the command `tkgi upgrade-cluster CLUSTER-NAME`.
   - You do not have to wait for each upgrade to complete before upgrading the next one.
   - The advantage of running each upgrade separately is that it makes troubleshooting easier. BOSH assigns a unique task ID to each cluster upgrade.

4. When the cluster upgrades are complete, run the command `tkgi clusters` and verify that they list the target version:

```
<table>
<thead>
<tr>
<th>PKS Version</th>
<th>Name</th>
<th>k8s Version</th>
<th>Plan Name</th>
<th>UUID</th>
<th>Status</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.8.0</td>
<td>pks-cluster-1-small</td>
<td>1.17.5</td>
<td>small</td>
<td>0bea03c8-aaf7-48c8-b249-814c0bc407b9</td>
<td>succeeded</td>
<td>UPGRADE</td>
</tr>
<tr>
<td>1.8.0</td>
<td>pks-cluster-2-medium</td>
<td>1.17.5</td>
<td>medium</td>
<td>5d9f4501-70cb-460b-9d78-0afbe074ceb8c</td>
<td>succeeded</td>
<td>UPGRADE</td>
</tr>
<tr>
<td>1.8.0</td>
<td>pks-cluster-3-large</td>
<td>1.17.5</td>
<td>large</td>
<td>b448117a-bb6f-49de-bc9b-452588bd44ef</td>
<td>succeeded</td>
<td>UPGRADE</td>
</tr>
</tbody>
</table>
```

### Verify TKGI Upgrade

1. To verify successful upgrade, create a test cluster:

   ```
   pks create-cluster pks-cluster-4-test --external-hostname pks-cluster-test --plan medium --num-nodes 3
   ```

2. Run `tkgi clusters` to verify that the new cluster is created with the appropriate version of TKGI and Kubernetes:

```
<table>
<thead>
<tr>
<th>PKS Version</th>
<th>Name</th>
<th>k8s Version</th>
<th>Plan Name</th>
<th>UUID</th>
<th>Status</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.8.0</td>
<td>pks-cluster-4-test</td>
<td>1.17.5</td>
<td>medium</td>
<td>5d9f4501-70cb-460b-9d78-0afbe074ceb8c</td>
<td>succeeded</td>
<td>CREATE</td>
</tr>
<tr>
<td>1.8.0</td>
<td>pks-cluster-1-small</td>
<td>1.17.5</td>
<td>small</td>
<td>0bea03c8-aaf7-48c8-b249-814c0bc407b9</td>
<td>succeeded</td>
<td>UPGRADE</td>
</tr>
<tr>
<td>1.8.0</td>
<td>pks-cluster-2-medium</td>
<td>1.17.5</td>
<td>medium</td>
<td>5d9f4501-70cb-460b-9d78-0afbe074ceb8c</td>
<td>succeeded</td>
<td>UPGRADE</td>
</tr>
<tr>
<td>1.8.0</td>
<td>pks-cluster-3-large</td>
<td>1.17.5</td>
<td>large</td>
<td>b448117a-bb6f-49de-bc9b-452588bd44ef</td>
<td>succeeded</td>
<td>UPGRADE</td>
</tr>
</tbody>
</table>
```

### (Optional) Upgrade NSX-T Data Center v2.5 to v3.0

**warning:** TKGI supports NSX-T v3.0 as a beta integration. Intermittent upgrade failures and scale problems may occur if you upgrade to NSX-T v3.0. Upgrading your NSX-T environment to v3.0 in a production or large-scale deployment is not recommended until a patch resolving these issues has been released.

If you are using NSX-T v2.5, you can upgrade to NSX-T v3.0. For a list of NSX-T 2.5 and 3.0 versions compatible with TKGI v1.8, see Product Snapshot in Release Notes for TKGI v1.8.

1. Confirm that you have upgraded all TKGI-provisioned Kubernetes clusters to TKGI v1.8 using `Upgrade all clusters errand` in Ops Manager or using the TKGI CLI.

   **Note:** This updates your Kubernetes clusters to the version of Kubernetes and version of NCP that are included with TKGI v1.8.
2. Confirm that your vSphere v6.5, v6.7, or v7.0 installation is on the supported version and patch for NSX-T v3.0.
   - 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. Upload the NSX-T upgrade bundle using the NSX-T Manager and proceed with the upgrade process by following the instructions in the UI.

   **warning**: TKGI v1.8 does not support Converged VDS, which runs one VDS for both vSphere v7 traffic and NSX-T v3.0 traffic, or VMware Cloud Foundation (VCF) v4, which requires Converged VDS. Use N-VDS for NSX-T v3.0 traffic.

For more information, refer to the NSX-T Data Center Upgrade Guide documentation.

4. If you made architectural changes to your NSX-T environment that affect TKGI, such as adding or updating a VIP address, or a load balancer for the NSX-T Management Cluster, update the BOSH Director and TKGI tiles with the new or updated IP addresses:
   - In the BOSH Director tile > vCenter Configuration pane, update NSX Address and NSX CA Cert.
   - In the TKGi tile > Networking pane, update NSX Manager hostname and NSX Manager CA Cert.

  **(Optional) Upgrade to vSphere 7**

  After upgrading TKGI and its Kubernetes clusters to v1.8 and NSX-T to v3.0, you can upgrade vSphere to v7. This upgrade includes upgrading each ESXi host and the vCenter Server Appliance.

  Upgrade vCenter first. Refer to Upgrading the vCenter Server Appliance in the vCenter documentation.

  After you have upgraded vCenter, upgrade each ESXi host. For more information, refer to the Upgrading ESXi hosts.
Troubleshoot the Upgrade

See Verifying Deployment Health for how to verify the health of your TKGI environment and gather information for troubleshooting cluster upgrades.

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

In this topic
About Cluster Upgrades
Set Workload Replicas
Define an Anti-Affinity Rule
Multi-AZ Worker
PersistentVolumes

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how you can maintain workload uptime for Kubernetes clusters deployed with VMware Tanzu Kubernetes Grid Integrated Edition.

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 Tanzu Kubernetes Grid Integrated Edition 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:

```yaml
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:

<table>
<thead>
<tr>
<th>Key-Value Pair</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>spec: replicas: 3</code></td>
<td>Set this value to at least 3 to have at least three instances of your workload running at any time.</td>
</tr>
<tr>
<td><code>app: APP-NAME</code></td>
<td>Use this app name when you define the anti-affinity rule later in the spec.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Key-Value Pair</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>podAntiAffinity: requiredDuringSchedulingIgnoredDuringExecution</td>
<td>- When you set <code>podAntiAffinity</code> to the <code>requiredDuringSchedulingIgnoredDuringExecution</code> 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.</td>
</tr>
<tr>
<td>matchExpressions:</td>
<td>This value matches <code>spec.template.metadata.labels.app</code>.</td>
</tr>
<tr>
<td>- key: &quot;app&quot;</td>
<td></td>
</tr>
<tr>
<td>values:</td>
<td>This value matches the <code>APP-NAME</code> you defined earlier in the spec.</td>
</tr>
<tr>
<td>- APP-NAME</td>
<td></td>
</tr>
</tbody>
</table>

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, VMware 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 Tanzu Kubernetes Grid Integrated Edition, see Configuring and Using PersistentVolumes.

For information about the supported storage topologies for Tanzu Kubernetes Grid Integrated Edition on vSphere, see PersistentVolume Storage Options on vSphere.

Please send any feedback you have to pks-feedback@pivotal.io.
Configuring the Upgrade Pipeline

In this topic

Download the Upgrade Pipeline

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to set up a Concourse pipeline to perform automatic upgrades of a VMware Tanzu Kubernetes Grid Integrated Edition installation.

When you configure the upgrade pipeline, the pipeline upgrades your installation when a new Tanzu Kubernetes Grid Integrated Edition release becomes available on VMware Tanzu 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 VMware Tanzu Network.

2. Navigate to the Platform Automation Tools product page to download the upgrade-tile pipeline.

   Note: If you cannot access Platform Automation Tools on VMware Tanzu Network, contact 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 Tanzu Kubernetes Grid Integrated Edition

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This section describes how to manage VMware Tanzu Kubernetes Grid Integrated Edition.

See the following topics:

- Monitor and Manage Tanzu Kubernetes Grid Integrated Edition in the Management Console
- Managing Tanzu Kubernetes Grid Integrated Edition Users
- Managing Kubernetes Cluster Options
- Adding Infrastructure Password Changes to the Tanzu Kubernetes Grid Integrated Edition Tile
- Shutting Down and Restarting Tanzu Kubernetes Grid Integrated Edition
- Deleting Tanzu Kubernetes Grid Integrated Edition

Please send any feedback you have to pks-feedback@pivotal.io.
Monitor and Manage Tanzu Kubernetes Grid Integrated Edition in the Management Console

In this topic

Obtain General Status Information
Obtain Deployment Metadata
View Component Deployment Status

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

After you have deployed VMware Tanzu Kubernetes Grid Integrated Edition on vSphere, you can use VMware Tanzu Kubernetes Grid Integrated Edition Management Console to perform the following operations:

- View the overall status of your deployment.
- View the deployment metadata and status of each of the components of your deployment.
- 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 Tanzu Kubernetes Grid Integrated Edition Deployment.
- Upgrade your deployment to a new version. For information about upgrading deployments, see Upgrade Tanzu Kubernetes Grid Integrated Edition Management Console.
- Patch the individual components of your deployment. For information about patching components, see Patch Tanzu Kubernetes Grid Integrated Edition Management Console Components.
- Delete Your Tanzu Kubernetes Grid Integrated Edition Deployment

For information about how to deploy the management console and install Tanzu Kubernetes Grid Integrated Edition, see Install on vSphere with the Management Console.

Obtain General Status Information

1. Go to the TKG Integrated Edition view of the management console.
2. Select the Summary tab for your Tanzu Kubernetes Grid Integrated Edition 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

Obtain Deployment Metadata

The deployment metadata provides credentials, certificates, and other metadata about your Tanzu Kubernetes Grid Integrated Edition deployment.

1. Expand **Configuration** and select **Deployment Metadata**.
2. Select the clipboard icon at the end of each row to copy the relevant value.
   For example, copy the Ops Manager password so that you can log in to the instance of Ops Manager that is running in your deployment.

View Component Deployment Status

You can see the status of the individual components of your deployment.

1. Expand Configuration and go to the TKGI Configuration view of the management console.

2. Select Deployment Status to see the status of the components.
3. Click the Download Logs button to download the log bundle for your Tanzu Kubernetes Grid Integrated Edition deployment.

Please send any feedback you have to pks-feedback@pivotal.io.
Reconfigure Your Tanzu Kubernetes Grid Integrated Edition Deployment

In this topic
Reconfigure Your Tanzu Kubernetes Grid Integrated Edition Deployment in the Wizard
Reconfigure Your Tanzu Kubernetes Grid Integrated Edition Deployment by Importing a YAML File
Which Options Can I Reconfigure?

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:


Reconfigure Your Tanzu Kubernetes Grid Integrated Edition Deployment in the Wizard

The procedure to reconfigure Tanzu Kubernetes Grid Integrated Edition deployments in the wizard is as follows.

1. Expand Configuration and go to the TKGI Configuration view of the management console.

2. Select Wizard to be taken to the configuration wizard for Tanzu Kubernetes Grid Integrated Edition.

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 Tanzu Kubernetes Grid Integrated Edition by Using the Configuration Wizard for 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 Tanzu Kubernetes Grid Integrated Edition.

Reconfigure Your Tanzu Kubernetes Grid Integrated Edition Deployment by Importing a YAML File

The procedure to reconfigure Tanzu Kubernetes Grid Integrated Edition deployments by importing an updated YAML file is as follows.

1. Expand Configuration and go to the TKGI Configuration view of the management console.

2. Select Wizard to be taken to the configuration wizard for Tanzu Kubernetes Grid Integrated Edition.

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 Tanzu Kubernetes Grid Integrated Edition 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 the 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition.

### Which Options Can I Reconfigure?

After the initial deployment of Tanzu Kubernetes Grid Integrated Edition, 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 Tanzu Kubernetes Grid Integrated Edition deployment.

<table>
<thead>
<tr>
<th>Section</th>
<th>Cannot Modify</th>
<th>Can Modify</th>
</tr>
</thead>
<tbody>
<tr>
<td>vCenter Account</td>
<td>vCenter Server instance, Datacenter</td>
<td>vCenter Server address, if it changes Username Password</td>
</tr>
<tr>
<td>Networking: Container Networking Interface</td>
<td>You cannot change between NSX-T Data Center (Automated NAT Deployment), NSX-T Data Center (Bring Your Own Topology), and Flannel.</td>
<td>None</td>
</tr>
<tr>
<td>Networking: NSX Manager Details</td>
<td>You cannot change the deployment to a different NSX Manager instance</td>
<td>NSX Manager address, if it changes Username Password</td>
</tr>
<tr>
<td>Networking: NSX-T Data Center (Automated NAT Deployment)</td>
<td>Tier0 Active Active Mode, Deployment CIDR, Deployment Network Reserved IP Range, Usable Range of Floating IPs</td>
<td>All other options</td>
</tr>
<tr>
<td>Networking: NSX-T Data Center (Bring Your Own Topology)</td>
<td>Network for TKGI Management Plane, Floating IP Pool ID, Deployment Network Reserved IP Range, NAT mode</td>
<td>All other options</td>
</tr>
<tr>
<td>Networking: Flannel</td>
<td>Deployment Network Reserved IP Range</td>
<td>Service Network Reserved IP Range</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Identity</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Availability Zones</td>
<td>Management availability zone</td>
<td></td>
</tr>
<tr>
<td>Plans</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Integrations</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Harbor</td>
<td>Harbor FQDN</td>
<td>Authentication mode</td>
</tr>
<tr>
<td>CEIP and Telemetry</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>

Please send any feedback you have to pks-feedback@pivotal.io.
Patch Tanzu Kubernetes Grid Integrated Edition Management Console Components

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

You can use VMware Tanzu Kubernetes Grid Integrated Edition Management Console on vSphere to update some of the components of your deployment individually when a new minor version of those components is available.

1. In Tanzu Kubernetes Grid Integrated Edition Management Console, go to Configuration > TKGI 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.

<table>
<thead>
<tr>
<th>COMPONENT</th>
<th>CURRENT VERSION</th>
<th>NEW VERSION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>gear2</td>
<td>1.6.0-rev2</td>
<td>1.6.0-rev3</td>
<td>Enterprise PKS Management Console Patch up-to-date.</td>
</tr>
</tbody>
</table>
3. When the patch imports are complete, select **Install Patch** to patch a component.

![VMware Enterprise PKS Component Patches](image)

If you are patching Tanzu Kubernetes Grid Integrated Edition Management Console itself, you will be automatically logged out during the patching process.

4. Log back in to the management console.

5. Click the help icon ? in the top banner and select **About** to check that the version of Tanzu Kubernetes Grid Integrated Edition Management Console has been updated.

6. Click **TKG Integrated Edition** to check the versions of the installed components.

Please send any feedback you have to pks-feedback@pivotal.io.
Delete Your Tanzu Kubernetes Grid Integrated Edition Deployment

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

**Page last updated:**

If you no longer require an Tanzu Kubernetes Grid Integrated Edition deployment on vSphere, you can use VMware Tanzu Kubernetes Grid Integrated Edition Management Console to delete it. The deletion process removes all objects from the vSphere inventory, and cleans up the objects in the Tanzu Kubernetes Grid Integrated Edition Management Console VM related to your deployment.


**Note:** You must use the TKGI CLI to delete any existing clusters and nodes before you can use the management console to delete your Tanzu Kubernetes Grid Integrated Edition deployment.

1. Go to the **TKG Integrated Edition** view of the management console.

2. Click the **Action** drop-down menu and select **Delete Tanzu Kubernetes Grid Integrated Edition Deployment**

3. Click **Delete** to confirm the deletion of the deployment.

Please send any feedback you have to pks-feedback@pivotal.io.
Managing Tanzu Kubernetes Grid Integrated Edition Users

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

**Page last updated:**

This section describes how to use either VMware Tanzu Kubernetes Grid Integrated Edition Management Console or Ops Manager to manage users.

See the following topics:

- Identity Management in the Management Console
- Identity Management in Ops Manager

Please send any feedback you have to pks-feedback@pivotal.io.
Identity Management in the Management Console

In this topic
Add Individual Users
Add User Groups
Remove Individual Users
Remove User Groups
Next Steps

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

On vSphere, you can add individual users or user groups to Tanzu Kubernetes Grid Integrated Edition Management Console. You can assign 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 roles that you can assign, see UAA Scopes for Tanzu Kubernetes Grid Integrated Edition Users

For information about the tasks that Cluster Managers can perform, see Tanzu Kubernetes Grid Integrated Edition Architecture
The TKGI Administrator role allows users to manage the Tanzu Kubernetes Grid Integrated Edition infrastructure.

Add Individual Users

The procedure to add individual users to Tanzu Kubernetes Grid Integrated Edition Management Console is as follows.

Note: This release of Tanzu Kubernetes Grid Integrated Edition 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.
3. Click **Add User**.

4. Enter a user name and enter and verify a password to create a new user account.

5. Assign a role to the user.
   - **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.

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 Tanzu Kubernetes Grid Integrated Edition Management Console is as follows.

1. Go to the Identity Management view of the management console.

2. Select the Groups tab.

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.
   - `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.
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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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**.

### Next Steps

- **Assign Resource Quotas to Users**
- **Working with Network Profiles**
Create Clusters in the Management Console

Please send any feedback you have to pks-feedback@pivotal.io.
Managing Tanzu Kubernetes Grid Integrated Edition Users in Ops Manager

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This section describes how to connect UAA to external user stores when using Ops Manager to configure VMware Tanzu Kubernetes Grid Integrated Edition and how to manage users with UAA.

See the following topics:

- Connecting Tanzu Kubernetes Grid Integrated Edition to an LDAP Server
- Configuring Okta as a SAML Identity Provider
- Configuring Azure Active Directory as a SAML Identity Provider
- Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider
- Managing Tanzu Kubernetes Grid Integrated Edition Users with UAA
- UAA Scopes for Tanzu Kubernetes Grid Integrated Edition Users
- OIDC Provider for Kubernetes Clusters

Please send any feedback you have to pks-feedback@pivotal.io.
Connecting Tanzu Kubernetes Grid Integrated Edition to an LDAP Server

In this topic
Overview
Integrate UAA with an LDAP Server
Complete Your Tile Configuration
Next Steps

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to connect VMware Tanzu Kubernetes Grid Integrated Edition to an external LDAP server.

Overview

User Account and Authentication (UAA), the identity management service for Tanzu Kubernetes Grid Integrated Edition, 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 Tanzu Kubernetes Grid Integrated Edition tile > UAA.

If you want to connect Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition > 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:** VMware recommends 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 Tanzu Kubernetes Grid Integrated Edition, 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. **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 Tanzu Kubernetes Grid Integrated Edition or a Kubernetes group.

   ![Note]
   - To map the groups under this search base to roles in Tanzu Kubernetes Grid Integrated Edition, follow the instructions in [Grant Tanzu Kubernetes Grid Integrated Edition Access to an External LDAP Group](#).

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

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

9. **First Name Attribute**, enter the attribute name in your LDAP directory that contains user first names. For example, `cn`.

10. **Last Name Attribute**, enter the attribute name in your LDAP directory that contains user last names. For example, `sn`.

11. **Email Attribute**, enter the attribute name in your LDAP directory that contains user email addresses. For example, `mail`.

12. **Email Domain(s)**, enter a comma-separated list of the email domains for external users who can receive invitations to Apps Manager.

13. **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.

14. **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.
15. Click **Save**.

### Complete Your Tile Configuration

- If you do not need to configure any other settings in the Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition tile, return to the **Installing Tanzu Kubernetes Grid Integrated Edition** topic for your IaaS and follow the instructions for the pane you want to configure:
  - Installing Tanzu Kubernetes Grid Integrated Edition on vSphere
  - Installing Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T
  - Installing Tanzu Kubernetes Grid Integrated Edition on GCP
  - Installing Tanzu Kubernetes Grid Integrated Edition on AWS
  - Installing Tanzu Kubernetes Grid Integrated Edition on Azure

### Next Steps

For information about creating Tanzu Kubernetes Grid Integrated Edition roles and managing Kubernetes cluster access, see:

- Setting Up Tanzu Kubernetes Grid Integrated Edition Admin Users for your IaaS
- Managing Cluster Access and Permissions

Please send any feedback you have to pks-feedback@pivotal.io.
Configuring Okta as a SAML Identity Provider

In this topic

Prerequisites
Configure SAML in Okta

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic explains how to configure single sign-on (SSO) between Okta and VMware Tanzu Kubernetes Grid Integrated Edition.

Prerequisites

To configure Okta to designate Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition, 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:

<table>
<thead>
<tr>
<th>Field</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single sign on URL</td>
<td>Enter <code>https://TKGI-API:8443/saml/SSO/alias/TKGI-API:8443</code>. For example:</td>
</tr>
<tr>
<td></td>
<td><code>https://api.tkgi.example.com:8443/saml/SSO/alias/api.tkgi.example.com:8443</code></td>
</tr>
</tbody>
</table>
Use this for Recipient URL and Destination URL

Ensure this checkbox is enabled.

<table>
<thead>
<tr>
<th>Audience URI (SP Entity ID)</th>
<th>Enter <a href="https://api.tkgi.example.com:8443">TKGI-API:8443</a>. For example: <a href="https://api.tkgi.example.com:8443">api.tkgi.example.com:8443</a>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name ID format</td>
<td>Select a name identifier format. By default, Tanzu Kubernetes Grid Integrated Edition uses <a href="https://api.tkgi.example.com:8443">EmailAddress</a>.</td>
</tr>
<tr>
<td>Attribute Statements</td>
<td>Enter any attribute statements that you want to map to users in the ID token. In Tanzu Kubernetes Grid Integrated Edition you can define first name, last name, and email attributes.</td>
</tr>
<tr>
<td>Group Attribute Statements</td>
<td>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 Tanzu Kubernetes Grid Integrated Edition.</td>
</tr>
</tbody>
</table>

**Note:** VMware 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](https://docs.okta.com) 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**.

Please send any feedback you have to pks-feedback@pivotal.io.
Configuring Azure Active Directory as a SAML Identity Provider

In this topic
- Prerequisites
- Configure SAML in Azure AD

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic explains how to configure single sign-on (SSO) between Azure Active Directory (Azure AD) and VMware Tanzu Kubernetes Grid Integrated Edition.

Prerequisites

To configure Azure AD to designate Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition, 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:
<table>
<thead>
<tr>
<th>Field</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier (Entity ID)</td>
<td>Enter <code>TKGI-API:8443</code>.</td>
</tr>
<tr>
<td></td>
<td>For example: api.tkgi.example.com:8443</td>
</tr>
<tr>
<td></td>
<td>For example: <a href="https://api.tkgi.example.com:8443/saml/SSO/alias/api.tkgi.example.com:8443">https://api.tkgi.example.com:8443/saml/SSO/alias/api.tkgi.example.com:8443</a></td>
</tr>
<tr>
<td></td>
<td>For example: <a href="https://api.tkgi.example.com:8443/saml/SSO/alias/api.tkgi.example.com:8443">https://api.tkgi.example.com:8443/saml/SSO/alias/api.tkgi.example.com:8443</a></td>
</tr>
</tbody>
</table>

**Note:** VMware 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**.

![User Attributes & Claims](image)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Claim Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>GivenName</td>
<td>user.givenname</td>
</tr>
<tr>
<td>Surname</td>
<td>user.surname</td>
</tr>
<tr>
<td>EmailAddress</td>
<td>user.mail</td>
</tr>
<tr>
<td>Name</td>
<td>user.userprincipalname</td>
</tr>
<tr>
<td>Unique User Identifier</td>
<td>user.userprincipalname</td>
</tr>
</tbody>
</table>

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, Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition tile. For more information, see Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider.
<table>
<thead>
<tr>
<th>SAML Signing Certificate</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Status</strong></td>
<td>Active</td>
</tr>
<tr>
<td><strong>Thumbprint</strong></td>
<td>4180083B5144B1426B29D022B17BE35C443D3AA38</td>
</tr>
<tr>
<td><strong>Expiration</strong></td>
<td>7/22/2022, 3:16:44 PM</td>
</tr>
<tr>
<td><strong>Notification Email</strong></td>
<td></td>
</tr>
<tr>
<td><strong>App Federation Metadata Url</strong></td>
<td><a href="https://login.microsoftonline.com/cSe8f8b0-ef">https://login.microsoftonline.com/cSe8f8b0-ef</a>...</td>
</tr>
<tr>
<td><strong>Certificate (Base64)</strong></td>
<td>Download</td>
</tr>
<tr>
<td><strong>Certificate (Raw)</strong></td>
<td>Download</td>
</tr>
<tr>
<td><strong>Federation Metadata XML</strong></td>
<td>Download</td>
</tr>
</tbody>
</table>

Please send any feedback you have to pks-feedback@pivotal.io.
Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider

In this topic
Overview
Prerequisites
Integrate UAA with a SAML IdP
Complete Your Tile Configuration
Next Steps

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to connect VMware Tanzu Kubernetes Grid Integrated Edition to a SAML identity provider (IdP).

Overview

User Account and Authentication (UAA), the identity management service for Tanzu Kubernetes Grid Integrated Edition, 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 Tanzu Kubernetes Grid Integrated Edition tile > UAA.

If you want to connect Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition tile, you must configure your IdP to designate Tanzu Kubernetes Grid Integrated Edition as a service provider (SP).

See the table below for information about industry-standard SAML IdPs and how to integrate them with Tanzu Kubernetes Grid Integrated Edition:

<table>
<thead>
<tr>
<th>Solution Name</th>
<th>Integration Guide</th>
</tr>
</thead>
<tbody>
<tr>
<td>Okta Single Sign-On</td>
<td>Configuring Okta as a SAML Identity Provider</td>
</tr>
<tr>
<td>Azure Active Directory</td>
<td>Configuring Azure Active Directory as a SAML Identity Provider</td>
</tr>
</tbody>
</table>

Integrate UAA with a SAML IdP

To integrate UAA with a SAML IdP:
1. In **Tanzu Kubernetes Grid Integrated Edition > UAA**, under **Configure your UAA user account store with either internal or external authentication mechanisms**, select **SAML Identity Provider**.

   ![Configuration form]

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 Ops Manager login page, which you can access at `https://TKGI-API:8443/login`.

4. Retrieve the metadata from your IdP. You recorded your IdP metadata when you configured your IdP to designate Tanzu Kubernetes Grid Integrated Edition 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:** VMware 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 Tanzu Kubernetes Grid Integrated Edition. 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 TKGI, see **Grant Tanzu Kubernetes Grid Integrated Edition Access to an External SAML Group** in **Managing Tanzu Kubernetes Grid Integrated Edition Users with UAA**.

10. By default, all SAML authentication requests from Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition tile, return to the **Installing Tanzu Kubernetes Grid Integrated Edition** topic for your IaaS and follow the instructions for the pane you want to configure:
  - Installing Tanzu Kubernetes Grid Integrated Edition on vSphere
  - Installing Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T
  - Installing Tanzu Kubernetes Grid Integrated Edition on GCP
  - Installing Tanzu Kubernetes Grid Integrated Edition on AWS
  - Installing Tanzu Kubernetes Grid Integrated Edition on Azure

### Next Steps

For information about creating Tanzu Kubernetes Grid Integrated Edition roles and managing Kubernetes cluster access, see:

- **Setting Up Tanzu Kubernetes Grid Integrated Edition Admin Users** for your IaaS
- **Managing Cluster Access and Permissions**

Please send any feedback you have to **pks-feedback@pivotal.io**.
Managing Tanzu Kubernetes Grid Integrated Edition Users with UAA

In this topic

Overview

UAA Scopes for Tanzu Kubernetes Grid Integrated Edition Users

Prerequisites

Log In as a UAA Admin

Grant Tanzu Kubernetes Grid Integrated Edition Access to an Individual User

Grant Tanzu Kubernetes Grid Integrated Edition Access to an External Group

Grant Tanzu Kubernetes Grid Integrated Edition Access to an External LDAP Group

Grant Tanzu Kubernetes Grid Integrated Edition Access to an External SAML Group

Grant Tanzu Kubernetes Grid Integrated Edition Access to a Client

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to manage users in VMware Tanzu Kubernetes Grid Integrated Edition with User Account and Authentication (UAA).

Overview

UAA is the identity management service for Tanzu Kubernetes Grid Integrated Edition. Tanzu Kubernetes Grid Integrated Edition includes a UAA server, which is hosted on the TKGI API 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 Tanzu Kubernetes Grid Integrated Edition Users

By assigning UAA scopes, you grant users the ability to create, manage, and audit Kubernetes clusters in Tanzu Kubernetes Grid Integrated Edition.

A UAA admin user can assign the following UAA scopes to Tanzu Kubernetes Grid Integrated Edition users:

- `pks.clusters.admin`: Accounts with this scope can create and access all clusters.
- `pks.clusters.manage`: Accounts with this scope can create and access their own 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 Tanzu Kubernetes Grid Integrated Edition, see UAA Scopes.

Prerequisites

Note:

As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.
Before managing users for Tanzu Kubernetes Grid Integrated Edition, you must connect to the TKGI API VM. To connect to the TKGI API VM, you need one of the following:

- SSH access to the Ops Manager VM
- A machine that can connect to your TKGI API VM

For instructions on how to connect to the TKGI control plane, see Connect to the TKGI API VM for your IaaS.

Log In as a UAA Admin

Before creating TKGI 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 Tanzu Kubernetes Grid Integrated Edition 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://TKGI-API:8443 --ca-cert CERTIFICATE-PATH
   ```

   Where:
   - **TKGI-API** is the domain name of your TKGI API server. You entered this domain name in the Tanzu Kubernetes Grid Integrated Edition tile > TKGI API > API Hostname (FQDN).
   - **CERTIFICATE-PATH** is the path to your Ops Manager root CA certificate. Provide this certificate to validate the TKGI 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.tkgi.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 Tanzu Kubernetes Grid Integrated Edition Access to an Individual User
To create a new UAA user with Tanzu Kubernetes Grid Integrated Edition 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:

   ```bash
   uaac user add USERNAME --emails USER-EMAIL -p USER-PASSWORD
   ```

   For example:

   ```bash
   $ uaac user add cody --emails cody@example.com -p password
   ```

   **Note:** If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 TKGI tile, select Settings > UAA > SAML, and locate the Provider Name.

   For information about configuring SAML, see Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider.

3. Assign a TKGI cluster scope to the new user by running the following command:

   ```bash
   uaac member add UAA-SCOPE USERNAME
   ```

   Where:

   - `UAA-SCOPE` is one of the UAA scopes described in UAA Scopes for Tanzu Kubernetes Grid Integrated Edition Users
   - `USERNAME` is the user that you created in the previous step.

   For example:

   ```bash
   $ 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 Tanzu Kubernetes Grid Integrated Edition Access to an External Group

Connecting Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition access to an external identity provider group, do one the following procedures:

- Grant Tanzu Kubernetes Grid Integrated Edition Access to an External LDAP Group
Grant Tanzu Kubernetes Grid Integrated Edition Access to an External SAML Group

Grant Tanzu Kubernetes Grid Integrated Edition Access to an External LDAP Group

To grant Tanzu Kubernetes Grid Integrated Edition 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 TKGI 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition Access to an External SAML Group

To grant Tanzu Kubernetes Grid Integrated Edition 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 TKGI 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 Tanzu Kubernetes Grid Integrated Edition 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 TKGI tile, select Settings > UAA > SAML, and locate the Provider Name.

   For example:
   ```
   $ uaac group map --name pks.clusters.manage tkgi-devs --origin my-sso.example.com
   ```

Grant Tanzu Kubernetes Grid Integrated Edition Access to a Client

To grant Tanzu Kubernetes Grid Integrated Edition 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:
uac 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 Tanzu Kubernetes Grid Integrated Edition Users, separated by a comma. For example:

  ```bash
  $ uac 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.
UAA Scopes for Tanzu Kubernetes Grid Integrated Edition Users

In this topic
Overview
UAA Scopes

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Overview

UAA is the identity management service for Tanzu Kubernetes Grid Integrated Edition.

By assigning UAA scopes, you grant users the ability to create, manage, and audit Kubernetes clusters in Tanzu Kubernetes Grid Integrated Edition.

A UAA admin user can assign the following UAA scopes to Tanzu Kubernetes Grid Integrated Edition users:

- **pks.clusters.admin**: Accounts with this scope can create and access all clusters.
- **pks.clusters.manage**: Accounts with this scope can create and access their own 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 Tanzu Kubernetes Grid Integrated Edition users a set of permissions for creating, managing, and auditing Tanzu Kubernetes Grid Integrated Edition-provisioned Kubernetes clusters. For information about the permissions, see the table below.

<table>
<thead>
<tr>
<th>Operation</th>
<th><strong>pks.clusters.admin</strong></th>
<th><strong>pks.clusters.manage</strong></th>
<th><strong>pks.clusters.admin.read</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Create, update, resize, and delete a cluster</td>
<td>Yes. Can create, modify, and delete all clusters.</td>
<td>Yes. Can create, modify, and delete only their own clusters.</td>
<td>No. Cannot create, modify, and delete clusters.</td>
</tr>
<tr>
<td>Get cluster credentials</td>
<td>Yes. Can retrieve cluster credentials for all clusters.</td>
<td>Yes. Can retrieve cluster credentials only for their own clusters.</td>
<td>No. Cannot retrieve cluster credentials.</td>
</tr>
<tr>
<td>Upgrade clusters</td>
<td>Yes. Can upgrade all clusters.</td>
<td>Yes. Can upgrade only their own clusters.</td>
<td>No. Cannot upgrade clusters.</td>
</tr>
<tr>
<td>Feature</td>
<td>Can list all clusters</td>
<td>Can list only their own clusters</td>
<td>Can list all clusters</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------</td>
<td>----------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>List clusters</td>
<td>Yes.</td>
<td>Yes.</td>
<td>Yes.</td>
</tr>
<tr>
<td>View cluster details</td>
<td>Yes.</td>
<td>Yes.</td>
<td>Yes.</td>
</tr>
<tr>
<td>Create and delete a compute profile</td>
<td>Yes.</td>
<td>No.</td>
<td>No.</td>
</tr>
<tr>
<td>Create and delete a network profile</td>
<td>Yes.</td>
<td>No.</td>
<td>No.</td>
</tr>
<tr>
<td>Create and delete a Kubernetes profile</td>
<td>Yes.</td>
<td>Yes.</td>
<td>No.</td>
</tr>
<tr>
<td>Create, update, and delete a quota</td>
<td>Yes.</td>
<td>No.</td>
<td>No.</td>
</tr>
</tbody>
</table>


Please send any feedback you have to pks-feedback@pivotal.io.
OIDC Provider for Kubernetes Clusters

In this topic

Overview
UAA as the Default OIDC Provider
Custom OIDC Provider
After You Configure Your OIDC Provider

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes the global default OpenID Connect (OIDC) provider setting that you can use for Kubernetes clusters in VMware Tanzu Kubernetes Grid Integrated Edition and how to override it for individual clusters.

Overview

Configuring an OIDC provider for TKGI-provisioned clusters enables Kubernetes to verify end-user identities based on the authentication performed by UAA or a custom OIDC provider.

You can use the following methods to configure an OIDC provider in Tanzu Kubernetes Grid Integrated Edition:

- Configure UAA as the default OIDC provider in the Tanzu Kubernetes Grid Integrated Edition tile > UAA. For more information, see UAA as the Default OIDC Provider below.
- Configure a custom OIDC provider by applying a Kubernetes profile to one or more TKGI-provisioned clusters. For more information, see Custom OIDC Provider below.

UAA as the Default OIDC Provider

The Tanzu Kubernetes Grid Integrated Edition tile > UAA > Configure created clusters to use UAA as the OIDC provider is a global setting for TKGI-provisioned clusters, described in the table below:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| Enabled | If you enable UAA as the OIDC provider, Kubernetes verifies end-user identities based on authentication executed by UAA as follows:  
- If you select Internal UAA, Kubernetes authenticates users against the internal UAA authentication mechanism.  
- If you select LDAP Server, Kubernetes authenticates users against the LDAP server.  
- If you select SAML Identity Provider, Kubernetes authenticates users against the SAML identity provider. |
| Disabled | If you do not enable UAA as the OIDC provider, Kubernetes authenticates users against its internal user management system. |

When you enable UAA as your OIDC provider, existing TKGI-provisioned clusters are upgraded to use OIDC. This invalidates your kubeconfig files. You must regenerate the files for all existing clusters.

Copyright © 2020 VMware, Inc. All Rights Reserved.
Custom OIDC Provider

You can configure one or more Kubernetes clusters to use a custom OIDC provider by creating and applying a Kubernetes profile to the clusters. This overrides the global Configure created clusters to use UAA as the OIDC provider setting in the Tanzu Kubernetes Grid Integrated Edition tile > UAA.

For instructions, see Add an OIDC Provider.

After You Configure Your OIDC Provider

If you want to give Kubernetes end users, such as developers, access to TKGI-provisioned clusters after you configure your OIDC provider, you must create Kubernetes role bindings for them.

For instructions, see Managing Cluster Access and Permissions.

Please send any feedback you have to pks-feedback@pivotal.io.
Managing Kubernetes Cluster Options

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

**Page last updated:**

This section describes how an Tanzu Kubernetes Grid Integrated Edition administrator can create and manage options for the Kubernetes clusters that Tanzu Kubernetes Grid Integrated Edition users provision.

See the following topics:

- CPU and Memory Quotas
- Network Profiles (NSX-T Only)
- Compute Profiles and Host Groups (vSphere Only)

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

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This section describes how to define and use resource quotas for Kubernetes clusters in VMware Tanzu Kubernetes Grid Integrated Edition.

See the following topics:

- Assign Resource Quotas to Users in the Management Console
- Managing Resource Usage with Quotas
- Viewing Usage Quotas

Please send any feedback you have to pks-feedback@pivotal.io.
Assign Resource Quotas to Users in the Management Console

After you have added users to your Tanzu Kubernetes Grid Integrated Edition deployment on vSphere and assigned the cluster manager or admin role to them, you can optionally assign resource quotas to them. By setting quotas, you can limit the amount of compute power and memory that those users can consume. You can also set a limit on the number of clusters that they can deploy.

1. Go to the Quotas view of the management console.

2. Click Add Quota and select the users to whom to apply the quota. Only users who have the cluster manager role appear in the list.
3. Enter maximums for the number of virtual CPUs and the amount of memory that the selected users can consume, and the number of clusters they can create.

You can also use the Unlimited toggles to allow the users unrestricted access to resources.

4. Click Save.

To update the resources assigned to a user, or to delete a quota, select the quota and click either Edit or Delete.

Note: Deleting a quota only deletes the quota. It does not delete the user account.

Next Steps
• Working with Network Profiles in the Management Console
• Create Clusters in the Management Console

Please send any feedback you have to pks-feedback@pivotal.io.
Managing Resource Usage with Quotas

Overview

As an Tanzu Kubernetes Grid Integrated Edition administrator, you can set a limit on each user’s total resource allocation within Tanzu Kubernetes Grid Integrated Edition.

You manage resources in Tanzu Kubernetes Grid Integrated Edition by defining quotas for individual users with the TKGI 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 Tanzu Kubernetes Grid Integrated Edition.

To review overall resource usage and for individual users, you access the TKGI 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

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

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.
The curl commands in this topic use an access token environment variable to authenticate into the TKGI API.

1. To export your access token into an environment variable, run the following command:

```
tkgi login -a TKGI-API -u USER-ID -p 'PASSWORD' -k; export YOUR-ACCESS-TOKEN=$(bosh int ~/.pks/creds.yml --path /access_token)
```

Where:

- **TKGI-API** is the FQDN of your TKGI API endpoint. For example, `api.tkgi.example.com`.
- **USER-ID** is your Tanzu Kubernetes Grid Integrated Edition user ID.
- **PASSWORD** is your Tanzu Kubernetes Grid Integrated Edition password.
- **YOUR-ACCESS-TOKEN** is the name of your access token environment variable.

For example:

```
s tkgi login -a tkgi.my.lab -u alana -p 'pswrdabc123....!' -k; export my_token=$(bosh int ~/.pks/creds.yml --path /access_token)
```

**Note:** If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 **TKGI CLI**. For information about configuring SAML, see **Connecting Tanzu Kubernetes Grid Integrated Edition 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 \n-H "Authorization: Bearer $YOUR-ACCESS-TOKEN" \n-H "Content-Type: application/json" \n-d '\n{"owner": "USER-ID", \n"limit": { \n"cpu": MAX-CPU, \n"memory": MAX-MEM, \n"cluster": MAX-CLUSTER \n}},\nhttps://TKGI-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.

TKGI-API is the FQDN of your TKGI API server.

For example:

```
$ user=exampleuser
$ tkgi login -a tkgi.my.lab -u Suser -p 'pswrdabc123...!' -k; export TOKEN=$(bosh int ~/.pks/creds.yml --path /access_token)
$ curl -k -X POST
-H "Authorization: Bearer STOKEN"
-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://TKGI-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.
- **TKGI-API** is the FQDN of your TKGI API server. For example, api.tkgi.example.com.

For example:
Delete a Quota

To delete a specific user's existing quota, run the following command:

```
$ curl -k -X DELETE -H "Authorization: Bearer $YOUR-ACCESS-TOKEN" \
https://TKGI-API:9021/v1/quotas/$USER-ID
```

Where:

- **$YOUR-ACCESS-TOKEN** is your access token environment variable.
- **TKGI-API** is the FQDN of your TKGI API server.
- **$USER-ID** is the user account ID to enforce the quota restriction on.

For example:

```
$ user=exampleuser
$ tkgi login -a tkgi.my.lab -u $user -p 'pswrdabc123...!' -k; export TOKEN=$(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 TKGI 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 $YOUR-ACCESS-TOKEN" \
https://TKGI-API:9021/v1/quotas/$USER-ID
```

Where:
YOUR-ACCESS-TOKEN is your access token environment variable.

TKGI-API is the FQDN of your TKGI API server.

USER-ID is the user account ID to report on.

For example:

```bash
$ user=exampleuser
$ tkgi login -a tkgi.my.lab -u exampleuser -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": "exampleuser",   "limit": {      "cpu": 2,      "memory": 1.0,      "cluster": 6   }}
```

View All Quotas

To list all current resource and cluster quota restrictions, run the following command:

```bash
curl -k -H "Authorization: Bearer $YOUR-ACCESS-TOKEN" 
 https://TKGI-API:9021/v1/quotas
```

Where:

- **YOUR-ACCESS-TOKEN** is your access token environment variable.
- **TKGI-API** is the FQDN of your TKGI API server.

For example:

```bash
$ user=exampleuser
$ tkgi login -a tkgi.my.lab -u exampleuser -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": "exampleuser",   "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 TKGI administrator.

View Usage

The TKGI 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:

```
```

Where:

- **YOUR-ACCESS-TOKEN** is your access token environment variable.
- **TKGI-API** is the FQDN of your TKGI 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:

```
```

Where:

- **YOUR-ACCESS-TOKEN** is your access token environment variable.
- **TKGI-API** is the FQDN of your TKGI API server.

For example:
$ user=exampleuser
$ tkgi login -a tkgi.my.lab -u user -p 'pswrdabc123...!' -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
Overview
Set up Your API Access Token
View Quotas
View Usage
Error Message When You Exceed Cluster Quota

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

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 Tanzu Kubernetes Grid Integrated Edition using the Tanzu Kubernetes Grid Integrated Edition API.

Overview
Your Tanzu Kubernetes Grid Integrated Edition 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 TKGI 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 TKGI API endpoints.

1. To export your access token into an environment variable, run the following command:

```bash
tkgi login -a TKGI-API -u USER-ID -p 'PASSWORD' -k; 
export YOUR-ACCESS-TOKEN=$(bosh int ~/.pks/creds.yml --path /access_token)
```

Where:
- **TKGI-API** is the FQDN of your TKGI API endpoint. For example, `api.tkgi.example.com`.
- **USER-ID** is your Tanzu Kubernetes Grid Integrated Edition user ID.
- **PASSWORD** is your Tanzu Kubernetes Grid Integrated Edition password.
- **YOUR-ACCESS-TOKEN** is the name of your access token environment variable.
For example:

```bash
$ tkgi login -a tkgi.my.lab -u alana -p 'psswrdbabc123...!' -k;
export my_token=$(bosh int ~/.pks/creds.yml --path /access_token)
```

Note: If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 TKGI CLI. For information about configuring SAML, see Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider

### View Quotas

The TKGI 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:

```bash
curl -k -H "Authorization: Bearer $YOUR-ACCESS-TOKEN"
https://TKGI-API:9021/v1/quotas/USER-ID
```

Where:

- **YOUR-ACCESS-TOKEN** is your access token environment variable.
- **TKGI-API** is the FQDN of your TKGI API endpoint. For example, `api.tkgi.example.com`.
- **USER-ID** is your Tanzu Kubernetes Grid Integrated Edition user ID.

For example:

```bash
$ user=exampleuser
$ tkgi login -a tkgi.my.lab -u $user -p 'psswrdbabc123...!' -k; export TOKEN=$(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 TKGI API `usages` endpoint reports your actual resource usage in the JSON format.

To view your current allocated resource usage, run the following command:

```bash
curl -k -H "Authorization: Bearer $YOUR-ACCESS-TOKEN"
https://TKGI-API:9021/v1/usages/USER-ID
```

Where:
• **YOUR-ACCESS-TOKEN** is your access token environment variable.
• **TKGI-API** is the FQDN of your TKGI API endpoint. For example, `api.tkgi.example.com`.
• **USER-ID** is your Tanzu Kubernetes Grid Integrated Edition user ID.

For example:

```
s user=exampleuser
tkgi login -a tkgi.my.lab -u Suser -p 'passwordabc123...!' -k; export TOKEN=$(bosh int ~/.pks/creds.yml --path /access_token)
```

```
[{
  "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 TKGI administrator.
```

Please send any feedback you have to pks-feedback@pivotal.io.
Network Profiles (NSX-T Only)

_aNote:_ As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

**Page last updated:**

This section describes how to define and use network profiles for VMware Tanzu Kubernetes Grid Integrated Edition clusters deployed on NSX-T with vSphere.

See the following topics:

- Working with Network Profiles in the Management Console
- Working with Network Profiles in Ops Manager
- **Network Profile Use Cases**
  - Size a Load Balancer
  - Customize Pod Networks
  - Customize Node Networks
  - Customize Floating IP Pools
  - Configure Bootstrap NSGroups
  - Configure Edge Router Selection
  - Specify Nodes DNS Servers
  - Configure DNS for Pre-Provisioned IPs
  - Configure the TCP Layer 4 Load Balancer
  - Configure the HTTP/S Layer 7 Ingress Controller
  - Define DFW Section Markers
  - Configure NCP Logging
  - Shared and Dedicated Tier-1 Router Topologies

Please send any feedback you have to pks-feedback@pivotal.io.
Creating and Managing Network Profiles in the Management Console

In this topic

Using Network Profiles
Requirements for Network Profiles
Create Cluster with Network Profile
Define Network Profile
Delete Network Profile
Advanced Network Parameters
Container Networks Parameters

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

You can add, view and remove network profiles using the Tanzu Kubernetes Grid Integrated Edition Management Console on vSphere.

Using Network Profiles

Network profiles let you customize the NSX-T infrastructure networking and the runtime NCP networking for Kubernetes clusters provisioned by Tanzu Kubernetes Grid Integrated Edition. For example, using a network profile you can change the size of the control plane load balancer, add an additional subnet for nodes, and enable the use of a third party ingress controller. For a complete list of use cases, see Network Profile Use Cases.

Requirements for Network Profiles

Network profiles are supported in NSX mode only; there is no support for Flannel mode. In addition, only management console root and tkgi.clusters.admin users can create, view, and delete network profiles. Cluster managers can use a network profile when creating a cluster, either using the management console or the TKGI CLI.

Create Cluster with Network Profile

Use the Tanzu Kubernetes Grid Integrated Edition Management Console to create a cluster with network profile.

1. Select the Create Cluster tab.
2. Select the Network Profile to use.
3. Click Show More to view the profile.
Define Network Profile

Use the Tanzu Kubernetes Grid Integrated Edition Management Console to define a network profile.

**NOTE:** You must be at the console home page to view the Network Profiles tab.

1. Select the Network Profiles tab.
2. Click Create Profile.
3. Enter a Name for the profile.
4. Enter a suitable Description for the profile.
5. Optionally you can set up Parameters for Advanced Network or Container Network.
6. Configure the new profile as needed, or use the default values.
7. Click Save.

View a larger version of this image
New Network Profile
Create a new network profile to be used when creating clusters.

Name: my-network-profile

Description: Custom network profile for load balancers

Optional Parameters
- Load Balancer Size: Large
- Pod IP Block IDs: IP Block ip-bk-1
- Pod Subnet Prefix: 16
- Pod Routable: Yes
- Floating Pool IDs: floating-ip-pool
- T0 Router ID: T0 Router 001
- Master VMS NSGroup ID: NS Group name 1
- Node IP Block IDs: IP Block ip-bk-2
- Node Routable: Yes
Delete Network Profile

Use the Tanzu Kubernetes Grid Integrated Edition Management Console to delete network profile.

**NOTE:** You cannot delete a network profile that is in use by a cluster.

1. Select the **Network Profiles** tab.
2. Select the network profile to remove.
3. Click **Delete**.
4. Confirm deletion.
Advanced Network Parameters

The table lists and describes the available network profile options for customizing NSX-T.

<table>
<thead>
<tr>
<th>Profile Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load Balancer Size</td>
<td>Size of the control plane load balancer: Small, Medium, Large.</td>
</tr>
<tr>
<td>Pod IP Block IDs</td>
<td>Array of Pod IP Block UUIDs defined in NSX-T.</td>
</tr>
<tr>
<td>Pod Subnet Prefix</td>
<td>Size of the Pods IP Block subnet.</td>
</tr>
<tr>
<td>Pod Routability</td>
<td>Make routable the custom Pods subnet: Yes or No.</td>
</tr>
<tr>
<td>Floating Pool IDs</td>
<td>Array of floating IP pool UUIDs defined in NSX-T.</td>
</tr>
<tr>
<td>T0 Router ID</td>
<td>Tenant Tier-0 Router UUID defined in NSX-T.</td>
</tr>
<tr>
<td>Master VMs NSGroup IDs</td>
<td>Namespace Group UUID as defined in NSX-T.</td>
</tr>
<tr>
<td>Node IP Block IDs</td>
<td>Array of Node IP Block UUIDs defined in NSX-T.</td>
</tr>
<tr>
<td>Node Routable</td>
<td>Make routable the custom Node subnet: Yes or No.</td>
</tr>
<tr>
<td>Node Subnet Prefix</td>
<td>Size of the Node IP Block subnet.</td>
</tr>
<tr>
<td>Nodes DNS</td>
<td>Array of DNS server IP addresses for lookup of Kubernetes nodes and pods.</td>
</tr>
<tr>
<td>DNS Lookup Mode</td>
<td>DNS lookup for the API LB (API) and ingress controller (API_INGRESS).</td>
</tr>
<tr>
<td>Ingress Prefix</td>
<td>Ingress controller hostname prefix for DNS lookup.</td>
</tr>
<tr>
<td>Single Tier Topology</td>
<td>Use a single Tier-1 Router per cluster: Yes or No.</td>
</tr>
<tr>
<td>Infrastructure Networks</td>
<td>Array of IP addresses and subnets for use with a single tier topology in a multi-T0 environment.</td>
</tr>
<tr>
<td>Custom Infrastructure Networks</td>
<td>Comma-separated array of custom IP addresses or network CIDRs to be used for Infrastructure Networks.</td>
</tr>
</tbody>
</table>
# Container Networks Parameters

The table lists and describes the available network profile options for customizing NCP.

<table>
<thead>
<tr>
<th>Profile Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use NSX-T L4 Virtual Server for K8s Load Balancer</td>
<td>Use NSX-T layer 4 virtual server for each Kubernetes service of type LoadBalancer: Yes or No.</td>
</tr>
<tr>
<td>Use NSX-T L7 Virtual Server as the Ingress Controller for K8s Cluster</td>
<td>Use NSX-T layer 7 virtual server as the ingress controller for the Kubernetes cluster: Yes or No.</td>
</tr>
<tr>
<td>Use Same Source IP for Calling Clients</td>
<td>Use the same source IP for calling clients: Insert or Replace.</td>
</tr>
<tr>
<td>Ingress controller IP address</td>
<td>IP address to use for the ingress controller.</td>
</tr>
<tr>
<td>NCP Log Level</td>
<td>Configure NCP log levels: INFO, WARNING, DEBUG, ERROR, CRITICAL.</td>
</tr>
<tr>
<td>Log Dropped Firewall Traffic</td>
<td>Log dropped firewall traffic: Yes or No.</td>
</tr>
<tr>
<td>Ingress Persistence Type</td>
<td>Specify the ingress persistence type: none, cookie, source_ip.</td>
</tr>
<tr>
<td>Persistence Timeout Interval in Seconds</td>
<td>Persistence timeout interval in seconds.</td>
</tr>
<tr>
<td>Maximum Number of L4 Servers Per Cluster</td>
<td>Limit the number of L4 virtual servers per cluster.</td>
</tr>
<tr>
<td>L4 Persistence Type</td>
<td>Connection stickiness based on source_ip.</td>
</tr>
<tr>
<td>L4 Load Balancer Behavior</td>
<td>Customize the layer 4 load balancer behavior: round_robin, least_connection, ip_hash, weighted_round_robin</td>
</tr>
<tr>
<td>Top Section-id for Distributed Firewall Section</td>
<td>UUID of the top section-id for the distributed firewall (DFW) section as defined in NSX-T.</td>
</tr>
<tr>
<td>Bottom Section-id for Distributed Firewall Section</td>
<td>UUID of the bottom section-id for the distributed firewall (DFW) section as defined in NSX-T.</td>
</tr>
</tbody>
</table>

Please send any feedback you have to pks-feedback@pivotal.io.
Creating and Managing Network Profiles with the CLI

In this topic
Create a Network Profile
   Network Profile Format
   Network Profile Parameters
The create-network-profile Command
Manage Network Profiles
   Delete a Network Profile
   Cluster Manager Operations
   Limitation: Pod IP Block Changes
Network Profile Use Cases

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:


For more information on how to use network profiles, see Using Network Profiles.

Create a Network Profile

To create a network profile in Tanzu Kubernetes Grid Integrated Edition, you:

1. Define a network profile in a JSON configuration file, following the Network Profile Format below.

2. Use the TKGI CLI to define the network profile within Tanzu Kubernetes Grid Integrated Edition, as described in The create-network-profile Command, below.

Network Profile Format

To create a network profile, you must first define it as a JSON file that specifies network parameters, listed in Network Profile Parameters below.

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.
Network Profile Parameters

The network profile JSON can include the following top-level parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the network profile.</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>Description of the network profile.</td>
</tr>
</tbody>
</table>

**WARNING:** None of the parameters defined under `cni_configurations` can be updated.
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameters</td>
<td>Map</td>
<td>One or more name-value pairs.</td>
</tr>
<tr>
<td>lb_size</td>
<td>String</td>
<td>Size of the NSX-T load balancer service: small (default), medium, and large.</td>
</tr>
<tr>
<td>pod_ip_block_ids</td>
<td>String</td>
<td>Array of Pod IP Block UUIDs.</td>
</tr>
<tr>
<td>pod_subnet_prefix</td>
<td>Integer</td>
<td>Size of the Pods IP Block subnet.</td>
</tr>
<tr>
<td>pod_routable</td>
<td>Boolean</td>
<td>Make routable the Pods subnet. Default is false.</td>
</tr>
<tr>
<td>fip_pool_ids</td>
<td>String</td>
<td>Array of floating IP pool UUIDs defined in NSX-T.</td>
</tr>
<tr>
<td>t0_router_id</td>
<td>String</td>
<td>Tenant Tier-0 Router UUID defined in NSX-T.</td>
</tr>
<tr>
<td>master_vms_nsgroup_id</td>
<td>String</td>
<td>Namespace Group UUID as defined in NSX-T.</td>
</tr>
<tr>
<td>nodes_dns</td>
<td>String</td>
<td>Array (up to 3) of DNS server IP addresses for lookup of Kubernetes nodes and pods.</td>
</tr>
<tr>
<td>dns_lookup_mode</td>
<td>String</td>
<td>DNS lookup mode for Kubernetes API load balancer (API) and ingress controller (API_INGRESS).</td>
</tr>
<tr>
<td>ingress_prefix</td>
<td>String</td>
<td>Ingress controller hostname prefix for DNS lookup.</td>
</tr>
<tr>
<td>singleTierTopology</td>
<td>Boolean</td>
<td>Use a single Tier-1 Router per cluster (shared). Default is true.</td>
</tr>
<tr>
<td>infrastructure_networks</td>
<td>String</td>
<td>Array of IP addresses and subnets for Node Networks for use with a Shared Tier-1 topology in a Multi-Tier-0 environment.</td>
</tr>
<tr>
<td>cni_configurations</td>
<td>Map</td>
<td>Map containing key-value pairs for configuring NCP (see table below).</td>
</tr>
</tbody>
</table>

The network profile JSON accepts the following parameters for `cni_configurations`:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>String</td>
<td>Only the constant nsxt is accepted.</td>
</tr>
<tr>
<td>parameters</td>
<td>Map</td>
<td>One or more name-value pairs for NCP settings.</td>
</tr>
<tr>
<td>x_forwarded_for</td>
<td>String</td>
<td>Use the same source IP for calling clients. Accepts &quot;insert&quot; and &quot;replace&quot;.</td>
</tr>
<tr>
<td>nsxt_lb</td>
<td>Boolean</td>
<td>Use NSX-T layer 4 virtual server for each Kubernetes service of type LoadBalancer. Default is true.</td>
</tr>
<tr>
<td>nsxt_ingress_controller</td>
<td>Boolean</td>
<td>Use NSX-T layer 7 virtual server as the ingress controller for the Kubernetes cluster. Default is true.</td>
</tr>
<tr>
<td>ingress_ip</td>
<td>String</td>
<td>IP address to use for the ingress controller.</td>
</tr>
<tr>
<td>log_settings</td>
<td>Map</td>
<td>Parameters for configuring NCP logging.</td>
</tr>
<tr>
<td>log_level</td>
<td>String</td>
<td>Accepted values: &quot;INFO&quot;, &quot;WARNING&quot;, &quot;DEBUG&quot;, &quot;ERROR&quot;, and &quot;CRITICAL&quot;.</td>
</tr>
<tr>
<td>log_dropped_traffic</td>
<td>Boolean</td>
<td>Log dropped firewall traffic. Default is false.</td>
</tr>
<tr>
<td>ingress_persistence_settings</td>
<td>String</td>
<td>Parameters for customizing Layer 7 persistence.</td>
</tr>
<tr>
<td>persistence_type</td>
<td>String</td>
<td>Specify the ingress persistence type: none, cookie, or source_ip.</td>
</tr>
<tr>
<td>persistence_timeout</td>
<td>Integer</td>
<td>Persistence timeout interval in seconds.</td>
</tr>
<tr>
<td>max_l4_lb_service</td>
<td>Integer</td>
<td>Limit the maximum number of layer 4 virtual servers per cluster. Minimum is 1.</td>
</tr>
<tr>
<td>l4_persistence_type</td>
<td>String</td>
<td>Connection stickiness based on source_ip.</td>
</tr>
</tbody>
</table>
Layer 4 load balancer behavior: `round_robin` (default), `least_connection`, `ip_hash`, `weighted_round_robin`.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>l4_lb_algorithm</code></td>
<td>String</td>
<td>UUID of the top <code>section-id</code> for the distributed firewall (DFW) section as defined in NSX-T.</td>
</tr>
<tr>
<td><code>top_firewall_section_marker</code></td>
<td>String</td>
<td>UUID of the bottom <code>section-id</code> for the distributed firewall (DFW) section as defined in NSX-T.</td>
</tr>
</tbody>
</table>

The create-network-profile Command

After a network profile is defined in a JSON file as described in Network Profile Format, an Tanzu Kubernetes Grid Integrated Edition administrator can create the network profile by running the following TKGI CLI command:

```
tkgi 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:

```
$ tkgi 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 TKGI administrator.
```

After an administrator creates a network profile, cluster managers can create clusters with it or assign it to existing clusters. For more information, see the Using Network Profiles topic.

Manage Network Profiles

Tanzu Kubernetes Grid Integrated Edition administrators can delete network profiles. Administrators can also perform the same operations that cluster managers use to list network profiles and manage how clusters use them.

Delete a Network Profile

To delete a network profile, run the following command:

```
tkgi 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 TKGI administrator.

Cluster Manager Operations

The following sections link to operations that both Tanzu Kubernetes Grid Integrated Edition administrators and cluster managers can perform on network profiles, documented in the Using Network Profiles (NSX-T Only) topic.

- List Network Profiles
- Create a Cluster with a Network Profile
- Assign a Network Profile to an Existing Cluster
  - This operation can assign a network profile to a cluster that does not have one, or change a cluster’s existing profile.
  - You cannot change a cluster’s network profile to remove pod IP block IDs. For details, see Limitation: Pod IP Block Changes

Limitation: Pod IP Block Changes

You cannot remove a cluster’s pod IP block IDs by creating and assigning a new network profile with `pod_ip_block_ids` array values removed.

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

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.

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.

Network Profile Use Cases

Network profiles let you customize NSX-T configuration parameters at the time of cluster creation. Use cases for network profiles include:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size a Load Balancer</td>
<td>Customize the size of the NSX-T load balancer service that is created when a Kubernetes cluster is provisioned.</td>
</tr>
<tr>
<td>Customize Pod Networks</td>
<td>Customize Kubernetes Pod Networks, including IP addresses, subnet size, and routability.</td>
</tr>
<tr>
<td>Topic</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Customize Node Networks</td>
<td>Customize Kubernetes Node Networks, including the IP addresses, subnet size, and routability.</td>
</tr>
<tr>
<td>Customize Floating IP Pools</td>
<td>Specify a custom floating IP pool.</td>
</tr>
<tr>
<td>Configure Bootstrap NSGroups</td>
<td>Specify an NSX-T Namespace Group where Kubernetes master nodes will be added to during cluster creation.</td>
</tr>
<tr>
<td>Configure Edge Router Selection</td>
<td>Specify the NSX-T Tier-0 router where Kubernetes node and Pod networks will be connected to.</td>
</tr>
<tr>
<td>Specify Nodes DNS Servers</td>
<td>Specify one or more DNS servers for Kubernetes clusters.</td>
</tr>
<tr>
<td>Configure DNS for Pre-Provisioned IPs</td>
<td>Configure DNS lookup of the Kubernetes API load balancer or ingress controller.</td>
</tr>
<tr>
<td>Configure the TCP Layer 4 Load Balancer</td>
<td>Configure layer 4 TCP load balancer settings; use third-party load balancer.</td>
</tr>
<tr>
<td>Configure the HTTP/S Layer 7 Ingress Controller</td>
<td>Configure layer 7 HTTP/S ingress controller settings; use third-party ingress controller.</td>
</tr>
<tr>
<td>Define DFW Section Markers</td>
<td>Configure top or bottom section markers for explicit DFW rule placement.</td>
</tr>
<tr>
<td>Configure NCP Logging</td>
<td>Configure NCP logging.</td>
</tr>
<tr>
<td>Dedicated Tier-1 Topology</td>
<td>Use dedicated Tier-1 routers, rather than a shared router, for each cluster’s Kube node, Namespace, and NSX-T load balancer.</td>
</tr>
</tbody>
</table>

Please send any feedback you have to pks-feedback@pivotal.io.
Size a Load Balancer

In this topic

Load Balancer Sizing

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition:

- 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 Tanzu Kubernetes Grid Integrated Edition, you can only use the large NSX Edge Node VM or the bare metal NSX Edge Node with Tanzu Kubernetes Grid Integrated Edition.

The following network profile, np-lb-med, defines a medium load balancer:

```json
{
   "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:

```json
{
   "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.
Customize Pod Networks

In this topic
Pod Subnet Prefix
Routable Pod Networks

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to define network profiles for pod networks.

Custom Pod Networks


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 Tanzu Kubernetes Grid Integrated Edition tile with a custom IP block. To use a custom pods network, do the following after you deploy Tanzu Kubernetes Grid Integrated Edition:
1. Define a custom IP block in NSX-T. For more information, see Creating NSX-T Objects for Tanzu Kubernetes Grid Integrated Edition.

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:

```json
{
  "description": "Example network profile with 2 non-routable pod networks",
  "name": "non-routable-pod",
  "parameters": {
    "pod_ip_block_ids": [
      "ebe78a74-a5d5-4dde-ba76-9cf067ee55",
      "ebe78a74-a5d5-4dde-ba76-9cf067ee56"
    ]
  }
}
```

**Note:** You cannot use the same Pod IP Block ID (UUID) that is specified in the TKGI Tile. Create a new Pod IP Block ID (UUID) that is not referenced in the TKGI 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 Tanzu Kubernetes Grid Integrated Edition.

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:

```json
{
  "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-ba76-9cf067ee55",
      "ebe78a74-a5d5-4dde-ba76-9cf067ee56"
    ]
  }
}
```

**Note:** You cannot customize the size of the Pod IP Block ID (UUID) that is specified in the TKGI 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 TKGI 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. Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition. 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 Tanzu Kubernetes Grid Integrated Edition:

1. Define a routable IP block in NSX-T. For more information, see Creating NSX-T Objects for Tanzu Kubernetes Grid Integrated Edition.

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:

```json
{
    "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 TKGi Tile. Create a new Pod IP Block ID (UUID) that is not referenced in TKGi Tile and use it to define a network profile.
Please send any feedback you have to pks-feedback@pivotal.io.
Customize Node Networks

In this topic

Configurable Node Network IP Blocks

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

### Page last updated:

This topic describes how to define network profiles for customizing Kubernetes node networks provisioned with VMware Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T.

#### Configurable Node Network IP Blocks

The **Nodes IP Block** is used by Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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, Tanzu Kubernetes Grid Integrated Edition 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 TKGI 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 TKGI Management Plane components. See [Create Management Plane in Installing and Configuring NSX-T Data Center v3.0 for TKGI](https://docs.vmware.com/en/VMware-Tanzu-Kubernetes-Grid-Integrated-Edition/v3.0/hcl/ks/ks.html#Create-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 TKGI 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 Tanzu Kubernetes Grid Integrated Edition administrator has configured the default in the TKGI 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.

```
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
  }
}
```

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.

Please send any feedback you have to pks-feedback@pivotal.io.
Customize Floating IP Pools

In this topic

Custom Floating IP Pool

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to define network profiles for custom floating IP pools.

Custom Floating IP Pool

To deploy Tanzu Kubernetes Grid Integrated Edition 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 TKGI 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 Tanzu Kubernetes Grid Integrated Edition.

You can define a network profile that specifies a custom floating IP pool to use instead of the default pool specified in the Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition. 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:

   ```json
   {
   "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.

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.
Please send any feedback you have to pks-feedback@pivotal.io.
Configure Bootstrap NSGroups

In this topic

Bootstrap Security Group

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to define network profiles for Kubernetes clusters provisioned with VMware Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T.

Bootstrap Security Group

Most of the NSX-T virtual interface tags used by Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition.

To address this bootstrap issue, Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition. 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:

   ```json
   {
       "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.
Configure Edge Router Selection

This topic describes how to define network profiles for Kubernetes clusters provisioned with VMware Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T.

Edge Router Selection

Using Tanzu Kubernetes Grid Integrated Edition 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 TKGI 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.

---

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

Copyright © 2020 VMware, Inc. All Rights Reserved.
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": ["f766f7-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-15b5-9052-a0e224256126b7",
        "fip_pool_ids": ["4ec625f-009a-79b7-ac22-40d37598c0d1"],
        "pod_ip_block_ids": ["91e7a3a1-5e12-4912-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-009a-79b7-ac22-40d37598e0d1"],
        "pod_routable": "true",
        "pod_ip_block_ids": ["f766f7-aaf1-49b2-d023-90e272e600ba", "6af46fd-cccc-4332-92d2-d918adc007ed"]
    }
}
```
np_customer_B.json
{
    "description": "network profile for Customer B",
    "name": "network-profile-Customer-B",
    "parameters": {
        "lb_size": "small",
        "t0_router_id": "a4c766cc-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-d023-90c272260090",
            "6fa46fd4-ecce-4332-92d2-d918adccce0"
        ]
    }
}
Specify Nodes DNS Servers

In this topic

DNS Configuration for Kubernetes Clusters

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to define network profiles for Kubernetes clusters provisioned with VMware Tanzu Kubernetes Grid Integrated Edition 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 TKGI 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition in a multi-tenant environment with multiple Tier-0 routers and a single TKGI 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.
Configure DNS for Pre-Provisioned IPs

In this topic
About DNS Lookup of Pre-Provisioned IP Addresses
DNS Lookup Parameters
Example API Load Balancer Lookup
Performing DNS Lookup of the Ingress Controller Using Network Profile
Setting the Master Node IP Address on the Command Line

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

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 Tanzu Kubernetes Grid Integrated Edition environment on NSX-T, when you provision a Kubernetes cluster using the command `tkgi 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 TKGI 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, TKGI will perform a lookup of the pre-provisioned IP address for the Kubernetes API load balancer. If the mode is API_INGRESS, TKGI 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 TKGI 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 TKGI 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 TKGi 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:

```
$ tkg 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 flag. For example:

```
$ tkg 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.
Configure the TCP Layer 4 Load Balancer

In this topic

Overview
Configure the TCP Ingress Controller Network Profile

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to define network profile to configure the NSX-T Load Balancer for VMware Tanzu Kubernetes Grid Integrated Edition.

Overview

The NSX-T Load Balancer is a logical load balancer that handles a number of functions using virtual servers and pools.

The NSX-T load balancer creates a load balancer service for each Kubernetes cluster provisioned by Tanzu Kubernetes Grid Integrated Edition with NSX-T. For each load balancer service, NCP, by way of the CRD, creates corresponding NSXLoadBalancerMonitor objects.

By default Tanzu Kubernetes Grid Integrated Edition deploys the following NSX-T virtual servers for each Kubernetes cluster:

- One TCP layer 4 load balancer virtual server for the Kubernetes API server.
- One TCP layer 4 auto-scaled load balancer virtual server for each Kubernetes service resource of type: LoadBalancer.
- Two HTTP/HTTPS layer 7 ingress routing virtual servers. These virtual server are attached to the Kubernetes Ingress Controller cluster load balancer service and can be manually scaled. Tanzu Kubernetes Grid Integrated Edition uses Kubernetes custom resources to monitor the state of the NSX-T load balancer service and scale the virtual servers created for ingress.

For information about configuring TCP layer 4 ingress controller see Configure the TCP Ingress Controller Network Profile below.

For information about configuring layer 7 ingress routing load balancers see Scaling the HTTP/S Layer 7 Ingress Load Balancers Using the LoadBalancer CRD. For information about configuring the layer 7 ingress controller see Defining Network Profiles for the HTTP/S Layer 7 Ingress Controller.

For more information about the NSX-T Load Balancer, see NSX-T Load Balancer in the VMware documentation.

For more information about Kubernetes custom resources, see Custom resources in the Kubernetes documentation.

Configure the TCP Ingress Controller Network Profile

The TCP layer 4 virtual server provisioned for each Kubernetes service is controlled by the parameters exposed in a network profile.

Note: 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.
NSX-T TCP Ingress Controller Network Profile Configuration

The NSX Ingress Controller is configured using the `ncp.ini` network profile configuration file.

The TCP Ingress Controller network profile has the following format:

```
{
    "name": "network_profile",
    "description": "DESCRIP",
    "parameters": {
        "cni_configurations": {
            "type": "nsxt",
            "parameters": {
                "nsx_lb": NSX-LB,
                "x_forwarded_for": "FORWARD-TYPE",
                "max_l4_lb_service": MAX-SERVERS,
                "l4_persistence_type": "source_ip",
                "l4_algorithm": "LB-BEHAVE"
            }
        }
    }
}
```

Where:

- **DESCRIP** is your description for this network profile configuration.
- **NSX-LB** is your preference for whether the NSX-T Load Balancer is used for your Kubernetes clusters. For more information see Configure Which NSX Load Balancer to Use below.
- **FORWARD-TYPE** (Optional) is your request header original client source IP. For more information see the Configure the Client Source IP Address, below.
- **MAX-SERVERS** is your maximum number of layer 4 virtual servers per cluster. For more information see the Configure the Maximum Number of Layer 4 Load Balancer Virtual Servers, below.
- **LB-BEHAVE** (Optional) is your load balancer behavior. For more information see the Configure the Layer 4 Load Balancer Algorithm, below.

For example:

```
{
    "name": "network_profile",
    "description": "DESCRIP",
    "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"
            }
        }
    }
}
```

The following table describes the Ingress Controller configuration parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>

Copyright © 2020 VMware, Inc. All Rights Reserved.
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:

<table>
<thead>
<tr>
<th>nsx_lb setting</th>
<th>nsx_ingress_controller: true</th>
<th>nsx_ingress_controller: false</th>
</tr>
</thead>
<tbody>
<tr>
<td>nsx_lb: true</td>
<td>Use the NSX-T Layer 4 LoadBalancer and the NCP-provisioned Layer 7 Ingress Controller.</td>
<td>Use the NSX-T Layer 4 LoadBalancer and a third-party Ingress Controller, such as NGINX.</td>
</tr>
<tr>
<td>nsx_lb: false</td>
<td>Invalid configuration. You cannot disable the NSX-T Load Balancer and use NCP as the Ingress Controller. The network profile will fail validation.</td>
<td>Use a third-party load balancer and a third-party ingress controller.</td>
</tr>
</tbody>
</table>

Configure Which NSX Load Balancer to Use

The `nsx_lb` flag controls whether to deploy either the NSX-T Load Balancer or a third-party load balancer, such as Nginx. The `nsx_lb` parameter accepts `true` or `false`. The default setting is `true` and the NSX-T Load Balancer is deployed. To use a third party load balancer, set this parameter to `false`.

For example:
Since the \texttt{nsx_ingress_controller} is a component of the NSX-T Load Balancer, if you choose to use a third-party load balancer (\texttt{nsx_lb: false}), you must also explicitly disable the \texttt{nsx_ingress_controller}. See Defining Network Profiles for the HTTP/S Layer 7 Ingress Controller.

Configure the 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 \texttt{x_forwarded_for} parameter can be enabled to ensure that the client IP address will be set in the HTTP request header. The \texttt{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 \texttt{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.

For example, with the following network profile the source IP address of the client will be appended to the existing set of client source IP addresses:

```json
{
  "name": "example-network-profile",
  "description": "x_forwarded_for insert",
  "parameters": {
    "cni_configurations": {
      "type": "nsxt",
      "parameters": {
        "x_forwarded_for": "insert"
      }
    }
  }
}
```

For example, with the following network profile the existing source IP address of the client will replace all other client source IP entries:

```json
{
  "name": "example-network-profile",
  "description": "x_forwarded_for replace",
  "parameters": {
    "cni_configurations": {
      "type": "nsxt",
      "parameters": {
        "x_forwarded_for": "replace"
      }
    }
  }
}
```

\textbf{Note:} The \texttt{nsx_lb} parameter maps to the \texttt{use_native_loadbalancer} parameter in NCP.ini.
Configure the Maximum 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 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_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_service` parameter to limit the number of layer 4 virtual servers to 100 per cluster:

```json
{
  "name": "example-network-profile",
  "description": "max_l4_service",
  "parameters": {
    "cni_configurations": {
      "type": "nsxt",
      "parameters": {
        "max_l4_service": 100
      }
    }
  }
}
```

Configure the 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`. 
Configure the 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 one of the following values:

- "round_robin" (default)
- "least_connection"
- "ip_hash"
- "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_persistence_type",
  "parameters": {
    "cni_configurations": {
      "type": "nsxt",
      "parameters": {
        "l4_persistence_type": "source_ip"
      }
    }
  }
}
```

Please send any feedback you have to pks-feedback@pivotal.io.
Configure the HTTP/S Layer 7 Ingress Controller

In this topic

Overview

Configure the HTTP/HTTPS Ingress Controller Network Profile

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to define network profiles for Kubernetes clusters provisioned with VMware Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T.

Overview

The NSX-T Load Balancer is a logical load balancer that handles a number of functions using virtual servers and pools.

The NSX-T load balancer creates a load balancer service for each Kubernetes cluster provisioned by Tanzu Kubernetes Grid Integrated Edition with NSX-T. For each load balancer service, NCP, by way of the CRD, creates corresponding NSXLoadBalancerMonitor objects.

By default Tanzu Kubernetes Grid Integrated Edition deploys the following NSX-T virtual servers for each Kubernetes cluster:

- One TCP layer 4 load balancer virtual server for the Kubernetes API server.
- One TCP layer 4 auto-scaled load balancer virtual server for each Kubernetes service resource of type: LoadBalancer.
- Two HTTP/HTTPS layer 7 ingress routing virtual servers. These virtual server are attached to the Kubernetes Ingress Controller cluster load balancer service and can be manually scaled. Tanzu Kubernetes Grid Integrated Edition uses Kubernetes custom resources to monitor the state of the NSX-T load balancer service and scale the virtual servers created for ingress.

For information about configuring the layer 7 ingress controller see Defining Network Profiles for the HTTP/S Layer 7 Ingress Controller, below. For information about configuring layer 7 ingress routing load balancers see Scaling the HTTP/S Layer 7 Ingress Load Balancers Using the LoadBalancer CRD.

For information about configuring TCP layer 4 ingress routing load balancers see Defining Network Profiles for the TCP Layer 4 Load Balancer.

For more information about the NSX-T Load Balancer, see NSX-T Load Balancer in the VMware documentation.

For more information about Kubernetes custom resources, see Custom resources in the Kubernetes documentation.

Configure the HTTP/HTTPS Ingress Controller Network Profile

The HTTP/HTTPS layer 7 virtual servers provisioned for each Kubernetes service are controlled by the parameters exposed in a network profile.
The NSX Ingress Controller is configured using the `ncp.ini` network profile configuration file.

The HTTP/HTTPS Ingress Controller network profile has the following format:

```json
{
  "name": "ncp_network_profile",
  "description": "DESCRIP",
  "parameters": {
    "cni_configurations": {
      "type": "nsxt",
      "parameters": {
        "nsx_lb": "NSX-LB",
        "nsx_ingress_controller": "NCP-IC",
        "ingress_ip": "IP-ADDRESS",
        "ingress_persistence_settings": {
          "persistence_type": "PERS-TYPE",
          "persistence_timeout": "TIMEOUT"
        }
      }
    }
  }
}
```

Where:

- `DESCRIP` is your description for this network profile configuration.
- `NSX-LB` is your preference for whether the NSX-T Load Balancer is used for your Kubernetes clusters. For more information see Configure the NSX Ingress Controller, below.
- `NCP-IC` is your preference for whether the NCP is used as the Ingress Controller for your Kubernetes clusters.
- `IP-ADDRESS` is the IP address to use for ingress controller load balancer. For more information see Configure the Ingress IP, below.
- `PERS-TYPE` is the persistence type to use for ingress controller load balancer. For more information see Configure the Ingress Persistence Settings, below.
- `TIMEOUT` is the persistence timeout to use for ingress controller load balancer. For more information see Configure the Ingress Persistence Settings, below.

For example:

```json
{
  "name": "ncp_network_profile",
  "description": "Example network profile for ingress controller",
  "parameters": {
    "cni_configurations": {
      "type": "nsxt",
      "parameters": {
        "nsx_lb": "true",
        "nsx_ingress_controller": "false",
        "ingress_ip": "192.168.160.212",
        "ingress_persistence_settings": {
          "persistence_type": "cookie",
          "persistence_timeout": 1
        }
      }
    }
  }
}
```

The following table describes the Ingress Controller configuration parameters:
### Parameter Details

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>User-defined name of the network profile.</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>User-defined description for the network profile.</td>
</tr>
<tr>
<td>parameters</td>
<td>Map</td>
<td>One or more name-value pairs.</td>
</tr>
<tr>
<td>cni_configurations</td>
<td>Map</td>
<td>Map containing two key-value pairs: type and parameters.</td>
</tr>
<tr>
<td>type</td>
<td>Constant String</td>
<td>Only <code>nsxt</code> is accepted.</td>
</tr>
<tr>
<td>parameters</td>
<td>Map</td>
<td>Map containing one or more key-value pairs for NCP settings.</td>
</tr>
<tr>
<td>nsx_lb</td>
<td>Boolean</td>
<td>Flag to control if the NSX-T Load Balancer is used for the Kubernetes cluster.</td>
</tr>
<tr>
<td>nsx_ingress_controller</td>
<td>Boolean</td>
<td>Flag to control if NCP is used as the Ingress Controller for the Kubernetes cluster.</td>
</tr>
<tr>
<td>ingress_ip</td>
<td>String</td>
<td>IP address to use for ingress controller load balancer.</td>
</tr>
<tr>
<td>ingress_persistence_settings</td>
<td>Map</td>
<td>Holds parameters for customizing Layer 7 persistence.</td>
</tr>
<tr>
<td>persistence_type</td>
<td>String</td>
<td>Valid values are <code>cookie</code> or <code>source_ip</code>. An empty value is not accepted.</td>
</tr>
<tr>
<td>persistence_timeout</td>
<td>Integer</td>
<td>Value that is equal to 1 or larger. Empty value is not accepted.</td>
</tr>
</tbody>
</table>

#### nsx_lb Setting

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:

<table>
<thead>
<tr>
<th>nsx_lb setting</th>
<th>nsx_ingress_controller: true</th>
<th>nsx_ingress_controller: false</th>
</tr>
</thead>
<tbody>
<tr>
<td>nsx_lb: true</td>
<td>Use the NSX-T Layer 4 LoadBalancer and the NCP-provisioned Layer 7 Ingress Controller.</td>
<td>Use the NSX-T Layer 4 LoadBalancer and a third-party Ingress Controller, such as NGINX.</td>
</tr>
<tr>
<td>nsx_lb: false</td>
<td>Invalid configuration. You cannot disable the NSX-T Load Balancer and use NCP as the Ingress Controller. The network profile will fail validation.</td>
<td>Use a third-party load balancer and a third-party ingress controller.</td>
</tr>
</tbody>
</table>

#### Configure the 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:

- The following network profile uses the NSX-T Load Balancer and a third-party ingress controller:
The following network profile uses a third party load balancer and a third-party ingress controller:

```json
{
  "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
      }
    }
  }
}
```

You should not disable `nsx_lb` and use the NCP Ingress Controller. Using the NCP Ingress Controller with `nsx_lb` disabled is invalid.

For example, the following is invalid:

```json
{
  "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
      }
    }
  }
}
```

Configure the 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`: 

```json
{
  "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
      }
    }
  }
}
```
An invalid IP address is rejected with an invalid parameter value error.

For example:

- The following network profile parameters cannot be parsed because the "ingress_ip" configuration specifies an invalid IP address:

  ```json
  {
    "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 "ingress_ip" configuration is not a string and the JSON input is invalid:

  ```json
  {
    "name": "example-network-profile",
    "description": "ingress_ip-ERROR",
    "parameters": {
      "cni_configurations": {
        "type": "nsxt",
        "parameters": {
          "ingress_ip": 192.168.160.212
        }
      }
    }
  }
  ```

Configure the 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`
- `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.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>persistence_type</td>
<td>String</td>
<td>Valid values are <code>cookie</code> or <code>source_ip</code>. An empty value is not accepted.</td>
</tr>
<tr>
<td>persistence_timeout</td>
<td>Integer</td>
<td>Value that is equal to 1 or larger. Empty value is not accepted.</td>
</tr>
</tbody>
</table>

For example:

- **Network profile for ingress_persistence_settings**:

```json
{
  "name": "example_network_profile",
  "description": "ingress_persistence_settings",
  "parameters": {
    "cni_configurations": {
      "type": "nsxt",
      "parameters": {
        "ingress_ip": "192.168.160.212"
      }
    }
  }
}
```

- **Network profile for ingress_persistence_settings**:

```json
{
  "name": "example_network_profile",
  "description": "ingress_persistence_settings",
  "parameters": {
    "cni_configurations": {
      "type": "nsxt",
      "parameters": {
        "ingress_ip": "192.168.160.212"
      }
    }
  }
}
```

Please send any feedback you have to pks-feedback@pivotal.io.
Define DFW Section Markers

In this topic

About DFW Section Markers
Top Firewall Section Marker
Bottom Firewall Section Marker

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

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.
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.

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

In this topic

About Logging for NCP Configurations
Parameters for NCP Logging
Example Network Profile for NCP Logging
Log Settings
Log Level
Log Dropped Traffic
enable_err_crd

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to define network profiles for logging NCP configurations.

About Logging for NCP 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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>User-defined name of the network profile.</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>User-defined description for the network profile.</td>
</tr>
<tr>
<td>parameters</td>
<td>Map</td>
<td>One or more name-value pairs.</td>
</tr>
<tr>
<td>cni_configurations</td>
<td>Map</td>
<td>Map containing two key-value pairs: type and parameters.</td>
</tr>
<tr>
<td>type</td>
<td>Constant String</td>
<td>Only next is accepted.</td>
</tr>
<tr>
<td>parameters</td>
<td>Map</td>
<td>Map containing one or more key-value pairs for NCP settings.</td>
</tr>
<tr>
<td>log_settings</td>
<td>Map</td>
<td>Holds parameters for configuring NCP logging.</td>
</tr>
<tr>
<td>log_level</td>
<td>String Enumeration</td>
<td>“INFO”, “WARNING”, “DEBUG”, “ERROR”, “CRITICAL”</td>
</tr>
<tr>
<td>log_dropped_traffic</td>
<td>Boolean</td>
<td>Default is false. Set to true to log dropped firewall traffic.</td>
</tr>
</tbody>
</table>

Example Network Profile for NCP Logging

The following network profile is an example that illustrates the parameters exposed for NCP logging.
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:

<table>
<thead>
<tr>
<th>ncp.ini key</th>
<th>Default log level</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>coe.nsxlib_loglevel</code></td>
<td>INFO</td>
</tr>
<tr>
<td><code>coe.loglevel</code></td>
<td>NONE</td>
</tr>
<tr>
<td><code>k8s.loglevel</code></td>
<td>NONE</td>
</tr>
</tbody>
</table>

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:
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": "log_settings",
  "parameters": {
    "cni_configurations": {
      "type": "nsxt",
      "parameters": {
        "log_settings": {
          "log_level": "DEBUG",
          "log_dropped_traffic": true
        }
      }
    }
  }
}
```

Please send any feedback you have to pks-feedback@pivotal.io.
Shared and Dedicated Tier-1 Router Topologies

In this topic

- Shared Tier-1 Topology
  - Comparison to Dedicated Tier-1
- Dedicated Tier-1 Topology
  - Network Profile for Dedicated Tier-1 Topology
- Implementing a Shared Tier-1 Topology in a Multi-Tier-0 Environment

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes shared and dedicated Tier-1 router topologies for Tanzu Kubernetes Grid Integrated Edition Kubernetes clusters on vSphere with NSX-T.

Shared Tier-1 topology is the default. This topic also explains how to define a network profile that overrides this default, to specify Dedicated Tier-1 topology for Tanzu Kubernetes Grid Integrated Edition clusters.

Shared Tier-1 Topology

By default, Kubernetes clusters in Tanzu Kubernetes Grid Integrated Edition with NSX-T have the Shared Tier-1 network topology, in which each cluster shares a single Tier-1 router for its external-facing components: the Kubernetes node, namespace, and NSX-T load balancer.

Note: The Shared Tier-1 topology requires NSX-T Data Center v2.5.

This topology uses a single, shared Tier-1 switch and router for each Kubernetes cluster. 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
Comparison to Dedicated Tier-1

Unlike the Dedicated Tier-1 Topology, 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 not 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 TKGI as the number of NSX-T objects allocated per Kubernetes cluster is drastically reduced, in comparison to dedicated Tier-1. 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.

Dedicated Tier-1 Topology

When you provision a Kubernetes cluster with a network profile that overrides the Shared Tier-1 topology, Tanzu Kubernetes Grid Integrated Edition 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 above, the result is that a given Kubernetes cluster will run several Tier-1 switches and routers in its topology.

Network Profile for Dedicated Tier-1 Topology

To create clusters with Dedicated Tier-1 topology, you define and use a network profile that overrides the default Shared Tier-1 topology by setting the `single_tier_topology` key to `false`.

Shown below is an example network profile that disables the Shared Tier-1 Router for Kubernetes clusters:

```json
{
  "name": "example-network-profile-shared-t1",
  "description": "Shared-Tier-1 topology network profile",
  "parameters": {
    "single_tier_topology": false
  }
}
```
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, Tanzu Kubernetes Grid Integrated Edition 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).

```json
{
  "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

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This section describes how to use compute profiles and host groups for VMware Tanzu Kubernetes Grid Integrated Edition clusters on vSphere.

See the following topics:

- Using Compute Profiles
- Using vSphere Host Groups with Tanzu Kubernetes Grid Integrated Edition

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

In this topic
Overview
Prerequisites
Define a Compute Profile
Retrieve the BOSH CPI ID
Create a Compute Profile
List Compute Profiles
Create a Cluster with a Compute Profile
Resize a Cluster with a Compute Profile
List Clusters with Compute Profile
Delete Compute Profiles
Example Compute Profiles

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

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 Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T integration.

The procedures in this topic use the TKGI API.

Overview

Compute profiles allow you to specify which vSphere resources are used when deploying Kubernetes clusters in a TKGI 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 TKGI control plane is smaller.

⚠️ Note: Compute profiles only override the AZ values specified in Tanzu Kubernetes Grid Integrated Edition 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 TKGI API.

You can customize any number of compute profiles.

Prerequisites

Before you define a compute profile, you must have the following:

- A TKGI UAA admin access token to provide when calling the TKGI API. For instructions that explain how to export your access token to an environment variable, see Export TKGI 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:

   ```json
   {
       "name": "PROFILE-NAME",
       "description": "PROFILE-DESCRIPTION",
       "parameters": {
           "master": { },
           "azs": [{
               "name": "AZ-NAME",
               "cpi": "CPI-NAME",
               "cloud_properties": {
                   "datacenters": [{
                       "name": "DC-NAME",
                       "clusters": [{
                           "CLUSTER-NAME": {
                               "resource_pool": "RESOURCE-POOL-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`.

   ```java
   Copyright © 2020 VMware, Inc. All Rights Reserved. 666 1.8
   ```
PROFILE-DESCRIPTION describes the compute profile. 

AZ-NAME is a name for the availability zone (AZ) where you want to deploy 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 Tanzu Kubernetes Grid Integrated Edition 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 datacenter 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`:

```json
{
  "name": "dev",
  "description": "For development clusters",
  "parameters": {
    "azs": [
      {
        "cpi": "abc012abc345abc567de",
        "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 Tanzu Kubernetes Grid Integrated Edition deployment.

1. Locate the credentials that were used to import the Ops Manager .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: YOUR-CPI-NAME
   ```


Create a Compute Profile

Use the JSON file that contains your compute profile parameters to make a request to the TKGI API. The TKGI API applies the parameters to your TKGI deployment as a compute profile.

To create the compute profile, run the following command:

```bash
```

Where:

- **TKGI-API** is the FQDN of your TKGI API endpoint. For example, `api.tkgi.example.com`.
- **YOUR-ACCESS-TOKEN** is your TKGI UAA admin access token specified as an environment variable. For information on how to export this token as an environment variable, see Export Note: If you define multiple AZs in the `azs` array, you can specify different AZs for worker node VMs and master node VMs.

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.

Copyright © 2020 VMware, Inc. All Rights Reserved.
TKGI API Token

- `/compute-profiles/dev-clusters.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 TKGI deployment by making a request to the TKGI API.

To list compute profiles, run the following command:

```
curl -s -k "https://TKGI-API:9021/v1/compute-profiles" 
-H "cache-control: no-cache" 
-H "authorization: Bearer $YOUR-ACCESS-TOKEN" 
-H "Content-Type: application/json"
```

Where:

- `TKGI-API` is the FQDN of your TKGI API endpoint. For example, `api.tkgi.example.com`.
- `YOUR-ACCESS-TOKEN` is your TKGI UAA admin access token specified as an environment variable. For information on how to export this token as an environment variable, see Export TKGI API Token.

For example:

```
curl -s -k "https://api.tkgi.example.com:9021/v1/compute-profiles" 
-H "cache-control: no-cache" 
-H "authorization: Bearer $YOUR-ACCESS-TOKEN" 
-H "Content-Type: application/json"
```

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 TKGI 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 Tanzu Kubernetes Grid Integrated Edition 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 -X POST "https://TKGI-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:
TKGI-API is the FQDN of your TKGI API endpoint. For example, api.tkgi.example.com.

YOUR-ACCESS-TOKEN is your TKGI UAA admin access token specified as an environment variable. For information on how to export this token as an environment variable, see Export TKGI 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:

```json
{
  "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": ",",
  "compute_profile": ",",
  "compute_profile_params": {"vcenter_cloud_provider_config_key": null, "unset_http_proxy": null, "insecure_registries": null}
}
```

Resizing a Cluster with a Compute Profile

Currently, you cannot use the TKGI 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 `tkgi 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.

Listing Clusters with Compute Profile

When you execute the `tkgi 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:

```bash
curl -o -4 -H "authorization: Bearer $YOUR-ACCESS-TOKEN" "https://TKGI-API:9021/v1/clusters"
```

Where:

- **TKGI-API** is the FQDN of your TKGI API endpoint. For example, api.tkgi.example.com.
- **$YOUR-ACCESS-TOKEN** is your TKGI UAA admin access token specified as an environment variable. For information on how to export this token as an environment variable, see Export TKGI API Token.

The output includes compute profile information.

For example:
Delete Compute Profiles

You can delete compute profiles in your TKGI deployment by making a request to the TKGI 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

`tkgi delete cluster` command.

To delete a compute profile, run the following command:

```bash
curl -X DELETE "https://TKGI-API:9021/v1/compute-profiles"
-H "cache-control: no-cache"
-H "authorization: Bearer $YOUR-ACCESS-TOKEN"
-H "Content-Type: application/json"
```

Where:

- **TKGI-API** is the FQDN of your TKGI API endpoint. For example, api.tkgi.example.com.
- **YOUR-ACCESS-TOKEN** is your TKGI UAA admin access token specified as an environment variable. For information on how to export this token as an environment variable, see Export TKGI 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:

```json
{
    "name": "dev",
    "description": "For development clusters",
    "parameters": {
        "api": {
            "ip": "<ip01.example.com>",
            "cloud_properties": {
                "datacenter": "my-dc",
                "resource_pools": ["my-res-pool1"]
            }
        },
        "master": {
            "ip": "<ip01.example.com>",
            "max-worker": 1
        },
        "worker": {
            "instance": 1
        }
    }
}
```

**prod-clusters.json**

The following example compute profile for a single cluster uses one AZ and one resource pool:

```json
{
    "name": "prod",
    "description": "For production clusters",
    "parameters": {
        "api": {
            "ip": "<ip01.example.com>",
            "cloud_properties": {
                "datacenter": "my-dc",
                "resource_pools": ["my-res-pool1"]
            }
        },
        "master": {
            "ip": "<ip01.example.com>",
            "max-worker": 1
        },
        "worker": {
            "instance": 1
        }
    }
}
```
The following example compute profile uses two AZs, with one cluster in the first AZ and three clusters in the second AZ:

```json
{
    "name": "prod",
    "description": "For production clusters",
    "parameters": {
        "z1": {
            "name": "z1",
            "cpi": "abc012abc345abc567des",
            "cloud_properties": {
                "datacenters": [
                    {
                        "name": "my-dc",
                        "clusters": [
                            {
                                "cluster1": {
                                    "resource_pool": "rp3"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        "z2": {
            "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 Tanzu Kubernetes Grid Integrated Edition

In this topic

About vSphere Host Groups
Host Group Use Cases for Tanzu Kubernetes Grid Integrated Edition
Defining a Host Group in vSphere
Using a Host Group with Tanzu Kubernetes Grid Integrated Edition

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

Topic provided by VMware

This topic describes how to use vSphere Host Groups with VMware Tanzu Kubernetes Grid Integrated Edition.

About vSphere Host Groups

In vSphere, a cluster is a collection of ESXi 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.

Tanzu Kubernetes Grid Integrated Edition users can define host groups in vSphere, then in the TKGI 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 Tanzu Kubernetes Grid Integrated Edition

This subsection describes use cases for using host groups with Tanzu Kubernetes Grid Integrated Edition.

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 TKGI.

Using Host Group as a New AZ in BOSH
Previously, the two types of AZs available with TKG 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 Tanzu Kubernetes Grid Integrated Edition, 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 Tanzu Kubernetes Grid Integrated Edition**

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 Tanzu Kubernetes Grid Integrated Edition.

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. (Optional) If you are using a host group with vSAN stretched clusters, set the **VM-Host Affinity Rule** dropdown to **SHOULD**. This setting maintains high availability by letting TKGi restart VMs in another host group if their AZ fails. TKGi ignores this setting if the vSAN cluster has no host group configured. For more information, see [Ability to Set the VM-Host Affinity Rule to “Should” for Clusters in vSphere® in the Ops Manager v2.9 Release Notes](#).

7. Click **Save**.

---

Please send any feedback you have to [pks-feedback@pivotal.io](mailto:pks-feedback@pivotal.io).
Adding Infrastructure Password Changes to the Tanzu Kubernetes Grid Integrated Edition Tile

In this topic
Manage Your Service Account Passwords
  Step 1: Update Your Service Account Passwords
  Step 2: Deploy Your New Service Account Passwords
Manage Your NSX Manager Password (vSphere and vSphere with NSX-T only)

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to manage VMware Tanzu Kubernetes Grid Integrated Edition after changing a BOSH Director or Tanzu Kubernetes Grid Integrated Edition service account password.

Manage Your Service Account Passwords

When you installed Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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.
To update Tanzu Kubernetes Grid Integrated Edition with a new **Master Node Service Account** password, perform the following steps:

1. Access the **Installation Dashboard** in Ops Manager.
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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition, perform the following steps:

1. Access the **Installation Dashboard** in Ops Manager.

2. Click **Review Pending Changes**.

3. In the **Errands** section for Tanzu Kubernetes Grid Integrated Edition, select **Update all clusters errand**.

4. Click **Apply Changes** to update the Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition. 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 Tanzu Kubernetes Grid Integrated Edition

In this topic

Shutdown Sequence and Tasks
- Step 1: Disable BOSH Resurrection
- Step 2: Shut Down Customer Apps
- Step 3: Shut Down Kubernetes Clusters
- Step 4: Shut Down the Tanzu Kubernetes Grid Integrated Edition API and Database VMs
- Step 5: Shut Down VMware Harbor Registry (vSphere Only)
- Step 6: Shut Down BOSH Director
- Step 7: Shut Down Ops Manager
- Step 8: Shut Down NSX-T Components (vSphere NSX-T Only)
- Step 9: Shut Down vCenter Server (vSphere Only)
- Step 11: Shut Down ESXi Hosts (vSphere NSX-T Only)

Startup Sequence and Tasks
- Step 1: Start ESXi Hosts (vSphere NSX-T Only)
- Step 2: Start vCenter (vSphere Only)
- Step 3: Start NSX-T Components (vSphere NSX-T Only)
- Step 4: Start Ops Manager
- Step 5: Start the BOSH Director
- Step 6: Start the Control Plane VMs
- Step 7: Start Harbor Registry (vSphere Only)
- Step 8: Start the Kubernetes Clusters
- Step 9: Start Customer Apps
- Step 10: Re-enable BOSH Resurrection

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic lists and describes the shutdown and startup sequence for VMware Tanzu Kubernetes Grid Integrated Edition including Tanzu Kubernetes Grid Integrated Edition-provisioned Kubernetes cluster nodes, TKGI components, and (vSphere only) vSphere hosts.

Many of these operations use your IaaS dashboard, such as vSphere Client, Azure Portal, AWS Management Console, or GCP Console.

Shutdown Sequence and Tasks

To perform a graceful shutdown of all Kubernetes, Tanzu Kubernetes Grid Integrated Edition, and infrastructure components, complete the following tasks in sequence.

Step 1: Disable BOSH Resurrection

If you have the Enable VM Res resurrector Plugin checkbox selected in the BOSH Director tile > Director Config pane, you must turn
BOSH resurrection off before restarting TKGI, to prevent BOSH from recreating VMs.

To do this, run the command `bosh update-resurrection off`.

Step 2: Shut Down Customer Apps


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 Tanzu Kubernetes Grid Integrated Edition-provisioned Kubernetes clusters following the procedure defined in the How to shutdown and startup a Multi Master TKGI 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition cluster. For example:

   `$ bosh -d service-instance_aa1234567bc8de9f0a1c stop master`

4. Using your IaaS dashboard, 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.
Step 4: Shut Down the Tanzu Kubernetes Grid Integrated Edition API and Database VMs

To shutdown Tanzu Kubernetes Grid Integrated Edition control plane TKGI API and TKGI Database VMs, complete the following:

1. **Stop Tanzu Kubernetes Grid Integrated Edition Control Plane Processes**

2. **Shut Down the Tanzu Kubernetes Grid Integrated Edition API and Database VMs**

Stop Tanzu Kubernetes Grid Integrated Edition Control Plane Processes

To stop Tanzu Kubernetes Grid Integrated Edition control plane processes and services, do the following:

1. Using the BOSH CLI, retrieve the BOSH deployment ID of your Tanzu Kubernetes Grid Integrated Edition deployment by running the following command:

   ```
   bosh deployments
   ```

   The Tanzu Kubernetes Grid Integrated Edition deployment ID is `pivotal-container-service-` followed by a unique BOSH-generated hash.

2. Using the BOSH CLI, stop the TKGI 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 Tanzu Kubernetes Grid Integrated Edition deployment.

   For example:

   ```
   $ bosh -d pivotal-container-service-1bf7b02738056cdc37e6 stop
   ```
Shut Down the Tanzu Kubernetes Grid Integrated Edition API and Database VMs

To shut down the TKGI API and TKGI Database VMs, do the following:

1. Run the `bosh vms` command to list your Tanzu Kubernetes Grid Integrated Edition control plane VMs:

   ```
   bosh -d pivotal-container-service-DEPLOYMENT-ID vms
   ```

   Where `DEPLOYMENT-ID` is the BOSH-generated ID of your Tanzu Kubernetes Grid Integrated Edition deployment.

   For example:

   ```
   $ bosh -d pivotal-container-service-1bf7b02738056cdc37e6 vms
   ```

2. From the `bosh vms` output, record:
   - The TKGI API VM name, listed under `Instances` as `pivotal-container-service/` followed by a unique BOSH-generated hash
   - The TKGI Database VM name, listed under `Instances` as `pks-db/` followed by a unique BOSH-generated hash

3. Using your IaaS dashboard, locate and gracefully shut down your Tanzu Kubernetes Grid Integrated Edition control plane VMs:
   - The TKGI API VM
   - The TKGI Database VM

View a larger version of this image.

Step 5: Shut Down VMware Harbor Registry (vSphere Only)

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:
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.

![Image of vCenter console](image)

View a larger version of this image.

Step 6: Shut Down BOSH Director

Using your IaaS dashboard, locate and gracefully shut down the BOSH Director VM.
Step 7: Shut Down Ops Manager

Using your IaaS dashboard, locate and gracefully shut down the Ops Manager VM.
Step 8: Shut Down NSX-T Components (vSphere NSX-T Only)

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

Step 9: Shut Down vCenter Server (vSphere Only)

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.
Step 11: Shut Down ESXi Hosts (vSphere NSX-T Only)

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.

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.

Startup Sequence and Tasks

To restart all Kubernetes, Tanzu Kubernetes Grid Integrated Edition, and infrastructure components, complete the following tasks in the sequence presented.

Step 1: Start ESXi Hosts (vSphere NSX-T Only)

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 (vSphere Only)

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 (vSphere NSX-T Only)

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 your IaaS dashboard, 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 your IaaS dashboard, 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 Ops Manager 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 Control Plane VMs

To start the TKGI Control Plane, do the following:

1. Using your IaaS dashboard, power on the TKGI Database VM.
2. Run `bosh start` to start the TKGI Database VM:
   
   ```
   bosh -d TKGI-DB-VM-ID start
   ```
Where **TKGI-DB-VM-ID** is the name of the TKGI Database VM listed by the `bosh vms` command.

3. Using your IaaS dashboard, power on the TKGI API VM.

4. Run `bosh start` to start the TKGI API VM:

   ```bash
   bosh -d TKGI-API-VM-ID start
   ```

   Where **TKGI-API-VM-ID** is the name of the TKGI API VM.

Step 7: Start Harbor Registry (vSphere Only)

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:

   ```bash
   bosh -d harbor-container-registry-DEPLOYMENT-ID start
   ```

   Where **DEPLOYMENT-ID** is the BOSH-generated ID of your Harbor Registry deployment. For example:

   ```bash
   $ 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 CLI, run `ssh` to access the first Tanzu Kubernetes Grid Integrated Edition master node and start etcd.

2. Using the BOSH CLI, start the next Tanzu Kubernetes Grid Integrated Edition master node.

3. Using the BOSH CLI, start all remaining Tanzu Kubernetes Grid Integrated Edition master nodes including the master where you started etcd.


For exact Kubernetes node startup instructions, refer to the How to shutdown and startup a Multi Master TKGI cluster knowledge base article.

Step 9: Start Customer Apps


Step 10: Re-enable BOSH Resurrection

Turn BOSH resurrection back on by running the command `bosh update-resurrection on`.

Please send any feedback you have to pks-feedback@pivotal.io.
Deleting Tanzu Kubernetes Grid Integrated Edition

In this topic
Delete the Tanzu Kubernetes Grid Integrated Edition Tile

This topic explains how to delete the Tanzu Kubernetes Grid Integrated Edition tile.

Delete the Tanzu Kubernetes Grid Integrated Edition Tile

To delete the Tanzu Kubernetes Grid Integrated Edition tile, perform the following steps:

1. Navigate to the Ops Manager Installation Dashboard.

2. Click the trash can icon on the Tanzu Kubernetes Grid Integrated Edition tile.

3. Click Confirm.

4. Click Review Pending Changes.

5. (Optional) By default, deleting the Tanzu Kubernetes Grid Integrated Edition tile also deletes all the clusters created by Tanzu Kubernetes Grid Integrated Edition. 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

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This section describes how to manage VMware Tanzu Kubernetes Grid Integrated Edition-provisioned Kubernetes clusters and workloads.

See the following topics:

- Create and Manage Clusters in the Management Console
- Managing Clusters with the CLI
- Supporting Clusters with the CLI
- Deploying Workloads with the CLI
- Load Balancing and Ingress
- Load Balancing and Ingress with NSX-T

Please send any feedback you have to pks-feedback@pivotal.io.
Create and Manage Clusters in the Management Console

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This section describes how to create clusters in the VMware Tanzu Kubernetes Grid Integrated Edition Management Console on vSphere. Cluster creation in the management console involves the following actions:

- Add users and user groups to VMware Tanzu Kubernetes Grid Integrated Edition, and assign roles to them so that they can create and manage clusters. For information about user management, see Identity Management in the Management Console.
- Set resource quotas to limit the amount of compute power and memory that users can consume. For information about quotas, see Assign Resource Quotas to Users.
- Create network profiles, so that you can customize the networking for different types of cluster. For information about network profiles, see Working with Network Profiles.
- When you have set up users and configured network profiles, you can easily Create Clusters in the Management Console, applying network and Kubernetes profiles to the clusters.
- After you create clusters, you can Monitor and Manage Clusters, Nodes, and Namespaces in the Management Console.


For information about how to deploy the management console and install Tanzu Kubernetes Grid Integrated Edition, see Install on vSphere with the Management Console.

Please send any feedback you have to pks-feedback@pivotal.io.
Create Clusters in the Management Console


When you deploy Kubernetes clusters from the management console, you select the following pre-existing resources to configure your cluster:

- A plan from the list of plans that were defined when the management console was deployed.
- An optional network profile. Network profiles allow cluster administrators and cluster managers to customize the networking for different types of Kubernetes cluster. For information about how to create network profiles in the management console, see Working with Network Profiles.
- An optional Kubernetes profile. Kubernetes profiles enable cluster administrators and cluster managers to customize Kubernetes component settings for any clusters that they provision. You create Kubernetes profiles outside of the management console, by using the TKGi CLI to define the Kubernetes profile in your Tanzu Kubernetes Grid Integrated Edition instance..

1. Go to the **TKG Integrated Edition** view of the management console.

2. Select the **Clusters** tab and click **Create Cluster**.

3. Use the **Plan** drop-down menu to select one of the plans that were configured during the deployment of Tanzu Kubernetes Grid Integrated Edition Management Console.

   The plan defines the set of resources that the Kubernetes cluster will use. A summary of the selected plan appears as you hover over each option.
4. Enter a name and a host name for the cluster, and specify the number of worker nodes to create.

5. Use the Network Profile drop-down menu to select an existing network profile for the cluster to use.

If you have not created any network profiles, the management console uses the default network profile. In this case, Tanzu Kubernetes Grid Integrated Edition Management Console configures networking for you, based on the plan that you selected.

6. Optionally use the Kubernetes Profile drop-down menu to select an existing Kubernetes profile for the cluster to use.

If you have not created any Kubernetes profiles, the management console uses the default Kubernetes profile. In this case, Tanzu Kubernetes Grid Integrated Edition Management Console configures the cluster for you, based on the plan that you selected.
7. Click **Save** to deploy your cluster. You can follow the progress of the deployment of your cluster in the **Clusters** tab.

![Cluster deployment progress](image)

8. When your cluster is running, optionally select it in the **Clusters** tab and click **Update**.

![Cluster update](image)

9. Optionally update the number of worker nodes, change the network or Kubernetes profile, or expand **Advanced Settings** to update the settings for node drain and the pod shutdown grace period.
To delete a cluster that you no longer require, select it in the Clusters tab and click Delete.

Next Steps

- Monitor and Manage Clusters, Nodes, and Namespaces in the Management Console
- Connect to Clusters in Kubernetes Dashboard
- Connect to Clusters with kubectl

Please send any feedback you have to pks-feedback@pivotal.io.
Monitor and Manage Clusters, Nodes, and Namespaces in the Management Console

In this topic

Obtain Cluster Information
Connect to Clusters in Kubernetes Dashboard
Connect to Clusters with kubectl
Obtain Node Information

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

You can find general information about your deployment on vSphere, and information about all of the clusters and nodes running it, in the **TKG Integrated Edition** view of Tanzu Kubernetes Grid Integrated Edition Management Console.

Obtain Cluster Information

1. Go to the **TKG Integrated Edition** view of the management console.

2. Select the **Clusters** tab to see detailed information about all of the clusters running in this instance.

3. 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 Storage panel, expand **Persistent Volume Claims** to see the volumes that your cluster is using.
- In the Nodes panel, expand **Masters and Workers** and select the VM links to go to those VMs in the vSphere inventory.

4. Select the **Nodes** tab to see details of all of the nodes that are running in that cluster.

5. Select the **Namespaces** tab to see the status and networking details of all of the namespaces that are running in that cluster.
Connect to Clusters in Kubernetes Dashboard


2. Select the Clusters tab for your Tanzu Kubernetes Grid Integrated Edition instance.

3. Select a cluster.

4. Select Open Kubernetes Dashboard for instructions about how to access the cluster by using Kubernetes Dashboard.

Connect to Clusters with kubectl


2. Select the Clusters tab for your Tanzu Kubernetes Grid Integrated Edition instance.

3. Select a cluster.

4. Select Access Cluster for instructions about how to access the cluster by using kubectl.

5. Click Email to send the instructions to users who need to use kubectl to connect to this cluster.
Obtain Node Information


2. Select the Nodes tab to see detailed information about all of the nodes running in this instance. This tab shows the general status, type, name, IP, and availability zone for all of the nodes that are running in your Tanzu Kubernetes Grid Integrated Edition instance.

3. Click the links in the Kubernetes Clusters columns to go the Summary tab of that cluster.

4. Click the links in the VM Name column to be taken to the node VMs in the vSphere inventory.

Please send any feedback you have to pks-feedback@pivotal.io.
Managing Clusters with the CLI

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

**Page last updated:**

This section describes how to use the CLI to manage Kubernetes clusters in VMware Tanzu Kubernetes Grid Integrated Edition.

See the following topics:

- Logging in to Tanzu Kubernetes Grid Integrated Edition
- Creating Clusters
- Using Kubernetes Profiles
- Using Network Profiles
- Viewing Cluster Lists
- Viewing Cluster Details
- Viewing Cluster Plans
- Scaling Existing Clusters
- Upgrading Clusters
- Deleting Clusters

Please send any feedback you have to pks-feedback@pivotal.io.
Logging in to Tanzu Kubernetes Grid Integrated Edition

In this topic
Overview
Prerequisites
Log in to the TKGI CLI
Log in to the TKGI CLI as an Automated Client
Export TKGI API Access Token

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to log in to VMware Tanzu Kubernetes Grid Integrated Edition.

Overview

To manage Tanzu Kubernetes Grid Integrated Edition-deployed clusters, you use the TKGI Command Line Interface (TKGI CLI). When you log in to Tanzu Kubernetes Grid Integrated Edition successfully for the first time, the TKGI 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 `TKGI_HOME` environment variable to override this location and store `creds.yml` in any directory on your system.

Prerequisites

Before you can log in to Tanzu Kubernetes Grid Integrated Edition, you must have the following:

- A running Tanzu Kubernetes Grid Integrated Edition environment, including an external load balancer configured to forward traffic to the TKGI API endpoint. See the Installing Tanzu Kubernetes Grid Integrated Edition section for your cloud provider.
- A username and password that has access to the TKGI API. See Managing Tanzu Kubernetes Grid Integrated Edition Users with UAA.
- The TKGI CLI installed on your local system. See Installing the TKGI CLI

Log in to the TKGI 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 TKGI CLI:

```
tkgi login -a TKGI-API -u USERNAME -p PASSWORD --ca-cert CERT-PATH
```

Replace the placeholder values in the command as follows:
TKGI-API is the domain name for the TKGI API that you entered in Ops Manager > Tanzu Kubernetes Grid Integrated Edition > TKGI API > API Hostname (FQDN). For example, api.tkgi.example.com.

USERNAME and PASSWORD belong to the account you created in the Grant Tanzu Kubernetes Grid Integrated Edition Access to an Individual User section of Managing Tanzu Kubernetes Grid Integrated Edition Users with UAA. If you do not use -p to provide a password, the TKGI CLI prompts for the password interactively. VMware 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 TKGI API certificate with SSL.

For example:

```
$ tkgi login -a api.tkgi.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:

```
$ tkgi login -a api.tkgi.example.com -u alana -k
```

**Note:** If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 TKGI CLI. For information about configuring SAML, see Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider.

Log in to the TKGI CLI as an Automated Client

To log in to the TKGI CLI as an automated client for a script or service, run the following command:

```
tkgi login -a TKGI-API --client-name CLIENT-NAME --client-secret CLIENT-SECRET --ca-cert CERTIFICATE-PATH
```

Where:

- **TKGI-API** is the domain name for the TKGI API that you entered in Ops Manager > Tanzu Kubernetes Grid Integrated Edition > TKGI API > API Hostname (FQDN). For example, api.tkgi.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 TKGI API certificate with SSL.

For example:

```
$ tkgi login -a api.tkgi.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 Tanzu Kubernetes Grid Integrated Edition Access to a Client in Managing
Export TKGI API Access Token

This procedure stores a TKGI API access token as an environment variable that you can use when executing TKGI API calls on the command line.

1. To export your access token into an environment variable, run the following command:

```
tkgi login -a TKGI-API -u USER-ID -p 'PASSWORD' -k; 
export YOUR-ACCESS-TOKEN=$(bosh int ~/.pks/creds.yml --path /access_token)
```

Where:
- **TKGI-API** is the FQDN of your TKGI API endpoint. For example, `api.tkgi.example.com`.
- **USER-ID** is your Tanzu Kubernetes Grid Integrated Edition user ID.
- **PASSWORD** is your Tanzu Kubernetes Grid Integrated Edition password.
- **YOUR-ACCESS-TOKEN** is the name of your access token environment variable.

For example:

```
$ tkgi login -a tkgi.my.lab -u alana -p 'psswrdabc123...!' -k; 
export my_token=$(bosh int ~/.pks/creds.yml --path /access_token)
```

**Note:** If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 *TKGI CLI*. For information about configuring SAML, see [Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider](#).

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

In this topic
Overview
Configure Cluster Access
vSphere with NSX-T
  GCP, AWS, Azure, or vSphere without NSX-T
Create a Kubernetes Cluster
Identify Kubernetes Cluster Master VMs
Next Steps

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to create a Kubernetes cluster with VMware Tanzu Kubernetes Grid Integrated Edition using the TKGI Command Line Interface (TKGI CLI).

Overview

Use the TKGI CLI to create Kubernetes clusters in your Tanzu Kubernetes Grid Integrated Edition environment.

To create a Tanzu Kubernetes Grid Integrated Edition Kubernetes cluster, do the following:

- Configure Cluster Access
- Create a Kubernetes Cluster
- Identify Kubernetes Cluster Master VMs

The `tkgi create-cluster` command creates a Kubernetes cluster with TKGI compatibility matching the TKGI version of the current TKGI control plane.

Configure Cluster Access

Cluster access configuration differs by the type of Tanzu Kubernetes Grid Integrated Edition deployment.

vSphere with NSX-T

Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition Deployments with NSX-T.

Note: For a complete list of the objects that Tanzu Kubernetes Grid Integrated Edition creates by default when you create a Kubernetes cluster on vSphere with NSX-T, see vSphere with NSX-T Cluster Objects.

Copyright © 2020 VMware, Inc. All Rights Reserved.
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 Tkgi CLI commands on the cluster from your local workstation. For more information, see Load Balancers in Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition cluster load balancer, see the following:

- Creating and Configuring a GCP Load Balancer for Tanzu Kubernetes Grid Integrated Edition Clusters
- Creating and Configuring an AWS Load Balancer for Tanzu Kubernetes Grid Integrated Edition Clusters
- Creating and Configuring an Azure Load Balancer for Tanzu Kubernetes Grid Integrated Edition Clusters

Create the Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition Access to an Individual User section of Managing Tanzu Kubernetes Grid Integrated Edition Users with UAA.

2. On the command line, run the following command to log in:

```
tkgi login -a TKGI-API -u USERNAME -k
```

Where:

- **TKGI-API** is the domain name for the TKGI API that you entered in Ops Manager > Tanzu Kubernetes Grid Integrated Edition > TKGI API > API Hostname (FQDN). For example, api.tkgi.example.com.
- **USERNAME** is your user name.

See Logging in to Tanzu Kubernetes Grid Integrated Edition for more information about the `tkgi login` command.

**Note:** If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 TKGI CLI. For information about configuring SAML, see Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider

3. To create a cluster run the following command:
tkgi create-cluster CLUSTER-NAME \ 
   --external-hostname HOSTNAME \ 
   --plan PLAN-NAME \ 
   [--num-nodes WORKER-NODES] \ 
   [--network-profile NETWORK-PROFILE-NAME] \ 
   [--tags TAGS]

Where:

- **CLUSTER-NAME** is your unique name for your cluster.

![Note: The **CLUSTER-NAME** must not contain special characters such as &. The TKGI 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.](image)

- **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 `tkgi 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.

- (Optional) (Azure and vSphere only) **TAGS** are the labels and metadata values to apply to the VMs created in the cluster. Specify the tags as **key:value** pairs. For more information about tagging see Tagging Clusters.

For example:

```
$ tkgi 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.](image)

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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition topic for your IaaS, such as Installing Tanzu Kubernetes Grid Integrated Edition on vSphere.

4. To track cluster creation, run the following command:

```
tkgi cluster CLUSTER-NAME
```

Where **CLUSTER-NAME** is the unique name for your cluster.
For example:

```bash
tkgi 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:

   ```bash
   bosh tasks
   ```


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 Tanzu Kubernetes Grid Integrated Edition 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 `tkgi 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 `tkgi cluster` command to locate the master node IP addresses and ports.


   ![Note](https://www.vmware.com/images/icons/exclamation-circle.png) **Note:** For clusters with multiple master node VMs, health checks on port 8443 are recommended.

7. To access your cluster, run the following command:

   ```bash
tkgi get-credentials CLUSTER-NAME
   ```

   Where `CLUSTER-NAME` is the unique name for your cluster.

   For example:
$ tkgi get-credentials tkgi-example-cluster

Fetching credentials for cluster tkgi-example-cluster.
Context set for cluster tkgi-example-cluster.

You can now switch between clusters by using:
$ kubectl config use-context <cluster-name>

The `tkgi get-credentials` command creates a local `kubeconfig` that allows you to manage the cluster. For more information about the `tkgi get-credentials` command, see `Retrieving Cluster Credentials and Configuration`.

**Note:** If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 `TKGI CLI`. For information about configuring SAML, see `Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider`.

8. To confirm you can access your cluster using the Kubernetes CLI, run the following command:

```bash
kubectl cluster-info
```


**Identify Kubernetes Cluster Master VMs**

**Note:** This section applies only to Tanzu Kubernetes Grid Integrated Edition deployments on GCP or on vSphere without NSX-T. Skip this section if your Tanzu Kubernetes Grid Integrated Edition deployment is on vSphere with NSX-T, AWS, or Azure. For more information, see `Load Balancers in Tanzu Kubernetes Grid Integrated Edition`.

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:

```bash
tkgi login -a TKGI-API -u USERNAME -k
```

Where:

- **TKGI-API** is the domain name for the TKGI API that you entered in `Ops Manager > Tanzu Kubernetes Grid Integrated Edition > TKGI API > API Hostname (FQDN)`. For example, `api.tkgi.example.com`.
- **USERNAME** is your user name.

See `Logging in to Tanzu Kubernetes Grid Integrated Edition` for more information about the `tkgi login` command.

**Note:** If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 `TKGI CLI`. For information about configuring SAML, see `Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider`.

---

**Copyright © 2020 VMware, Inc. All Rights Reserved.**
2. To locate the cluster ID and master node IP addresses, run the following command:

```
tkgi 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 tkgi 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 tkgi -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 tkgi -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 Creating and Configuring a GCP Load Balancer for Tanzu Kubernetes Grid Integrated Edition Clusters.

Next Steps

If you did not tag your new cluster during creation, tag your cluster’s VMs now. If your Tanzu Kubernetes Grid Integrated Edition deployment is on:

- **AWS**: Tag your subnets with your new cluster’s unique identifier before adding the subnets to the Tanzu Kubernetes Grid Integrated Edition workload load balancer. After you complete the Create a Kubernetes Cluster procedure above, follow the
instructions in AWS Prerequisites in *Deploying and Exposing Basic Linux Workloads*.

- **Azure, vSphere, or** vSphere with NSX-T: You can use the TKGI CLI to tag clusters by following the steps in Tagging Clusters.
- **GCP**: You can tag your clusters using your IaaS-provided management console.

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

In this topic

Overview

Who Creates and Manages Kubernetes Profiles
Validated vs Experimental Customizations
"k8s" to "kubernetes" Alias in TKG CLI

Create a Kubernetes Profile

Kubernetes Profile Format
Kubernetes Profile Parameters
The create-k8s-profile Command

Manage Kubernetes Profiles

List Kubernetes Profiles
Delete a Kubernetes Profile
View Kubernetes Profile Details
Create a Cluster with a Kubernetes Profile
Assign a Kubernetes Profile to an Existing Cluster

Kubernetes Profile Use Cases

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:


This topic also lists verified use cases for Kubernetes profiles on Tanzu Kubernetes Grid Integrated Edition.

Overview

Kubernetes profiles enable cluster administrators and cluster managers to customize Kubernetes component settings for any clusters that they provision.

To use Kubernetes profiles, Tanzu Kubernetes Grid Integrated Edition users:

1. Create JSON configuration files that set configuration options for any Kubernetes components, such as `kube-apiserver` on the control plane or the `kubelet` on each node.
2. Use the TKGI CLI to create the Kubernetes profile in Tanzu Kubernetes Grid Integrated Edition.
3. Use the TKGI CLI to apply the profile to clusters.

Verified uses for Kubernetes profiles include encrypting secrets in an etcd database, adding an OIDC provider, and using a `ResourceQuota` admission control plugin.

Who Creates and Manages Kubernetes Profiles
Users with the `pks.clusters.admin` or `pks.clusters.manage` roles can create and use Kubernetes profiles.

If user with the `pks.clusters.admin-read-only` role attempts to create a Kubernetes profile, they see the following error:

```
You do not have enough privileges to perform this action. Please contact the TKGI administrator.
```

Validated vs Experimental Customizations

A Kubernetes profile configures settings for Kubernetes components in two JSON code blocks, `customizations` and `experimental_customizations`. See Kubernetes Profile Format for details. The code blocks differ as follows:

- **customizations** block
  - TKGI checks the validity of configurations in this block. If you run `tkgi create-k8s-profile` on a profile with invalid configurations in `customizations`, the command returns an error.
  - The TKGI team supports clusters configured with tested, validated parameters in this block.

- **experimental_customizations** block
  - **warning:** Experimental customizations are not validated or supported.
  - TKGI imposes no restrictions on the contents of this block.
  - Configurations in this block are neither tested nor supported.
  - If a customer wants to use an unsupported configuration, they should contact the TKGI team with the parameters that they want tested, validated, and supported.

"k8s" to "kubernetes" Alias in TKGI CLI

In the TKGI CLI, all commands that include `k8s-profile` are aliased to also use `kubernetes-profile`. For example, the `tkgi k8s-profiles` and `tkgi kubernetes-profiles` commands are equivalent.

For brevity, this documentation uses the `k8s` versions.

Create a Kubernetes Profile

To create a Kubernetes profile in Tanzu Kubernetes Grid Integrated Edition, you:

1. Define a Kubernetes profile in a JSON configuration file, following the Kubernetes Profile Format below.

2. Use the TKGI CLI to define the Kubernetes profile within Tanzu Kubernetes Grid Integrated Edition, as described in The create-network-profile Command, below.

Kubernetes Profile Format

To create a Kubernetes profile, you must first define it as a JSON file that specifies network parameters, listed in Kubernetes Profile Parameters below.

Here is the basic structure of a Kubernetes profile.
The Kubernetes profile JSON can include the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>String</td>
<td>Name of the Kubernetes profile.</td>
</tr>
<tr>
<td>description</td>
<td>String</td>
<td>Description of the Kubernetes profile.</td>
</tr>
<tr>
<td>customizations</td>
<td>Map</td>
<td>A block that defines supported, validated K8s options using the component, arguments, and file-arguments parameters below.</td>
</tr>
<tr>
<td>experimental_customizations</td>
<td>Map</td>
<td>A block that defines unsupported, unvalidated K8s options using the component, arguments, and file-arguments parameters below.</td>
</tr>
<tr>
<td>component</td>
<td>String</td>
<td>The name of a K8s component, e.g. kubelet, kube-apiserver.</td>
</tr>
<tr>
<td>arguments</td>
<td>Map</td>
<td>Parameters for each component, as one or more Key:Value pairs. Multiple values must be separated by commas, without spaces.</td>
</tr>
<tr>
<td>file-arguments</td>
<td>Map</td>
<td>Parameters whose values are stored as files on the local machine.</td>
</tr>
</tbody>
</table>

Parameters marked with an * are mandatory.

Note: If you specify the same parameter in both customizations and experimental_customizations, the one in customization takes precedence.

The create-k8s-profile Command
After defining a Kubernetes profile in a JSON file as described in Kubernetes Profile Format, a cluster administrator or manager creates the Kubernetes profile by running the following TKGI CLI command:

```bash
tkgi create-k8s-profile PATH-TO-YOUR-KUBERNETES-PROFILE-CONFIGURATION
```

Where `PATH-TO-YOUR-KUBERNETES-PROFILE-CONFIGURATION` is the path to the JSON file you created when defining the Kubernetes profile.

For example:

```json
cat profile3-docs.json
{
  "name": "my-profile3",
  "description": "My profile description",
  "customizations": [ 
    {
      "component": "kube-apiserver",
      "arguments": {
        "service-node-port-range": "30000-40000"
      }
    }
  ],
  "experimental_customizations": [ 
    {
      "component": "kubelet",
      "arguments": {
        "maximum-dead-containers": "1000",
        "feature-gates": "APIListChunking=true,AttachVolumeLimit=false"
      }
    }
  ]
}
```

```bash
user ~/workspace: ./tkgi create-k8s-profile k8s-profile3.json
Kubernetes profile my-profile3 successfully created
```

Manage Kubernetes Profiles

Tanzu Kubernetes Grid Integrated Edition cluster administrators and managers can perform the following operations on Kubernetes profiles and the clusters that use them.

List Kubernetes Profiles

To list available Kubernetes profiles, run the following command:

```bash
tkgi k8s-profiles
```

For example:

```bash
$ tkgi k8s-profiles
K8s-profile   Owner   Created Date
Basic-k8s-profile   Alana   Tue, 05 Nov 2019 16:28:15 PST
```

The command output differs by user role:  
- `pks.cluster.admin` see a list of Kubernetes profiles that all users created  
- `pks.cluster.manage` see a list of only the Kubernetes profiles that they created
Delete a Kubernetes Profile

To delete a Kubernetes profile, run the following command:

```bash
tkgi delete-k8s-profile KUBERNETES-PROFILE-NAME
```

Where `KUBERNETES-PROFILE-NAME` is the name of the Kubernetes profile you want to delete.

For example:

```bash
$ user ~/workspace: ./tkgi delete-k8s-profile my-profile3
Are you sure you want to delete the kubernetes profile my-profile3? (y/n): y
Deletion of my-profile3 completed
```

**Note:** You cannot delete a Kubernetes profile that is in use. Before deleting a Kubernetes profile, you must disassociate it from all clusters or delete all clusters it is associated with.

Both `pks.clusters.admin` and `pks.clusters.manage` users can delete Kubernetes profiles. If a `pks.clusters.admin-read-only` user attempts to delete a Kubernetes profile, they see the following error:

You do not have enough privileges to perform this action. Please contact the TKGI administrator.

View Kubernetes Profile Details

To view details of a Kubernetes profile, run the following command:

```bash
tkgi k8s-profile KUBERNETES-PROFILE-NAME
```

Where `KUBERNETES-PROFILE-NAME` is the name of the Kubernetes profile you want to view.

For example:

```bash
tkgi k8s-profile Basic-k8s-profile
Name: Basic-k8s-profile
Owner: Alana
Created Date: Tue, 05 Nov 2019 16:28:15 PST
Description: Kubernetes profile for customer A
...<KEY> : <VALUE>
```

- Users with the `pks.cluster.admin` can view the details of any Kubernetes profile; users with the `pks.cluster.manage` role can view details of Kubernetes profiles that they created.
- Once you create or update a cluster with an encryption profile, you cannot assign any other Kubernetes profiles to that cluster. Because decryption is not straightforward, applying another profile can have nondeterministic outcome.

Create a Cluster with a Kubernetes Profile

You can assign a Kubernetes profile to a Kubernetes cluster at the time of cluster creation.

To create an Tanzu Kubernetes Grid Integrated Edition-provisioned Kubernetes cluster with a Kubernetes profile, run the following
command:

```
tkgi create-cluster CLUSTER-NAME --external-hostname HOSTNAME --plan PLAN-NAME --kubernetes-profile KUBERNETES-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 Tanzu Kubernetes Grid Integrated Edition plan you want to use for your cluster.
- **KUBERNETES-PROFILE-NAME** is the name of the Kubernetes profile you want to use for your cluster.

Assign a Kubernetes Profile to an Existing Cluster

TKGI supports changing the Kubernetes profile for an already created cluster. You can use this procedure to:

- assign a Kubernetes profile to a cluster that does not have one, or
- change a cluster’s existing profile to a new one

This is the procedure to change a cluster’s Kubernetes profile:

1. Do one of the following
   - Choose a new Kubernetes profile for the cluster. See List Kubernetes profiles.
   - Define and create a new Kubernetes profile as described in Create a Kubernetes Profile.
     - The name of the new Kubernetes profile must be unique and different from the previously assigned Kubernetes profile.

2. Run the following command to update the cluster with the new Kubernetes profile:

```
tkgi update-cluster CLUSTER-NAME --kubernetes-profile NEW-KUBERNETES-PROFILE-NAME
```

Where:
- **CLUSTER-NAME** is the name of the existing Kubernetes cluster
- **NEW-KUBERNETES-PROFILE-NAME** is the name of the new Kubernetes profile you want to apply to the cluster.

Kubernetes Profile Use Cases

Kubernetes profiles let you customize Kubernetes configuration parameters at the time of cluster creation. Use cases for Kubernetes profiles include:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encrypt Secrets in an etcd Database</td>
<td>Use an encryption provider to encrypt secrets in a cluster’s etcd database.</td>
</tr>
<tr>
<td>Admission Control: ResourceQuota</td>
<td>Use the <a href="#">ResourceQuota</a> admission control plugin to restrict incoming requests by resource usage.</td>
</tr>
<tr>
<td>Set Service Node Port Range</td>
<td>Use <a href="#">service-node-port-range</a> to specify an IP range for NodePort services.</td>
</tr>
<tr>
<td>Add an OIDD Provider</td>
<td>Customize a cluster’s OIDD provider by deploying a <a href="#">dex</a> connector or other OIDD provider to its pod.</td>
</tr>
</tbody>
</table>
### Admission Control: ResourceQuota

To create a Kubernetes profile that includes the `ResourceQuota` admission control plugin:

- Follow the Create a Kubernetes Profile instructions.
- Include the following customizations in your profile configuration file:

```json
"customizations": [
  {
    "component": "kube-apiserver",
    "arguments": {
      "enable-admission-plugins": PLUGINS-LIST
    }
  }
],
```

Where `PLUGINS-LIST` is one of the following:

- The string `"ResourceQuota`.
- A comma-delimited string list of validated plugins that includes `ResourceQuota`.

For more information, see [ResourceQuota](https://kubernetes.io) in the Kubernetes documentation.

### Set Service Node Port Range

To create a Kubernetes profile that uses `service-node-port-range` for NodePort type services:

- Follow the Create a Kubernetes Profile instructions.
- Include the following customizations in your profile configuration file:

```json
"customizations": [
  {
    "component": "kube-apiserver",
    "arguments": {
      "service-node-port-range": PORT-RANGE
    }
  }
],
```

Where `PORT-RANGE` is a CIDR notation IP range from which to assign service cluster IPs, such as `30000-40000`.

If the specified `PORT-RANGE` is not valid, the `tkgi create-k8s-profile` command returns an error for `invalid value for service-node-port-range`.

For more information, see [Type NodePort](https://kubernetes.io) in the Kubernetes documentation.

### Restrict Request Header Names

To create a Kubernetes profile that uses `requestheader-allowed-names` for Apiserver client authentication:

- Follow the Create a Kubernetes Profile instructions.
Include the following `customizations` in your profile configuration file:

```json
"customizations": [
    {
        "component": "kube-apiserver",
        "arguments": {
            "requestheader-allowed-names": COMMON-NAMES
        }
    }
],
```

Where `COMMON-NAMES` is a string list of valid Common Name values in the signed client certificate, such as "cn1.com,c2.com".

For more information, see Kubernetes Apiserver Client Authentication in the Kubernetes documentation.

Please send any feedback you have to pks-feedback@pivotal.io.
Using Network Profiles (NSX-T Only)

In this topic
- How Network Profiles are Created
- List Network Profiles
- Create a Cluster with a Network Profile
  - Network Profile Update Validation
- Assign a Network Profile to an Existing Cluster

Network Profile Use Cases

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to use network profiles for Kubernetes clusters provisioned with VMware Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T integration.

Network profiles let you customize NSX-T configuration parameters.

How Network Profiles are Created

Tanzu Kubernetes Grid Integrated Edition cluster administrators can create and delete network profiles, as described in the Creating and Managing Network Profiles topic.

After an administrator creates a network profile, cluster managers can create clusters with it or assign it to existing clusters.

List Network Profiles

To list available network profiles, run the following command:

```
tkgi network-profiles
```

For example:

```
$ tkgi network-profiles

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lb-profile-medium</td>
<td>Network profile for medium size NSX-T load balancer</td>
</tr>
<tr>
<td>small-routable-pod</td>
<td>Network profile with small load balancer and two routable pod networks</td>
</tr>
</tbody>
</table>
```

Create a Cluster with a Network Profile

You can assign a network profile to a Kubernetes cluster at the time of cluster creation.

To create a Tanzu Kubernetes Grid Integrated Edition-provisioned Kubernetes cluster with a network profile, run the following

```
command:

```
tkgi 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 Tanzu Kubernetes Grid Integrated Edition 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.

Assign a Network Profile to an Existing Cluster

TKGI supports changing the network profile for an already created cluster. You can use this procedure to:

- assign a network profile to a cluster that does not have one, or
- change a cluster’s existing profile to a new one

**Note:** You cannot change a cluster’s network profile to remove pod IP block IDs. For more information, see Limitation: Pod IP Block Changes in Creating and Managing Network Profiles.

This is the procedure to change a cluster’s network profile:

1. Do one of the following
   - Choose a new network profile for the cluster. See List Network Profiles.
   - Have a Tanzu Kubernetes Grid Integrated Edition cluster administrator define and create a new network profile as described in Create a Network Profile in Creating and Managing Network Profiles.
     - The name of the new network profile must be unique and different from the previously assigned network profile.

2. Run the following command to update the cluster with the new network profile:

```
tkgi 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 `tkgi 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.

Network Profile Use Cases

Network profiles let you customize NSX-T configuration parameters for clusters when you create them or afterward. Use cases for network profiles include:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size a Load Balancer</td>
<td>Customize the size of the NSX-T load balancer service that is created when a Kubernetes cluster is provisioned.</td>
</tr>
<tr>
<td>Customize Pod Networks</td>
<td>Customize Kubernetes Pod Networks, including IP addresses, subnet size, and routability.</td>
</tr>
<tr>
<td>Customize Node Networks</td>
<td>Customize Kubernetes Node Networks, including the IP addresses, subnet size, and routability.</td>
</tr>
<tr>
<td>Customize Floating IP Pools</td>
<td>Specify a custom floating IP pool.</td>
</tr>
<tr>
<td>Configure Bootstrap NSGroups</td>
<td>Specify an NSX-T Namespace Group where Kubernetes master nodes will be added to during cluster creation.</td>
</tr>
<tr>
<td>Configure Edge Router Selection</td>
<td>Specify the NSX-T Tier-0 router where Kubernetes node and Pod networks will be connected to.</td>
</tr>
<tr>
<td>Specify Nodes DNS Servers</td>
<td>Specify one or more DNS servers for Kubernetes clusters.</td>
</tr>
<tr>
<td>Configure DNS for Pre-Provisioned IPs</td>
<td>Configure DNS lookup of the Kubernetes API load balancer or ingress controller.</td>
</tr>
<tr>
<td>Configure the TCP Layer 4 Load Balancer</td>
<td>Configure layer 4 TCP load balancer settings; use third-party load balancer.</td>
</tr>
<tr>
<td>Configure the HTTP/S Layer 7 Ingress Controller</td>
<td>Configure layer 7 HTTP/S ingress controller settings; use third-party ingress controller.</td>
</tr>
<tr>
<td>Define DFW Section Markers</td>
<td>Configure top or bottom section markers for explicit DFW rule placement.</td>
</tr>
<tr>
<td>Configure NCP Logging</td>
<td>Configure NCP logging.</td>
</tr>
<tr>
<td>Dedicated Tier-1 Topology</td>
<td>Use dedicated Tier-1 routers, rather than a shared router, for each cluster’s Kube node, Namespace, and NSX-T load balancer.</td>
</tr>
</tbody>
</table>

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

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

**Page last updated:**

Follow the steps below to view the list of deployed Kubernetes cluster with the TKGI CLI.

1. On the command line, run the following command to log in:

   ```
tkgi login -a TKGI-API -u USERNAME -k
   ```

   Where:
   - **TKGI-API** is the domain name for the TKGI API that you entered in *Ops Manager > Tanzu Kubernetes Grid Integrated Edition > TKGI API > API Hostname (FQDN)*. For example, `api.tkgi.example.com`.
   - **USERNAME** is your user name.

   See [Logging in to Tanzu Kubernetes Grid Integrated Edition](#) for more information about the `tkgi login` command.

   **Note:** If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 *TKGI CLI*. For information about configuring SAML, see [Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider](#).

2. Run the following command to view the list of deployed clusters, including cluster names and status:

   ```
tkgi clusters
   ```

   For example:

   ```
   $ tkgi clusters
<table>
<thead>
<tr>
<th>Name</th>
<th>Plan Name</th>
<th>UUID</th>
<th>Status</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster-one</td>
<td>small</td>
<td>881543kd-64fg-7826-hea6-3h7g1e04kh0e</td>
<td>succeeded</td>
<td>CREATE</td>
</tr>
<tr>
<td>cluster-two</td>
<td>small</td>
<td>951547dl-67kg-9631-bjub-7893e098br0q</td>
<td>succeeded</td>
<td>CREATE</td>
</tr>
</tbody>
</table>
   ```

   Please send any feedback you have to [pkgs-feedback@pivotal.io](mailto:pkgs-feedback@pivotal.io).
Viewing Cluster Details

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

Follow the steps below to view the details of an individual cluster using the TKGI CLI.

1. On the command line, run the following command to log in:

   ```
tkgi login -a TKGI-API -u USERNAME -k
   ```

   Where:
   - `TKGI-API` is the domain name for the TKGI that you entered in Ops Manager > Tanzu Kubernetes Grid Integrated Edition > TKGI API > API Hostname (FQDN). For example, `api.tkgi.example.com`.
   - `USERNAME` is your user name.

   See Logging in to Tanzu Kubernetes Grid Integrated Edition for more information about the `tkgi login` command.

   Note: If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 TKGI CLI. For information about configuring SAML, see Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider

2. Run the following command to view the details of an individual cluster:

   ```
tkgi cluster CLUSTER-NAME
   ```

   Replace `CLUSTER-NAME` with the unique name for your cluster. For example:

   ```
   $ tkgi 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:

   ```
tkgi cluster CLUSTER-NAME --details
   ```

   Replace `CLUSTER-NAME` with the unique name for your cluster. For example:

   ```
   $ tkgi cluster my-cluster --details
   ```

The following shows the sample output for an example Kubernetes cluster named `my-cluster`:
$ tkgi cluster my-cluster --details

Name: my-cluster
K8s Version: 1.17.5
Plan Name: small
UUID: 4b1a9b6d-3594-4ced-ad0f-22043b26480
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 IPs: 10.197.100.130

Network Profile Name:

NSXT Network Details:

<table>
<thead>
<tr>
<th>Load Balancer Size</th>
<th>(lb_size): &quot;small&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes DNS Setting</td>
<td>[&quot;10.142.7.2&quot;]</td>
</tr>
<tr>
<td>Node IP addresses</td>
<td>[no-nat] (node_routable): false</td>
</tr>
<tr>
<td>Nodes subnet prefix</td>
<td>(node_subnet_prefix): 24</td>
</tr>
<tr>
<td>POD IP addresses</td>
<td>[no-nat] (pod_routable): false</td>
</tr>
<tr>
<td>PODs subnet prefix</td>
<td>(pod_subnet_prefix): 24</td>
</tr>
<tr>
<td>NS Group ID of master VMs</td>
<td>(master_vms_nsgroup_id): &quot;&quot;</td>
</tr>
<tr>
<td>Tier 0 Router identifier</td>
<td>(t0_router_id): &quot;1e8371ac-1718-4617-8734-3975c6cd373b&quot;</td>
</tr>
<tr>
<td>Floating IP Pool identifiers</td>
<td>(fip_pool_ids): [&quot;901341c3-72e4-49d0-a3d1-66748664a062&quot;]</td>
</tr>
<tr>
<td>Node IP block identifiers</td>
<td>(node_ip_block_ids): [&quot;c5f0eb13-9691-4170-a9cd-c988f36ebd2&quot;]</td>
</tr>
<tr>
<td>POD IP block identifiers</td>
<td>(pod_ip_block_ids): [&quot;f562d2ca-9e8c-4c5f-98a3-e797f06b8d4&quot;]</td>
</tr>
</tbody>
</table>

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

The following image shows another example of tkgi cluster CLUSTER-NAME -- details output with NSX-T details:

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

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

Follow the steps below to view information about the available plans for deploying a cluster using the TKGI CLI.

1. On the command line, run the following command to log in:

   `tkgi login -a TKGI-API -u USERNAME -k`

   Where:
   - `TKGI-API` is the domain name for the TKGI API that you entered in Ops Manager > Tanzu Kubernetes Grid Integrated Edition > TKGI > API Hostname (FQDN). For example, `api.tkgi.example.com`.
   - `USERNAME` is your user name.

   See Logging in to Tanzu Kubernetes Grid Integrated Edition for more information about the `tkgi login` command.

   Note: If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 TKGI CLI. For information about configuring SAML, see Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider

2. Run the following command to view information about the available plans for deploying a cluster:

   `tkgi plans`

   The response lists details about the available plans, including plan names and descriptions:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>Default plan for K8s cluster</td>
</tr>
</tbody>
</table>

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

In this topic
Scale Horizontally by Changing the Number of Worker Nodes Using the TKGI CLI
Scale Vertically by Changing Cluster Node VM Sizes in the TKGI Tile

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

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. Tanzu Kubernetes Grid Integrated Edition 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 TKGi topic for your IaaS.

Scale Horizontally by Changing the Number of Worker Nodes Using the TKGI CLI

Follow the steps below to scale an existing cluster using the TKGI CLI by increasing or decreasing the number of worker nodes.

1. On the command line, run the following command to log in:

   ```
tkgi login -a TKGI-API -u USERNAME -k
   ```

   Where:
   - **TKGI-API** is the domain name for the TKGI API that you entered in Ops Manager > Tanzu Kubernetes Grid Integrated Edition > TKGI API > API Hostname (FQDN). For example, api.tkgi.example.com.
   - **USERNAME** is your user name.

   See Logging in to Tanzu Kubernetes Grid Integrated Edition for more information about the `tkgi login` command.

   Note: If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 TKGI CLI. For information about configuring SAML, see Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider.

2. To view the current number of worker nodes in your cluster, run the following command:

   ```
tkgi cluster CLUSTER-NAME
   ```
3. Run the following command:

```
tkgi 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 Tanzu Kubernetes Grid Integrated Edition tile in Ops Manager.

For example:

```
$ tkgi 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 TKGI 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 TKGI 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 TKGI 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 size from the Master/ETCD VM Type menu.
   - For Worker nodes, select the desired 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 TKGI cluster.

5. Click **Save** to preserve tile changes.
6. At the Installation Dashboard, click **Review Pending Changes**.
7. For the TKGI 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.
Tagging Clusters

Overview

IaaSes provide the ability for customers to “tag” VMs, databases, and other resources with custom labels and metadata values. Apply one or more tags to your clusters to simplify organizing, managing, searching for, and filtering resources within your IaaS-provided management console and other tools:

- If your Tanzu Kubernetes Grid Integrated Edition deployment is on Azure or vSphere, including vSphere with NSX-T, you can use the TKGI CLI to tag clusters by following the steps in Tag Your Clusters as They Are Created below.
- If your Tanzu Kubernetes Grid Integrated Edition deployment is on AWS or GCP, you can tag your clusters using your IaaS-provided management console.

**Note:** Tanzu Kubernetes Grid Integrated Edition Cluster tagging requires Ops Manager v2.8.0 or later.

Tag Your Clusters as They Are Created

To apply tags to your cluster’s VMs, include the `--tags` parameter in your `tkgi create-cluster` command line, and specify the desired tags as a comma-delimited list of key:value pairs.

```
tkgi create-cluster CLUSTER-NAME --tags "TAGS"
```

Where:

- `CLUSTER-NAME` is the name of the cluster to create.
- `TAGS` is a comma-delimited list of key:value pairs to apply to the cluster.

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic explains how to tag new and existing clusters using the TKGI CLI.
For example:

$ tkgi create-cluster my-cluster --tags "status:billable"
$ tkgi create-cluster my-cluster --tags "status:non-billable,region:northwest"
$ tkgi create-cluster my-cluster --tags "client:example.com, costcenter:pettycash"

Tag Your Existing Clusters

To apply tags to your existing cluster’s VMs, run the `tkgi update-cluster` command line, and specify the `--tags` parameter and a comma-delimited list of `key:value` pairs of the tags to apply to the cluster.

```
tkgi update-cluster CLUSTER-NAME --tags "TAGS"
```

Where:

- **CLUSTER-NAME** is the name of the cluster to tag.
- **TAGS** is a comma-delimited list of `key:value` pairs.

For example:

```
$ tkgi update-cluster my-cluster --tags "client:tinymegacorp"
$ tkgi update-cluster my-cluster --tags "client:example.com, costcenter:pettycash"
$ tkgi update-cluster my-cluster --tags "status:non-billable, region:northwest"
```

Modify Cluster Tags

You can also use `tkgi update-cluster` to modify your cluster’s existing tags. When you modify cluster tags you completely replace all of the cluster’s existing tags with the specified tags.

Modify Your Existing Tags

To modify your cluster’s existing tags do the following:

1. Retrieve the cluster’s existing tags list string by running `tkgi cluster`. For information on `tkgi cluster` Review Your Tags below.

2. Modify the tags list string by doing one of the following:
   - To add a new tag, append it to existing tags list string.
   - To modify an existing tag, modify it within the tags list string.
   - To remove an existing tag, delete it from within the tags list string.

3. Run the following command:

```
tkgi update-cluster CLUSTER-NAME --tags "TAGS"
```

Where **TAGS** is a comma-delimited list of revised `key:value` pairs.

*Note: On Azure, `tkgi update-cluster` cannot remove tags from your IaaS. For more information, see Azure-Specific Tagging Limitations below.*
Remove All Tags From Your Cluster

To remove all of your cluster’s existing tags do the following:

1. Run the following `tkgi update-cluster --tags` on your command line:

```
$ tkgi update-cluster CLUSTER-NAME --tags ""
```

Where `CLUSTER-NAME` is the cluster to remove tags from.

**Review Your Tags**

To review the tags applied to a cluster, run `tkgi cluster`.

For example:

```
$ tkgi cluster my-cluster
Name: my-cluster
Plan Name: large
UUID: 01a234ba-d56e-7f89-01a2-3b4ede5b789
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
Tags: client:tinymegacorp,costcenter:pettycash
```

The `tkgi cluster` function returns only the custom tags you’ve applied to the cluster using the TKGI CLI. To display all of the tags applied to your cluster VMs use your IaaS-provided management console.

**Note:** Do not use the IaaS-provided management console to modify your custom tags. Custom tag alterations you’ve applied via the management console will be overwritten when you next run `tkgi update-cluster`.

**Tagging Rules**

The tagging you apply must adhere to the following rules:

- Tag names and values should not include either `\`, `\` or `/`.
- Surrounding double quotes are required if there are one or more spaces in your tag list, such as a space after a comma delimiter.
- Tag names and values must adhere to the tagging rules of the IaaS hosting your Tanzu Kubernetes Grid Integrated Edition environment.

For information about IaaS-specific tagging rules see the following:

- Azure: See [Use tags to organize your Azure resources](https://docs.microsoft.com/en-us/azure/azure-resource-manager/) in the Azure documentation.
- vSphere: See [vSphere Tags and Attributes](https://docs.vmware.com/en/vsphere/7.0/com/vmware/vsphere_datacenter_client/html/index.html) in the vSphere documentation.

**Tagging Limitations**
Cluster tagging has the following limitations:

Tags Reserved for BOSH

The BOSH layer applies 10 system-level metadata tags to each cluster, including `deployment`, `director`, `id`, `index`, `instance_group`, `job`, and `name`. These reserved tags impose the following limitations:

- The maximum number of custom tags you can apply to a cluster is 10 less than the maximum number of tags supported by your IaaS.
  - For example: Azure limits tagging to a maximum of 50 tags per entity. Therefore, if your Tanzu Kubernetes Grid Integrated Edition environment is hosted on Azure, apply fewer than 40 custom tags to your clusters.
- You cannot set or change BOSH system-level tags using the TKGI CLI. If you use the TKGI CLI to create tags with those reserved names, they are ignored.
- When you try to change system-level tags with the TKGI CLI, the output of `tkgi cluster create-cluster` or `update-cluster` command, but BOSH overrides and ignores these settings.

Azure-Specific Tagging Limitations

The following are the known Azure-specific IaaS tagging limitations:

- `tkgi update-cluster` cannot remove tags from your Azure clusters. This limitation is due to an issue in the Azure CPI for BOSH which is used by `tkgi cli` for Azure IaaS tagging.
- To remove an IaaS tag from an Azure cluster do the following:
  1. Perform the removal steps described in Modify Existing Tags above.
  2. Remove unwanted tags through the Azure portal.

For information about additional Azure-specific tagging limitations see Use tags to organize your Azure resources in the Azure documentation.

vSphere-Specific Tagging Limitations

The following are the known vSphere-specific IaaS tagging limitations:

- `tkgi update-cluster` applies tagging to vSphere entities as vSphere Custom Attributes. This limitation is due to an issue in the vSphere CPI which is used by `tkgi cli` for vSphere IaaS tagging.
  - vSphere Custom Attribute tagging is applied to VMs only. Disks and other resources are not tagged.
  - A vSphere Custom Attribute applied to a single VM is also visible on all other VMs, but as an empty property.

For information about additional vSphere-specific tagging limitations see vSphere Tags and Attributes in the vSphere documentation.

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

Overview
Upgrading a TKGI-provisioned Kubernetes cluster updates the Tanzu Kubernetes Grid Integrated Edition version and the Kubernetes version of the cluster. TKGI-provisioned Kubernetes clusters upgrade when:

- You run `tkgi upgrade-cluster` or `tkgi upgrade-clusters` as described in `Upgrade Clusters` below.

For example, running `tkgi upgrade-cluster` upgrades the cluster you specify to your current version of Tanzu Kubernetes Grid Integrated Edition and to the version of Kubernetes that is included with your current version of Tanzu Kubernetes Grid Integrated Edition.

⚠️ warning: Do not change the number of master/etcd nodes for any plan that was used to create currently-running clusters. Tanzu Kubernetes Grid Integrated Edition does not support changing the number of master/etcd nodes for plans with existing clusters.

---

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:


For information about how to upgrade TKGI-provisioned clusters through the Tanzu Kubernetes Grid Integrated Edition tile, see Verify Errand Configuration in one of the following topics:

- Upgrading Tanzu Kubernetes Grid Integrated Edition (Flannel Networking)
- Upgrading Tanzu Kubernetes Grid Integrated Edition (NSX-T Networking)


---

Copyright © 2020 VMware, Inc. All Rights Reserved.
Prerequisites

Before upgrading TKGI-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 Tanzu Kubernetes Grid Integrated Edition.

2. Verify that your Kubernetes environment is healthy. For information, see Verifying Deployment Health.

3. Install the TKGI CLI. For information, see Installing the TKGI CLI.


Upgrade Clusters

You can 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 Cluster


To upgrade an individual Kubernetes cluster, run the following command:

```
tkgi 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 `tkgi upgrade-cluster` command, see `tkgi upgrade-cluster` in the TKGI CLI documentation.

Upgrade Multiple Clusters


To upgrade a single cluster, see Upgrade a Single Kubernetes Cluster above.

For more information about the `tkgi upgrade-clusters` command, see `tkgi upgrade-clusters` in the TKGI CLI documentation.
Upgrade Clusters in Parallel

To upgrade multiple Kubernetes clusters, run the following command:

```
$ tkgi 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, Tanzu Kubernetes Grid Integrated Edition uses the default `max-in-flight` value of 1 and the clusters are upgraded serially.

For example:

```
$ tkgi 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-001
Finished upgrading cluster: k8-cluster-001
Finished upgrading cluster: k8-cluster-002
Started upgrading cluster: k8-cluster-002
Upgrade task d772aba0-2670-4fba-b26c-044b19d6ab60 is done.

Upgrade 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:

```
$ tkgi 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:

```
$ tkgi 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: cc31a1bb-380a-453f-aff0-835fa1ce6ac

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

Upgrading cluster failed: k8-cluster-dev

Upgrade task cc31a1bb-380a-453f-aff0-835fa1ce6ac is done.

---

**Manage Your Cluster Upgrade Job**

You can use the TKGI CLI to monitor and manage your Tanzu Kubernetes Grid Integrated Edition-provisioned Kubernetes cluster upgrade jobs.

**Monitor Your Clusters**

To review the status of the actions being performed on your clusters, run the following command:

```
tkgi clusters
```

For example:
$ tkgi clusters

Upgrade is available to TKGI Version: 1.8.0-build.16

<table>
<thead>
<tr>
<th>TKGI Version</th>
<th>Name</th>
<th>k8s Version</th>
<th>Plan</th>
<th>Name</th>
<th>UUID</th>
<th>Status</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.8.0-build.16</td>
<td>k8-cluster-000</td>
<td>1.17.5</td>
<td>small</td>
<td></td>
<td>9527ebaa-e2fa-422f-a52b-de3c30e39a4</td>
<td>succeeded</td>
<td>UPGRADE</td>
</tr>
<tr>
<td>1.8.0-build.16</td>
<td>k8-cluster-001</td>
<td>1.17.5</td>
<td>small</td>
<td></td>
<td>9527ebaa-e2fa-422f-a52b-de3c30e39a4</td>
<td>failed</td>
<td></td>
</tr>
<tr>
<td>1.8.0-build.16</td>
<td>k8-cluster-002</td>
<td>1.17.5</td>
<td>small</td>
<td></td>
<td>9527ebaa-e2fa-422f-a52b-de3c30e39a4</td>
<td>in progress</td>
<td>UPGRADE</td>
</tr>
<tr>
<td>1.8.0-build.16</td>
<td>k8-cluster-003</td>
<td>1.17.5</td>
<td>small</td>
<td></td>
<td>9527ebaa-e2fa-422f-a52b-de3c30e39a4</td>
<td>queued</td>
<td></td>
</tr>
</tbody>
</table>

Monitor Your Cluster Upgrade Job

To review the status of your upgrade-clusters job, run the following command:

```
tkgi task TASKID
```

Where **TASKID** is the ID of the task that was returned when you ran `tkgi upgrade-clusters`.

For example:

```
$ tkgi task cc31a1bb-380a-453f-afa0-8355fa1ce6ac

Your upgrade task is: done

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Start time</th>
<th>End time</th>
<th>isCanary</th>
</tr>
</thead>
<tbody>
<tr>
<td>k8-cluster-000</td>
<td>succeeded</td>
<td>Mon, 14 Oct 2019 12:00:00 PDT</td>
<td>Mon, 14 Oct 2019 12:19:54 PDT</td>
<td>true</td>
</tr>
<tr>
<td>k8-cluster-001</td>
<td>failed</td>
<td>Mon, 14 Oct 2019 12:20:00 PDT</td>
<td>---</td>
<td>true</td>
</tr>
</tbody>
</table>
```

Stop Your Cluster Upgrade Job

To cancel a running upgrade-clusters job, run the following TKGI CLI command:

```
tkgi cancel-task TASKID
```

Where **TASKID** is the ID of the task that was returned when you ran `tkgi upgrade-clusters`.

⚠️ **warning**: `tkgi cancel-task` does not cancel cluster upgrades currently in progress. This command only cancels a job’s pending cluster upgrades.

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

In this topic
- Delete Cluster
- Verify Cluster Deletion
- Delete Cluster without Prompt

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to delete a Kubernetes cluster deployed by VMware Tanzu Kubernetes Grid Integrated Edition. Running the command automatically deletes all cluster objects.

`tkgi delete-cluster`

If you are using Tanzu Kubernetes Grid Integrated Edition 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 TKGI CLI.

1. On the command line, run the following command to log in:

   
   ```
   tkgi login -a TKGI-API -u USERNAME -k
   ```

   Where:
   - **TKGI-API** is the domain name for the TKGI API that you entered in Ops Manager > Tanzu Kubernetes Grid Integrated Edition > TKGI API > API Hostname (FQDN). For example, `api.tkgi.example.com`
   - **USERNAME** is your user name.

   See Logging in to Tanzu Kubernetes Grid Integrated Edition for more information about the `tkgi login` command.

   **Note:** If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 TKGI CLI. For information about configuring SAML, see Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider

2. Run `tkgi delete-cluster CLUSTER-NAME` to delete a cluster. Replace **CLUSTER-NAME** with the unique name for your cluster. For example:

   ```
   $ tkgi delete-cluster my-cluster
   ```

3. Confirm cluster deletion by entering `y`, or cancel cluster deletion by entering `n`.

Note: If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 TKGI CLI. For information about configuring SAML, see Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider
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 TKGI CLI.

1. To verify cluster deletion, run `tkgi cluster CLUSTER-NAME`. Replace `CLUSTER-NAME` with the unique name for your cluster.

   For example:

   ```
   $ tkgi cluster my-cluster
   Name: my-cluster
   Plan Name: small
   UUID: 106abc7-5eb-4c54-a800-a32ecf7a593
   Last Action: DELETE
   Last Action State: in progress
   Last Action Description: Instance deletion in progress
   Kubernetes Master Host: my-cluster.tkgi.local
   Kubernetes Master Port: 8443
   Worker Nodes: 3
   Kubernetes Master IP(s): 10.196.219.88
   Network Profile Name:
   ```

   While Tanzu Kubernetes Grid Integrated Edition is deleting the cluster, the value for `Last Action Description` is `Instance deletion in progress`.

2. Continue running the `tkgi cluster CLUSTER-NAME` command to track cluster deletion. The cluster is deleted when the CLI returns `Error: Cluster CLUSTER-NAME not found`.


   ![Note](image)

   If the cluster is not deleted, see Cluster Deletion Fails in Troubleshooting.

Delete Cluster without Prompt

If you do not want the TKGI CLI to prompt you to confirm cluster deletion, use the `--non-interactive` flag.

For example:

```
$ tkgi delete-cluster my-cluster --non-interactive
```

![Note](image)

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

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This section describes how to support Kubernetes clusters provisioned by VMware Tanzu Kubernetes Grid Integrated Edition.

See the following topics:

- Retrieving Cluster Credentials and Configuration
- Managing Cluster Access and Permissions
- Using Admission Control Plugins for Tanzu Kubernetes Grid Integrated Edition Clusters
- Getting Started with VMware Harbor Registry
- Configuring Tanzu Kubernetes Grid Integrated Edition Clusters with Private Docker Registry CA Certificates (Beta)
- PersistentVolume Storage Options on vSphere
- Configuring and Using PersistentVolumes

Note: Tanzu Kubernetes Grid Integrated Edition does not currently support the Kubernetes Service Catalog and the GCP Service Broker.

Please send any feedback you have to pks-feedback@pivotal.io.
Load Balancing and Ingress

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

This section describes how to create and configure load balancers for VMware Tanzu Kubernetes Grid Integrated Edition clusters. See the following topics:

- Creating and Configuring a GCP Load Balancer for Tanzu Kubernetes Grid Integrated Edition Clusters
- Creating and Configuring an AWS Load Balancer for Tanzu Kubernetes Grid Integrated Edition Clusters
- Creating and Configuring an Azure Load Balancer for Tanzu Kubernetes Grid Integrated Edition Clusters
- Configuring Ingress Routing

Please send any feedback you have to pks-feedback@pivotal.io.
Load Balancing and Ingress with NSX-T

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

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

- Monitoring Ingress Resources
- Viewing and Troubleshooting the Health Status of Cluster Network Objects
- Configuring Ingress Resources and Load Balancer Services
- Defining a Network Profile for Load Balancer Sizing
- Scaling the HTTP/S Layer 7 Ingress Load Balancers Using the LoadBalancer CRD
- Defining Network Profiles for the HTTP/S Layer 7 Ingress Controller
- Defining Network Profiles for the TCP Layer 4 Load Balancer
- Using Ingress URL Rewrite

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

In this topic
- Overview
- Monitor the NSX-T Load Balancer Service

Page last updated: 
This topic describes how to monitor the health status of the NSX-T ingress load balancer resources.

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Note: This feature requires NCP v2.5.1 or later.

Overview

The NSX-T Load Balancer is a logical load balancer that handles a number of functions using virtual servers and pools.

The NSX-T load balancer creates a load balancer service for each Kubernetes cluster provisioned by Tanzu Kubernetes Grid Integrated Edition with NSX-T. For each load balancer service, NCP, by way of the CRD, creates corresponding NSXLoadBalancerMonitor objects.

By default Tanzu Kubernetes Grid Integrated Edition deploys the following NSX-T virtual servers for each Kubernetes cluster:

- One TCP layer 4 load balancer virtual server for the Kubernetes API server.
- One TCP layer 4 auto-scaled load balancer virtual server for each Kubernetes service resource of `type: LoadBalancer`.
- Two HTTP/HTTPS layer 7 ingress routing virtual servers. These virtual server are attached to the Kubernetes Ingress Controller cluster load balancer service and can be manually scaled. Tanzu Kubernetes Grid Integrated Edition uses Kubernetes custom resources to monitor the state of the NSX-T load balancer service and scale the virtual servers created for ingress.

For information about scaling TCP layer 4 ingress controller see Defining Network Profiles for the TCP Layer 4 Load Balancer.

For information about configuring layer 7 ingress routing load balancers see Scaling the HTTP/S Layer 7 Ingress Load Balancers Using the LoadBalancer CRD. For information about configuring the layer 7 ingress controller see Defining Network Profiles for the HTTP/S Layer 7 Ingress Controller.

For more information about the NSX-T Load Balancer, see NSX-T Load Balancer in the VMware documentation.

For more information about Kubernetes custom resources, see Custom resources in the Kubernetes documentation.

Monitor 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.

The NSXLoadBalancerMonitor returns statistics showing the number of connections and throughput of the virtual servers for each type
of load balancer.

In addition to connections and throughput statistics the NSXLoadBalancerMonitor CRD returns two health scores for the current performance of load balancers:

- **servicePressureIndex** which represents an overall health score for the NSX-T load balancer service.
- **infraPressureIndex** which represents the health 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:

- 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. For more information see Defining Network Profiles for the TCP Layer 4 Load Balancer.
- If the health score is poor for the layer 7 ingress load balancers, you can use the Scaling the HTTP/S Layer 7 Ingress Load Balancers Using the LoadBalancer CRD to manually scale ingress.

The table below summarizes the actions that you can take based on the health scores.

<table>
<thead>
<tr>
<th>servicePressureIndex</th>
<th>infraPressureIndex</th>
<th>Cluster Manager</th>
<th>Infrastructure Admin</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOW or WARM</td>
<td>LOW or WARM</td>
<td>NONE</td>
<td>Move the LBS from the CRITICAL Edge Node to another Edge Node.</td>
</tr>
<tr>
<td>LOW or WARM</td>
<td>HIGH</td>
<td>Alert infra admin</td>
<td></td>
</tr>
<tr>
<td>HIGH</td>
<td>LOW or WARM</td>
<td>Resolve the LBS health score by Scaling the HTTP/S Layer 7 Ingress Load Balancers Using the LoadBalancer CRD and, if necessary, by increasing the size of the LBS using Defining Network Profiles for the TCP Layer 4 Load Balancer.</td>
<td>NONE</td>
</tr>
<tr>
<td>HIGH</td>
<td>HIGH</td>
<td>Alert infra admin; Resolve the LBS health score by Scaling the HTTP/S Layer 7 Ingress Load Balancers Using the LoadBalancer CRD and, if necessary, by increasing the size of the LBS using Defining Network Profiles for the TCP Layer 4 Load Balancer.</td>
<td>Move the LBS from the CRITICAL Edge Node to another Edge Node.</td>
</tr>
</tbody>
</table>

Monitor Your NSX-T Load Balancer Service Using the NSXLoadBalancerMonitor CRD

To monitor your NSX-T Load Balancer Service using the NSXLoadBalancerMonitor CRD, complete the following procedure.

1. To view the NSXLoadBalancerMonitor CRD, run the following command:

   ```bash
   kubectl get crd
   ```

2. To determine the UUID of the NSX-T load balancer deployed for the cluster, run the following command:

   ```bash
   kubectl get nsxlbmonitors
   ```
3. To view statistics, throughput, and health score for all virtual servers deployed by a specific load balancer service, run the following command:

```
kubectl describe nsxlbmonitors UUID-OF-LOAD-BALANCER
```

Where `UUID-OF-LOAD-BALANCER` is your load balancer’s UUID.

For example:

```
$ kubectl describe nsxlbmonitor f61a8cec-28eb-4b0c-bf4a-906f3ce2d8e6
Name:    f61a8cec-28eb-4b0c-bf4a-906f3ce2d8e6
Namespace: <none>
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:
  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>
```

Please send any feedback you have to pks-feedback@pivotal.io.
Viewing and Troubleshooting the Health Status of Cluster Network Objects

In this topic

About the NSX Errors CRD
Errors Reported by the NSX Errors CRD
NSX Errors CRD Example

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

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 TKGI-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 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.

```bash
# 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.

```bash
# 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.

```bash
# 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-abd8-000c29e02b4c
  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.
Configuring Ingress Resources and Load Balancer Services

In this topic

Kubernetes Ingress Rules
The NSX-T Load Balancer Service

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes example configurations for ingress routing (Layer 7) and load balancing (Layer 4) for Kubernetes clusters deployed by VMware Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T integration.

Note: The examples in this topic are based on NCP v2.3.2.

Kubernetes Ingress Rules

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.

You define ingress resource configuration in the manifest of your Kubernetes deployment. 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. You can use wildcard DNS entries to route traffic to the exposed ingress resource.

When you define two ingress rules with the same hostname, include both the hostname and path in the ingress rules to avoid ambiguity.

Rules:

- If multiple ingress rules use the same hostname and the same path, the first rule you create takes priority.
- If an ingress rule that includes only a hostname precedes a rule that includes both the same hostname and a path, the first rule takes priority.

For example:

- The following NSX ingress rule includes both a host and a path specification. The rule matches `host: test.com` and `path:/testpath` in the incoming request:

  Ingress Rule Example 1
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: svc-ingress1
spec:
rules:
- host: test.com
  http:
    paths:
    - path: /testpath
      backend:
        serviceName: svc1
        servicePort: 80

The following NSX ingress rule includes only a host specification. The rule matches all requests where the host and path fields match.

### Ingress Rule Example 2

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: svc-ingress2
spec:
rules:
- host: test.com
  http:
    paths:
    - path: /testpath
      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 **svc-ingress2** serves the **test.com/testpath** URI because inbound requests hit the **host: test.com** NSX ingress rule first.

For more information about Kubernetes ingress resources, see Ingress in the Kubernetes documentation.

### The NSX-T Load Balancer Service

NSX-T supports autoscaling, which spins up a new Kubernetes **type: LoadBalancer** service if the previous one has reached its scale limit. The NSX-T load balancer that is automatically provisioned by Tanzu Kubernetes Grid Integrated Edition 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.

The following is the format for the Kubernetes **LoadBalancer** service definition:
kind: Service
apiVersion: v1
metadata:
  name: SERVICE-NAME
spec:
type: LoadBalancer
selector:
  app: APP-NAME
ports:
- protocol: PROTOCOL
  port: PORT
targetPort: TARGET-PORT
name: PORT-NAME

Where:

- **SERVICE-NAME** is the name for your load balancer service.
- **APP-NAME** is the name of your app serviced by the load balancer service.
- **PROTOCOL** (Optional) is the network protocol to service. If the protocol is not specified it defaults to TCP. For more information about supported protocols, see Supported protocols in the Kubernetes documentation.
- **PORT** is the listening port. An integer value is supported. For example, 80.
- **TARGET-PORT** is the target port. Either an integer or a string value is supported. For example, 8080 or http.
- **PORT-NAME** (Optional) is the port name. Kubernetes requires the port name be specified for multi-port services.

For example, the following is a service definition for an Tanzu Kubernetes Grid Integrated Edition-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
```

**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.

For more information about the Kubernetes Service definition see Type LoadBalancer in the Kubernetes documentation.

When deploying a Kubernetes LoadBalancer service, NSX-T automatically creates a new virtual IP address (VIP) on the existing load balancer.

Please send any feedback you have to pks-feedback@pivotal.io.
Scaling the HTTP/S Layer 7 Ingress Load Balancers Using the LoadBalancer CRD

In this topic
Overview
Prerequisites
Scale Ingress Load Balancer Resources

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to scale ingress resources.

Note: This feature requires NCP v2.5.1 or later.

Overview

The NSX-T Load Balancer is a logical load balancer that handles a number of functions using virtual servers and pools.

The NSX-T load balancer creates a load balancer service for each Kubernetes cluster provisioned by Tanzu Kubernetes Grid Integrated Edition with NSX-T. For each load balancer service, NCP, by way of the CRD, creates corresponding NSXLoadBalancerMonitor objects.

By default Tanzu Kubernetes Grid Integrated Edition deploys the following NSX-T virtual servers for each Kubernetes cluster:

- One TCP layer 4 load balancer virtual server for the Kubernetes API server.
- One TCP layer 4 auto-scaled load balancer virtual server for each Kubernetes service resource of `type: LoadBalancer`.
- Two HTTP/HTTPS layer 7 ingress routing virtual servers. These virtual server are attached to the Kubernetes Ingress Controller cluster load balancer service and can be manually scaled. Tanzu Kubernetes Grid Integrated Edition uses Kubernetes custom resources to monitor the state of the NSX-T load balancer service and scale the virtual servers created for ingress.

For information about configuring layer 7 ingress routing load balancers see Determine Your Load Balancer’s Status, below. For information about configuring the layer 7 ingress controller see Defining Network Profiles for the HTTP/S Layer 7 Ingress Controller.

For information about configuring TCP layer 4 ingress controller see Defining Network Profiles for the TCP Layer 4 Load Balancer.

For more information about the NSX-T Load Balancer, see NSX-T Load Balancer in the VMware documentation.

For more information about Kubernetes custom resources, see Custom resources in the Kubernetes documentation.

Prerequisites

Before scaling your ingress load balancers you should understand your load balancer’s status. Use the NSXLoadBalancerMonitor CRD
to monitor your NSX-T load balancer service, including traffic, usage and health score information. The NSXLoadBalancerMonitor CRD provides information for the health of the NSX-T load balancer service, and the NSX-T Edge Node running the load balancer.

For more information about monitoring using the NSXLoadBalancerMonitor CRD see Monitoring Ingress Resources.

Scale Ingress Load Balancer Resources

The LoadBalancer CRD provides you with an interactive method to scale the load balancer for ingress routing.

Create a New Ingress Load Balancer

Use the LoadBalancer CRD to create a new ingress load balancer.

1. To configure a new ingress load balancer, configure a new YAML file as follows:

   ```yaml
   apiVersion: vmware.com/v1alpha1
   kind: LoadBalancer
   metadata:
     name: LB-NAME
   spec:
     httpConfig: HTTP-CONFIG
     virtualIP: IP-ADDRESS
     port: PORT
     tls:
       port: TLS-PORT
       secretName: SECRET-NAME
       secretNamespace: SECRET-NAMESPACE
       xForwardedFor: FORWARD-TYPE
     affinity:
       type: IP-SOURCE
       timeout: TIMEOUT
     size: SIZE
     virtualNetwork: NETWORK-NAME
   status:
     httpVirtualIP: V-IP-ADDRESS
   ```

Where:

- **LB-NAME** is the display name of the loadBalancer.
- **HTTP-CONFIG** (Optional) is the config to support http/https route on the loadBalancer. Set as `httpConfig: {}` to apply default settings.
- **IP-ADDRESS** (Optional) is the virtual IP address. Defaults to `auto_allocate`.
- **PORT** (Optional) is the port. Defaults to `80`.
- **TLS-PORT** (Optional) is the TLS port. Defaults to `443`.
- **SECRET-NAME** (Optional) is the TLS secret name. Defaults to `nil`.
- **SECRET-NAMESPACE** (Optional) is the TLS secret namespace. Defaults to `nil`. You must deploy the new ingress load balancer in the same namespace where you deploy the ingress resource.
- **FORWARD-TYPE** (Optional) is the forward type. Supported values are: `INSERT` and `REPLACE`. Defaults to `nil`.
- **IP-SOURCE** (Optional) is the source IP. Supported values are: `sourceIP` and `cookie`.
- **TIMEOUT** (Optional) is the connection timeout. Defaults to `10800`.
- **SIZE** (Optional) is the ingress load balancer size. Supported values are: `SMALL` and `MEDIUM`. Defaults to `SMALL`.
- **NETWORK-NAME** (Optional) is the virtual network name. Defaults to `nil`.
- **V-IP-ADDRESS** is the external IP address for http/https virtual server. The external IP address can be auto-allocated or user specified.

2. To create a new ingress load balancer run the following command:
kubectl apply -f YAML-FILE

Where **YAML-FILE** is the filename of the load balancer configuration YAML file.

For example:

```yaml
# kubectl apply -f lb.yaml
apiVersion: vmware.com/v1alpha1
kind: LoadBalancer
metadata:
  name: cluster1_lbs0
spec:
  httpConfig:
    virtualIP:
      port: 233
    tls:
      port: 2333
      secretName: default_secret
      secretNamespace: default
      xForwardedFor: INSERT
      affinity:
        type: source_ip
        timeout: 100
      size: MEDIUM
    virtualNetwork: virtualnetwork1
status:
  httpVirtualIP: <realized external ip>
```

Configure Your Kubernetes Ingress Resource to Use the New Ingress Load Balancer

Annotate the Kubernetes ingress resource with the newly created ingress load balancer. NCP will attach the ingress rules to the scaled out load balancer.

1. To configure a Kubernetes ingress resource with the new ingress load balancer, configure a new YAML file as follows:

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ING-NAME
  annotations:
    nsx/loadbalancer: LB-NAME
spec:
  rules:
  - host: HOST-NAME
    http:
      paths:
      - path: HTTP-PATH
        backend:
          serviceName: SERVICE-NAME
          servicePort: SERVICE-PORT
```

Where:
- **ING-NAME** is the name of the ingress resource.
- **LB-NAME** is the display name of the loadBalancer.
- **HOST-NAME** is the host name.
- **HTTP-PATH** is the HTTP path.
SERVICE-NAME is the http backend service name.

SERVICE-PORT is the http backend service port.

2. To annotate the Kubernetes ingress resource with the newly created ingress load balancer, run the following command:

```bash
cubectl apply -f YAML-FILE
```

Where YAML-FILE is the filename of a the Kubernetes ingress resource configuration YAML file.

For example:

```yaml
# kubectl apply -f ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: svc-ingress1
  annotations:
    nsx/loadbalancer: cluster1_lbs0
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.
Using Ingress URL Rewrite

In this topic

About Support for URL Rewrite for Ingress Resources

URL Rewrite Example

---

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

---

**Page last updated:**

This topic describes how to perform URL rewrite for Kubernetes ingress resources.

---

About Support for URL Rewrite for Ingress Resources

Tanzu Kubernetes Grid Integrated Edition 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.

```yaml
apiVersion: networking.k8s.io/v1
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.

---

Copyright © 2020 VMware, Inc. All Rights Reserved.
Creating and Configuring an AWS Load Balancer for Tanzu Kubernetes Grid Integrated Edition Clusters

In this topic

Prerequisite
Configure AWS Load Balancer
  Step 1: Define Load Balancer
  Step 2: Assign Security Groups
  Step 3: Configure Security Settings
  Step 4: Configure Health Check
  Step 5: Add EC2 Instances
(Optional) Step 6: Add Tags
  Step 7: Review and Create the Load Balancer
  Step 8: Create a Cluster
  Step 9: Point the Load Balancer to All Master VMs
Reconfigure AWS Load Balancer

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to configure a Amazon Web Services (AWS) load balancer for your VMware Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition deployment see Load Balancers in TKGI.

You can use an AWS Tanzu Kubernetes Grid Integrated Edition cluster load balancer to secure and facilitate access to a Tanzu Kubernetes Grid Integrated Edition cluster from outside the network. You can also reconfigure your AWS Tanzu Kubernetes Grid Integrated Edition cluster load balancers.

Using an AWS Tanzu Kubernetes Grid Integrated Edition cluster load balancer is optional, but adding one to your Kubernetes cluster can make it easier to manage the cluster using the TKGI API and kubectl.

Note: If Kubernetes master node VMs are recreated for any reason, you must reconfigure your AWS TKGI cluster load balancers to point to the new master VMs.

Prerequisite

The version of the TKGI CLI you are using must match the version of the Tanzu Kubernetes Grid Integrated Edition tile that you are installing.

Note: This procedure uses example commands which you should modify to represent the details of your Tanzu Kubernetes
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:
   a. **Load Balancer name**: Name the load balancer. VMware 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`.
   b. **Create LB inside**: Select the VPC where you installed Ops Manager.
   c. **Create an internal load balancer**: Do not enable this checkbox. The cluster load balancer must be internet-facing.
7. 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.
8. Under Select Subnets, select the public subnets for your load balancer in the availability zones where you want to create the load balancer.

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 **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 `tkgi create-cluster` command. For example:
$ k8s 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.

Please send any feedback you have to pks-feedback@pivotal.io.
Creating and Configuring an Azure Load Balancer for Tanzu Kubernetes Grid Integrated Edition Clusters

In this topic
Prerequisites
Create and Configure a Load Balancer
   Create Load Balancer
   Create Backend Pool
   Create Health Probe
   Create Load Balancing Rule
   Create Inbound Security Rule
   Verify Hostname Resolution
Reconfigure Load Balancer

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to create and configure an Azure load balancer for your VMware Tanzu Kubernetes Grid Integrated Edition 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 TKGI 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 Tanzu Kubernetes Grid Integrated Edition 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 TKGI API 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition deployed.
   g. **Resource group**: Select the resource group which has Tanzu Kubernetes Grid Integrated Edition deployed.
   h. **Location**: Select the location group which has Tanzu Kubernetes Grid Integrated Edition 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 TKGI 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 TKGI API is deployed. If you deployed Tanzu Kubernetes Grid Integrated Edition 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.

Please send any feedback you have to pks-feedback@pivotal.io.
Creating and Configuring a GCP Load Balancer for Tanzu Kubernetes Grid Integrated Edition Clusters

In this topic
Overview
Prerequisites
Configure GCP Load Balancer
  Create a GCP Load Balancer
  Create a DNS Entry
  Create the Cluster
  Configure Load Balancer Back End
  Create a Network Tag
  Create Firewall Rules
Access the Cluster
Reconfigure Load Balancer

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to configure a Google Cloud Platform (GCP) load balancer for a Kubernetes cluster deployed by VMware Tanzu Kubernetes Grid Integrated Edition.

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 TKGI-deployed cluster from outside the network using the TKGI 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 TKGI 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 TKGI API. For more information, see Creating a GCP Load Balancer for the TKGI API.
- The version of the TKGI CLI you are using must match the version of the Tanzu Kubernetes Grid Integrated Edition tile that you are installing.

Configure GCP Load Balancer

Follow the procedures in this section to create and configure a load balancer for TKGI-deployed Kubernetes clusters using GCP. Modify the example commands in these procedures to match your Tanzu Kubernetes Grid Integrated Edition installation.

Create a GCP Load Balancer

To create a GCP load balancer for your TKGI 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 TKGI 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 Tanzu Kubernetes Grid Integrated Edition manually:
Select the zone you used when you created the TKGI API DNS entry. See the Create a DNS Entry section in Creating a GCP Load Balancer for the TKGI API.

If you installed Tanzu Kubernetes Grid Integrated Edition using Terraform:
Run terraform output and locate the value for dnsManagedZone.

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 TKGI cluster load balancer 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 TKGI cluster hostname from the above step as the external hostname when you run the tkgi 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 TKGI by running the following command:

   tkgi login -a TKGI-API -u USERNAME -k

   Where:
   • TKGI-API is the domain name for the TKGI API that you entered in Ops Manager > Tanzu Kubernetes Grid Integrated Edition > TKGI API > API Hostname (FQDN). For example, api.tkgi.example.com.
   • USERNAME is your user name.

   Note: If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 TKGI CLI. For information about configuring SAML, see Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider

   2. Locate the master node IP addresses by running the following command:

   tkgi cluster CLUSTER-NAME
Where \texttt{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.


4. From the sidebar menu, navigate to \texttt{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 \texttt{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.

   \textbf{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.

\textbf{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 \texttt{Compute Engine > VM instances}.

2. Filter to find the master instances of your cluster. Type \texttt{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 \texttt{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 Tanzu Kubernetes Grid Integrated Edition 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 `tkgi get-credentials CLUSTER-NAME`.

   Where `CLUSTER-NAME` is the unique name for your cluster.

   For example:

   ```
   $ tkgi get-credentials tkgi-example-cluster
   Fetching credentials for cluster tkgi-example-cluster.
   Context set for cluster tkgi-example-cluster.
   ```

   The `tkgi get-credentials` command creates a local `kubeconfig` that enables you to manage the cluster. For more information about the `tkgi get-credentials` command, see [Retrieving Cluster Credentials and Configuration](#).

   **Note:** If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 [TKGI CLI](#). For information about configuring SAML, see [Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider](#).

2. Run `kubectl cluster-info` to confirm you can access your cluster using the Kubernetes CLI.

   See [Managing Tanzu Kubernetes Grid Integrated Edition](#) 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.
Configuring Ingress Routing

In this topic
- Overview
- Prerequisites
- Deploy a Kubernetes Ingress Controller
- Configure DNS
- (Optional) Configure TLS
- Deploy an App to the Cluster

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic provides resources for configuring an ingress controller on VMware Tanzu Kubernetes Grid Integrated Edition.

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 TKGI-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 TKGI cluster in the Knowledge Base.

To deploy an open source ingress controller to a TKGI 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:

   ```
   tkgi get-credentials CLUSTER-NAME
   ```

   Where `CLUSTER-NAME` is the name of your TKGI-deployed Kubernetes cluster.

   For example:

   ```
   $ tkgi get-credentials tkgi-example-cluster
   Fetching credentials for cluster tkgi-example-cluster.
   Context set for cluster tkgi-example-cluster.
   You can now switch between clusters by using:
   $ kubectl config use-context <cluster-name>
   ```

   Note: If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 TKGI CLI. For information about configuring SAML, see Connecting Tanzu Kubernetes Grid Integrated Edition 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.
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](https://istio.io) 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:

   ```bash
   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:

   ```bash
   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:

   ```bash
   openssl req -x509 \
   -nodes -newkey rsa:4096 \
   -keyout KEY-PATH.pem \
   -out CERT-PATH.pem \
   -days 365 \
   -subj "CN=*.TKGI.EXAMPLE.COM"
   ```

   Where:
2. Upload your TLS certificate and key to your ingress controller namespace by running the following command:

```bash
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:

```bash
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:

   ```bash
   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:

   ```bash
   kubectl create -f INGRESS-CONTROLLER.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:

   ```yaml
   spec:
     tls:
     - secretName: INGRESS-CERT
       rules:
       - host: INGRESS.TKGI.EXAMPLE.COM
   ```
Where:

- **INGRESS-CERT** is the name of the Kubernetes secret that contains your TLS certificate and key pair.
- **INGRESS.TKGI.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.
Using Admission Control Plugins for Tanzu Kubernetes Grid Integrated Edition Clusters

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

This section describes how to enable, use, and disable admission control plugins for VMware Tanzu Kubernetes Grid Integrated Edition clusters.

For more information about Admission Controllers, see Using Admission Controllers in the Kubernetes documentation.

Tanzu Kubernetes Grid Integrated Edition supports three admission control plugins. See the following topics for details on each:

- Enabling the DenyEscalatingExec Admission Plugin
- Enabling the SecurityContextDeny Admission Plugin for Tanzu Kubernetes Grid Integrated Edition Clusters

To disable one or more Admission Plugins, refer to the following topic:

- Disabling Admission Control Plugins for Tanzu Kubernetes Grid Integrated Edition Clusters

Please send any feedback you have to pks-feedback@pivotal.io.
Enabling and Configuring Pod Security Policies

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 Kubernetes 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 Kubernetes documentation.

Default Pod Security Policies in TKGI

Tanzu Kubernetes Grid Integrated Edition ships with two default PSPs: PKS Privileged and PKS Restricted.

<table>
<thead>
<tr>
<th>PSP</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKS Privileged</td>
<td>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.</td>
</tr>
<tr>
<td>PKS Restricted</td>
<td>Restricts privileged access to pod containers.</td>
</tr>
</tbody>
</table>

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.
Enabling Pod Security Policies in TKGI

Enabling PSPs is done during configuration of Tanzu Kubernetes Grid Integrated Edition in the Plan section of the tile configuration. Refer to the TKGI 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.

**New Installations of TKGI: PSPs Are Not Enabled for Any Plan**

For new Tanzu Kubernetes Grid Integrated Edition 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 TKGI.

**Upgrade of TKGI: PSPs Must Be Enabled per Plan**

On upgrade of Tanzu Kubernetes Grid Integrated Edition, 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 TKGI Command Line Interface (TKGI CLI). For instructions on upgrading individual Kubernetes clusters, see Upgrading Clusters.

**Configuring the pks-restricted PSP for Developers to Use with TKGI**

This section describes how to define and configure a pod security policy for developers to use to access a TKGI 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: policy/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.alpha.kubernetes.io/allowedProfileNames": "docker/default", "seccomp.security.alpha.kubernetes.io/defaultProfileName": "docker/default"}
    "creationTimestamp": "2019-03-28T02:33:07Z"
    "name": "pks-restricted"
    "namespace": "default"
    "uid": "d25bb0eb-5101-11e9-abc1-0a834658b12e"

spec:
  allowPrivilegeEscalation: false
  fsGroup:
    ranges:
      - min: 1
        max: 65535
  requiredDropCapabilities: 
    - ALL
  runAsUser:
    rule: MustRunAsNonRoot
  seLinux:
    rule: RunAsAny
  supplementalGroups:
    ranges:
      - min: 1
        max: 65535
    rule: MustRunAs
  volumes:
    - configMap
    - emptyDir
    - downwardAPI
    - secret
    - 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: pks-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 of 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 TKGI, 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:
**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 `pks-restricted` `ClusterRole`, which provides permissions to use the `pks-restricted` `PSP`:

**Example of binding a service account user:**

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: psp:restricted
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: psp:restricted
subjects:
- kind: ServiceAccount
  name: cluster-user
```

**Example of binding an externally authenticated user named jane:**

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: psp:restricted
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: psp:restricted
subjects:
- kind: User
  name: jane
```

**Example of binding externally authenticated group members:**

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: psp:restricted
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: psp:restricted
subjects:
- kind: Group
  name: developers
```

**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
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>PRIV</th>
<th>CAPS</th>
<th>SELINUX</th>
<th>RUNAUSER</th>
<th>FGROUP</th>
<th>RUGROUP</th>
<th>READONLYROOTFS</th>
<th>VOLUMES</th>
</tr>
</thead>
<tbody>
<tr>
<td>pks-privileged</td>
<td>true</td>
<td>*</td>
<td>RunAsAny</td>
<td>RunAsAny</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>Disk,glusterfs,iscsi,nfs,persistentVolumeClaim,projected,portworxVolume,quobyte,rbd,scaleIO,secret,storageos,vsphereVolume</td>
</tr>
<tr>
<td>pks-restricted</td>
<td>false</td>
<td>RunAsAny</td>
<td>RunAsAny</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>configMap,emptyDir,projected,secret,downtowndAPpersistentVolumeClaim</td>
</tr>
</tbody>
</table>

To view the details of a particular PSP:

```
kubectl describe psp <pss-name>
```
For example:

```bash
kubectl describe psp pks-privileged
Name: pks-privileged
Settings:
  Allow Privileged: true
  Allow Privilege Escalation: true
  Default AddCapabilities: <none>
  Required DropCapabilities: <none>
  Allowed Volume Types: <none>
  Allow Host Network: true
  Allow Host PID: 0-65535
  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>
```

```bash
kubectl describe psp pks-restricted
Name: pks-restricted
Settings:
  Allow Privileged: false
  Allow Privilege Escalation: false
  Default AddCapabilities: <none>
  Required DropCapabilities: ALL
  Allowed Volume Types: <none>
  Allow Host Network: false
  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:

```bash
kubectl get rolebinding
```

To view the cluster role bindings:

```bash
kubectl get clusterrolebindings
```

In the results, the `service-account` is a service account.

<table>
<thead>
<tr>
<th>NAME</th>
<th>AGE</th>
<th>SECRET</th>
</tr>
</thead>
<tbody>
<tr>
<td>9a605c3d-b2f3-4961-b2a4-804885d85eba-cluster-admin</td>
<td>23s</td>
<td></td>
</tr>
<tr>
<td>9e495241-65b0-43bb-ac3a-120cb4ee6533-cluster-admin</td>
<td>2d1h</td>
<td></td>
</tr>
<tr>
<td>c990e9cc-29fc-4696-8673-fc69a93904f4-cluster-admin</td>
<td>2d1h</td>
<td></td>
</tr>
<tr>
<td>cluster-admin</td>
<td>8d</td>
<td></td>
</tr>
<tr>
<td>system:basic-user</td>
<td>8d</td>
<td></td>
</tr>
<tr>
<td>telemetry-agent</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To view service accounts:

```bash
kubectl get sa
```

To view the YAML for a cluster role binding:

```bash
kubectl get clusterrolebinding 9a605c3d-b2f3-4961-b2a4-804885d85eba-cluster-admin -o yaml
```
To view a cluster role:

```
kubectl get clusterrole cluster-admin
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>cluster-admin</td>
<td>8d</td>
</tr>
</tbody>
</table>

To view the YAML for a cluster role:

```
kubectl get clusterrole cluster-admin -o yaml
```

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 ReplicationController, 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

Tanzu Kubernetes Grid Integrated Edition 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
About the SecurityContextDeny Admission Plugin
When to Enable the SecurityContextDeny Admission Plugin
Impact of Enabling the SecurityContextDeny Admission Plugin
Enabling the SecurityContextDeny Admission Plugin

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 TKGI Command Line Interface (TKGI CLI).
Enabling the SecurityContextDeny Admission Plugin

To enable the SecurityContextDeny admission plugin:

1. In the TKGI tile, select the desired Plan, such as Plan 1.
2. At the bottom of the configuration panel, select the SecurityContextDeny option.
3. Click Save.
5. For Tanzu Kubernetes Grid Integrated Edition, 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 TKGI Command Line Interface (TKGI 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

About the DenyEscalatingExec Admission Plugin
When to Enable the DenyEscalatingExec Admission Plugin
Impact of Enabling the DenyEscalatingExec Admission Controller
Enabling the DenyEscalatingExec Admission Plugin

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

Topic provided by VMware

This section describes how and when to enable the DenyEscalatingExec admission controller for VMware Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition tile, select the desired Plan, such as Plan 1.

2. At the bottom of the configuration panel, select the DenyEscalatingExec option.

3. Click Save.


5. For Tanzu Kubernetes Grid Integrated Edition, 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 TKGI Command Line Interface (TKGI 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 Tanzu Kubernetes Grid Integrated Edition Clusters

In this topic
Disabling a Single Admission Control Plugin
Disabling an Orphaned Admission Control Plugin

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

Topic provided by VMware

This section describes how to disable one or more admission control plugins for VMware Tanzu Kubernetes Grid Integrated Edition clusters. For more information, see Using Admission Control Plugins for Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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 TKGI Command Line Interface (TKGI 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 Tanzu Kubernetes Grid Integrated Edition tile by doing one of the following options:
   a. Option 1: Use the Ops Manager API:
      i. In a terminal, run the following command:


      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 Tanzu Kubernetes Grid Integrated Edition tile.

      ii. Copy the BOSH deployment name including the GUID from the URL:

         ![Ops Manager UI](image)

         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 Tanzu Kubernetes Grid Integrated Edition 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:

```bash
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 Tanzu Kubernetes Grid Integrated Edition section and select **SEE CHANGES**.
   d. Verify that the admission control plugins are displayed as removed in the **Manifest** section. For example:

```
manifest:
  instance_groups:
    - name: pivotal-container-service
      jobs:
        - name: pks-nsx-t-osb-proxy
          properties:
            plans:
              - name: small
                properties:
                  enabled_admission_plugins:
                  - pod_security_policy
                  - deny_escalating_exec
  service_catalog:
    plans:
      - name: small
        properties:
          enabled_admission_plugins:
          - pod_security_policy
          - deny_escalating_exec
```

Copyright © 2020 VMware, Inc. All Rights Reserved.
7. Click **Apply Changes**.

Please send any feedback you have to pks-feedback@pivotal.io.
Retrieving Cluster Credentials and Configuration

In this topic
Retrieve Cluster Credentials
Run kubectl Commands

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to use the `tkgi get-credentials` command in VMware Tanzu Kubernetes Grid Integrated Edition using the TKGI Command Line Interface (TKGI CLI).

The `tkgi 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 `tkgi get-credentials CLUSTER-NAME`, TKGI sets the context to the cluster you provide as the `CLUSTER-NAME`. TKGI 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 `tkgi get-credentials`. For example:

```
$ KUBECONFIG=/path/to/my-cluster.config tkgi 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:

```
tkgi login -a TKGI-API -u USERNAME -k
```

Where:

- `TKGI-API` is the domain name for the TKGI API that you entered in Ops Manager > Tanzu Kubernetes Grid Integrated Edition > TKGI API > API Hostname (FQDN). For example, `api.tkgi.example.com`.
- `USERNAME` is your user name.

See Logging in to Tanzu Kubernetes Grid Integrated Edition for more information about the `tkgi login` command.
Run the following command:

```
$ tkgi get-credentials CLUSTER-NAME
```

Where `CLUSTER-NAME` is the unique name for your cluster.

For example:

```
$ tkgi get-credentials tkgi-example-cluster
Fetching credentials for cluster tkgi-example-cluster.
Context set for cluster tkgi-example-cluster.
You can now switch between clusters by using:
$ kubectl config use-context <cluster-name>
```

**Note:** If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 TKGI CLI. For information about configuring SAML, see Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider.

---

**Note:** If you enable OpenID Connect (OIDC) in the Tanzu Kubernetes Grid Integrated Edition tile, TKGI requires your password to run the `tkgi get-credentials CLUSTER-NAME` command. This allows TKGI to retrieve valid tokens for the `kubectl config` file. You can provide your password at the prompt or as the `TKGI_USER_PASSWORD` environment variable. For more information, see the Configure OpenID Connect section of Installing Tanzu Kubernetes Grid Integrated Edition for your IaaS.

---

**Note:** If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 TKGI CLI. For information about configuring SAML, see Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider.

---

### Run kubectl Commands

After TKGI populates your `kubectl` config, 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
Overview
Example Workflow
Prerequisites
Grant Cluster Access to a User
Obtain Cluster Access as a User
Grant Cluster Access to a Group

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to grant Kubernetes cluster access and namespace permissions to Kubernetes users in VMware Tanzu Kubernetes Grid Integrated Edition.

Overview

Tanzu Kubernetes Grid Integrated Edition admin users can grant Kubernetes users, such as developers, permissions to specific clusters.

If you are an Tanzu Kubernetes Grid Integrated Edition 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 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 user or group access to an Tanzu Kubernetes Grid Integrated Edition-provisioned cluster, Kubernetes users can connect to the cluster through the Kubernetes CLI (kubectl). Kubernetes users cannot create, resize, or delete clusters.

Example Workflow

The following diagram outlines the workflow to grant cluster access to users who belong to an identity provider group:
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 Tanzu Kubernetes Grid Integrated Edition, you must have the following:

- Fully qualified domain name (FQDN) of your TKGI deployment.
- OpenID Connect (OIDC) provider for your Kubernetes clusters, configured using one or both of the following:
  - Global OIDC provider configuration for all clusters in `Ops Manager Installation Dashboard > Tanzu Kubernetes Grid Integrated Edition > Settings > UAA > Configure created clusters to use UAA as the OIDC provider`. For instructions, see `UAA in the Installing topic for your IaaS`.
  - Custom OIDC provider configuration for individual clusters through a Kubernetes profile. For instructions, see `Add an OIDC Provider`.

**Grant Cluster Access to a User**

To grant cluster access to a user, do the following:

1. Log in to Tanzu Kubernetes Grid Integrated Edition by running following command:

   ```markdown
tkig login -u USERNAME -p PASSWORD -a TKGI-API --ca-cert CERT-PATH
   ```

   Where:
   - **USERNAME** is your cluster admin username:
     - If you use LDAP or SAML for UAA, this is your LDAP or SAML username.
     - If you do not use LDAP or SAML for UAA, this is your UAA username.
• PASSWORD is your cluster admin password.
• TKGI-API is the FQDN you use to access the TKGI API.
• CERT-PATH is the path to your root CA certificate. Provide the certificate to validate the TKGI API certificate with SSL.

**Note:** If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 **TKGI CLI**. For information about configuring SAML, see **Connecting Tanzu Kubernetes Grid Integrated Edition 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:

   ```bash
tkgi get-credentials CLUSTER-NAME
   ```

   This step creates a **ClusterRoleBinding** for the cluster admin.

   **Note:** If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 **TKGI CLI**. For information about configuring SAML, see **Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider**

3. When prompted, re-enter your password.

4. Create a YAML file for either **Role** or **ClusterRole**. Use the following example as a template:

   ```yaml
   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:
     ```yaml
     ["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:

   ```bash
   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:

```yaml
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.
  - If you use LDAP or SAML for UAA, this is the LDAP or SAML username.
  - If you do not use LDAP or SAML for UAA, this is the UAA username.

**Note:** If you configured an OIDC username prefix in Ops Manager Installation Dashboard > Tanzu Kubernetes Grid Integrated Edition > Settings > UAA or in a Kubernetes profile, you must prepend `USERNAME` with the prefix you configured. For more information, see UAA in the Installing topic for your IaaS and Add an OIDC Provider.

- **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 created in the previous step.

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:

   - TKGI API FQDN
   - Cluster name

**Obtain Cluster Access as a User**

To obtain access to a Tanzu Kubernetes Grid Integrated Edition-provisioned cluster, the end user must do the following:

1. Fetch the kubeconfig file by running one of the following command:

   - If you want to validate the TKGI API certificate with SSL, run the following command:

     ```
tkgi get-kubeconfig CLUSTER-NAME -u USERNAME -a TKGI-API --ca-cert CERT-PATH
     ```
Where:

- **CLUSTER-NAME** is the cluster name provided by the cluster admin.
- **USERNAME** is the Kubernetes end user username.

- If the Tanzu Kubernetes Grid Integrated Edition deployment you are targeting uses LDAP or SAML for UAA, this is the LDAP or SAML username.
- If your Tanzu Kubernetes Grid Integrated Edition deployment you are targeting does not use LDAP or SAML for UAA, this is the UAA username.

- **TKGI-API** is the FQDN you use to access the TKGI API.
- **CERT-PATH** is the path to your root CA certificate. Provide the certificate to validate the TKGI API certificate with SSL.

For example:

```bash
$ tkgi get-kubeconfig my-cluster -u naomi -a api.tkgi.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:

```bash
tkgi get-kubeconfig CLUSTER-NAME -u USERNAME -a TKGI-API -k
```

Where `-k` is the shortcut flag to skip SSL validation.

For example:

```bash
$ tkgi get-kubeconfig my-cluster -u naomi -a api.tkgi.example.com -k
```

**Note:** If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 **TKGI CLI**. For information about configuring SAML, see [Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider](#).

2. When prompted, enter your password.

3. The TKGI 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
  - 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 a UAA or an identity provider group by creating a ClusterRoleBinding or RoleBinding for that group. You can grant access to an identity provider group only 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 > Tanzu Kubernetes Grid Integrated Edition > 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 Tanzu Kubernetes Grid Integrated Edition tile. To do this, review External Groups Whitelist in Ops Manager Installation Dashboard > Tanzu Kubernetes Grid Integrated Edition > Settings > UAA.

To grant cluster access to a 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:

```yaml
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 group name. This name is case sensitive.

- If you use LDAP or SAML for UAA, enter the LDAP or SAML group name.
- If you do not use LDAP or SAML for UAA, enter the UAA group name.

**Note:** If you configured an OIDC groups prefix in *Ops Manager Installation Dashboard > Tanzu Kubernetes Grid Integrated Edition > Settings > UAA* or in a Kubernetes profile, you must prepend `NAME-OF-GROUP` with the prefix you configured. For more information, see UAA in the *Installing* topic for your IaaS and Add an OIDC Provider.

- **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.
Getting Started with VMware Harbor Registry

In this topic
Overview
Install Harbor
Use Harbor
Manage Harbor

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:
This topic describes VMware Harbor Registry, an enterprise-class image registry server that stores and distributes container images for VMware Tanzu Kubernetes Grid Integrated Edition.

Overview

As an enterprise private registry, Harbor also offers enhanced performance and improved security. By configuring Harbor with Tanzu Kubernetes Grid Integrated Edition, you can apply enterprise features to your image registry, such as security, identity, and management.

You can install Harbor alongside Tanzu Kubernetes Grid Integrated Edition on vSphere, Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure.

Install Harbor
To install Harbor, do the following:

1. Install Tanzu Kubernetes Grid Integrated Edition. For more information, see the Installing Tanzu Kubernetes Grid Integrated Edition topic for your cloud provider.

2. Install Harbor. For more information, 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. For more information, see Create Projects in the Harbor documentation.

2. Create a Harbor project that contains all repositories for your app. For more information, see Create 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). For more information, 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. For more information, see Managing Helm Charts in the Harbor documentation.

- Install Clair to enable vulnerability scanning for images stored in Harbor. For more information, see Step 8: Configure Container Vulnerability Scanning Using Clair in Installing and Configuring VMware Harbor Registry.

For more information about managing images in Harbor, see the 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 Cluster Access to Private Docker Registries (Beta)

In this topic
Overview
Prerequisites
Set up Your API Access Token
Create a Cluster with SSL CA Certificates
Update a Cluster with SSL CA Certificates
SSL CA Certificate Formats

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to configure VMware Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition Kubernetes clusters to authenticate into your private Docker registries by configuring your clusters with SSL CA certificates.

You can configure both new and existing Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition if you want to apply a single Harbor Registry certificate to all of your Tanzu Kubernetes Grid Integrated Edition clusters.

Warning: Configuring Tanzu Kubernetes Grid Integrated Edition clusters with private Docker registry CA certificates is

Copyright © 2020 VMware, Inc. All Rights Reserved.
Prerequisites

Before configuring Tanzu Kubernetes Grid Integrated Edition 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 TKGI 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 TKGI API endpoints.

1. To export your access token into an environment variable, run the following command:

```bash
tkgi login -a TKGI-API -u USER-ID -p \'PASSWORD\' -k; 
export YOUR-ACCESS-TOKEN=$(bosh int ~/.pks/creds.yml --path /access_token)
```

Where:

- **TKGI-API** is the FQDN of your TKGI API endpoint. For example, `api.tkgi.example.com`.
- **USER-ID** is your Tanzu Kubernetes Grid Integrated Edition user ID.
- **PASSWORD** is your Tanzu Kubernetes Grid Integrated Edition password.
- **YOUR-ACCESS-TOKEN** is the name of your access token environment variable.

For example:

```bash
$ tkgi login -a tkgi.my.lab -u alana -p 'passwrdabc123....!' -k; 
export my_token=$(bosh int ~/.pks/creds.yml --path /access_token)
```

💡 Note: If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 TKGI CLI. For information about configuring SAML, see Connecting Tanzu Kubernetes Grid Integrated Edition 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 TKGI 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://TKGI-API:9021/v1/clusters \
-H Accept: application/json \
-H "Authorization: Bearer $YOUR-ACCESS-TOKEN" \
-H 'Content-Type: application/json' \
-H 'Host: TKGI-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-CERTIFICATE"
      }
    ]
  }
}'

Where:

- **TKGI-API** is the FQDN of your TKGI API endpoint. For example, api.tkgi.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-CERTIFICATE** is the CA certificate that corresponds to **DOMAIN-NAME**. For more information about using a CA certificate in a TKGI 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 TKGI 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://TKGI-API:9021/v1/clusters/CLUSTER-NAME \
-H Accept: application/json \
-H "Authorization: Bearer $YOUR-ACCESS-TOKEN" \
-H 'Content-Type: application/json' \
-H 'Host: TKGI-API:9021' \
-d '{
  "custom_ca_certs": [
    { "domain_name": "DOMAIN-NAME",
      "ca_cert": "CA-CERTIFICATE"
    }
  ]
}'
```
Where:
is the FQDN of your TKGI API endpoint. For example, api.tkgi.example.com .
YOUR-ACCESS-TOKEN is the name of your access token environment variable.
CLUSTER-NAME is the name of your cluster.
TKGI-API

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 TKGI API command, see Prepare a Certificate String for Command Line Use, below.
DOMAIN-NAME

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 TKGI API command, as in the TKGI API commands above, your certificate
string must be provided without newline wrapping.



Note: The TKGI API does not validate certificate strings for correctness. Ensure your certificate string is free of newline
characters before using the certificate string in a TKGI 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.
Please send any feedback you have to pks-feedback@pivotal.io.

Copyright © 2020 VMware, Inc. All Rights Reserved.

814

1.8


Cloud Native Storage (CNS) on vSphere

In this topic

Overview

Prerequisites for CNS with TKGI

Manually Install CSI on a TKGI Cluster

Step 1: Create a TKGI cluster

Step 2: Create a CSI Secret

Step 3: Create RBAC for CSI Access

Step 4: Install the vSphere CSI driver

Verify a CSI Installation

Verify that CSI Deployed Successfully

Verify that All Pods Can Access vCenter

Verify that CSI Custom Resource Definitions are Working

Verify that ProviderID was Added to Nodes

Create a vsphere Storage Class

Configuration File Templates

RBAC Manifest

CSI Driver Manifest

DaemonSet Manifest

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic explains how to integrate Cloud Native Storage (CNS) with VMware Tanzu Kubernetes Grid Integrated Edition with a Cloud Storage Interface (CSI). This integration enables TKGI clusters to use external container storage.

Overview

Cloud Native Storage (CNS) provides comprehensive data management for stateful, containerized apps, enabling apps to survive restarts and outages. Stateful containers can use vSphere storage primitives such as standard volume, persistent volume, and dynamic provisioning, independent of VM and container lifecycle.

vSphere storage backs the volumes, and you can set a storage policy directly on the volumes. After you create the volumes, you can use the vSphere client to review the volumes and their backing virtual disks, and monitor their storage policy compliance.

For more information, see Getting Started with VMware Cloud Native Storage.

Prerequisites for CNS with TKGI

- **vSphere** v6.7U3 or later
  - vSphere definition

- **NSX-T** version compatible with vSphere version above
  - NSX-T v2.4.0 and later are compatible with vSphere v6.7U3
  - See the VMware Product Interoperability Matrices for other version compatibilities

- **TKGI** v1.7 or later
  - Support upgrading virtual hardware version on Kubernetes cluster VMs

- **Firewall and network configuration**
  - Enable the following components to access vCenter:
    - Cluster master nodes
    - Cluster worker nodes, so their CSI components can provision their disks
    - All Pods running CSI components

- **TKGI plan configuration**:
  - In the TKGI tile, configure a Plan with the Allow Privileged checkbox enabled, so containers run in privileged mode

Copyright © 2020 VMware, Inc. All Rights Reserved.
Manually Install CSI on a TKGI Cluster

Step 1: Create a TKGI cluster

1. Create a TKGI cluster:

   tkgi create-cluster tkgi-cluster-5-shared-t1 --external-hostname tkgi-cluster-5-shared-t1 --plan large --num-nodes 3 --network-profile single-tier-profile

2. Make sure that all VMs in the Kubernetes cluster have hardware compatible with VMware version 15.

Step 2: Create a CSI Secret

1. Create the following configuration file `csi-vsphere.conf` anywhere in your system:

   ```
   [Global]
   cluster-id = CLUSTER-ID
   [VirtualCenter "10.1.1.1"]
   insecure-flag = "true"
   user = "administrator@vsphere.local"
   password = "VMware1!"
   port = "443"
   datacenters = "vSAN_Datacenter"
   ```

   Where `CLUSTER-ID` is an unique identifier, such as the cluster name.

2. Create a secret based on the configuration file:

   ```
   > kubectl create secret generic vsphere-config-secret --from-file=csi-vsphere.conf --namespace=kube-system
   secret/vsphere-config-secret created
   ```

3. Confirm that the secret exists:

   ```
   > kubectl get secret/vsphere-config-secret -n kube-system
   NAME                      TYPE     DATA AGE
   vsphere-config-secret     Opaque  1    37s
   ```

Step 3: Create RBAC for CSI Access

1. Create a manifest `vsphere-csi-controller-rbac.yaml` that defines role-based access control (RBAC) of the CSI with a `ServiceAccount`, `ClusterRole`, and `ClusterRoleBinding`.

   * As a template, use the RBAC Manifest template below.

Copyright © 2020 VMware, Inc. All Rights Reserved.
2. Create the RBAC objects:

```bash
kubectl apply -f vsphere-csi-controller-rbac.yaml
```

serviceaccount/vsphere-csi-controller created
clusterrole.rbac.authorization.k8s.io/vsphere-csi-controller-role created
clusterrolebinding.rbac.authorization.k8s.io/vsphere-csi-controller-binding created

Step 4: Install the vSphere CSI driver

1. Create a manifest `vsphere-csi-controller-ss.yaml` that defines StatefulSet and CSIDriver objects for installing the CSI controller.

As a template, use the CSI Driver Manifest template below.

2. Create the CSI driver objects:

```bash
kubectl apply -f vsphere-csi-controller-ss.yaml
```

statefulset.apps/vsphere-csi-controller created
csidriver.storage.k8s.io/csi.vsphere.vmware.com created

3. Create a manifest `vsphere-csi-node-ds.yaml` that defines the DaemonSet for the CSI controller.

As a template, use the DaemonSet Manifest template below.

4. Create the DaemonSet:

```bash
kubectl apply -f vsphere-csi-node-ds.yaml
```

daemonset.apps/vsphere-csi-node created

5. Verify that CNS works for the cluster by following the Verify a CSI Installation steps below.

Verify a CSI Installation

Verify that All Pods Can Access vCenter

After CSI is installed in the Kubernetes cluster, you see these Pods in the kube-system namespace:

```bash
kubectl get po -n kube-system
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>vsphere-csi-controller-0</td>
<td>5/5</td>
<td>24h</td>
</tr>
<tr>
<td>vsphere-csi-node-0</td>
<td>3/3</td>
<td>12m</td>
</tr>
<tr>
<td>vsphere-csi-node-nodes1</td>
<td>3/3</td>
<td>12m</td>
</tr>
<tr>
<td>vsphere-csi-node-kube</td>
<td>3/3</td>
<td>12m</td>
</tr>
<tr>
<td>vsphere-csi-node-pods</td>
<td>3/3</td>
<td>12m</td>
</tr>
</tbody>
</table>

All these Pods must be able to access vCenter. This means that the Floating IP address allocated to the SNAT rule for this namespace in the T0 (or T1 if shared T1 model is used) must be able to reach vCenter.

Verify that CSI Custom Resource Definitions are Working
kubectl get CSINode
NAME																																			CREATED	AT
3a0ea98b-879e-4f7a-abbd-e3ad426c8a1b			2020-03-05T22:29:00Z
c4e3819f-00fc-457b-beda-26fbd599c53		2020-03-05T22:28:57Z
c9b4f441-4c08-43cf-bb17-8e8f02d676a4	2020-03-05T22:29:00Z

kubectl describe CSINode
Name:									3a0ea98b-879e-4f7a-abbd-e3ad426c8a1b
Namespace:
Labels:
Annotations:
API Version:		storage.k8s.io/v1beta1
Kind:									CSINode
Metadata:
Creation Timestamp:	2020-03-05T22:29:00Z
Owner References:
API Version:		v1
Kind:									Node
Name:									3a0ea98b-879e-4f7a-abbd-e3ad426c8a1b
UID:									2ab6f1b2-227-41a6-878b-419717c6b070
Resource Version:	153666
Self Link:
ispersicrok8s.lis.v1beta1/csinodes/3a0ea98b-879e-4f7a-abbd-e3ad426c8a1bUID:									79144892-453-4f7a-8242-a248b93260a
Spec:
Drivers:
Name:									csi.vsphere.vmware.com
Node ID:								3a0ea98b-879e-4f7a-abbd-e3ad426c8a1b
Topology Keys:
Events:
Name:									4c3819f-00fc-457b-beda-26fbd599c53
Namespace:
Labels:
Annotations:
API Version:		storage.k8s.io/v1beta1
Kind:									CSINode
Metadata:
Creation Timestamp:	2020-03-05T22:28:57Z
Owner References:
API Version:		v1
Kind:									Node
Name:									4c3819f-00fc-457b-beda-26fbd599c53
UID:									442cdc30-2e2b-4e7f-ac2b-17e667f18688
Resource Version:	153646
Self Link:
ispersicrok8s.lis.v1beta1/csinodes/4c3819f-00fc-457b-beda-26fbd599c53UID:									acbc421b-cf7f-415d-bc10-1316b28dbd47
Spec:
Drivers:
Name:									csi.vsphere.vmware.com
Node ID:								4c3819f-00fc-457b-beda-26fbd599c53
Topology Keys:
Events:
Name:									c9b4f441-4c08-43cf-bb17-8e8f02d676a4
Namespace:
Labels:
Annotations:
API Version:		storage.k8s.io/v1beta1
Kind:									CSINode
Metadata:
Creation Timestamp:	2020-03-05T22:29:00Z
Owner References:
API Version:		v1
Kind:									Node
Name:									c9b4f441-4c08-43cf-bb17-8e8f02d676a4
UID:									c4d0f53-83f3-466e-bc85-8f61d499f58a
Resource Version:	153663
Self Link:
ispersicrok8s.lis.v1beta1/csinodes/c9b4f441-4c08-43cf-bb17-8e8f02d676a4UID:									69538e9-90ac-4c70-bf90-af124348c5
Spec:
Drivers:
Name:									csi.vsphere.vmware.com
Node ID:								c9b4f441-4c08-43cf-bb17-8e8f02d676a4
Topology Keys:
Events:
> kubectl get csidrivers
NAME	CREATED AT

> kubectl describe csidrivers
Name: csi.vsphere.vmware.com
Namespace:
Labels:
Annotations: kubectl.kubernetes.io/last-applied-configuration

[API Version: "storage.k8s.io/v1", "kind": "CSIDriver", "metadata": {"annotations": [], "name": "csi.vsphere.vmware.com"}, "spec": {"attachRequired": true, "podInfoOnMount": false}]

Verify that ProviderID was Added to Nodes

> kubectl describe nodes | grep "ProviderID"
ProviderID: vsphere://421025c3-0ce4-8cff-8229-1a2ec0bf2d97
ProviderID: vsphere://42109234-71ec-3f26-5ddd-9c97c5a02fe9
ProviderID: vsphere://4210ecc1-e7d8-a13b-10c5f2086b8b36e

Create a vSphere Storage Class

Create the following YAML:

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: demo-sts-storageclass
annotations:
  storagesclass.kubernetes.io/is-default-class: "true"
provisioner: csi.vsphere.vmware.com
parameters:
datastoreurl: /datastore/vmfs/volumes/vsan:52d8eb48424b0f493-41523be9cd4f7b7
```

For a non-vSAN datastore, the datastoreurl value looks like: `ds://vmfs/volumes/5e66e525-8e46bd39-c184-005056ae28de/`. You can find the datastoreurl value in vCenter:

Configuration File Templates

RBAC Manifest

Define RBAC for CSI with a `vsphere-csi-controller-rbac.yaml` file that looks like this:
CSI Driver Manifest

Define a CSI driver with a `vsphere-csi-controller-ss.yaml` file that looks like this:

```yaml
kind: StatefulSet
apiVersion: apps/v1
metadata:
  name: vsphere-csi-controller
  namespace: kube-system
spec:
  serviceName: vsphere-csi-controller
  replicas: 1
  updateStrategy:
    type: "RollingUpdate"
  selector:
    matchLabels:
      app: vsphere-csi-controller
  template:
    metadata:
      labels:
        app: vsphere-csi-controller
        role: vsphere-csi
    spec:
      serviceAccountName: vsphere-csi-controller
      dnsPolicy: "Default"
      containers:
        - name: csi-attacher
          image: quay.io/k8scsi/csi-attacher:v1.1.1
          args:
            - "--v=4"
            - "--timeout=300s"
            - "--csi-address=$(ADDRESS)"
          env:
            - name: ADDRESS
              value: /csi/csi.sock
          volumeMounts:
            - mountPath: /csi
              name: socket-dir
            - name: vsphere-csi-controller
              image: gcr.io/cloud-provider-vsphere/csi/release/driver:v1.0.2
          lifecycle:
            preStop:
```

---

Copyright © 2020 VMware, Inc. All Rights Reserved.
---

apiVersion: storage.k8s.io/v1beta1
kind: CSIDriver
metadata:
  name: csi.vsphere.vmware.com
spec:
  attachRequired: true
  podInfoOnMount: false
Define a DaemonSet with a `vsphere-csi-node-ds.yaml` file that looks like this:

```yaml
# eks-cloud-provider-vsphere-csi-node-daemonset.yaml

kind: DaemonSet
apiVersion: apps/v1
metadata:
  name: vsphere-csi-node
  namespace: kube-system
spec:
  selector:
    matchLabels:
      app: vsphere-csi-node
  updateStrategy:
    type: "RollingUpdate"
  template:
    metadata:
      labels:
        app: vsphere-csi-node
        role: vsphere-csi
    spec:
      dnsPolicy: "Default"
      containers:
        - name: node-driver-registrar
          image: quay.io/k8scsi/csi-node-driver-registrar:v1.1.0
          lifecycle:
            preStop:
              exec:
                command:
                - /bin/sh
                - -c
          args:
            - "--v=5"
            - "--csi-address=$(ADDRESS)"
            - "--kubernetes-registration-path=$(DRIVER_REG_SOCK_PATH)"
          env:
            - name: ADDRESS
              value: /csi/csi.sock
            - name: DRIVER_REG_SOCK_PATH
              value: /var/vcap/data/kubelet/plugins_registry/csi.vsphere.vmware.com/csi.sock
          securityContext:
            privileged: true
            volumeMounts:
              - name: plugin-dir
                mountPath: /csi
              - name: registration-dir
                mountPath: /registration
          image: gcr.io/cloud-provider-vsphere/csi/release/driver:v1.0.2
          imagePullPolicy: "Always"
          env:
            - name: NODE_NAME
              valueFrom:
                fieldRef:
                  fieldPath: spec.nodeName
            - name: CSI_ENDPOINT
              value: unix:///csi/csi.sock
            - name: X_CSI_MODE
              value: "node"
            - name: X_CSI_SPEC_REQ_VALIDATION
              value: "false"
            - name: VSPHERE_CSI_CONFIG
              value:="/etc/cloud/csi-vsphere.conf" # here csi-vsphere.conf is the name of the file used for creating secret using "--from-file" flag
          args:
            - "--v=4"
          securityContext:
            privileged: true
            capabilities:
              add: ["SYS_ADMIN"]
            allowPrivilegeEscalation: true
            volumeMounts:
              - name: vsphere-config-volume
                mountPath: /etc/cloud
                readOnly: true
              - name: plugin-dir
                mountPath: /csi
              - name: pod-volume-dir
                mountPath: /var/vcap/data/kubelet
              - name: device-dir
                mountPath: /dev
          ports:
            - name: healthz
              containerPort: 9808
              protocol: TCP
            - name: healthz
              containerPort: 9808
              protocol: TCP
          livenessProbe:
            httpGet:
              path: /healthz
              port: healthz
            initialDelaySeconds: 10
```

---

Copyright © 2020 VMware, Inc. All Rights Reserved.
timeoutSeconds: 3
periodSeconds: 5
failureThreshold: 3
- name: liveness-probe
  image: quay.io/k8scsi/livenessprobe:v1.1.0
  args:
    - "--csi-address=$ADDRESS"
  env:
    - name: ADDRESS
      value: /csi/csi.sock
  volumeMounts:
  - name: plugin-dir
    mountPath: /csi
  volumes:
  - name: vSphere-config-volume
    secret:
      secretName: vSphere-config-secret
    - name: registration-dir
      hostPath:
        path: /var/vcap/data/kubelet/plugins_registry
        type: DirectoryOrCreate
      - name: plugin-dir
        hostPath:
        path: /var/vcap/data/kubelet/plugins_registry/csi.vsphere.vmware.com
        type: DirectoryOrCreate
      - name: pods-mount-dir
        hostPath:
        path: /var/vcap/data/kubelet
        type: Directory
      - name: device-dir
        hostPath:
        path: /dev

Please send any feedback you have to pks-feedback@pivotal.io.
PersistentVolume Storage Options on vSphere

In this topic
Considerations for Running Stateful Apps in Kubernetes
Persistent Volume Provisioning Support in Kubernetes
vSphere Support for Static and Dynamic PVs
Single vSphere Compute Cluster with vSAN Datastore
Single vSphere Compute Cluster with File System Datastore
Multiple vSphere Compute Clusters Each with vSAN Datastore
Multiple vSphere Compute Clusters Each with File System Datastore
Multiple vSphere Compute Clusters with Local vSAN and Shared File System Datastore
Multiple vSphere Compute Clusters with Shared File System Datastore

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes options for configuring VMware Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition, 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 Tanzu Kubernetes Grid Integrated Edition support for PVs in Kubernetes when deployed on vSphere:

<table>
<thead>
<tr>
<th>Storage Mechanism</th>
<th>vSAN datastore</th>
<th>File system datastore (VMFS/NFS over iSCSI/FC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single vSphere compute cluster with single datastore</td>
<td>Both static and dynamic PV provisioning are supported.</td>
<td>Both static and dynamic PV provisioning are supported.</td>
</tr>
<tr>
<td>Single AZ and resource pool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiple vSphere compute clusters each with local datastore</td>
<td>Neither static nor dynamic PV provisioning are supported.</td>
<td>Neither static nor dynamic PV provisioning are supported.</td>
</tr>
<tr>
<td>Multiple AZs each using separate resource pool</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiple vSphere compute clusters with shared datastore</td>
<td>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.</td>
<td>Both static and dynamic PV provisioning are supported.</td>
</tr>
<tr>
<td>Multiple AZs using a shared resource pool</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Note: This information assumes that the underlying vSphere infrastructure is a single locality environment where all vSphere
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, Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition foundation, create an AZ that is mapped directly to the single cluster.
- If you use multiple Tanzu Kubernetes Grid Integrated Edition 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, Tanzu Kubernetes Grid Integrated Edition 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, Tanzu Kubernetes Grid Integrated Edition supports both static and dynamic PV provisioning. Dynamic PV provisioning is recommended.
In this topology, multiple vSphere compute clusters are used to host all Kubernetes clusters. A unique shared datastore that uses NFS, or VMFS over iSCSI/FC is used across all compute clusters. In the above diagram, this datastore is labeled **Shared Datastore1**.

One or more AZs can be instantiated. Each AZ is mapped to a compute cluster. The AZ is bound to a failure domain which is typically the physical rack where the compute cluster is hosted.

You can have multiple shared datastores connected across all the vSphere compute clusters. PVs, backed by respective VMDK files, can then be dispatched across those datastores to mitigate the impact of datastore failure. For StatefulSets, all PVs used by different instances of the replica land in the same datastore.

This topology has the following failover scenarios:

- **ESXi hosts are down**: No impact on PVs.
- **One shared datastore is down**: PVs on the down datastore are unreachable.

Please send any feedback you have to pks-feedback@pivotal.io.
Configuring and Using PersistentVolumes

In this topic

Provision a Static PV
  - Provision a Static PV for a Deployment Workload
  - Provision a Static PV for a StatefulSets Workload

Provision a Dynamic PV
  - Provision a Dynamic PV for Deployment Workloads
  - Provision a Dynamic PV for StatefulSets Workloads
  - Specify a Default StorageClass
  - Provision Dynamic PVs for Use with Tanzu Kubernetes Grid Integrated Edition

Page last updated:

This topic describes how to provision static and dynamic PersistentVolumes (PVs) for VMware Tanzu Kubernetes Grid Integrated Edition 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:

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:
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:

```yaml
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:

```yaml
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:

```yaml
apiVersion: apps/v1
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
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
sphereVolume:
  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:

```yaml
apiVersion: 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.

Tanzu Kubernetes Grid Integrated Edition supports dynamic PV provisioning by providing StorageClasses for all supported cloud providers, as well as an example PVC.
Provision a Dynamic PV for Deployment Workloads

**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.

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:

   ```yaml
   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:

   ```yaml
   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:

   ```yaml
   apiVersion: apps/v1
   kind: Deployment
   metadata:
     name: redis-master
   ...  
   spec:
     template:
       spec:
         volumes:
         - name: redis-master-data
           persistentVolumeClaim:
             claimName: redis-master-claim
   ```

   **Note:** The examples in this section use the vSphere provisioner. Refer to the Kubernetes documentation for information about provisioners for other cloud providers.
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:

```yaml
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:

```yaml
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:
Provision Dynamic PVs for Use with Tanzu Kubernetes Grid Integrated Edition

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:
     
     ```
     wget https://raw.githubusercontent.com/cloudfoundry-incubator/kubo-ci/master/specs/storage-class-azure.yml
     ```
     ```
     For Azure disk storage:
     wget https://raw.githubusercontent.com/cloudfoundry-incubator/kubo-ci/master/specs/storage-class-azure.yml
     ```
     ```
     For Azure file storage:
     ```
   
   - 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-disk
     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:

   ```
   kubectl create -f example-pvc.yaml
   ```

Note: The above example uses the vSphere provisioner. Refer to the Kubernetes documentation for information about provisioners for other cloud providers.

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

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

**Page last updated:**

This section describes how to deploy workloads to Kubernetes clusters provisioned by VMware Tanzu Kubernetes Grid Integrated Edition.

See the following topics:

- Deploying and Exposing Basic Linux Workloads
- Deploying and Exposing Basic Windows Workloads (Beta)
- Adding Custom Linux Workloads
- Using Helm with Tanzu Kubernetes Grid Integrated Edition

Please send any feedback you have to pks-feedback@pivotal.io.
Deploying and Exposing Basic Linux Workloads

In this topic

Overview

Prerequisites

vSphere without NSX-T Prerequisites

GCP, AWS, Azure, and vSphere with NSX-T Prerequisites

AWS Prerequisites

Deploy Workloads on vSphere with NSX-T

Deploy Workloads on GCP, AWS, or Azure, Using a Public-Cloud External Load Balancer

Deploy AWS Workloads Using an Internal Load Balancer

Deploy Workloads for a Generic External Load Balancer

Deploy Workloads without a Load Balancer

---

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

---

Page last updated:

This topic describes how to configure, deploy, and expose basic workloads in VMware Tanzu Kubernetes Grid Integrated Edition.

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 Tanzu Kubernetes Grid Integrated Edition 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 TKGI.
- 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

---

Copyright © 2020 VMware, Inc. All Rights Reserved.
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 `tkgi 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.

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kubernetes.io/cluster/service-instance</td>
<td>empty</td>
</tr>
<tr>
<td>CLUSTER-UUID</td>
<td></td>
</tr>
</tbody>
</table>

   **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:
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:

   ```bash
   kubectl apply -f SERVICE-CONFIG
   ```

   Where `SERVICE-CONFIG` is your workload’s Kubernetes service configuration. For example:

   ```bash
   $ 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:

   ```bash
   kubectl get svc SERVICE-NAME
   ```

   Where `SERVICE-NAME` is the specified service `name` of your workload configuration. For example:

   ```bash
   $ 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:

```bash
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:

   ```yaml
   ---
   apiVersion: v1
   kind: Service
   metadata:
     labels:
     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 kuboc-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:

```bash
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:

```bash
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:

```bash
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:

   ```yaml
   annotations:
     service.beta.kubernetes.io/aws-load-balancer-internal: 0.0.0.0/0
   ```

   For example:

   ```yaml
   ---
   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:

   ```bash
   kubectl apply -f SERVICE-CONFIG
   ```

   Where `SERVICE-CONFIG` is the Kubernetes service configuration of your workload.

   **For example:**

   ```bash
   $ 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**

Copyright © 2020 VMware, Inc. All Rights Reserved.
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 your 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:
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:

   ```bash
to kubectl apply -f SERVICE-CONFIG
   
   Where `SERVICE-CONFIG` is the Kubernetes service configuration of your workload.
   
   For example:
   ```

   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 of your workload configuration.

For example:

```bash
kubectl get svc nginx
```

3. Find the node port number in the `3XXX` 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:

```yaml
---
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.
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 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 3XXX 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.

Please send any feedback you have to pks-feedback@pivotal.io.
Deploying and Exposing Basic Windows Workloads (Beta)

In this topic
Overview
Prerequisites
Access Your Windows-Based Cluster
Deploy a Windows Worker Pod
  Configure a Pod Deployment Manifest
  Deploy the Pod
Deploy a Service to a Windows Worker Pod
  Configure a Service Manifest
  Deploy the Service

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes deploying Windows worker-based Kubernetes clusters in VMware Tanzu Kubernetes Grid Integrated Edition.

Overview

In Tanzu Kubernetes Grid Integrated Edition, 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition tile. For instructions on configuring the Tanzu Kubernetes Grid Integrated Edition 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:

```bash
tkgi clusters
```

For example:

```
$ tkgi clusters

<table>
<thead>
<tr>
<th>Name</th>
<th>Plan Name</th>
<th>UUID</th>
<th>Status</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>windows-k8s</td>
<td>Plan-11-Windows-Beta 881543kd-64fg-7826-3h7g104fkh0e</td>
<td>succeeded</td>
<td>Create</td>
<td></td>
</tr>
<tr>
<td>second-windows-k8s</td>
<td>Plan-11-Windows-Beta 951547dl-67kg-9631-bju8-7h93a98br0q</td>
<td>succeeded</td>
<td>Create</td>
<td></td>
</tr>
</tbody>
</table>
```

Only clusters configured on Plans 11, 12, or 13 are Windows-based.

2. To access your Windows-based cluster, run the following command:

```bash
tkgi get-credentials CLUSTER-NAME
```

Where `CLUSTER-NAME` is the name of your Windows-based cluster.

For example:

```
$ tkgi 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 `tkgi get-credentials` command creates a local `kubeconfig`, allowing you to manage the cluster from the command line. For more information about the `tkgi get-credentials` command, see Retriving Cluster Credentials and Configuration.

3. To verify you have established access to the correct cluster, run the following command:

```bash
kubectl cluster-info
```

4. (Optional) To review the existing pods in the cluster, run the following command:

```bash
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: apps/v1
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: apps/v1
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              READYINESS GATES
   win-webserver-795g866cd7-58oct  1/1      Running 0 88s 10.200.42.4 0983934a-6d69-8e5g-g3k1-98r8r56l345j <none>  <none>
   ```

   In the preceding example a new pod is created, and creation and status of the new pod and new nodes verified.

   **Note:** The ping command does not work reliably for Windows workers. For more information, see Pinging Windows Workers Does Not Work in Release Notes.

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:

   ```yaml
   ---
   apiVersion: v1
   kind: Service
   metadata:
     name: APP-NAME
   spec:
     selector:
       app: APP-NAME
     ports:
       - port: 80
         targetPort: 80
         protocol: TCP
         name: http
     type: NodePort
   ```

   Where `APP-NAME` is the name of your Windows service.

   For example:
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
Create YAML Configuration
Apply Custom Workloads

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to add custom workloads to VMware Tanzu Kubernetes Grid Integrated Edition 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:

```yaml
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:
    metadata: # create pods using pod definition in this template
      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 Tanzu Kubernetes Grid Integrated Edition tile.

For more information, see the Plans section of the Installing Tanzu Kubernetes Grid Integrated Edition topic for your IaaS. For example, Plans in Installing Tanzu Kubernetes Grid Integrated Edition on vSphere.
Please send any feedback you have to pks-feedback@pivotal.io.
Using Helm with Tanzu Kubernetes Grid Integrated Edition

In this topic
Overview
Install and Configure Helm 3
Install and Configure Helm 2

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

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 Tanzu Kubernetes Grid Integrated Edition.

Helm 3 requires less configuration than Helm 2.

Overview
Helm 3 and Helm 2 include the following components:

<table>
<thead>
<tr>
<th>Component</th>
<th>Role</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>helm</td>
<td>Client</td>
<td>Runs on your local workstation</td>
</tr>
<tr>
<td>tiller (Helm 2 only)</td>
<td>Server</td>
<td>Runs inside your Kubernetes cluster</td>
</tr>
</tbody>
</table>

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 TKGI documentation.

Install and Configure Helm 2
To use Helm 2 with Tanzu Kubernetes Grid Integrated Edition, 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:

```yaml
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.
This section describes how to monitor VMware Tanzu Kubernetes Grid Integrated Edition (TKGI) environments.

See the following topics:

- Monitoring TKGI and TKGI-Provisioned Clusters
- Monitoring Workers and Workloads
- Accessing Dashboard
- Viewing Usage Data from the Billing Database

Please send any feedback you have to pks-feedback@pivotal.io.
Monitoring TKGI and TKGI-Provisioned Clusters

In this topic

Overview

Logs: Syslog and vRLI

Syslog

vRealize Log Insight (vSphere Only)

Metrics: Telegraf

---

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic lists VMware Tanzu Kubernetes Grid Integrated Edition (TKGI) components and integrations you can use to capture logs and metrics about TKGI and TKGI-provisioned cluster VMs.

For information about monitoring Kubernetes workloads, see Monitoring Workers and Workloads.

Overview

To monitor TKGI and TKGI-provisioned cluster VMs, you can enable one or more of the following components and integrations in the Tanzu Kubernetes Grid Integrated Edition tile > Host Monitoring:

<table>
<thead>
<tr>
<th>Name</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syslog</td>
<td>See Syslog below.</td>
</tr>
<tr>
<td>VMware vRealize Log Insight</td>
<td>See vRealize Log Insight (vSphere Only) below.</td>
</tr>
<tr>
<td>Telegraf (metrics)</td>
<td>See Telegraf below.</td>
</tr>
</tbody>
</table>

These components and integrations are visible only to TKGI 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 Tanzu Kubernetes Grid Integrated Edition on vSphere
- Installing Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T
- Installing Tanzu Kubernetes Grid Integrated Edition on GCP
- Installing Tanzu Kubernetes Grid Integrated Edition on AWS
- Installing Tanzu Kubernetes Grid Integrated Edition on Azure

Logs: Syslog and vRLI

You can configure Syslog or vRealize Log Insight (vSphere only) to publish logs from the TKGI control plane and TKGI-provisioned cluster VMs.
You might need to inspect Syslog or vRealize Log Insight (vRLI) logs when troubleshooting or auditing your TKGI environment. For information about key TKGI events and the log entries they generate, see Auditing Tanzu Kubernetes Grid Integrated Edition Logs.

**Syslog**

Syslog sends log messages from all BOSH-deployed VMs in a TKGI 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.

**vRealize Log Insight (vSphere Only)**

The vRealize Log Insight (vRLI) integration for TKGI 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.

**Metrics: Telegraf**

The Telegraf agent sends metrics from TKGI API, 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 more the following sources:

<table>
<thead>
<tr>
<th>Source</th>
<th>Includes metrics from...</th>
</tr>
</thead>
<tbody>
<tr>
<td>TKGI API</td>
<td>- Node Exporter (Prometheus)</td>
</tr>
<tr>
<td>Master nodes</td>
<td>One or more of the following:</td>
</tr>
<tr>
<td>(not visible to cluster users)</td>
<td>- Node Exporter (Prometheus)</td>
</tr>
<tr>
<td></td>
<td>- Kubernetes API server</td>
</tr>
<tr>
<td></td>
<td>- Kubernetes controller manager</td>
</tr>
<tr>
<td></td>
<td>- etcd</td>
</tr>
<tr>
<td>Worker nodes</td>
<td>One or more of the following:</td>
</tr>
<tr>
<td></td>
<td>- Node Exporter (Prometheus)</td>
</tr>
<tr>
<td></td>
<td>- kubelet</td>
</tr>
</tbody>
</table>

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 Tanzu Kubernetes Grid Integrated Edition Logs

In this topic
TKGI API events
  - Cluster Creation
  - Cluster Deletion
  - Successful Login
  - Unsuccessful Login
  - Successful Cluster Credential Retrieval
  - User Creation
  - User Deletion
  - Telemetry Collection

Kubernetes Audit Log Events

Related Links

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic summarizes key auditable events in TKGI, 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.

TKGI API events

The following log entry examples are produced by TKGI API events and correspond to key actions taken by a user logged into the TKGI CLI.

Cluster Creation

| Description | A user has issued a create cluster command. |
| Identifying String | Action 'create-cluster' |
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 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

**Successful Login**

**UserAuthenticationSuccess**

**Description**
A user has successfully logged into Tanzu Kubernetes Grid Integrated Edition.

**Identifying String**
UserAuthenticationSuccess

**Example Log Entries**
User Authentication Failure

**Description**
A user has failed a login attempt into Tanzu Kubernetes Grid Integrated Edition.

**Identifying String**
UserAuthenticationFailure

**Example Log Entries**
```
```

Successful Cluster Credential Retrieval

**Client Authentication Success**

**Description**
A user has successfully gained access to a cluster in Tanzu Kubernetes Grid Integrated Edition.

**Identifying String**
ClientAuthenticationSuccess

**Example Log Entries**
```
```

User Creation

**User Created Event**

**Description**
An administrator has successfully created a new user for Tanzu Kubernetes Grid Integrated Edition.

**Identifying String**
UserCreatedEvent

**Example Log Entries**
```
```

User Deletion

**User Deleted Event**

**Description**
An administrator has successfully deleted a user for Tanzu Kubernetes Grid Integrated Edition.

**Identifying String**
UserDeletedEvent
Example Log Entries


Telemetry Collection

Telemetry Ping

Description

The optional telemetry system has successfully reached an external host for collecting product data for Tanzu Kubernetes Grid Integrated Edition.

To learn more about the Tanzu Kubernetes Grid Integrated Edition 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.

```json
{
  "kind": "Event",
  "apiVersion": "audit.k8s.io/v1",
  "level": "Request",
  "auditID": "dc2cb4e9-4b85-42da-82a3-5ee47091207d",
  "stage": "ResponseStarted",
  "requestURI": "/apis/policy/v1beta1/poddisruptionbudgets?resourceVersion=370506&timeout=7m54s&timeoutSeconds=474&watch=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",
  "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 Tanzu Kubernetes Grid Integrated Edition.
- For information about downloading TKGI 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

Overview
Download Logs

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic explains how to download logs from BOSH-deployed VMs in your VMware Tanzu Kubernetes Grid Integrated Edition environment using the BOSH Command Line Interface (CLI).

Overview

In Tanzu Kubernetes Grid Integrated Edition, you can download logs from any BOSH-deployed VM, such as the TKGI API VM or Kubernetes cluster VMs.

You might need to download these logs when troubleshooting or auditing your TKGI 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 tkgi 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:
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 tkgi -d service-instance_ae681cd1-7ff4-4661-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 Tanzu Kubernetes Grid Integrated Edition.

Please send any feedback you have to pks-feedback@pivotal.io.
Monitoring Master/etcd Node VMs

In this topic
Overview
Collect Metrics Using Telegraf
  Create a Configuration File
  Configure Telegraf in the Tile
Troubleshoot etcd

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Overview

This topic describes how platform operators can monitor and retrieve metrics from master/etcd node VMs in VMware Tanzu Kubernetes Grid Integrated Edition deployments.

About Node Exporter

Node Exporter exports hardware and operating system metrics in Prometheus format.

In Tanzu Kubernetes Grid Integrated Edition, you can enable the Node Exporter BOSH job separately on master nodes, worker nodes, and the TKGI API VM. These settings are located in the Host Monitoring pane of the Tanzu Kubernetes Grid Integrated Edition tile.

Node Exporter exposes metrics on localhost only. For a list of Node Exporter metrics, see the Node Exporter GitHub repository.

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.


Create a Configuration File

To connect a third-party monitoring service to Tanzu Kubernetes Grid Integrated Edition, 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 Tanzu Kubernetes Grid Integrated Edition 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, VMware recommends adding tags to filter your metrics in your monitoring service.

Configure Telegraf in the Tile

To configure Telegraf in the Tanzu Kubernetes Grid Integrated Edition tile, follow the instructions in the `Installing` topic for your IaaS. For example, if you are installing Tanzu Kubernetes Grid Integrated Edition on vSphere, follow the instructions in the Telegraf section of `Installing Tanzu Kubernetes Grid Integrated Edition on vSphere`.

Troubleshoot etcd

VMware recommends working with 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
Overview
Sink Resources
Supported Integrations

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic lists VMware Tanzu Kubernetes Grid Integrated Edition (TKGI) components and integrations you can use to capture logs and metrics about your Kubernetes worker nodes and workloads.

For information about monitoring TKGI and TKGI-provisioned cluster VMs, see Monitoring TKGI and TKGI-Provisioned Clusters.

Overview

To monitor Kubernetes worker nodes and workloads in your TKGI deployment, you can enable one or more of the following components and integrations in the Tanzu Kubernetes Grid Integrated Edition tile > In-Cluster Monitoring:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sink resources</td>
<td>TKGI component</td>
<td>See Sink Resources below.</td>
</tr>
<tr>
<td>Wavefront</td>
<td>Integration</td>
<td>See Supported Integrations below.</td>
</tr>
<tr>
<td>cAdvisor</td>
<td>Integration</td>
<td>See Supported Integrations below.</td>
</tr>
</tbody>
</table>

When running on worker nodes, these components and integrations are visible to both TKGI 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 Tanzu Kubernetes Grid Integrated Edition on vSphere
- Installing Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T
- Installing Tanzu Kubernetes Grid Integrated Edition on GCP
- Installing Tanzu Kubernetes Grid Integrated Edition on AWS
- Installing Tanzu Kubernetes Grid Integrated Edition on Azure

Sink Resources

In TKGI, 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition](#)
- Sink resource types, outputs, and identifying strings: [Monitoring Clusters with Log Sinks](#)

### Supported Integrations

TKGI 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 Tanzu Kubernetes Grid Integrated Edition

In this topic
Overview
Sink Types
Sink Architecture
  Log Sink Architecture
  Metric Sink Architecture

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how VMware Tanzu Kubernetes Grid Integrated Edition (TKGI) implements sinks for collecting logs and metrics from Kubernetes worker nodes and workloads.

For step-by-step instructions on creating sinks in TKGI, see Creating and Managing Sink Resources

Overview

A sink collects logs or metrics about Kubernetes worker nodes in a TKGI 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.

<table>
<thead>
<tr>
<th>Sink Type</th>
<th>Sink Resource</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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](https://tools.ietf.org/html/rfc5424)  
  - WebHook  
  - Fluent Bit output plugins |
## Sink Architecture

TKGI-provisioned Kubernetes clusters include an observability manager that manages log sink and metric sink configurations within a cluster.

The following diagram details TKGI cluster observability architecture:

<table>
<thead>
<tr>
<th>Sink Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| Log sink    | 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](https://tools.ietf.org/html/rfc5424)  
- WebHook  
- Fluent Bit output plugins |
| Metric sink | Collects and writes metrics from a cluster to specified outputs using input and output plugins. |
| Metric sink | Collects and writes metrics from a namespace within a cluster to specified outputs using input and output plugins. |

In the *Tanzu Kubernetes Grid Integrated Edition* 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 TKGI log sink aggregates workload logs and forwards them to a common log destination.

The following diagram details TKGI log sink architecture:

![Log Sink Architecture Diagram]

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 TKGI metric sink aggregates workload metrics and forwards them to a common metrics destination.

The following diagram details TKGI 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 Tanzu Kubernetes Grid Integrated Edition 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

Overview

Sinks collect logs and metrics about Kubernetes worker nodes in your TKGI 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 Tanzu Kubernetes Grid Integrated Edition

Prerequisites

Before creating a sink resource:


2. Configure sink resources in the Tanzu Kubernetes Grid Integrated Edition 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 Metrics below, select the **Enable node exporter on workers** checkbox.

For more information about these configuration settings, see the TKGI installation topic for your IaaS:

- Installing Tanzu Kubernetes Grid Integrated Edition on vSphere
- Installing Tanzu Kubernetes Grid Integrated Edition on vSphere with NSX-T Integration
- Installing Tanzu Kubernetes Grid Integrated Edition on GCP
- Installing Tanzu Kubernetes Grid Integrated Edition on AWS
- Installing Tanzu Kubernetes Grid Integrated Edition 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 TKGI do not support UDP connections.

**Note:** TKGI 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:

   ```bash
   kubectl apply -f YOUR-SINK.yml
   ```

   Where **YOUR-SINK.yml** is the name of your YAML file. For example:

   ```bash
   $ 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:

   ```yaml
   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:
Unsecured ClusterLogSink and LogSink Log Forwarding

By default, TKGI 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 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.

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:

```yaml
apiVersion: pksapi.io/v1beta1
kind: ClusterMetricSink
metadata:
  name: http
spec:
  inputs:
  - type: http
    url: https://example.com
    method: POST
    data_format: json
  outputs:
    - type: YOUR-OUTPUT-PLUGIN
```

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 **Tanzu Kubernetes Grid Integrated Edition** tile > **In-Cluster Monitoring**.

2. Create a YAML file in the following format:

```yaml
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

Overview

Log Sinks

Log Format

Notable Kubernetes API Events

Related Links

---

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes the log sink resources you can use to monitor Kubernetes clusters provisioned by VMware Tanzu Kubernetes Grid Integrated Edition and their workloads.

**Overview**

You can use the following sink resources to collect logs from your Kubernetes clusters.

<table>
<thead>
<tr>
<th>Sink Resource</th>
<th>Sink Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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](https://tools.ietf.org/html/rfc5424)  
- 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](https://tools.ietf.org/html/rfc5424)  
- 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

---

Copyright © 2020 VMware, Inc. All Rights Reserved.
Log Format

In Tanzu Kubernetes Grid Integrated Edition, 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**, Tanzu Kubernetes Grid Integrated Edition 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"]
```

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 Tanzu Kubernetes Grid Integrated Edition.

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**

<table>
<thead>
<tr>
<th>Description</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identifying String</strong></td>
<td>Error:ErrImagePull</td>
</tr>
<tr>
<td><strong>Example Sink Log Entry</strong></td>
<td>Jan 25 10:18:58 gke-bf-test-default-pool-aa8027bc-rnf6 k8s.event/default/test-669d4d66b9-zd9h4/: Error: ErrImagePull</td>
</tr>
</tbody>
</table>

### Malfunctioning Containers

**CrashLoopBackOff**

<table>
<thead>
<tr>
<th>Description</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identifying String</strong></td>
<td>Back-off restarting failed container</td>
</tr>
<tr>
<td><strong>Example Sink Log Entry</strong></td>
<td>Jan 25 09:26:44 cluster-name k8s.event/monitoring/cost-analyzer-prometheus-se: Back-off restarting failed container</td>
</tr>
</tbody>
</table>

### Successful Scheduling of Containers

**ContainerCreated**
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.

**Example Sink Log Entries**

<table>
<thead>
<tr>
<th>Time</th>
<th>Event Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 25 09:14:55</td>
<td>cluster-name 35.239.18.250 k8s.event/rocky-raccoon/logspewer-6b58b6689d/ Created pod: logspewer-6b58b6689d-sr96t</td>
</tr>
<tr>
<td>Jan 25 09:14:55</td>
<td>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</td>
</tr>
<tr>
<td>Jan 25 09:14:55</td>
<td>cluster-name k8s.event/rocky-raccoon/logspewer-6b58b6689d-mdk: Killing container with id docker://logspewer:Need to kill Pod</td>
</tr>
<tr>
<td>Jan 25 09:14:56</td>
<td>cluster-name k8s.event/rocky-raccoon/logspewer-6b58b6689d-sr9: Container image &quot;oratos/logspewer:v0.1&quot; already present on machine</td>
</tr>
<tr>
<td>Jan 25 09:14:56</td>
<td>cluster-name k8s.event/rocky-raccoon/logspewer-6b58b6689d-sr9: Created container</td>
</tr>
<tr>
<td>Jan 25 09:14:56</td>
<td>cluster-name k8s.event/rocky-raccoon/logspewer-6b58b6689d-sr9: Started container</td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Time</th>
<th>Event Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 25 10:51:48</td>
<td>gke-bf-test-default-pool-aa8027bc-rnf6 k8s.event/default/test2-5c87bf4b65-7fdtd/: 0/1 nodes are available: 1 Insufficient cpu.</td>
</tr>
</tbody>
</table>

**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 Tanzu Kubernetes Grid Integrated Edition](#)
  
  See this topic for conceptual information about sinks.

[Please send any feedback you have to pks-feedback@pivotal.io](mailto:pks-feedback@pivotal.io).
Accessing Dashboard

In this topic

Overview
Install Dashboard
Access Credentials
  Configure Kubeconfig Access Credentials
  Request Bearer Token Access Credentials
Access Dashboard
Use Dashboard

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to access Dashboard, a web-based Kubernetes UI, for your VMware Tanzu Kubernetes Grid Integrated Edition deployment.

warning: For security reasons, clusters created with TKGI v1.7 and later do not have Kubernetes Dashboard automatically installed. See the Release Notes for more information.

Overview

Kubernetes provides Dashboard to manage Kubernetes clusters and applications, and to review the state of Kubernetes cluster resources.

Install Dashboard

Install Dashboard on clusters running TKGI by following the Deploying the Dashboard UI instructions in the Kubernetes documentation.

Kubernetes dashboard is automatically installed on clusters created with versions prior to TKGI v1.7.

Access Credentials

You must have either a Kubeconfig or Bearer Token access credential to access Dashboard.

Configure Kubeconfig Access Credentials

You can use the TKGI 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 TKGI CLI:

  ```
tkgi get-credentials CLUSTER-NAME
  ```

  Where **CLUSTER-NAME** is the name of your cluster.

  For example:

  ```
$ tkgi get-credentials tkgi-bosh

Fetching credentials for cluster tkgi-bosh.
Context set for cluster tkgi-bosh.
  ```

  **Note:** If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 **TKGI CLI**. For information about configuring SAML, see **Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider**

- Request a Kubeconfig access credential and assign to your Kubernetes configuration:

  ```
KUBECONFIG=CONFIG-FILE tkgi 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 Tanzu Kubernetes Grid Integrated Edition 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 **TKGI CLI**. For information about configuring SAML, see **Connecting Tanzu Kubernetes Grid Integrated Edition 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 == "tkgi-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-ac11-43f9-99a7-87a5u4fbe44b | 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-ac11-43f9-99a7-87a5u4fbe44b | 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.

Please send any feedback you have to pks-feedback@pivotal.io.
This topic describes how operators can view VMware Tanzu Kubernetes Grid Integrated Edition pod usage information from the billing database.

### About the Billing Database

The Tanzu Kubernetes Grid Integrated Edition billing 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.

You can use this data to calculate billed usage, perform customer chargebacks, generate usage reports, and perform other functions.

### Usage Data Format

This section describes the usage data records you can view in the Tanzu Kubernetes Grid Integrated Edition billing database. The agent pod collects usage data for the deployment and sends the data to the Tanzu Kubernetes Grid Integrated Edition aggregator agent. The aggregator agent then stores the data in the Tanzu Kubernetes Grid Integrated Edition billing database. You can access the billing database from the TKGI API VM.

The following is an example of a pod usage data table:

```
+--------------------------------------+---------------------+---------------------+--------------+--------------+-------------------------------------------------------+
| id                                  | first_seen          | last_seen           | name         | service_instance_id    | service_instance_id                                    |
| 12a34b6e-7890-13c4-de56-7890a123b4c  | 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:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique record identifier</td>
</tr>
<tr>
<td>first_seen</td>
<td>The date when the pod was first recorded to the database</td>
</tr>
<tr>
<td>last_seen</td>
<td>The date when the pod was most recently recorded to the database</td>
</tr>
<tr>
<td>namespace</td>
<td>The namespace where the pod is deployed</td>
</tr>
<tr>
<td>name</td>
<td>The name of the pod</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>service_instance_id</td>
<td>The cluster where the pod is deployed</td>
</tr>
</tbody>
</table>

### 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 **Tanzu Kubernetes Grid Integrated Edition** 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 TKGI network. If your system is outside the network, SSH into a TKGI DB VM, whether it is a singleton or one of multiple DB VMs. For more information, see [SSH into a TKGI DB VM](Using_BOSH_Diagnostic_Commands_in_Tanzu_Kubernetes_Grid_Integrated_Edition).
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 TKGI DB VM**
     ```
     mysql -h 127.0.0.1 --port 3333 -u billing -p BILLING-PASSWORD
     ```
     Where `BILLING-PASSWORD` is the billing database password you located in the steps above.
   - **Run the following command to connect using the MySQL socket file from inside the TKGI DB VM**
     ```
     mysql --socket=/var/vcap/sys/run/pxc-mysql/mysqld.sock -u billing -p BILLING-PASSWORD
     ```
     Where `BILLING-PASSWORD` is the billing database password you located in the steps above.
   - **Run the following command to connect by IP address from outside the TKGI DB VM**
     ```
     mysql -h IP-ADDRESS --port 3306 -u billing -p BILLING-PASSWORD
     ```
     Where:
     - `IP-ADDRESS` is the IP address that you located in the steps above.
     - `BILLING-PASSWORD` is the billing database password you located in the steps above.
8. 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)
   ```
9. View the raw pod usage data in the **pods** table by running:
   - **select * from pods**
   For example:
MariaDB [billing]>

```
select *
from pods;
```

```
+--------------------------------------+---------------------+---------------------+--------------+--------------+-------------------------------------------------------+
<table>
<thead>
<tr>
<th>id</th>
<th>first_seen</th>
<th>last_seen</th>
<th>namespace</th>
<th>name</th>
<th>service_instance_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>12a345b6-7890-13c4-de5f-67890a123b4c</td>
<td>2019-01-07 13:57:03</td>
<td>2019-01-08 11:34:33</td>
<td>my-namespace</td>
<td>my-pod</td>
<td>service-instance_a12b3456-78cd-90e1-fa2b-3456c7890def0</td>
</tr>
<tr>
<td>ac203f27-104b-11e9-b520-42010a000b0a</td>
<td>2019-01-04 18:09:04</td>
<td>2019-01-07 14:09:03</td>
<td>my-namespace</td>
<td>my-other-pod</td>
<td>service-instance_a12b3456-78cd-90e1-fa2b-3456c7890def0</td>
</tr>
</tbody>
</table>
+---------------------------------------+---------------------+---------------------+--------------+--------------+-------------------------------------------------------+
2 rows in set (0.00 sec)
```

10. (Optional) For information about running additional queries against the billing database, see the following articles in the 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 Tanzu Kubernetes Grid Integrated Edition

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

**Page last updated:**

This section describes how to back up and restore the VMware Tanzu Kubernetes Grid Integrated Edition control plane and Tanzu Kubernetes Grid Integrated Edition clusters. Tanzu Kubernetes Grid Integrated Edition uses the BOSH Backup and Restore (BBR) framework to back up and restore the TKGI 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
- Tanzu Kubernetes Grid Integrated Edition control plane API VM and its ETCD database
- Tanzu Kubernetes Grid Integrated Edition control plane database VM (MySQL)
- Tanzu Kubernetes Grid Integrated Edition 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.

For more information about installing and using BBR, see the following topics:

- Installing BOSH Backup and Restore
- Backing Up Tanzu Kubernetes Grid Integrated Edition
- Restore the BOSH Director
- Restore the Tanzu Kubernetes Grid Integrated Edition Control Plane
- Restore Tanzu Kubernetes Grid Integrated Edition Clusters

For information about troubleshooting BBR, see BBR Logging.

If you are using vSphere to run Tanzu Kubernetes Grid Integrated Edition, 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
Overview
Prerequisite
Configure Your Jumpbox
Transfer BBR to Your Jumpbox

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 Tanzu Kubernetes Grid Integrated Edition deployment.

For more information about using BOSH Backup and Restore, see:

- To perform a backup, see Backing Up Tanzu Kubernetes Grid Integrated Edition
- To perform a restore of the BOSH Director, see Restore the BOSH Director.

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 TKGI 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 Tanzu Kubernetes Grid Integrated Edition Release Notes. To download a BBR installation file, see BOSH Backup and Restore on the VMware Tanzu 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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 VMware Tanzu 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 VMware Tanzu Network.
   - `JUMPBOX-USER` is the ssh username for connecting to the jumpbox.
   - `JUMPBOX-ADDRESS` is the IP address, or hostname, of the jumpbox.

Please send any feedback you have to pks-feedback@pivotal.io.
Backing Up Tanzu Kubernetes Grid Integrated Edition

This topic describes how to use BOSH Backup and Restore (BBR) to back up the VMware Tanzu Kubernetes Grid Integrated Edition Control Plane and its cluster deployments.

Overview

The BOSH Director, Tanzu Kubernetes Grid Integrated Edition 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 TKGI Control Plane, see Restore the Tanzu Kubernetes Grid Integrated Edition Control Plane
- To perform a restore of a cluster deployment, see Restore Tanzu Kubernetes Grid Integrated Edition Clusters

To view the BBR release notes, see the Cloud Foundry documentation, BOSH Backup and Restore Release Notes.

Recommendations

VMware recommends:

- Follow the full procedure documented in this topic when creating a backup. This ensures that you always have a consistent backup of Ops Manager and Tanzu Kubernetes Grid Integrated Edition to restore from.
- Back up frequently, especially before upgrading your Tanzu Kubernetes Grid Integrated Edition deployment.
- For BOSH v270.0 and above (currently in Ops Manager 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 Tanzu Kubernetes Grid Integrated Edition 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
- Tanzu Kubernetes Grid Integrated Edition control plane API VM and its ETCD database
• Tanzu Kubernetes Grid Integrated Edition control plane database VM (MySQL)
• Tanzu Kubernetes Grid Integrated Edition 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.

Prepare to Back Up

Before you use BBR to either back up TKGI or restore TKGI from backup, follow these steps to retrieve deployment information and credentials:

• 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 Names

Verify Your BBR Version

Before running BBR, verify that the installed version of BBR is compatible with your deployment’s current Tanzu Kubernetes Grid Integrated Edition release.


2. To verify the currently installed BBR version, run the following command:

   ```bash
   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.

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 Ops Manager API**.

2. Retrieve the **Bbr Ssh Credentials** by running the following command:

   ```bash
   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:

   ```bash
   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:

   ```bash
   $ 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:

   ```bash
   curl "https://OPS-MAN-FQDN/api/v0/deployed/director/credentials/bbr_ssh_credentials" \
   -X GET \n   -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 **Tanzu Kubernetes Grid Integrated Edition** 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`
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 Tanzu Kubernetes Grid Integrated Edition 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 Names

To locate and record a cluster deployment name, follow the steps below for each cluster:

1. On the command line, run the following command to log in:

   ```
   tkgi login -a TKGI-API -u USERNAME -k
   ```

Note: You must use BOSH credentials that limit the scope of BBR activity to your cluster deployments.
Where:

- **TKGI-API** is the domain name for the TKGI API that you entered in Ops Manager > Tanzu Kubernetes Grid Integrated Edition > TKGI API > API Hostname (FQDN). For example, `api.tkgi.example.com`.
- **USERNAME** is your user name.

See Logging in to Tanzu Kubernetes Grid Integrated Edition for more information about the `tkgi login` command.

**Note:** if your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 TKGI CLI. For information about configuring SAML, see Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider.

2. Identify the cluster ID:

```
tkgi 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 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 Tanzu Kubernetes Grid Integrated Edition

To back up your Tanzu Kubernetes Grid Integrated Edition 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 if 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.
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:

     ```bash
     ssh -4 -D SOCKS-PORT -inc 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:

     ```bash
     $ ssh -4 -D 12345 -inc 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:

     ```bash
     export BOSH_ALL_PROXY=socks5://localhost:SOCKS-PORT
     ```

     Where **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:

     ```bash
     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:

     ```bash
     $ 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, you 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 Tanzu Kubernetes Grid Integrated Edition.
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.
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 Tanzu Kubernetes Grid Integrated Edition BOSH Director

To back up BOSH Director you will validate your current configuration, then execute the `bbr` backup command.

Validate the Tanzu Kubernetes Grid Integrated Edition 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 10.10.0.5 --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 Tanzu Kubernetes Grid Integrated Edition BOSH Director

1. If the pre-backup check succeeds, run the BBR backup command from your jumpbox to back up the TKGI 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 `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:
To back up your Tanzu Kubernetes Grid Integrated Edition Control Plane you will validate the Control Plane, then execute the following sequence:

1. Validate the Tanzu Kubernetes Grid Integrated Edition Control Plane

   * Locate the Tanzu Kubernetes Grid Integrated Edition Deployment Name


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.

3. If the backup command fails, perform the following actions:
   a. Run the command again, adding the **--debug** flag to enable debug logs.
   b. Follow the steps in Recover from a Failing Command.

Back Up the Tanzu Kubernetes Grid Integrated Edition Control Plane

To back up your Tanzu Kubernetes Grid Integrated Edition Control Plane you will validate the Control Plane, then execute the **bbr** backup command.

Locate the Tanzu Kubernetes Grid Integrated Edition Deployment Name

Locate and record your Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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-service
   ```

   ```
   pivotal-container-service-51f08f6402aaa960f041
   service-instance_4ffeb5b5-5182-4faa-9d92-696d97cc9ae1
   backup-and-restore-sdk/1.8.0
   bosh-dns/1.10.0
   bosh-google-kvm-ubuntu-xenial-go_agent/250.25
   ```

   3. Review the returned output. The Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition Control Plane

1. To confirm that your TKGI control plane is reachable and has a deployment that can be backed up, run the BBR pre-backup check command:

   ```
   $ bbr director --host 10.0.0.5 --username bbr
   --private-key-path private-key.pem
   --ca-cert PATH-TO-BOSH-SERVER-CERT
   --target BOSH-TARGET --username BOSH-CLIENT --deployment DEPLOYMENT-NAME
   ```

   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-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 Tanzu Kubernetes Grid Integrated Edition BOSH deployment name that you located in the Locate the Tanzu Kubernetes Grid Integrated Edition 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.

   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 bbr director --host 10.0.0.5 --username bbr
   --private-key-path private-key.pem
   --ca-cert /var/tempest/workspaces/default/root_ca_certificate
   --target bosh.example.com --username admin --deployment cf-acceptance-0
   ```

   ```
   pivotal-container-service-51f08f6402aaa960f041
   service-instance_4ffeb5b5-5182-4faa-9d92-696d97cc9ae1
   backup-and-restore-sdk/1.8.0
   bosh-dns/1.10.0
   bosh-google-kvm-ubuntu-xenial-go_agent/250.25
   ```

   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 Tanzu Kubernetes Grid Integrated Edition Control Plane

If the pre-backup check succeeds, run the BBR backup command.

1. To back up the TKGI control plane, run the following BBR backup command from your jumpbox:

```bash
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
```

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 Tanzu Kubernetes Grid Integrated Edition BOSH deployment name that you located in the Locate the Tanzu Kubernetes Grid Integrated Edition 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 TKGI 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 TKGI Control Plane in the case of its loss. The backup artifact created by this process contains credentials that you should keep secret.

For example:

```bash
$ BOSH_CLIENT_SECRET=p455w0rd	nohup bbr deployment --all-deployments --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 TKGI cluster deployments you should verify that they can be backed up.

Verify Your Cluster Deployments

To verify that you can reach your TKGI 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:

```bash
BOSH_CLIENT_SECRET=TKGI-UAA-CLIENT-SECRET
bbr deployment --all-deployments --target BOSH-TARGET --username TKGI-UAA-CLIENT-NAME --ca-cert PATH-TO-BOSH-SERVER-CERT
pre-backup-check
```

Where:
- `TKGI-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.
- `TKGI-UAA-CLIENT-NAME` is the value you recorded for `uaa_client_name` in Download the UAA Client Credentials above.
- `PATH-TO-BOSH-CA-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
nohup 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:

```
[21:51:31] -------------------------
[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 TKGI cluster.

4. If the pre-backup-check command fails, do one or more of the following:

- Make sure you are using the correct Tanzu Kubernetes Grid Integrated Edition 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 TKGI 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 TKGI UAA credentials that you recorded in Download the UAA Client Credentials. These credentials limit the scope of the backup to cluster deployments only.

1. To back up all cluster deployments, run the following command from your jumpbox:

   ```
   $ BOSH_CLIENT_SECRET=TKGI-UAA-CLIENT-SECRET
   nohup bbr deployment
   --all-deployments --target BOSH-TARGET --username TKGI-UAA-CLIENT-NAME
   --ca-cert PATH-TO-BOSH-SERVER-CERT
   backup --with-manifest [--artifact-path]
   ```

   Where:
   - `TKGI-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 the `bbr` commands.
   - `TKGI-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 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 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=TKGI-UAA-CLIENT-SECRET
  nohup bbr deployment
      --deployment CLUSTER-DEPLOYMENT-NAME
      --target BOSH-DIRECTOR-IP
      --username TKGI-UAA-CLIENT-NAME
      --ca-cert PATH-TO-BOSH-SERVER-CERT
  backup
      [--with-manifest]
      [--artifact-path]
```

Where:

- **TKGI-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-DIRECTOR** 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.
- **TKGI-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-90101112131415
      --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 Tanzu Kubernetes Grid Integrated Edition on vSphere with Flannel or NSX-T networking, you must back up your vCenter in addition to completing the BBR procedures above.

For Tanzu Kubernetes Grid Integrated Edition deployments with NSX-T networking, you must also back up the NSX Manager.

To complete the backup of your Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition

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 Tanzu Kubernetes Grid Integrated Edition, test your backup artifact by following the procedures in:
   - Restore the Tanzu Kubernetes Grid Integrated Edition BOSH Director
   - Restore the Tanzu Kubernetes Grid Integrated Edition Control Plane
   - Restore Tanzu Kubernetes Grid Integrated Edition 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.

**Note:** 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 TKGI BOSH Director backup failed,** run the following BBR cleanup script command to clean up:

  ```bash
  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 shown 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:
  ```bash
  bbr director --host 10.0.0.5
  --username bbr --private-key-path private-key.pem
  backup-cleanup
  ```

- **If the TKGI control plane or TKGI clusters backups fail,** run the following BBR cleanup script command to clean up:

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 Tanzu Kubernetes Grid Integrated Edition BOSH deployment name that you located in the Locate the Tanzu Kubernetes Grid Integrated Edition 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=ps55word 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.

<table>
<thead>
<tr>
<th>Value</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td>1</td>
<td>General failure</td>
</tr>
<tr>
<td>8</td>
<td>The post-backup unlock failed. One of your deployments might be in a bad state and require attention.</td>
</tr>
<tr>
<td>16</td>
<td>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.</td>
</tr>
</tbody>
</table>

Please send any feedback you have to pks-feedback@pivotal.io.
Restoring Tanzu Kubernetes Grid Integrated Edition

In this topic
Overview
Compatibility of Restore
Prepare to Restore a Backup
Transfer Artifacts to Your Jumpbox
Restore the BOSH Director
  Deploy Ops Manager
  Import Installation Settings
  (Optional) Configure Ops Manager for New Resources
  Remove BOSH State File
  Deploy the BOSH Director
  Restore the BOSH Director
  Remove All Stale Deployment Cloud IDs
  Restore the Tanzu Kubernetes Grid Integrated Edition Control Plane
    Determine the Required Stemcell
    Upload Stemcells
    Redeploy the Tanzu Kubernetes Grid Integrated Edition Control Plane
    Restore the TKGI Control Plane
  Redeploy and Restore Clusters
    Redeploy Clusters
    Restore Clusters
  Register Restored Worker VMs
    Delete Nodes
    Restart kubelet
  Resolve a Failing BBR Restore Command
  Cancel a Restore
  Clean Up After a Failed Restore

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to use BOSH Backup and Restore (BBR) to restore the BOSH Director, VMware Tanzu Kubernetes Grid Integrated Edition control plane, and Kubernetes 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 Tanzu Kubernetes Grid Integrated Edition 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.

If you are running Tanzu Kubernetes Grid Integrated Edition 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.

Prepare to Restore a Backup

Before you use BBR to either back up TKGI or restore TKGI from backup, follow these steps to retrieve deployment information and credentials:

- 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 Names

Verify Your BBR Version

Before running BBR, verify that the installed version of BBR is compatible with your deployment’s current Tanzu Kubernetes Grid Integrated Edition release.


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.

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 Ops Manager 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:

   ```bash
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 **Tanzu Kubernetes Grid Integrated Edition** 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:

   ```bash
   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 Tanzu Kubernetes Grid Integrated Edition 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 Names

To locate and record a cluster deployment name, follow the steps below for each cluster:

1. On the command line, run the following command to log in:

   ```
tkgi login -a TKGI-API -u USERNAME -k
   ```

   Where:

   - **TKGI-API** is the domain name for the TKGI API that you entered in **Ops Manager > Tanzu Kubernetes Grid**
Integrated Edition > TKGI API > API Hostname (FQDN). For example, api.tkgi.example.com.

- USERNAME is your user name.

See Logging in to Tanzu Kubernetes Grid Integrated Edition for more information about the `tkgi login` command.

**Note:** If your operator has configured Tanzu Kubernetes Grid Integrated Edition 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 TKGI CLI. For information about configuring SAML, see Connecting Tanzu Kubernetes Grid Integrated Edition to a SAML Identity Provider.

2. Identify the cluster ID:

```
tkgi 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 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, Tanzu Kubernetes Grid Integrated Edition 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:
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 Tanzu Kubernetes Grid Integrated Edition Ops Manager to restore the installation settings artifacts saved when following the Export Installation Settings backup procedure steps. To redeploys 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 Tanzu Kubernetes Grid Integrated Edition by following the installation instructions specific to your IaaS in Installing Tanzu Kubernetes Grid Integrated Edition.

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 before instructed to in the steps Deploy the BOSH Director or Redeploy the Tanzu Kubernetes Grid Integrated Edition 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 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:
   - **Project ID** to reflect the GCP project ID.
   - **Default Deployment Tag** to reflect the environment name.
   - **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 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 your jumpbox.

To restore the BOSH Director:

1. Ensure the Tanzu Kubernetes Grid Integrated Edition BOSH Director backup artifact is in the folder from which you run BBR.

2. Run the BBR restore command to restore the TKGI 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 TKGI 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
```
3. If your BOSH Director restore 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 Resolve a Failing BBR Restore Command below.

   Be sure to complete the steps in Clean Up After a Failed Restore below.

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:

   ```shell
   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:

   ```shell
   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 Tanzu Kubernetes Grid Integrated Edition Control Plane

You must redeploy the Tanzu Kubernetes Grid Integrated Edition tile before restoring the Tanzu Kubernetes Grid Integrated Edition control plane. By redeploying the Tanzu Kubernetes Grid Integrated Edition tile you create the VMs that constitute the control plane deployment.

To redeploy the Tanzu Kubernetes Grid Integrated Edition tile, do the following:

- **Determine the Required Stemcell needed by the tile.**
- **Upload that stemcell as described in Upload Stemcells.**
- **Redeploy the Tanzu Kubernetes Grid Integrated Edition Control Plane**
- **Restore the TKGI Control Plane from a BBR backup on top of the deployment.**
Determine the Required Stemcell

Do either the following procedures to determine the stemcell that TKGI uses:

- **Review the Stemcell Library:**
  1. Open Ops Manager.
  2. Click **Stemcell Library**.
  3. Record the TKGI 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:

     ```
     8 bosh deployments
    Using environment '10.0.0.5' as user 'director' (bosh.*.read, 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 TKGI 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**

To upload the stemcell used by your Tanzu Kubernetes Grid Integrated Edition tile:

1. **Download the stemcell from VMware Tanzu Network**.

2. Run the following command to upload the stemcell used by TKGI:

   ```
   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.
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 TKGI stemcell.

Redeploy the Tanzu Kubernetes Grid Integrated Edition Control Plane

To redeploy your Tanzu Kubernetes Grid Integrated Edition tile's control plane:

1. From the Ops Manager Installation Dashboard, navigate to **VMware Tanzu Kubernetes Grid Integrated Edition > Resource Config**.
2. Ensure the **Upgrade all clusters** errand is **Off**.
3. Ensure that all other errands needed by your system are set to run.
4. Return to the Ops Manager Installation Dashboard.
5. Click **Review Pending Changes**.
6. Review your changes. For more information, see **Reviewing Pending Product Changes**.
7. Click **Apply Changes** to redeploy the control plane.

Restore the TKGI Control Plane

Restore the Tanzu Kubernetes Grid Integrated Edition control plane by running BBR commands on your jumpbox.

To restore the Tanzu Kubernetes Grid Integrated Edition control plane:

1. Ensure the Tanzu Kubernetes Grid Integrated Edition deployment backup artifact is in the folder from which you run BBR.
2. Run the BBR restore command to restore the TKGI 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 Tanzu Kubernetes Grid Integrated Edition 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 TKGI control plane backup that you want to restore.

   For example:
$ BOSH_CLIENT_SECRET=ps555w0rd
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

Note: The BBR restore command can take a long time to complete. The command above 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.

3. If your Tanzu Kubernetes Grid Integrated Edition control plane restore 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 Resolve a Failing BBR Restore Command below.

Be sure to complete the steps in Clean Up After a Failed Restore below.

Redeploy and Restore Clusters

After restoring the Tanzu Kubernetes Grid Integrated Edition control plane, perform the following steps to redeploy the TKGI-provisioned Kubernetes clusters and restore their state from backup.

Redeploy Clusters

Before restoring your TKGI-provisioned clusters, you must redeploy them to BOSH. To redeploy TKGI-provisioned clusters:

- If you want to redeploy all clusters simultaneously, see Redeploy All Clusters.
- If you want to redeploy one cluster at a time, see Redeploy a Single Cluster.

Redeploy All Clusters

To redeploy all clusters:

1. In Ops Manager, navigate to the Tanzu Kubernetes Grid Integrated Edition tile.
2. Click Errands.
3. Ensure the Upgrade all clusters errand is On. This errand redeployes all your TKGI-provisioned clusters.
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.

Redeploy a Single Cluster

To redeploy a TKGI-provisioned cluster through the TKGI CLI:

1. Identify the names of your TKGI-provisioned clusters:
2. For each cluster you want to redeploy, run the following command:

```
tkgi upgrade-cluster CLUSTER-NAME
```

Where **CLUSTER-NAME** is the name of your Kubernetes cluster. For more information, see Upgrade Clusters.

-provisioned cluster through the BOSH CLI:

1. Identify the names of your cluster deployments:

   ```
   bosh deployments
   ```

1. For each cluster you want to redeploy, do the following:

   1. Download the cluster deployment manifest:

   ```
   bosh -d DEPLOYMENT-NAME manifest /tmp/YOUR-DEPLOYMENT-MANIFEST.yml
   ```

   Where:
   * `DEPLOYMENT-NAME` is the name of your Kubernetes cluster deployment.
   * `YOUR-DEPLOYMENT-MANIFEST` is the name of your Kubernetes cluster deployment manifest.

   1. Deploy the cluster:

   ```
   bosh -d DEPLOYMENT-NAME deploy YOUR-DEPLOYMENT-MANIFEST.yml
   ```

   Where:
   * `DEPLOYMENT-NAME` is the name of your Kubernetes cluster deployment.
   * `YOUR-DEPLOYMENT-MANIFEST` is the name of your Kubernetes cluster deployment manifest.

   1. Run all errands needed for your cluster:

   ```
   bosh -d DEPLOYMENT-NAME run-errand ERRAND-NAME
   ```

   Where:
   * `DEPLOYMENT-NAME` is the name of your Kubernetes cluster deployment.
   * `ERRAND-NAME` is the name of the errand you want to run.

   To list errands available in your deployment, use the `bosh errands` command.

   For example:

   ```
   bosh -d service-instance_8de000ff-a87a-4930-81ba-106ed42c2471e run-errand apply-addons
   ```

Restore Clusters

After redeploying your TKGi-provisioned clusters, restore their stateless workloads and cluster state from backup by running the BOSH restore command from your jumpbox. Stateless workloads are tracked in the cluster etcd database, which BBR backs up.

⚠️ **warning:** BBR does not back up persistent volumes, load balancers, or other IaaS resources.

⚠️ **warning:** When you restore a cluster, etcd is stopped in the API server. During this process, only currently-deployed clusters
To restore a cluster:

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:

   ```bash
   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 Retrieve Your Cluster Deployment Names 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 your deployment backup. Make sure you have transfer your artifact into your jumpbox as described in Transfer Artifacts to Jumpbox above.

   For example:

   ```bash
   $ 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. To cancel a running `bbr restore`, see Cancel a Restore below.

5. After you restore a Kubernetes cluster, you must register its workers with their master nodes by following the Register Restored Worker VMs steps below.

---

Note: The BBR restore command can take a long time to complete. The BBR restore command above 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.
6. If your Tanzu Kubernetes Grid Integrated Edition cluster restore 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 Resolve a Failing BBR Restore Command below.

Be sure to complete the steps in Clean Up After a Failed Restore below.

Register Restored Worker VMs

After restoring a Kubernetes cluster, you must register all of the cluster’s worker nodes with their master nodes. To register cluster worker nodes, complete the following:

1. **Delete Nodes**

2. **Restart kubelet**

Delete Nodes

To delete a cluster’s restored nodes:

1. To determine your cluster’s namespace, run the following command:

   ```bash
toilet get all --all-namespaces
   ```

2. To retrieve the list of worker nodes in the cluster, run the following command:

   ```bash
toilet get nodes -o wide
   ```

   Document the worker node names listed in the `NAME` column. The worker nodes should all be listed with a status of `NotReady`.

3. To delete a node, run the following:

   ```bash
toilet delete node NODE-NAME
   ```

   Where `NODE-NAME` is a node `NAME` returned by the `kubectl get nodes` command.

4. Repeat the preceding `kubectl delete node` step for each of your cluster’s nodes.

Restart kubelet

To restart `kubelet` on your worker node VMs:

1. To restart `kubelet` on all of your cluster’s worker node VMs, run the following command:

   ```bashosh ssh -d DEPLOYMENT-NAME worker -c '/var/vcap/bin/monit restart kubelet'
   ```

   Where `DEPLOYMENT-NAME` is the cluster BOSH deployment name that you recorded in Retrieve Your Cluster Deployment Names above.

2. To confirm all worker nodes in your cluster have been restored to a `Ready` state, run the following command:
Resolve a Failing BBR Restore Command

To resolve a failing BBR restore 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 \n--host BOSH-DIRECTOR-IP \n--username bbr \n--private-key-path PRIVATE-KEY-FILE \nrestore-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:
To resolve issues following a failed control plane restore, run the following BBR command:

```bash
$ nohup bbr director --target 10.0.0.5 --username bbr --private-key-path private.pem restore-cleanup
```

To resolve issues following a failed cluster restore, run the following BBR command:

```bash
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](#).
- **PATH-TO-BOSH-CA-CERT** is the path to the root CA certificate that you downloaded in [Download the Root CA Certificate](#).

For example:

```bash
$ BOSH_CLIENT_SECRET=p455w0rd
bbr deployment
--target bosh.example.com
--username admin
--deployment pivotal-container-service-453f2f
--ca-cert bosh.ca.crt
restore-cleanup
```

Copyright © 2020 VMware, Inc. All Rights Reserved.
$ 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

Understand Logging

⚠ Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

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.
Tanzu Kubernetes Grid Integrated Edition Security

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This section includes security topics for VMware Tanzu Kubernetes Grid Integrated Edition.

See the following topic:


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

In this topic
- Security Issues in Kubernetes
- Security Issues from CFF
- Security Issues in VMware NSX
- Security Issues in VMware Harbor

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes the processes for disclosing security issues and releasing related fixes for VMware Tanzu Kubernetes Grid Integrated Edition, Kubernetes, VMware NSX, and VMware Harbor.


Security fixes are provided in accordance with the Ops Manager Security Overview and 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

VMware follows 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, Tanzu Kubernetes Grid Integrated Edition releases fixes according to the Ops Manager Security Overview and Policy.

Security Issues from CFF

VMware follows the Cloud Foundry Foundation (CFF) responsible disclosure process to report and address suspected security issues.

This process is discussed in Cloud Foundry Security.

When the Cloud Foundry Foundation releases security fixes, Tanzu Kubernetes Grid Integrated Edition releases fixes according to the Ops Manager Security Overview and 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 Tanzu Kubernetes Grid Integrated Edition

**Note:** As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

**Page last updated:**

This topic is intended to provide assistance when diagnosing and troubleshooting issues installing or using VMware Tanzu Kubernetes Grid Integrated Edition.

See the following sections:

- General Troubleshooting
- Using BOSH Diagnostic Commands in Tanzu Kubernetes Grid Integrated Edition
- Verifying Deployment Health
- Service Interruptions
- Troubleshooting the Management Console

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

In this topic
TKGI API is Slow or Times Out
All Cluster Operations Fail
Cluster Creation Fails
Cannot Re-Create a Cluster that Failed to Deploy
Cannot Access Add-On Features or Functions
Resurrecting VMs Causes Incorrect Permissions in vSphere HA
Worker Node Hangs Indefinitely
Cannot Authenticate to an OpenID Connect-Enabled Cluster
Error: Login Failed
Error: Failed Jobs
Error: No Such Host
Error: FailedMount
Error: Plan Not Found

TKGI API is Slow or Times Out

Symptom
When you run TKGI CLI commands, the TKGI API times out or is slow to respond.

Explanation
The TKGI 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.
4. For the TKGI API 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

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:
Changes.

8. Click **Apply Changes**.

All Cluster Operations Fail

**Symptom**

All TKGI CLI cluster operations fail including attempts to create or delete clusters with `tkgi create-cluster` and `tkgi delete-cluster`.

The output of `tkgi 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 Tanzu Kubernetes Grid Integrated Edition for more information about logging in to the BOSH Director.

2. In the BOSH command output, locate a task that attempted to perform a cluster operation, such as cluster creation or deletion.

3. To retrieve more information about the task, run the following command:

   ```
   bosh -e MY-ENVIRONMENT task TASK-NUMBER
   ```

   Where:
   - `MY-ENVIRONMENT` is the name of your BOSH environment.
   - `TASK-NUMBER` is the number of the task that attempted to create the cluster.

   For example:

   ```
   $ bosh -e tkgi 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:
7. Make a note of the plan assigned to the failing node.

Solution

1. In the Tanzu Kubernetes Grid Integrated Edition 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 TKGI Tile.

Cluster Creation Fails

Symptom

When creating a cluster, you run `tkgi 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 Tanzu Kubernetes Grid Integrated Edition for more information about logging in to the BOSH Director.

2. In the BOSH command output, locate the task that attempted to create the cluster.

3. To retrieve more information about the task, run the following command:

   ```
   bosh -e MY-ENVIRONMENT task TASK-NUMBER
   ```

   Where:
   - `MY-ENVIRONMENT` is the name of your BOSH environment.
   - `TASK-NUMBER` is the number of the task that attempted to create the cluster.

   For example:

   ```
   $ bosh -e tkgi 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 `tkgi create-cluster` to attempt creating the cluster again.

Explanation

Tanzu Kubernetes Grid Integrated Edition does not automatically clean up the failed BOSH deployment. Running `tkgi 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 `tkgi 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 Tanzu Kubernetes Grid Integrated Edition.

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 TKGI command:

   ```
   tkgi delete-cluster CLUSTER-NAME
   ```

   Where `CLUSTER-NAME` is the name of your Tanzu Kubernetes Grid Integrated Edition cluster.

4. To re-create the cluster, run the following TKGI command:

   ```
   tkgi create-cluster CLUSTER-NAME
   ```

   Where `CLUSTER-NAME` is the name of your Tanzu Kubernetes Grid Integrated Edition cluster.

Cannot Access Add-On Features or Functions

Symptom

You cannot access a feature or function provided by a Kubernetes add-on.

For example, pods cannot resolve DNS names, and error messages report the service `CoreDNS` is invalid. If `CoreDNS` is not deployed, the cluster typically fails to start.
**Explanation**

Kubernetes features and functions are provided by Tanzu Kubernetes Grid Integrated Edition add-ons. DNS resolution, for example, is provided by the CoreDNS service.

To enable these add-ons, Ops Manager must run scripts after deploying Tanzu Kubernetes Grid Integrated Edition. 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/](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 `tkgi delete-cluster` on the command line to delete the cluster. For more information, see Deleting Clusters.

10. On the command line, enter `tkgi 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 BOSH documentation.

**Solution**

---

Copyright © 2020 VMware, Inc. All Rights Reserved.
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 TKGI Tile in Upgrading Tanzu Kubernetes Grid Integrated Edition (Flannel Networking)**.

#### Explanation

During the Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition, see **Prepare to Upgrade in Upgrading Tanzu Kubernetes Grid Integrated Edition (Flannel Networking)**.

- Configure the default node drain behavior by doing the following:
1. Navigate to **Ops Manager Installation** > **Tanzu Kubernetes Grid Integrated Edition** > **Plans**.
2. Set the default node drain behavior by configuring the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node Drain Timeout</td>
<td>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.</td>
</tr>
<tr>
<td>Pod Shutdown Grace</td>
<td>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.</td>
</tr>
<tr>
<td>Force node to drain even if it has running pods not managed by a ReplicationController, ReplicaSet, Job, DaemonSet or StatefulSet.</td>
<td>If you enable this configuration, the node still drains when pods are not managed by a ReplicationController, ReplicaSet, Job, DaemonSet or StatefulSet.</td>
</tr>
<tr>
<td>Force node to drain even if it has running DaemonSet-managed pods.</td>
<td>If you enable this configuration, the node still drains when pods are managed by a DaemonSet.</td>
</tr>
<tr>
<td>Force node to drain even if it has running running pods using emptyDir.</td>
<td>If you enable this configuration, the node still drains when pods are using a emptyDir volume.</td>
</tr>
<tr>
<td>Force node to drain even if pods are still running after timeout.</td>
<td>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.</td>
</tr>
</tbody>
</table>

**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 TKGI CLI to configure node drain behavior. To configure the default node drain behavior with the TKGI CLI, run `tkgi update-cluster` with an action flag. You can view the current node drain behavior with `tkgi cluster --details`. For more information, see **Configure Node Drain Behavior in Upgrade Preparation Checklist for Tanzu Kubernetes Grid Integrated Edition v1.8**.

---

### 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 TKGI CLI to v1.2.0 or later. To download the TKGI CLI, navigate to VMware Tanzu Network. For more information, see [Installing the TKGI CLI](#).

2. Obtain a kubeconfig file that contains the new tokens by running the following command:

   ```bash
   tkgi get-credentials CLUSTER-NAME
   ```

   Where **CLUSTER-NAME** is the name of your cluster.

   For example:

   ```bash
   $ tkgi get-credentials tkgi-example-cluster
   Fetching credentials for cluster tkgi-example-cluster.
   Context set for cluster tkgi-example-cluster.
   You can now switch between clusters by using:
   $ kubectl config use-context <cluster-name>
   ```

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: Login Failed**

**Symptom**

TKGI 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 Tanzu Kubernetes Grid Integrated Edition deployment. As a result, the persistent disk on the TKGI Database VM runs out of space.

**Solution**

1. Check the total number of pods in your Tanzu Kubernetes Grid Integrated Edition 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 TKGI Database VM.

3. If available disk space is low, increase the amount of persistent disk storage on the TKGI Database VM depending on the number of pods in your Tanzu Kubernetes Grid Integrated Edition 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 TKGI Database VM as follows:

<table>
<thead>
<tr>
<th>Number of Pods</th>
<th>Persistent Disk Requirements (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000 pods</td>
<td>20</td>
</tr>
<tr>
<td>5,000 pods</td>
<td>100</td>
</tr>
<tr>
<td>10,000 pods</td>
<td>200</td>
</tr>
<tr>
<td>50,000 pods</td>
<td>1,000</td>
</tr>
</tbody>
</table>

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 Tanzu Kubernetes Grid Integrated Edition, 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/](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 Tanzu Kubernetes Grid Integrated Edition, follow the steps below:
   - a. On the command line, enter `tkgi delete-cluster` to delete the cluster. For more information, see **Deleting Clusters**.
   - b. Enter `tkgi 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: 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 TKGI 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 `tkgi 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 Tanzu Kubernetes Grid Integrated Edition

In this topic

Overview
Log in to the BOSH Director VM
SSH into the TKGI API VM
SSH into the TKGI Database VM
SSH into a Kubernetes Cluster VM
View Log Files

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes how to access information about your VMware Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition deployment. For example, you can access Tanzu Kubernetes Grid Integrated Edition log files after SSHing into the TKGI API or a Kubernetes cluster VM:

1. SSH into the TKGI 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 Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition environment, run the following command:

   ```bash
   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, `tkgi`.
   - `BOSH-DIRECTOR-IP` is the BOSH Director IP address you located in the first step. For example, `10.0.0.3`.

Copyright © 2020 VMware, Inc. All Rights Reserved.
For example:

```
$ bosh alias-env tkgi -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 tkgi 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 Ops Manager documentation.

**SSH into the TKGI API VM**

To SSH into the TKGI 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 TKGI deployment name, run the following command:

```
bosh -e ENVIRONMENT deployments
```

Where `ENVIRONMENT` is your BOSH environment alias.

For example:

```
$ bosh -e tkgi deployments
```

Your TKGI deployment name begins with `pivotal-container-service` and includes a BOSH-generated identifier.

3. To identify your TKGI API VM name, run the following command:

```
bosh -e ENVIRONMENT -d DEPLOYMENT vms
```

Where:

- `ENVIRONMENT` is the BOSH environment alias.
- `DEPLOYMENT` is your TKGI deployment name.

For example:

```
$ bosh -e tkgi -d pivotal-container-service-a1b2c333d444e5f6a77 vms
```

Your TKGI API VM name begins with `pivotal-container-service` and includes a BOSH-generated identifier.

**Note:** The TKGI API VM identifier is different from the identifier in your TKGI deployment name.
4. To SSH into the TKGI API VM, run the following command:

```bash
bosh -e ENVIRONMENT -d DEPLOYMENT ssh TKGI-API-VM
```

Where:
- `ENVIRONMENT` is the BOSH environment alias.
- `DEPLOYMENT` is your TKGI deployment name.
- `TKGI-API-VM` is your TKGI API VM name.

For example:

```bash
$ bosh -e tkgi -d pivotal-container-service-a1b2c333d444e5f66a77 ssh pivotal-container-service/000a1111-222b-3333-4cc5-de66f7a8899b
```

### SSH into the TKGI Database VM

To SSH into a TKGI Database 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 TKGI deployment name, run the following command:

   ```bash
   bosh -e ENVIRONMENT deployments
   ```

   Where `ENVIRONMENT` is your BOSH environment alias.

   For example:

   ```bash
   $ bosh -e tkgi deployments
   ```

   Your TKGI deployment name begins with `pivotal-container-service` and includes a BOSH-generated identifier.

3. To identify your TKGI Database VM name, run the following command:

   ```bash
   bosh -e ENVIRONMENT -d DEPLOYMENT vms
   ```

   Where:
   - `ENVIRONMENT` is the BOSH environment alias.
   - `DEPLOYMENT` is your TKGI deployment name.

   For example:

   ```bash
   $ bosh -e tkgi -d pivotal-container-service-a1b2c333d444e5f66a77 vms
   ```

   Your TKGI Database VM name begins with `pks-db` and includes a BOSH-generated identifier.

4. To SSH into the TKGI Database VM, run the following command:

   ```bash
   bosh -e ENVIRONMENT -d DEPLOYMENT ssh TKGI-DB-VM
   ```
Where:

- **ENVIRONMENT** is the BOSH environment alias.
- **DEPLOYMENT** is your TKGI deployment name.
- **TKGI-DB-VM** is the name of the TKGI Database VM to SSH into.

For example:

```
$ bosh -e tkgi \
-d pivotal-container-service-a1b2c333d444e5f6a77 \
ssh pks-db/000a4444-555b-6666-4cc5-de66f8a9900b
```

**SSH into a Kubernetes Cluster VM**

Each Kubernetes cluster corresponds to a BOSH deployment. To SSH into a TKGI-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 tkgi 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 tkgi -d service-instance_ae681cd1-7f84-4661-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 tkgi -d service-instance_ae681cd1-7f84-4661-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 Tanzu Kubernetes Grid Integrated Edition deployment. To access Tanzu Kubernetes Grid Integrated Edition log files, SSH into the TKGI API VM, 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.

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

In this topic
Verify Kubernetes Node and Pod Health
Verify Kubernetes Cluster Health
Retrieve Cluster Upgrade Task ID
Verify NCP Health (NSX-T Only)

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:
This topic describes how to check the health of your Tanzu Kubernetes Grid Integrated Edition deployment and the nodes, pods, and clusters that it hosts.

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:

   ```bash
token = `bosh -e ENVIRONMENT login`
   ```

   Where `ENVIRONMENT` is the alias you set for your BOSH Director. For more information, see Using BOSH Diagnostic Commands in Tanzu Kubernetes Grid Integrated Edition.

   For example:

   ```bash
   $ bosh -e tkgi login
   ```

2. To verify that all nodes are in a ready state, run the following command for all Kubernetes contexts:

   ```bash
   kubectl get nodes
   ```

3. To verify that all pods are running, run the following command for all Kubernetes contexts:

   ```bash
   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:

   ```bash
token = `bosh -e ENVIRONMENT login`
   ```

   Where `ENVIRONMENT` is the alias you set for your BOSH Director. For more information, see Using BOSH Diagnostic Commands in Tanzu Kubernetes Grid Integrated Edition.

   For example:

   ```bash
   $ bosh -e tkgi login
   ```

2. To get the deployment name of a target Kubernetes cluster, run the following command:

   ```bash
   bosh wards deploy
   ```

   ```bash
   deployment = `bosh wards deploy`
   ```

   ```bash
   for deployment in deployments:
       print deployment
   ```
bosh deployments

For example:

```
$ bosh deployments
Using environment `30.0.0.10` as client `ops_manager`

<table>
<thead>
<tr>
<th>Name</th>
<th>Release(s)</th>
<th>Stemcell(s)</th>
<th>Team(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>harbor-container-registry-b4023f685720f2b27399</td>
<td>bosh-dns/1.10.0</td>
<td>bosh-vsphere-exi-ubuntu-xenial-go_agent/170.15 -</td>
<td></td>
</tr>
<tr>
<td>harbor-container-registry/1.7.3-build.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bosh-dns/1.10.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bosh-vsphere-exi-ubuntu-xenial-go_agent/170.15 -</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>backup-and-restore-sdk/1.8.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bosh-vsphere-exi-ubuntu-xenial-go_agent/170.15 -</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pivotal-container-service-7e64d53fc57050385690</td>
<td>bosh-dns/1.10.0</td>
<td>bosh-vsphere-exi-ubuntu-xenial-go_agent/170.15 -</td>
<td></td>
</tr>
<tr>
<td>backup-and-restore-sdk/1.8.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bosh-vsphere-exi-ubuntu-xenial-go_agent/170.15 -</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k8s-api/50.0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k8s-nsx-v1.19.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k8s-telemetry/2.0.0-build.113</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>syslog/11.4.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uaa/64.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wavefront-proxy/0.9.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>service-instance_8de000ff-a87a-4930-81ba-106d42e2471e</td>
<td>bosh-dns/1.10.0</td>
<td>bosh-vsphere-exi-ubuntu-xenial-go_agent/170.15</td>
<td>pivots-container-service-7e64d53fc57050385690</td>
</tr>
<tr>
<td>bpsms/0.13.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cf-mysql/36.14.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cf-or-etc/1.8.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>docker/35.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>harbor-container-registry/1.7.3-build.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kubos/0.25.9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kubos-service-adapter/1.3.3-build.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k8s-nx-ceti/2.3.1.10692340</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nsx-vk/v1.0.20.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sink-resources-release/0.1.15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>syslog/11.4.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wavefront-proxy/0.9.0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3 deployments
```

In the example above, `service-instance_8de000ff-a87a-4930-81ba-106d42e2471e` is the Kubernetes cluster deployment name.

**Note:** If you have deployed multiple Kubernetes clusters, determine the UUID using `tkg 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:
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:
To retrieve the BOSH task ID for a recent cluster upgrade:

1. **(Optional)** To retrieve a list of clusters, run the `bsh clusters` command. For example:

```bash
$ bosh -d service-instance_f8de00ff-8a71-4930-81ba-106d42e2471e instances --ps
```

Deployment `service-instance_f8de00ff-8a71-4930-81ba-106d42e2471e`

<table>
<thead>
<tr>
<th>Instance</th>
<th>Process</th>
<th>Process State</th>
<th>AZ</th>
<th>IPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>master/b6d2e3b3-4f3d-4c73-d99370e2d6</td>
<td>bosh-dns</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>master/b6d2e3b3-4f3d-4c73-d99370e2d6</td>
<td>bosh-dns-healthcheck</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>master/b6d2e3b3-4f3d-4c73-d99370e2d6</td>
<td>bosh-dns-resolvconf</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>master/b6d2e3b3-4f3d-4c73-d99370e2d6</td>
<td>etcd</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>master/b6d2e3b3-4f3d-4c73-d99370e2d6</td>
<td>kube-api-server</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>master/b6d2e3b3-4f3d-4c73-d99370e2d6</td>
<td>kube-controller-manager</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>master/b6d2e3b3-4f3d-4c73-d99370e2d6</td>
<td>kube-scheduler</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>master/b6d2e3b3-4f3d-4c73-d99370e2d6</td>
<td>ncp</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>master/b6d2e3b3-4f3d-4c73-d99370e2d6</td>
<td>pks-helps-bosh-dns-resolvconf</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d4e3f0a2-b9d7-4801-892a-3950875026</td>
<td>blackbox</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d4e3f0a2-b9d7-4801-892a-3950875026</td>
<td>bosh-dns</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d4e3f0a2-b9d7-4801-892a-3950875026</td>
<td>bosh-dns-healthcheck</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d4e3f0a2-b9d7-4801-892a-3950875026</td>
<td>bosh-dns-resolvconf</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d4e3f0a2-b9d7-4801-892a-3950875026</td>
<td>docker</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d4e3f0a2-b9d7-4801-892a-3950875026</td>
<td>kube-proxy</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d4e3f0a2-b9d7-4801-892a-3950875026</td>
<td>kubelet</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d4e3f0a2-b9d7-4801-892a-3950875026</td>
<td>nsm-kube-proxy</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d4e3f0a2-b9d7-4801-892a-3950875026</td>
<td>nsm-node-agent</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d4e3f0a2-b9d7-4801-892a-3950875026</td>
<td>ovs-vswitchd</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d4e3f0a2-b9d7-4801-892a-3950875026</td>
<td>ovsdb-server</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d4e3f0a2-b9d7-4801-892a-3950875026</td>
<td>pks-helps-bosh-dns-resolvconf</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d7f9f44b-33d0-9f8b-0b7f-69e502</td>
<td>blackbox</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d7f9f44b-33d0-9f8b-0b7f-69e502</td>
<td>bosh-dns</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d7f9f44b-33d0-9f8b-0b7f-69e502</td>
<td>bosh-dns-healthcheck</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d7f9f44b-33d0-9f8b-0b7f-69e502</td>
<td>bosh-dns-resolvconf</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d7f9f44b-33d0-9f8b-0b7f-69e502</td>
<td>docker</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d7f9f44b-33d0-9f8b-0b7f-69e502</td>
<td>kube-proxy</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d7f9f44b-33d0-9f8b-0b7f-69e502</td>
<td>kubelet</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d7f9f44b-33d0-9f8b-0b7f-69e502</td>
<td>nsm-kube-proxy</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d7f9f44b-33d0-9f8b-0b7f-69e502</td>
<td>nsm-node-agent</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d7f9f44b-33d0-9f8b-0b7f-69e502</td>
<td>ovs-vswitchd</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d7f9f44b-33d0-9f8b-0b7f-69e502</td>
<td>ovsdb-server</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/d7f9f44b-33d0-9f8b-0b7f-69e502</td>
<td>pks-helps-bosh-dns-resolvconf</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/e5e25e44-7500-4d19-990b-89546118c502</td>
<td>blackbox</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/e5e25e44-7500-4d19-990b-89546118c502</td>
<td>bosh-dns</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/e5e25e44-7500-4d19-990b-89546118c502</td>
<td>bosh-dns-healthcheck</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/e5e25e44-7500-4d19-990b-89546118c502</td>
<td>bosh-dns-resolvconf</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/e5e25e44-7500-4d19-990b-89546118c502</td>
<td>docker</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/e5e25e44-7500-4d19-990b-89546118c502</td>
<td>kube-proxy</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/e5e25e44-7500-4d19-990b-89546118c502</td>
<td>kubelet</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/e5e25e44-7500-4d19-990b-89546118c502</td>
<td>nsm-kube-proxy</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/e5e25e44-7500-4d19-990b-89546118c502</td>
<td>nsm-node-agent</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/e5e25e44-7500-4d19-990b-89546118c502</td>
<td>ovs-vswitchd</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/e5e25e44-7500-4d19-990b-89546118c502</td>
<td>ovsdb-server</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>worker/e5e25e44-7500-4d19-990b-89546118c502</td>
<td>pks-helps-bosh-dns-resolvconf</td>
<td>running</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

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 Tanzu Kubernetes Grid Integrated Edition environment:

   ```bash
   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:

   ```bash
   $ bosh -e tkgi -d service-instance_8de6000ff-a87a-4930-81ba-106d42c2471e instances --ps
   
   Using environment '30.0.0.10' as client 'ops_manager'
   ```

   **Task 678. Done**

   Deployment 'service-instance_8de6000ff-a87a-4930-81ba-106d42c2471e'

<table>
<thead>
<tr>
<th>Instance</th>
<th>Process</th>
<th>Process State</th>
<th>AZ</th>
<th>IPs</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>apply-addons</td>
<td>cddf06a9e-d3b1-4482-b63c-431dbbb3730c26 -</td>
<td></td>
<td></td>
<td></td>
<td>AZ-K8S:</td>
</tr>
<tr>
<td>master-366d5c263-1682-4e79-87b6-15930125761d</td>
<td></td>
<td>running</td>
<td></td>
<td></td>
<td>40.0.2.2</td>
</tr>
<tr>
<td></td>
<td>blackbox</td>
<td>running</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bosh-dns</td>
<td>running</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bosh-dns-healthcheck</td>
<td>running</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bosh-dns-resolverconf</td>
<td>running</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>etcd</td>
<td>running</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>kube-apiserver</td>
<td>running</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>kube-controller-manager</td>
<td>running</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>kube-scheduler</td>
<td>running</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ncp</td>
<td>running</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pks-helpers-bosh-dns-resolverconf</td>
<td>running</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 tkgi -d service-instance_8de000ff-a87a-4930-81ba-106d42c2471e ssh master/b6d3c261-1682-4c79-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

- Stemcell or Service Update
- VM Process Failure on a Cluster Master
- VM Process Failure on a Cluster Worker
- VM Process Failure on the TKGI API VM
- VM Failure
- AZ Failure
- Region Failure

### Note:
As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

This topic describes events in the lifecycle of a Kubernetes cluster deployed by VMware Tanzu Kubernetes Grid Integrated Edition that can cause temporary service interruptions.

#### Stemcell or Service 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 TKGI API VM

A process, such as the TKGI API server, crashes on the pivotal-container-service VM.

Impact

- **TKGI control plane:** Depending on the process and what it was doing, the TKGI control plane may experience 60-120 seconds of downtime. Until the process resumes, the following can occur:
  - The TKGI API or UAA may be inaccessible
  - Use of the TKGI 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 TKGI control plane recovers automatically and the TKGI CLI resumes working.

VM Failure

An Tanzu Kubernetes Grid Integrated Edition 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 documentation for process failures in the cluster worker, cluster master, and TKGI API 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. VMware 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 Tanzu Kubernetes Grid Integrated Edition components offline.

Impact

The entire Tanzu Kubernetes Grid Integrated Edition deployment and all services are unavailable. The extent of the downtime is unknown.

Required Actions

The TKGI control plane can be restored using BOSH Backup and Restore (BBR). Each cluster may need to be restored manually from backups.

Please send any feedback you have to pks-feedback@pivotal.io.
Troubleshooting Tanzu Kubernetes Grid Integrated Edition Management Console

In this topic
Deployment of the Tanzu Kubernetes Grid Integrated Edition Management Console Fails
Deployment of Tanzu Kubernetes Grid Integrated Edition from the Management Console Fails
Tanzu Kubernetes Grid Integrated Edition Management Console Cannot Retrieve Cluster Data in a Multi-Tier0 Topology
Obtain the vRealize Log Insight Agent ID for Tanzu Kubernetes Grid Integrated Edition Management Console

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

The following sections describe how to troubleshoot failures to deploy the VMware Tanzu Kubernetes Grid Integrated Edition Management Console and of Tanzu Kubernetes Grid Integrated Edition instances from the management console.

For information about how to deploy the management console and install Tanzu Kubernetes Grid Integrated Edition, see Install on vSphere with the Management Console.

Deployment of the Tanzu Kubernetes Grid Integrated Edition Management Console Fails

Problem
Tanzu Kubernetes Grid Integrated Edition Management Console VM fails to deploy from the OVA template.

Solution
1. Use SSH to log in to the Tanzu Kubernetes Grid Integrated Edition Management Console 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 tkgi-mgmt-server > server.log
   ```

3. If the logs do not provide the solution, delete the management console VM from vCenter Server and attempt to deploy it again.

Deployment of Tanzu Kubernetes Grid Integrated Edition from the Management Console Fails

Problem
Tanzu Kubernetes Grid Integrated Edition fails to deploy from the management console.

Solution
1. Follow the procedure in Delete Your Tanzu Kubernetes Grid Integrated Edition Deployment to cleanly remove all Tanzu Kubernetes Grid Integrated Edition components from vSphere and to clean up related objects in the management console VM.


Tanzu Kubernetes Grid Integrated Edition Management Console Cannot Retrieve Cluster Data in a Multi-Tier0 Topology

Problem
In a deployment to a multiple-tier0 topology, Tanzu Kubernetes Grid Integrated Edition Management Console cannot display cluster information when you go to TKG Integrated Edition > 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 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, Tanzu Kubernetes Grid Integrated Edition 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 Tanzu Kubernetes Grid Integrated Edition 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 VM. For example, run the following command:

```bash
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 Tanzu Kubernetes Grid Integrated Edition Management Console

If you enabled integration with VMware vRealize Log Insight, Tanzu Kubernetes Grid Integrated Edition Management Console generates a unique vRealize Log Insight agent ID for the management console VM. You must provide this agent ID to vRealize Log Insight so that it can pull the appropriate logs from the management console.

You obtain the vRealize Log Insight agent ID as follows:

1. Use SSH to log in to the Tanzu Kubernetes Grid Integrated Edition Management Console VM as `root` user.

2. Run the following command to obtain the ID:

```bash
grep LOGINSIGHT_ID /etc/vmware/environment | cut -d= -f2
```

The resulting ID will be similar to `59debec7-daba-4770-9d21-2268d743843`.

3. Log in to the vRealize Log Insight Web user interface as administrator and add the agent ID to your list of agents.

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

In this topic

Connect to Operations Manager with SSH
Log In to the Operations Manager UI

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

When you use Tanzu Kubernetes Grid Integrated Edition Management Console to deploy Tanzu Kubernetes Grid Integrated Edition on vSphere, it deploys 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

Tanzu Kubernetes Grid Integrated Edition Management Console generates an SSH private key to control SSH access to the Operations Manager VM when you deploy Tanzu Kubernetes Grid Integrated Edition.

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 ~/tkgi_om.key.


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


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.


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.

Please send any feedback you have to pks-feedback@pivotal.io.
Using the BOSH CLI

In this topic

Using the BOSH CLI from the Tanzu Kubernetes Grid Integrated Edition Management Console VM
Using BOSH SSH

Note: As of v1.8, Enterprise PKS has been renamed to VMware Tanzu Kubernetes Grid Integrated Edition. Some screenshots in this documentation do not yet reflect the change.

Page last updated:

After you deploy Tanzu Kubernetes Grid Integrated Edition from Tanzu Kubernetes Grid Integrated Edition Management Console on vSphere, you can use the BOSH CLI from both the management console VM and the Ops Manager VM.

Using the BOSH CLI from the Tanzu Kubernetes Grid Integrated Edition Management Console VM

You can use the BOSH CLI from the Tanzu Kubernetes Grid Integrated Edition Management Console VM.

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 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 VM.
   
   ```
   export <bosh_cli_invocation_value>
   ```

Using BOSH SSH

The management console VM does not support the use of the `bosh ssh` command to connect to the BOSH VM from the management console VM. To connect to the BOSH VM by using `bosh ssh`, you must use the BOSH CLI from the Ops Manager VM.

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.
TKGI CLI

In this topic
Overview
TKGI CLI Commands
tkgi cancel-task
tkgi cluster
tkgi clusters
tkgi create-cluster
tkgi create-kubernetes-profile
tkgi create-network-profile
tkgi delete-cluster
tkgi delete-kubernetes-profile
tkgi delete-network-profile
tkgi get-credentials
tkgi get-kubeconfig
tkgi kubernetes-profile
tkgi kubernetes-profiles
tkgi login
tkgi logout
tkgi network-profile
tkgi network-profiles
tkgi plans
tkgi resize
tkgi task
tkgi tasks
tkgi update-cluster
tkgi upgrade-cluster
tkgi upgrade-clusters

Overview

This topic describes how to use the VMware Tanzu Kubernetes Grid Integrated Edition Command Line Interface (TKGI CLI) to interact with the TKGI API.

The TKGI CLI is a command-line tool to manage Tanzu Kubernetes Grid Integrated Edition provisioned Kubernetes clusters. Use the TKGI CLI to create, manage, and delete Kubernetes clusters.

To deploy workloads to a Kubernetes cluster, use kubectl, the Kubernetes CLI.

The TKGI CLI was previously named the PKS CLI, and both CLIs accept the same commands and arguments.

Tanzu Kubernetes Grid Integrated Edition v1.8 is compatible with both the TKGI and the PKS CLIs. Enterprise PKS v1.7 and earlier versions are compatible with only the PKS CLI.

If you are using the PKS CLI, substitute pks where the commands below use tkgi.

TKGI CLI Commands

Current Version: 1.8.0-build.72

tkgi cancel-task

Cancel a task

Synopsis

Cancel a task.

tkgi cancel-task <task-id> [flags]
tkgi cancel-task 0941fc83-b254-41a0-a505-14b04919e2cd

Options

- -h, --help    help for cancel-task

tkgi 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.

tkgi cluster [flags]

Examples

tkgi cluster my-cluster

Options

- --details Show details
- -h, --help help for cluster
- --json Return the TKGI-API output as json

tkgi clusters

Show all clusters created with TKGI

Synopsis

This command describes the clusters created via TKGI, and the last action taken on the cluster.

tkgi clusters [flags]

Examples

tkgi clusters

Options

- -h, --help help for clusters
- --json Return the TKGI-API output as json

tkgi 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).

tkgi create-cluster <cluster-name> [flags]

Examples

tkgi 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 TKGI-API output as json

- `--kubernetes-profile` string  
  Optional, kubernetes profile name

- `--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

- `--plan` string  
  Preconfigured plans. Run tkgi plans for more details

- `--tags` ClusterTag  
  Optional, (Azure, Vsphere Only) Add Tags for VMs as a list of k/v pairs (eg. "key1:val1,key2:val2,keyWithoutVal")

- `--wait`  
  Wait for the operation to finish

### tkgi create-kubernetes-profile

Create a kubernetes profile

**Synopsis**

Create kubernetes profile requires a path to the profile JSON file.

```
tkgi create-kubernetes-profile <kubernetes-profile-JSON-path> [flags]
```

**Examples**

```
tkgi create-kubernetes-profile my-profile.json
```

**Options**

- `-h`, `--help` help for create-kubernetes-profile

### tkgi create-network-profile

Create a network profile

**Synopsis**

Create network profile requires a path to the profile JSON file (Only applicable for NSX-T).

```
tkgi create-network-profile <network-profile-JSON-path> [flags]
```

**Examples**

```
tkgi create-network-profile my-network-profile.json
```

**Options**

- `-h`, `--help` help for create-network-profile

### tkgi delete-cluster

Deletes a kubernetes cluster, requires cluster name

**Synopsis**

Delete-cluster requires a cluster name.

```
tkgi delete-cluster <cluster-name> [flags]
```

**Examples**

```
tkgi 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

**tkgi delete-kubernetes-profile**

Delete a kubernetes profile

Synopsis

Deletes kubernetes profile. Requires a kubernetes profile name. Cannot be deleted if profile in use.

```
tkgi delete-kubernetes-profile <profile-name> [flags]
```

Examples

```
tkgi delete-kubernetes-profile my-k8s-profile
```

Options

-h, --help  help for delete-kubernetes-profile
--non-interactive  Don't ask for user input

**tkgi 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.

```
tkgi delete-network-profile PROFILE_NAME [flags]
```

Examples

```
tkgi delete-network-profile my-network-profile
```

Options

-h, --help  help for delete-network-profile
--non-interactive  Don't ask for user input

**tkgi 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.

Use the **--sso** flag if the TKGi tile is configured with SAML.

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

```
tkgi get-credentials <cluster-name> [flags]
```

Examples
tkgi get-credentials my-cluster

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

**tkgi 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 the TKGI tile is configured with SAML.

**tkgi get-kubeconfig <cluster-name> -u username -p password -a api [flags]**

Examples

- tkgi get-kubeconfig my-cluster -u username -p password -a 192.168.1.1
- tkgi get-kubeconfig my-cluster --sso -a 192.168.1.1

Options

- **-a, --api** string: API
- **--ca-cert** string: Path to CA Cert for TKGI API
- **-h, --help**: help for get-kubeconfig
- **-p, --password** string: Password
- **--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

**tkgi kubernetes-profile**

View a kubernetes profile

Synopsis

View saved kubernetes profile configuration.

**tkgi kubernetes-profile <profile-name> [flags]**

Examples

- tkgi kubernetes-profile custom-profile-1

Options

- **-h, --help**: help for kubernetes-profile
- **--json**: Return the TKGI-API output as json

**tkgi kubernetes-profiles**

List kubernetes profiles
Synopsis

Lists and describes kubernetes profiles.

```
tkgi kubernetes-profiles [flags]
```

Examples

```
tkgi kubernetes-profiles
```

Options

```
-h, --help  help for kubernetes-profiles
--json     Return the TKGI-API output as json
```

tkgi login

Log in to TKGI

Synopsis

The login command requires -a to target the IP of your TKGI API, -u for username and -p for password.

Use the `--sso` flag if the TKGI tile is configured with SAML.

```
tkgi login [flags]
```

Examples

```
tkgi login -a <API> -u <USERNAME> -p <PASSWORD> [--ca-cert <PATH TO CERT> | -k]
tkgi login -a <API> --client-name <CLIENT NAME> --client-secret <CLIENT SECRET> [--ca-cert <PATH TO CERT> | -k]
tkgi login -a <API> --sso [--ca-cert <PATH TO CERT> | -k]
tkgi login -a <API> --sso-auto [--ca-cert <PATH TO CERT> | -k]
tkgi login -a <API> --sso-passcode <sso-passcode> [--ca-cert <PATH TO CERT> | -k]
```

Options

```
-a, --api string  The TKGI API server URI
--ca-cert string  Path to CA Cert for TKGI API
--client-name string  Client name
--client-secret string  Client secret
-h, --help  help for login
-p, --password string  Password
--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
```

tkgi logout

Log out of TKGI

Synopsis

Log out of TKGI. Does not remove kubeconfig credentials or kubectl access.

```
tkgi logout [flags]
```

Examples

```
tkgi logout
```

Copyright © 2020 VMware, Inc. All Rights Reserved.
### Options

- 
  -h, --help  help for logout

### tkg network-profile

**View a network profile**

**Synopsis**

View saved network profile configuration.

```
tkg network-profile <profile-name> [flags]
```

**Examples**

```
tkg network-profile large-lb-profile
```

**Options**

- 
  -h, --help  help for network-profile
  
  --json  Return the TKGI-API output as json

### tkg network-profiles

**Show all network profiles created with TKGi**

**Synopsis**

Lists and describes network profiles.

```
tkg network-profiles [flags]
```

**Examples**

```
tkg network-profiles
```

**Options**

- 
  -h, --help  help for network-profiles
  
  --json  Return the TKGI-API output as json

### tkg plans

**View the preconfigured plans available**

**Synopsis**

This command describes the preconfigured plans available.

```
tkg plans [flags]
```

**Examples**

```
tkg plans
```

**Options**

- 
  -h, --help  help for plans
  
  --json  Return the TKGI-API output as json
tkgi 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.

```
tkgi resize <cluster-name> [flags]
```

Examples

```
tkgi resize my-cluster --num-nodes 5
```

Options

- `-h`, `--help`   help for resize
- `--json`   Return the TKGI-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

tkgi task

View a task

Synopsis

View a status and details of a task.

```
tkgi task <task-id> [flags]
```

Examples

```
tkgi task 09416d3-3b54-41a0-a505-14b04919e2cd
```

Options

- `-h`, `--help`   help for task
- `--json`   Return the TKGI-API output as json

tkgi tasks

List tasks

Synopsis

List recent tasks. By default shows most recent 10 tasks.

```
tkgi tasks [flags]
```

Examples

```
tkgi tasks -l 10
```

Options

- `-h`, `--help`   help for tasks
- `--json`   Return the TKGI-API output as json
- `-l`, `--limit int32`   Action flag, Show limit number of recent tasks (default 10)
tkgi 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.

```
tkgi update-cluster <cluster-name> [flags]
```

Examples

```
tkgi update-cluster my-cluster --num-nodes 5
```

Options

- `--network-profile` string Action flag, Network profile name
- `--kubernetes-profile` string Optional, Kubernetes 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.
- `--tags` [ClusterTag] Action flag, (Azure, Vsphere Only) Add/Update/Delete Tags for VMs as a list of key value pairs (e.g. –tags "key1:val1,key2:val2,keyWithoutVal"). To delete all tags, pass an empty string (e.g. –tags "")
- `--tags` [ClusterTag] Action flag, (Azure, Vsphere Only) Add/Update/Delete Tags for VMs as a list of key value pairs (e.g. –tags "key1:val1,key2:val2,keyWithoutVal"). To delete all tags, pass an empty string (e.g. –tags "")
- `--non-interactive` Don’t ask for user input
- `--json` Return the TKGI-API output as json
- `--wait` Wait for the operation to finish
- `-h, --help` help for update-cluster

tkgi upgrade-cluster

Upgrades the Kubernetes cluster

Synopsis

Upgrades the Kubernetes cluster listed. You need to provide a single cluster name.

```
tkgi upgrade-cluster <cluster-one> [flags]
```

Examples

```
tkgi upgrade-cluster one-cluster
```

Options

- `-h, --help` help for upgrade-cluster
- `--json` Return the TKGI-API output as json
- `--non-interactive` Don’t ask for user input
- `--wait` Wait for the operation to finish

tkgi upgrade-clusters

Upgrades the Kubernetes clusters

Synopsis

Upgrades the Kubernetes clusters listed. Specify clusters or canaries with a comma separated list of names.

```
tkgi upgrade-clusters [flags]
```

Examples

```
tkgi upgrade-clusters --clusters <cluster-1>,<cluster-2>,<cluster-3> --canaries <cluster-3>,<cluster-4> --max-in-flight 2
```
### Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--canaries</td>
<td>Optional, list of clusters to be treated as canaries. Will upgrade sequentially before other clusters. Should be a comma separated list of names.</td>
</tr>
<tr>
<td>-c, --clusters</td>
<td>List of clusters to be upgraded. Should be a comma separated list of names.</td>
</tr>
<tr>
<td>-h, --help</td>
<td>help for upgrade-clusters</td>
</tr>
<tr>
<td>--json</td>
<td>Return the TKGI-API output as json</td>
</tr>
<tr>
<td>--max-in-flight</td>
<td>Optional, number of clusters to be upgraded in parallel (default 1)</td>
</tr>
<tr>
<td>--non-interactive</td>
<td>Don't ask for user input</td>
</tr>
<tr>
<td>--wait</td>
<td>Wait for the operation to finish</td>
</tr>
</tbody>
</table>

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