# About VMware Tanzu Kubernetes Grid

VMware Tanzu Kubernetes Grid 2

You can find the most up-to-date technical documentation on the VMware website at:

https://docs.vmware.com/

**VMware, Inc.**
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

# Contents

# About Tanzu Kubernetes Grid

Tanzu Kubernetes Grid is a turnkey solution for deploying, running, and managing enterprise-grade Kubernetes clusters for hosting applications.

To deploy and manage Kubernetes clusters, Tanzu Kubernetes Grid (TKG) uses a *management cluster* that takes requests from a client CLI or UI and executes them using Cluster API, a standard open-source tool for low-level infrastructure and Kubernetes cluster operations.*

The management cluster has two deployment options that run on different infrastructures using different sets of components:

- **Supervisor** is a management cluster that is deeply integrated into vSphere with Tanzu and also performs infrastructure-level functions apart from supporting TKG.
- **Standalone management cluster** is a management cluster that runs as dedicated VMs, to support TKG on multiple cloud infrastructures.

In both cases, the management cluster publishes an API that wraps and adds higher-level functionality to Cluster API. On the client side, the Tanzu CLI wraps and adds higher-level functionality to `kubectl` and `clusterctl`, the Kubernetes and Cluster API CLIs.

**TKG 2.x unifies the management cluster API and underlying object definitions for these two management cluster deployment options** and is supported in product releases as follows:

- **vSphere 8** supports the TKG 2.0 API and objects for the Supervisor.
- **TKG v2.2.x and Tanzu CLI v0.29.x** support the TKG 2.2 API and objects for standalone management clusters on vSphere 6.7, 7, and 8 without Supervisor, AWS, and Azure, and on vSphere 8 with Supervisor.
- **TKG v2.1.x and Tanzu CLI v0.28.x** support the TKG 2.1 API and objects for standalone management clusters on vSphere 6.7, 7, and 8 without Supervisor, AWS, and Azure, and on vSphere 8 with Supervisor.
- **TKG v1.6.1 and Tanzu CLI v0.25.4** (and TKG 1.6.0 and Tanzu CLI v0.25.0) support the TKG 2.0 API and objects on vSphere 8 with Supervisor, and legacy cluster infrastructure with standalone TKG 1.6.x management clusters on vSphere 6.7, 7, and 8 without Supervisor, AWS, and Azure.

**Important**

The vSphere with Tanzu Supervisor in vSphere 8 runs TKG v2.0. TKG v2.0 was not released as a downloadable product, and allows you to create and manage class-based workload clusters with the Supervisor in vSphere 8 only. Standalone management clusters that run TKG 2.x are available from TKG 2.1 onwards. Due to the earlier TKG version that is embedded in Supervisor, some of the features that are available if you are using a standalone management cluster are not available if you are using vSphere with Tanzu. TKG v2.2 or later will be embedded in Supervisor in a future vSphere update release. Consequently, the version of TKG that is embedded in the latest vSphere version at a given time may not be as recent as the latest standalone version of TKG. However, the version of the Tanzu CLI that ships with all TKG v2.x releases is fully supported for use with Supervisor in vSphere 8.

*Another product in the Tanzu portfolio, Tanzu Kubernetes Grid Integrated Edition, does not use Cluster API and is not covered by this publication.

## Where to Find TKG Documentation

TKG documentation is published in multiple locations based on management cluster deployment option:

- **Supervisor on vSphere 8**
  - **To set up and use TKG with a Supervisor on vSphere 8**, see Using Tanzu Kubernetes Grid 2 with vSphere with Tanzu.
  - **To use the Tanzu CLI with a Supervisor on vSphere 8**, see Creating and Managing TKG 2.2 Workload Clusters with the Tanzu CLI.
- **Standalone management cluster** with TKG v2.2 and v2.1 on vSphere 6.7, 7, and 8 without Supervisor, and on AWS or Azure
  - **To set up and use TKG with a standalone management cluster with the Tanzu CLI**, see Deploying and Managing Tanzu Kubernetes Grid 2.2 Standalone Management Clusters.
- **Tanzu CLI**
  - For the **Tanzu CLI command reference**, see VMware Tanzu CLI Reference.

## TKG Feature States

The following state definitions describe Tanzu Kubernetes Grid features, ordered by typical lifecycle; documentation covers features in the Technical Preview, Stable, and Deprecated states:

- **Experimental**:
  - Not supported, and use voids support for its environment
  - Not for production environments, but can be tried in development environments
  - May have listings marked as “Experimental” in the CLI, download sites, etc.
  - Not documented publicly, but may have documentation distributed privately
- **Technical Preview**:
  - Not supported, but does not void warranty for its environment
  - For pre-production environments or production environments without support for the feature
  - Documented publicly, labeled as “Technical Preview”
- **Stable**:
  - Fully tested and supported for production environments, except as noted in documentation
  - Documented publicly without labels
- **Deprecated**:
  - Tested and supported, but with end of support expected
  - Customers are encouraged to stop using the feature and to deactivate it in their

environments

- Documented as “Deprecated”

## Why Kubernetes?

Kubernetes is an open-source system that replicates apps and keeps them running and accessible through network outages, changes in demand, and other potential stressors. To do this, Kubernetes hosts apps in portable containers that run on any compatible VM or physical machine, where “compatible” means running an OS and Kubernetes version that can support the containerized apps.

Combine Kubernetes with an infrastructure that can configure, create, and manage VMs, storage, and other resources, and you can securely and reliably host apps anywhere that the infrastructure extends, including private datacenters, public clouds, and edge hardware.

# Tanzu Kubernetes Grid Glossary

The topic below defines the key terms and concepts of a Tanzu Kubernetes Grid (TKG) deployment. Other topics in this section provide references for key TKG elements and describe experimental TKG features.

## Tanzu Kubernetes Grid

Tanzu Kubernetes Grid (TKG) is a high-level, multicloud infrastructure for Kubernetes. Tanzu Kubernetes Grid allows you to make Kubernetes available to developers as a utility, just like an electricity grid. Operators can use this grid to create and manage Kubernetes clusters for hosting containerized applications, and developers can use it to develop, deploy, and manage the applications. For more information, see What Is Tanzu Kubernetes Grid?.

## Management Cluster

A management cluster is a Kubernetes cluster that deploys and manages other Kubernetes clusters, called workload clusters, that host containerized apps.

TKG users log in to the management cluster with the Tanzu CLI and the Kubernetes CLI (`kubectl`) and issue commands like `tanzu cluster create` to create a workload cluster, or `tanzu package install` to install a packaged service to the cluster for hosted apps to consume.

The management cluster runs Cluster API, Carvel tools, and other software to process these commands.

The management cluster is purpose-built for managing workload clusters and packaged services, and for running container networking and other system-level agents. VMware recommends never deploying workloads to the management cluster itself because:

- Management clusters and workload clusters have separate concerns.
- Management clusters are not backed up like workload clusters. To ensure resilience, management cluster configuration is best managed as code.

## Management Cluster Deployment Options

The management cluster has two deployment options that run on different infrastructures using different sets of components:

- **Supervisor** is a management cluster that is deeply integrated into vSphere with Tanzu and also performs infrastructure-level functions apart from supporting TKG.
- **Standalone management cluster** is a management cluster that runs as dedicated VMs, to support TKG on multiple cloud infrastructures. With this option, “Deploying TKG” means deploying a management cluster to an infrastructure such as vSphere, AWS, or Azure.

## Workload Cluster

Workload clusters deployed by Tanzu Kubernetes Grid are CNCF-conformant Kubernetes clusters where containerized apps and packaged services are deployed to and run.

Workload clusters are deployed by the management cluster and run on the same private or public cloud infrastructure.

You can have many workload clusters, and to match the needs of the apps that they host, workload clusters can run different Kubernetes versions and have different *topologies* that include customizable node types and node counts, diverse operating systems, processors, storage, and other resource settings and configurations.

For pod-to-pod networking, workload clusters use Antrea by default, and can also use Calico.

## Tanzu Kubernetes Grid Instance

A Tanzu Kubernetes Grid instance is a full deployment of Tanzu Kubernetes Grid, including the management cluster, the deployed workload clusters, and the packaged services that they run. You can operate many instances of Tanzu Kubernetes Grid, for different environments, such as production, staging, and test; for different IaaS providers, such as vSphere, Azure, and AWS; and for different failure domains, for example `Datacenter-1`, AWS `us-east-2`, or AWS `us-west-2`.

## Tanzu CLI

The Tanzu CLI enables the `tanzu` commands that deploy and operate TKG. For example:

- `tanzu cluster` commands communicate with a TKG management cluster to create and manage workload clusters that host containerized workloads.
- `tanzu package` commands install and manage packaged services that hosted workloads use.
- `tanzu apps` commands manage hosted workloads via Tanzu Application Platform running on workload clusters.
- (Standalone management cluster only) `tanzu management-cluster` (or `tanzu mc`) commands deploy TKG by creating a standalone management cluster on a target infrastructure, and then manage the deployment once the standalone management cluster is running.

The Tanzu CLI uses plugins to modularize and extend its capabilities.

With a standalone management cluster, the version of Kubernetes that the management cluster runs is the same as the Kubernetes version used by the Tanzu CLI.

See Tanzu CLI Architecture and Configuration for more information.

## Bootstrap Machine

The bootstrap machine is a laptop, host, or server on which you download and run the Tanzu CLI.

When you use the Tanzu CLI to deploy a standalone management cluster, it creates the management cluster as a `kind` cluster on the bootstrap machine before deploying it to the target infrastructure.

How you connect the Tanzu CLI to an existing management cluster depends on its deployment option:

- **Supervisor**: Connect the Tanzu CLI to the Supervisor
- **Standalone management cluster**: Add Existing Management Clusters to Your Tanzu CLI.

A bootstrap machine can be a local laptop, a jumpbox, or any other physical or virtual machine.

## Tanzu Kubernetes Release

To run safely and efficiently, Kubernetes apps typically need to be hosted on nodes with specific patch versions of both Kubernetes and a base OS, along with compatible versions of other components. These component versions change over time.

To facilitate currency, safety, and compatibility, VMware publishes Tanzu Kubernetes releases (TKrs), which package patch versions of Kubernetes with base OS versions that it can run on, along with other versioned components that support that version of Kubernetes and the workloads it hosts.

The management cluster uses TKrs to create workload clusters that run the desired Kubernetes and OS versions.

Each TKr contains everything that a specific patch version of Kubernetes needs to run on various VM types on various cloud infrastructures.

See Tanzu Kubernetes Releases and Custom Node Images for more information.

## Packages and Cluster Services

To provide hosted workloads with services such as authentication, ingress control, container registry, observability, service discovery, and logging, you can install Tanzu-packaged services, or *packages* to TKG clusters.

As an alternative to running separate instances of the same service in multiple workload clusters, TKG with a standalone management cluster supports installing some services to a shared services cluster, a special workload cluster that can publish its services to other workload clusters.

Tanzu-packaged services are bundled with the Carvel imgpkg tool and tested for TKG by VMware.

Such packages can include:

- Services for use by hosted applications
- Platform services for cluster administrators
- Services for both of the above

## Cluster Plans (TKG v1.x)

In Tanzu Kubernetes Grid v1.x, and in legacy TKC-based clusters supported by TKG 2.x, a cluster plan is a standardized configuration for workload clusters. The plan configures settings for the number of control plane nodes, worker nodes, VM types, etc.

TKG v1.x provides two default plans: `dev` clusters have one control plane node and one worker node, and `prod` clusters have three control plane nodes and three workers.

TKG 2.x supports more fine-grained configuration of cluster topology as described in Configure a Class-Based Workload Cluster.

## `ytt` Overlays (TKG v1.x)

Configuration settings for TKG clusters and plans come from upstream, open-source sources such as the Cluster API project and its IaaS-specific provider projects. These sources publish Kubernetes object specifications in YAML, with settings pre-configured.

TKG supports Carvel ytt overlays for customizing objects for your own installation non-destructively, retaining the original YAML specifications. This is useful when YAML customization files reside on the bootstrap machine, and changing them directly would destroy the local copy of the original, upstream configurations.

In TKG v1.x, installing the Tanzu CLI installs the cluster and cluster plan configuration files in the `~/.config/tanzu/tkg` directory of the bootstrap machine, and supports `ytt` overlays for these configurations.

`ytt` overlays specify how to change target settings in target locations within a source YAML file, to support any possible customization.

See Advanced TKC Configuration with ytt for more information.

## Tanzu Kubernetes Grid Installer (Standalone Management Cluster)

To deploy TKG with a standalone management cluster, the Tanzu Kubernetes Grid installer is a graphical wizard that you start up by running the command `tanzu mc create --ui`. The installer wizard runs on the bootstrap machine, and provides a user interface to guide you through the process of deploying a standalone management cluster.

## Cluster Configuration File (Legacy Supervisor and Standalone Management Cluster)

The Tanzu CLI uses a cluster configuration file to create clusters under the following circumstances:

- When connected to a Supervisor and creating a legacy, TKC-based workload cluster
- When creating a standalone management cluster from a cluster configuration file
- When connected to a standalone management cluster and creating a workload cluster

When the TKG Installer creates a standalone management cluster, it captures user input from the UI and writes the entered values out to a cluster configuration file. The TKG Installer then uses this cluster configuration file to deploy the standalone management cluster.

Required and optional variables in the cluster configuration file depend on the management cluster deployment option:

- **Supervisor**: See Configure a Supervisor-Deployed Cluster with a Configuration File
- **Standalone management cluster**: See Configuration File Variable Reference

## Tanzu Kubernetes Grid and Cluster Upgrades

Upgrading TKG means different things based on its deployment option:

- **Supervisor**: When you update vCenter, the **Supervisor** tab lets you update the Supervisor to the latest Kubernetes version.
- **Standalone management cluster**: After upgrading the Tanzu CLI, running `tanzu management-cluster upgrade` upgrades the standalone management cluster to the CLI's version of Kubernetes.

Upgrading workload clusters in Tanzu Kubernetes Grid means migrating its nodes to run on a base VM image with a newer version of Kubernetes. By default, workload clusters upgrade to the management cluster's native of Kubernetes version, but you can specify other, non-default Kubernetes versions to upgrade workload clusters to.

To find out which Kubernetes versions are available in Tanzu Kubernetes Grid, see:

- **Supervisor**: See Compatibility for VMware Tanzu Kubernetes releases.
- **Standalone management cluster**: See List Available Versions.

## Integrating Tanzu Mission Control

To integrate your management cluster with Tanzu Mission Control, a Kubernetes management platform with a UI console, see:

- On the management cluster side:
  - **Supervisor**: Integrate the Tanzu Kubernetes Grid on the Supervisor with Tanzu Mission Control
  - **Standalone management cluster**: Examine and Register a Newly-Deployed Standalone Management Cluster
- In Tanzu Mission Control: Register a Management Cluster with Tanzu Mission Control

# Workload Clusters

This topic describes the different types of workload clusters created by Tanzu Kubernetes Grid (TKG) and how they are configured and created.

## Workload Cluster Types: Class-based, TKC, and Plan-based

Tanzu Kubernetes Grid hosts three different types of workload clusters:

- **Class-based clusters**
  - Are Kubernetes objects of type `Cluster`
  - Are a new type of cluster introduced in vSphere with Tanzu 8 and TKG 2.x
  - Have basic topology defined in a `spec.topology` block
    - For example, number and type of worker and control plane nodes
  - Inherit configuration from `spec.topology.class` value
    - Refers to a `ClusterClass` object
    - On Supervisor, default `class` is `tanzukubernetescluster`
    - On a standalone management cluster, default `class` is `tkg-INFRASTRUCTURE-default-VERSION`, for example, `tkg-vsphere-default-v1.0.0`.
  - Can be created by using Supervisor in vSphere with Tanzu 8 or by using a standalone TKG v2.x management cluster on vSphere 6.7, 7, and 8 without a Supervisor, on AWS, or on Azure
- **TKC-based clusters (legacy)**
  - Are Kubernetes objects of type `TanzuKubernetesCluster`
  - Can be created by using a Supervisor Cluster on vSphere 7, or by Supervisor on vSphere 8 for legacy purposes
- **Plan-based clusters (legacy)**
  - Are Kubernetes objects of type `Cluster`
  - Can be created by using a standalone TKG v2.x or v1.x management cluster on vSphere 6.7, 7, and 8 without a Supervisor, on AWS, or on Azure

Note that Class-based clusters with `class: tanzukubernetescluster`, all lowercase, are different from TKC-based clusters, which have object type `TanzuKubernetesCluster`.

Class-based clusters are designed to replace the other two cluster types, by presenting the same API to both types of management cluster: Supervisors and standalone management clusters.

## Cluster Types and Cluster API

To create and manage workload clusters, management clusters run Cluster API software:

- Cluster API - open-source Kubernetes software for creating and managing Kubernetes clusters.

- Cluster API Provider software that runs on specific cloud or physical infrastructures as an interface to support Cluster API.
  - Most Cluster API Provider software projects are open-source, but some are proprietary.

The following table maps management and workload cluster types to the Cluster API providers that they use:

| TKG with... | uses Cluster API Provider... | on... | to create and manage workload clusters of type... | in product versions... |
|---|---|---|---|---|
| Supervisor | CAPW (proprietary) | vSphere | **Class-based `Cluster` objects** | TKG 2.x and vSphere with Tanzu 8 |
| | | | `TanzuKubernetesCluster` objects | vSphere with Tanzu 7 & 8 |
| Standalone management cluster | CAPA (OSS) | AWS | **Class-based `Cluster` objects** | TKG v2.x |
| | | | Plan-based `AWSCluster` objects | TKG v2.x and v1.x |
| | CAPZ (OSS) | Azure | **Class-based `Cluster` objects** | TKG v2.x |
| | | | Plan-based `AzureCluster` objects | TKG v2.x and v1.x |
| | CAPV (OSS) | vSphere | **Class-based `Cluster` objects** | TKG v2.x |
| | | | Plan-based `VSphereCluster` objects | TKG v2.x and v1.x |

## Cluster Types and Tanzu CLI Compatibility

The different versions of the Tanzu CLI that ship with different versions of Tanzu Kubernetes Grid allow you to create different types of cluster depending on whether you are using Supervisor on vSphere 8, a Supervisor Cluster on vSphere 7, or a standalone management cluster on vSphere 6.7, 7, and 8 without a Supervisor, on AWS, or on Azure.

| CLI Version | TKG Version | Create class-based clusters with ... | | | Create plan-based clusters with ... | | | Create `TanzuKubernetesClusters` with ... | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Standalone Management Cluster | Supervisor on vSphere 8 | Supervisor Cluster on vSphere 7 | Standalone Management Cluster | Supervisor on vSphere 8 | Supervisor Cluster on vSphere 7 | Standalone Management Cluster | Supervisor on vSphere 8 | Supervisor Cluster on vSphere 7 |
| v0.29.0 | 2.2.0 | ✓ | ✓ | x | ✓ | ✓ | x | x | ✓ | x |
| v0.28.1 | 2.1.1 | ✓ | ✓ | x | ✓ | ✓ | x | x | ✓ | x |
| v0.25.4 | 1.6.1 | x | ✓ | x | ✓ | ✓ | x | x | ✓ | ✓ |
| v0.25.0 | 1.6.0 | x | ✓ | x | ✓ | ✓ | x | x | ✓ | ✓ |
| v0.11.x | 1.5.x | x | x | x | ✓ | x | x | x | x | ✓ |

## Workload Cluster Object Subcomponents

Class-based clusters have the following high-level hierarchy of object types. The objects underlying `KubeAdmControlPlane` and `MachineDeployment` have the same types, but are typically different objects:

- `Cluster` - number and type of control plane and worker nodes set by `topology` block in spec

- `KubeAdmControlPlane` - defines the **control plane** nodes
  - IaaS-specific machine object - for example, `vSphereMachine`, `AWSMachine`, `DockerMachine`
  - `Machine` - generic object for node VM
  - `KubeAdmConfig` - Kubernetes configuration, including Kubernetes version, image repository, pre- and post- deploy hooks, etc.
- `MachineDeployment` - defines the **worker** nodes
  - IaaS-specific machine object
  - `Machine`
  - `KubeAdmConfig`

For more information, see CustomResourceDefinitions relationships in *The Cluster API Book*.

## Ways of Creating Workload Clusters

Depending on your installed environment, you can create Tanzu Kubernetes Grid workload clusters in multiple ways: with the Tanzu CLI, Tanzu Mission Control, and `kubectl`.

The following chart outlines how users can create different types of workload clusters on different infrastructures:

| Using the... | to create a... | takes config values from... | and config templates from... | Instructions |
|---|---|---|---|---|
| Tanzu CLI: `tanzu cluster create` | Class-based workload cluster (vSphere) | `Cluster` and underlying object specs | User, for example classycluster.yaml | Create a Class-based Cluster |
| | `TanzuKubernetesCluster` workload cluster (vSphere) | Cluster configuration file, local environment, (advanced) `ytt` overlays | `infrastructure-tkg-service-vsphere`* | (Legacy) Create a Plan-Based or a TKC Cluster |
| | Plan-based workload cluster (vSphere, AWS, Azure) | | `infrastructure-vsphere`, `infrastructure-aws`, `infrastructure-azure`* | |
| Tanzu Mission Control (TMC) | `TanzuKubernetesCluster` or plan-based workload cluster | TMC UI | Registered management cluster | Provisioning Workload Clusters |
| `kubectl apply` | Class-based or `TanzuKubernetesCluster` workload clusters (vSphere) | `Cluster` and underlying object specs | User, for example classycluster.yaml, tkc.yaml | Creating Workload Clusters Declaratively |

*Local directories under `.config/tanzu/tkg/providers/`

## About Legacy TKC and Plan-Based Cluster Configuration

When the Tanzu CLI creates a TKC-based workload cluster, it combines configuration values from the following:

- Live input at invocation
  - CLI input
- Environment variables

- `~/.config/tanzu/tkg/cluster-config.yaml` or other file passed to the CLI `--file` option
- Cluster plan YAML configuration files in `~/.config/tanzu/tkg/providers/infrastructure-tkg-service-vsphere`, as described in Plan Configuration Files below.
- Other, non-plan YAML configuration files under `~/.config/tanzu/tkg/providers`

Live input applies configuration values that are unique to each invocation, environment variables persist them over a terminal session, and configuration files and overlays persist them indefinitely. You can customize clusters through any of these sources, with recommendations and caveats described below.

See Configuration Value Precedence for how the `tanzu` CLI derives specific cluster configuration values from these multiple sources where they may conflict.

## Plan Configuration Files

The `~/.config/tanzu/tkg/providers/infrastructure-tkg-service-vsphere` directory contains TKC workload cluster plan configuration files that are named `cluster-template-definition-PLAN.yaml`. The configuration values for each plan come from these files and from the files that they list under `spec.paths`:

- Config files that ship with the `tanzu` CLI
- Custom files that users create and add to the `spec.paths` list
- ytt Overlays that users create or edit to overwrite values in other configuration files

## Files to Edit, Files to Leave Alone

To customize cluster plans via YAML, you edit files under `~/.config/tanzu/tkg/providers/infrastructure-tkg-service-vsphere`, but you should avoid changing other files.

**Files to Edit**

Workload cluster plan configuration file paths follow the form `~/.config/tanzu/tkg/providers/infrastructure-infrastructure-tkg-service-vsphere/VERSION/cluster-template-definition-PLAN.yaml`, where:

- `VERSION` is the version of the Cluster API Provider module that the configuration uses.
- `PLAN` is `dev`, `prod`, or a custom plan.

Each plan configuration file has a `spec.paths` section that lists source files and `ytt` directories that configure the cluster plan. For example:

```
apiVersion: providers.tanzu.vmware.com/v1alpha1
kind: TemplateDefinition
spec:
  paths:
    - path: providers/infrastructure-tkg-service-vsphere/v1.1.0/ytt
    - path: providers/ytt
    - path: bom
      filemark: text-plain
    - path: providers/config_default.yaml
```

These files are processed in the order listed. If the same configuration field is set in multiple files, the last-processed setting is the one that the `tanzu` CLI uses.

To customize your cluster configuration, you can:

- Create new configuration files and add them to the `spec.paths` list.
  - This is the easier method.
- Modify existing `ytt` overlay files.
  - This is the more powerful method, for people who are comfortable with `ytt`.

**Files to Leave Alone**

VMware discourages changing the following files under `~/.config/tanzu/tkg/providers`, except as directed:

- `base-template.yaml` files, in `ytt` directories
  - These configuration files use values from the Cluster API provider repos under Kubernetes SIGs, and other upstream, open-source projects, and they are best kept intact.
  - Instead, create new configuration files or see Clusters and Cluster Plans in *Advanced TKC Configuration with* `ytt` to set values in the `overlay.yaml` file in the same `ytt` directory.
- `~/.config/tanzu/tkg/providers/config_default.yaml` - Append only
  - This file contains system-wide defaults for Tanzu Kubernetes Grid.
  - Do not modify existing values in this file, but you can append a `User Customizations` section at the end.
  - Instead of changing values in this file, customize cluster configurations in files that you pass to the `--file` option of `tanzu cluster create`.
- `~/.config/tanzu/tkg/providers/config.yaml`
  - The `tanzu` CLI uses this file as a reference for all providers present in the `/providers` directory, and their default versions.

## Configuration Value Precedence

When the Tanzu CLI creates a TKC-based workload cluster, it it combines configuration values from multiple sources. If those sources conflict, it resolves conflicts in the following order of descending precedence:

| Processing layers, ordered by descending precedence | Source | Examples |
|---|---|---|
| 1. Cluster configuration variables set in your local environment | Set in shell. | `export WORKER_VM_CLASS=best-effort-large` |
| 2. Cluster configuration variables set in the Tanzu CLI, with `tanzu config set env.` | Set in shell; saved in the global Tanzu CLI configuration file, `~/.config/tanzu/config.yaml`. | `tanzu config set env.WORKER_VM_CLASS best-effort-large` |
| 3. Cluster configuration variables set in the cluster configuration file | Set in the file passed to the `--file` option of `tanzu cluster create`. File defaults to `~/.config/tanzu/tkg/cluster-config.yaml`. | `WORKER_VM_CLASS: best-effort-large` |
| 4. Factory default configuration values | Set in `providers/config_default.yaml`, but some fields are listed without default values. Do not modify this file. | `WORKER_VM_CLASS:` |

# Tanzu Kubernetes Releases and Custom Node Images

This topic outlines how Tanzu Kubernetes Grid uses Tanzu Kubernetes releases (TKrs) and VM images to deploy clusters that run nodes with specific Kubernetes versions and operating systems.

## Tanzu Kubernetes Releases

To support running diverse applications efficiently and reliably, you can customize Tanzu Kubernetes Grid (TKG) clusters to run its worker nodes and other VMs on different Kubernetes versions, operating systems, and operating system (OS) versions. For supported Kubernetes versions, VMware publishes Tanzu Kubernetes releases (TKrs), which associate a specific patch version of Kubernetes with compatible versions of a base OS plus compatible versions of additional components required by cluster nodes.

Each TKr contains everything that a specific patch version of Kubernetes needs to run on various VM types on various cloud infrastructures, and the management cluster uses a TKr to create a workload cluster that runs the desired Kubernetes and OS version.

How TKrs are published, how TKG uses them, and which Kubernetes versions they support depend on the management cluster deployment option:

- **TKG with Supervisor**:
  - **TKr format and location**: OVA templates in Tanzu content library
    - Update list of available TKrs with `kubectl get tanzukubernetesreleases`
    - See Using Content Libraries
  - **How TKrs work**, in *vSphere with Tanzu Installation and Configuration*:
    - Conceptual overview: About Tanzu Kubernetes release Distributions
    - Practical how-to: Topics under Creating and Managing Content Libraries for Tanzu Kubernetes releases
  - **Supported Kubernetes versions**:
    - See Compatibility for VMware Tanzu Kubernetes releases in *VMware Tanzu Kubernetes releases Release Notes*
- **TKG with standalone management cluster**:
  - **TKr format and location**: `TanzuKubernetesRelease` objects
    - The default TKr is defined by the TKr BoM file installed with the Tanzu CLI to the `~/.config/tanzu/tkr/bom` directory
    - List available TKrs with `tanzu kubernetes-release get`
  - **How TKrs work**, in *Tanzu Kubernetes Grid v2.2 Documentation*:
    - Conceptual overview: About Kubernetes Versions
    - Practical how-to: Multiple Kubernetes Versions

- **Supported Kubernetes versions**:
  - See **TKG with standalone management cluster**: Supported Kubernetes Versions in Tanzu Kubernetes Grid v2.2 in *VMware Tanzu Kubernetes Grid 2.2 Release Notes*

## Custom Node Images (Standalone Management Cluster)

For TKG with a standalone management cluster, TKrs distributed by VMware support a selection of OSes by default, as listed in the Target Operating Systems table below.

You can also create clusters based on additional OSes listed as **Custom image** in the table. To do this, you run Kubernetes Image Builder and create a custom TKr as described in Build Machine Images.

## Target Operating Systems (Standalone Management Cluster)

The following table shows the operating systems for cluster nodes that Tanzu Kubernetes releases support:

| | vSphere | AWS | Azure |
|---|---|---|---|
| Distributed with TKG | Ubuntu 20.04<br>Photon OS 3 | Ubuntu 20.04<br>Amazon Linux 2 | Ubuntu 20.04<br>Ubuntu 18.04 |
| Custom Image (see Build Machine Images) | Ubuntu 20.04<br>Ubuntu 18.04<br>RHEL 8<br>Photon OS 3<br>Windows 2019 | Ubuntu 20.04<br>Ubuntu 18.04<br>Amazon Linux 2 | Ubuntu 20.04<br>Ubuntu 18.04 |

# Security and Compliance

For Tanzu Kubernetes Grid (TKG) with a standalone management cluster, this topic lists resources for securing infrastructure and complying with network security policies.

For TKG with a Supervisor, see vSphere with Tanzu Security in the vSphere 8 documentation.

## Ports and Protocols

Networking ports and protocols used by Tanzu Kubernetes Grid are listed in the VMware Ports and Protocols tool.

For each internal communication path, VMware Ports and Protocols lists:

- Product
- Version
- Source
- Destination
- Ports
- Protocols
- Purpose
- Service Description
- Classification (Outgoing, Incoming, or Bidirectional)

You can use this information to configure firewall rules.

## Tanzu Kubernetes Grid Firewall Rules

| Name | Source | Destination | Service(:Port) | Purpose |
|---|---|---|---|---|
| workload-to-mgmt | workload cluster network | management cluster network | TCP:6443* | Allow workload cluster to register with management cluster |
| mgmt-to-workload | management cluster network | workload cluster network | TCP:6443*, 5556 | Allow management network to configure workload cluster |
| allow-mgmt-subnet | management cluster network | management cluster network | all | Allow all internal cluster communication |

| | | | | |
|---|---|---|---|---|
| allow-wl-subnet | workload cluster network | workload cluster network | all | Allow all internal cluster communication |
| jumpbox-to-k8s | jumpbox IP | management cluster network, workload cluster network | TCP: 6443* | Allow Jumpbox to create management cluster and manage clusters. |
| dhcp | any | NSX: any / no NSX: DHCP IP | DHCP | Allows hosts to get DHCP addresses. |
| mc-pods-internal | management cluster pod network | .1 address within `SERVICE_CIDR` and `CLUSTER_CIDR` | TCP: * | Allows internal traffic from pods on management cluster to Kubernetes API server and pods on workload clusters |
| to-harbor | management cluster network, workload cluster network, jumpbox IP | Harbor IP | HTTPS | Allows components to retrieve container images |
| to-vcenter | management cluster network, workload cluster network, jumpbox IP | vCenter IP | HTTPS | Allows components to access vSphere to create VMs and Storage Volumes |
| dns-ntp-outbound | management cluster network, workload cluster network, jumpbox IP | DNS, NTP servers | DNS, NTP | Core services |
| ssh-to-jumpbox | any | jumpbox IP | SSH | Access from outside to the jumpbox |
| | | **For External Identity Provider** | | |
| allow-pinniped | workload cluster network | management cluster network | TCP: 31234 | Allow Pinniped concierge on workload cluster to access Pinniped supervisor on management cluster, which may be running behind a `NodePort` or `LoadBalancer` service |
| bootstrap-allow-pinniped | Bootstrap machine** | management cluster network | TCP: 31234 | Allow the bootstrap machine to access Pinniped supervisor on management cluster, which may be running behind a `NodePort` or `LoadBalancer` service |
| | | **For NSX ALB*** ** | | |
| avi-nodeport | Avi SE | any workload cluster | TCP: 30000-32767 | Allow NSX ALB SE (service engine) traffic to pods, for clusters in `NodePort` mode (default) for both L4 and L7 Virtual Services |
| avi-nodeportlocal | Avi SE | any workload cluster | TCP: 61000-62000 | Allow NSX ALB SE traffic to pods, for clusters in `NodePortLocal` mode for both L4 and L7 Virtual Services |
| ako-to-avi | management cluster network, workload cluster network | Avi Controller** | TCP: 443 | Allow Avi Kubernetes Operator (AKO) and AKO Operator (AKOO) access to Avi Controller |

| | | | | |
|---|---|---|---|---|
| avi-to-nsxt | Avi Controller | VMware NSX (formerly NSX-T) | TCP: 443 | Allow Avi controller to access VMware NSX, if Avi uses the NSX-T Cloud connector |
| | | **Default deny-all rule** | | |
| deny-all | any | any | all | deny by default |

*You can override the port 6443 setting with the `CLUSTER_API_SERVER_PORT` or for environments with NSX Advanced Load Balancer, `VSPHERE_CONTROL_PLANE_ENDPOINT_PORT` cluster configuration variable.

** Bootstrap machine is where the Tanzu CLI commands are run. It can be a jumpbox (a remote machine to which you establish a connection through SSH), your local machine, or a CI/CD host.

***For additional firewall rules required for NSX Advanced Load Balancer, formerly known as Avi Vantage, see Protocol Ports Used by Avi Vantage for Management Communication in the Avi Networks documentation.

## CIS Benchmarking for Clusters

To assess cluster security, you can run Center for Internet Security (CIS) Kubernetes benchmark tests on clusters deployed by Tanzu Kubernetes Grid.

For benchmarking clusters in Tanzu Mission Control, see the **Expected Test Failures for CIS Benchmark Inspection on Provisioned Tanzu Kubernetes Clusters** table under About the CIS Benchmark Inspection and Provisioned Tanzu Kubernetes Clusters, in the Tanzu Mission Control documentation.

# Identity and Access Management

**Before** you can create clusters using Tanzu Kubernetes Grid (TKG), you must connect to your management cluster. How you do this depends on the type of the management cluster that you plan to use:

- **Tanzu Kubernetes Grid with a vSphere with Tanzu Supervisor in vSphere 8.** You can connect to the Supervisor using either of the following methods:
  - Connect to the Supervisor as a vCenter single sign-on user with the Kubernetes or Tanzu CLI. For instructions, see:
    - Connect to Supervisor as a vCenter Single Sign-On User with kubectl
    - Connect to Supervisor Using the Tanzu CLI and vCenter SSO Authentication
  - Integrate the Supervisor with an OIDC-compliant identity provider and then use the Tanzu CLI to connect to the Supervisor. For instructions on how to:
    - Integrate the Supervisor with an external identity provider, see Configure an External OIDC Provider and Register an External IDP with Supervisor
    - Use the Tanzu CLI after registering the identity provider, see Connect to Supervisor Using the Tanzu CLI and an External IDP
- **TKG with a standalone management cluster.** To connect to the standalone management cluster, you integrate the cluster with an OIDC or LDAP identity provider or use the build-in authentication mechanism provided by TKG. For instructions, see Identity and Access Management.

**After** you create a cluster using a Supervisor or standalone management cluster, you can connect to the cluster using:

- In TKG 2.x with Supervisor:
  - vCenter single sign-on and the vSphere plugin for `kubectl` by itself or in conjunction with the Tanzu CLI. For instructions, see Connecting to TKG Clusters on Supervisor Using vCenter SSO Authentication.
  - An external OIDC provider in conjunction with the Tanzu CLI. For instructions, see Connect to and Examine Workload Clusters.
- In TKG with a standalone management cluster:
  - The Tanzu CLI, with or without an external OIDC or LDAP identity provider. For instructions, see Configure RBAC.

For conceptual information about identity and access management in TKG, see:

- **Tanzu Kubernetes Grid with a vSphere with Tanzu Supervisor in vSphere 8:** About Identity and Access Management for TKG 2 Clusters on Supervisor in *Using Tanzu Kubernetes Grid with vSphere with Tanzu* in the vSphere 8 documentation.
- **TKG with a standalone management cluster:** About Identity and Access Management.

# Packages

This section provides an overview of packages you can install and configure on workload clusters in Tanzu Kubernetes Grid.

Installing a *package* on a workload cluster created by Tanzu Kubernetes Grid adds a functionality to the cluster. This functionality typically provides services to the workloads that the cluster hosts. For example, the Antrea package provides the Antrea container network interface (CNI), the Contour package ingress control services, the Harbor package a private container registry, and so on. Some packages support the operation of the cluster itself.

Internally, a package consists of configuration metadata and image references that inform the package manager what software the package contains and how to install it into a Kubernetes cluster. Packages are grouped into *package repositories*. Some packages are enabled in clusters automatically while others are installed explicitly by using the Tanzu CLI. For more information about how packages and package repositories are implemented in Tanzu Kubernetes Grid, see Carvel API Resources below.

## Types of Packages

Tanzu Kubernetes Grid includes the following types of packages:

- **Auto-managed packages.** These packages are installed and upgraded automatically by Tanzu Kubernetes Grid. See the Auto-Managed Packages section below.
- **CLI-managed packages.** These packages are installed and upgraded explicitly by using the Tanzu CLI. Located in the `tanzu-standard` package repository or in other repositories that you add to your clusters. See the CLI-Managed Packages section below.

## Auto-Managed Packages

These packages are typically required for basic cluster functionality. Tanzu Kubernetes Grid installs and upgrades them automatically when you create and upgrade a Kubernetes cluster.

**Note**

The `tanzu package` CLI plugin is intended only for CLI-managed packages. Do not use the `tanzu package` CLI plugin to install and manage auto-managed packages. Their lifecycle is managed by Tanzu Kubernetes Grid.

See View and Update Auto-Managed Package Configuration for how to view and update auto-managed package configuration.

## Installation and Version Updates

Tanzu Kubernetes Grid manages the lifecycle of auto-managed packages. This includes automatic package installation and version updates.

Auto-managed packages are installed during cluster creation. To determine which auto-managed packages to install in a workload cluster, Tanzu Kubernetes Grid reads the Tanzu Kubernetes release that is used to create the cluster and cluster-specific configuration information. When you upgrade a workload cluster, as part of the upgrade process, Tanzu Kubernetes Grid updates the versions of the auto-managed packages that are installed in the cluster.

## List of Auto-Managed Packages

The table below lists the auto-managed packages that TKG installs and what types of clusters they are installed in.

Auto-managed packages are installed from the `tanzu-core` repository and run in the following namespace, depending on TKG deployment type:

- **Supervisor** namespace: `vmware-system-tkg`
- **Standalone management cluster** namespace: `tkg-system`

| Package | Installed in (with Supervisor) | Installed in (with standalone management cluster) | Description |
|---|---|---|---|
| `ako-operator` | *Not installed* | Management (vSphere with NSX ALB only) | Provides VMware NSX Advanced Load Balancer. This package is installed if NSX Advanced Load Balancer is enabled. |
| `antrea` | Workload | Management and workload | Enables pod networking and enforces network policies for Kubernetes clusters. Installed by default, unless Calico is selected as the CNI provider. |
| `calico` | Workload | Management and workload | Enables pod networking and enforces network policies for Kubernetes clusters. Installed if Calico is selected as the CNI provider. Not supported on Windows. |
| `capabilities` | Workload | Workload | Enables the Capabilities API |
| `guest-cluster-auth-service` | Workload | *Not installed* | Manages vSphere single sign-on (SSO), which enables vSphere SSO users to access the target workload cluster |
| `kapp-controller` | Workload | Management and workload | Manages packages. |
| `load-balancer-and-ingress-service` (AKO) | *Not installed* | Management and workload (vSphere with NSX ALB only) | Provides L4+L7 load balancing for applications running in clusters created by Tanzu Kubernetes Grid; used for north-south traffic. This package is installed if NSX Advanced Load Balancer is enabled. |
| `metrics-server` | Workload | Management and workload | Provides Metrics Server |
| `pinniped` | Workload | Management and workload | Provides user authentication. Installed only if an identity provider is configured. Can be installed in a standalone management cluster after it is already created; see Enable and Configure Identity Management in an Existing Deployment. |
| `secretgen-controller` | Workload | Workload | Enables carvel-secretgen-controller |

| Package | Installed in (with Supervisor) | Installed in (with standalone management cluster) | Description |
|---|---|---|---|
| `tanzu-addons-manager` | Management | Management | Manages the lifecycle of `tanzu-core` packages. |
| `tkg-pkg` | *Not installed* | Management | Installs `tanzu-addons-manager`, `tkr-source-controller`, `ClusterClass` definitions, and other components that TKG standalone management clusters require. |
| `vsphere-cpi` | Workload | Management and workload (vSphere only) | Provides the vSphere Cloud Provider Interface |
| `vsphere-pv-csi` | Workload | *Not installed* | Provides the vSphere Cloud Storage Interface |

## CLI-Managed Packages

CLI-managed packages extend Kubernetes clusters created by Tanzu Kubernetes Grid. After creating a cluster, you can install packages from the `tanzu-standard` package repository or from package repositories that you add to the cluster.

For a list of CLI-managed packages, the package repositories where they are published, and how to install them, see Installing and Managing Packages with the Tanzu CLI in *Creating and Managing TKG 2.2 Workload Clusters with the Tanzu CLI*.

## Carvel API Resources

To make packages available in Kubernetes clusters, Tanzu Kubernetes Grid creates the following API resources in the target cluster:

- `PackageRepository`, or `pkgr`, represents a single package repository. It points `kapp-controller`, a package manager, to the package repository that is defined in the resource. A package repository contains `Package` and `PackageMetadata` resources. After a `PackageRepository` is created in your target cluster, `kapp-controller` can install any of the packages that the package repository contains. This API resource is used only for CLI-managed packages.
- `Package`, or `pkg`, contains version-specific information about a given package and defines how to install the package. `kapp-controller` uses the `Package` resource when installing the package.
- `PackageMetadata`, or `pkgm`, contains version-agnostic information about a given package.
- `PackageInstall`, or `pkgi`, represents an installed package in your target cluster. It also references the `Package` resource that was used to install the package.

You can list and view these resources by using the `kubectl api-resources` and `kubectl get` commands. For more information about the Carvel API resources, see Packaging in the Carvel documentation.

# Reference Architectures

VMware provides validated reference architectures and designs for deploying Tanzu Kubernetes Grid (TKG) in various environments.

## Supervisor on vSphere 8 Network Topology

For vSphere with Tanzu Supervisor network topologies, see Supervisor Networking in the vSphere documentation.

## Tanzu Reference Architecture

The VMware Tanzu Reference Architecture Documentation includes the following designs and procedures for deploying TKG with a standalone management cluster to various infrastructures as part of Tanzu for Kubernetes Operations.

- **Air-Gap Solution Architecture**: Provides separate reference designs and deployment steps for deploying Tanzu Kubernetes Grid in an air-gapped environment on the following platforms:
  - Tanzu Kubernetes Grid on AWS
  - Tanzu Kubernetes Grid on vSphere with vSphere networking
  - Tanzu Kubernetes Grid on vSphere with NSX-T networking

  See VMware Tanzu Kubernetes Grid 2.1 Air-Gapped Reference Design and Deployment.
- **VMware Tanzu Edge Solution Reference Architecture**: Provides a reference architecture and deployment steps for deploying Tanzu on the VMware Edge Compute Stack. The Tanzu components in the design and deployment include Tanzu Kubernetes Grid, Tanzu Mission Control, and Tanzu Observability.

  See VMware Tanzu Edge Solution Reference Architecture 1.0.
- **Tanzu for Kubernetes Operation Reference Architecture**: Provides a reference architecture and designs for deploying the components in Tanzu for Kubernetes Operations, including Tanzu for Kubernetes Grid, on the following platforms:
  - Public clouds such as VMware Cloud on AWS, AWS, and Microsoft Azure.
  - vSphere using vSphere networking or using NSX-T.
  - vSphere with Tanzu using vSphere networking or using NSX-T

  VMware Tanzu for Kubernetes Operations Reference Architecture provides a high-level architecture for deploying the Tanzu components that make up VMware Tanzu for Kubernetes Operations. The reference designs are tailored for deploying Tanzu for Kubernetes Operations on your SaaS or infrastructure of choice.

  See VMware Tanzu for Kubernetes Operations Reference Architecture 2.1.

## Automating Reference Design Deployment

You can automate a reference design deployment using Service Installer for VMware Tanzu.

Service Installer for VMware Tanzu automates the deployment of the reference designs for Tanzu for Kubernetes Operations on the following platforms:

- Tanzu Kubernetes Grid on VMware Cloud
- Tanzu Kubernetes Grid on vSphere with NSX-T
- Tanzu Kubernetes Grid on vSphere running Virtual Distributed Switch (VDS)
- Tanzu Kubernetes Grid Service on vSphere running Virtual Distributed Switch (VDS)
- Tanzu Kubernetes Grid on AWS (air-gapped and non air-gapped)

For more information, see Service Installer for VMware Tanzu the VMware Tanzu Reference Architecture Documentation page.