

vSphere with Tanzu Configuration and Management

Update 3

VMware vSphere 7.0

vCenter Server 7.0

VMware ESXi 7.0

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

<https://docs.vmware.com/>

VMware by Broadcom
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Copyright © 2019-2023 Broadcom. All Rights Reserved. The term “Broadcom” refers to Broadcom Inc. and/or its subsidiaries. For more information, go to <https://www.broadcom.com>. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies. [Copyright and trademark information](#).

Contents

vSphere with Tanzu Configuration and Management	11
1 Updated Information	12
2 vSphere with Tanzu Concepts	19
What Is vSphere with Tanzu?	19
What Is a vSphere Pod?	22
What Is a Tanzu Kubernetes Cluster?	24
When to Use vSphere Pods and Tanzu Kubernetes Clusters	26
Using Virtual Machines in vSphere with Tanzu	26
vSphere with Tanzu User Roles and Workflows	28
How Does vSphere with Tanzu Change the vSphere Environment?	39
Licensing for vSphere with Tanzu	40
3 vSphere with Tanzu Architecture and Components	43
vSphere with Tanzu Architecture	43
Tanzu Kubernetes Grid Service Architecture	47
Tanzu Kubernetes Cluster Tenancy Model	49
vSphere with Tanzu Authentication	50
vSphere with Tanzu Networking	52
vSphere with Tanzu Security	52
vSphere with Tanzu Storage	53
4 Networking for vSphere with Tanzu	56
Supervisor Cluster Networking	56
Tanzu Kubernetes Grid Service Cluster Networking	61
Configuring NSX for vSphere with Tanzu	62
System Requirements for Setting Up vSphere with Tanzu with NSX	65
Topologies for a Supervisor Cluster with NSX	70
Best Practice Considerations for Configuring the Supervisor Cluster with NSX	71
Install and Configure NSX for vSphere with Tanzu	71
Configuring vSphere Networking and NSX Advanced Load Balancer for vSphere with Tanzu	90
NSX Advanced Load Balancer Components	92
System Requirements for Setting Up vSphere with Tanzu with vSphere Networking and NSX Advanced Load Balancer	93
Topology for Supervisor Cluster with vSphere Networking and NSX Advanced Load Balancer	99
Install and Configure the NSX Advanced Load Balancer	100

Configuring vSphere Networking and HAProxy Load Balancer for vSphere with Tanzu	115
System Requirements for Setting Up vSphere with Tanzu with vSphere Networking and HAProxy Load Balancer	116
Topologies for Deploying the HAProxy Load Balancer	119
Create a vSphere Distributed Switch for a Supervisor Cluster for Use with HAProxy Load Balancer	127
Install and Configure the HAProxy Load Balancer	128

5 Configuring and Managing a Supervisor Cluster 134

Prerequisites for Configuring vSphere with Tanzu on a vSphere Cluster	135
Enable Workload Management with vSphere Networking	138
Enable Workload Management with NSX Networking	145
Assign the Tanzu Edition License to a Supervisor Cluster	148
Replace the VIP Certificate to Securely Connect to the Supervisor Cluster API Endpoint	149
Integrate the Tanzu Kubernetes Grid Service on the Supervisor Cluster with Tanzu Mission Control	149
Set the Default CNI for Tanzu Kubernetes Clusters	151
Add Workload Networks to a Supervisor Cluster Configured with VDS Networking	153
Change the Control Plane Size of a Supervisor Cluster	154
Change the Management Network Settings on a Supervisor Cluster	155
Change the Workload Network Settings on a Supervisor Cluster Configured with VDS Networking	155
Change Workload Network Settings on a Supervisor Cluster Configured with NSX-T Data Center	156
Resolving Errors Health Statuses on Supervisor Cluster During Initial Configuration Or Upgrade	157
Configuring HTTP Proxy Settings in vSphere with Tanzu for use with Tanzu Mission Control	161
Streaming Logs of the Supervisor Cluster Control Plane to a Remote rsyslog	165
Registering Tanzu Mission Control Self-Managed with Supervisor Cluster	167

6 Creating and Managing Content Libraries in vSphere with Tanzu 169

Creating and Managing Content Libraries for Tanzu Kubernetes releases	169
About Tanzu Kubernetes release Distributions	169
Create, Secure, and Synchronize a Subscribed Content Library for Tanzu Kubernetes releases	170
Create, Secure, and Synchronize a Local Content Library for Tanzu Kubernetes releases	173
Migrate Tanzu Kubernetes Clusters to a New Content Library	177
Import the HAProxy OVA to a Local Content Library	178
Creating and Managing Content Libraries for Stand-Alone VMs in vSphere with Tanzu	179
Create a Content Library for Stand-Alone VMs in vSphere with Tanzu	179
Populate a Content Library with VM Images for Stand-Alone VMs in vSphere with Tanzu	182
Associate a VM Content Library with a Namespace in vSphere with Tanzu	183

Manage VM Content Libraries on a Namespace in vSphere with Tanzu 184

7 Configuring and Managing vSphere Namespaces 186

- Create and Configure a vSphere Namespace 186
- Set Resource Limits to a vSphere Namespace 190
- Configure Object Limitations on a vSphere Namespace 191
- Monitor and Manage Resources in a vSphere Namespace 192
- Configure a vSphere Namespace for Tanzu Kubernetes releases 193
- Add Security Policies to an NSX Supervisor Cluster Namespace 195
 - Create a Security Policy 195
- Provision a Self-Service Namespace Template 196
 - Create and Configure a Self-Service Namespace Template 197
 - Deactivate a Self-Service Namespace 198
 - Create a Self-Service Namespace 199
 - Create a Self-Service Namespace with Annotations and Labels 199
 - Update a Self-Service Namespace Using kubectl annotate and kubectl label 201
 - Update a Self-Service Namespace Using kubectl edit 202
 - Delete a Self-Service Namespace 204

8 Managing Supervisor Services with vSphere with Tanzu 205

- Add a Supervisor Service to vCenter Server 207
- Install a Supervisor Service on Supervisor Clusters 209
- Access the Management Interface of a Supervisor Service on the Supervisor Cluster 211
- Add a New Version to a Supervisor Service 211
- View Supervisor Services Installed on a Supervisor Cluster 212
- Deactivate a Supervisor Service or a Version 213
- Activate a Supervisor Service Version on vCenter Server 214
- Uninstall a Supervisor Service from a Supervisor Cluster 215
- Delete a Supervisor Service Version 215
- Delete a Supervisor Service 216

9 Connecting to vSphere with Tanzu Clusters 218

- Download and Install the Kubernetes CLI Tools for vSphere 218
- Configure Secure Login for vSphere with Tanzu Clusters 221
- Connect to the Supervisor Cluster as a vCenter Single Sign-On User 222
- Authenticating with Tanzu Kubernetes Clusters 223
- Connect to a Tanzu Kubernetes Cluster as a vCenter Single Sign-On User 224
- Connect to the Tanzu Kubernetes Cluster Control Plane as the Administrator 226
- SSH to Tanzu Kubernetes Cluster Nodes as the System User Using a Private Key 227
- SSH to Tanzu Kubernetes Cluster Nodes as the System User Using a Password 230
 - Create a Linux Jump Host VM 231

Grant Developer Access to Tanzu Kubernetes Clusters 233

10 Using Persistent Storage in vSphere with Tanzu 235

- How vSphere with Tanzu Integrates with vSphere Storage 239
- Functionality Supported by vSphere CNS-CSI and Paravirtual CSI in vSphere with Tanzu 242
- Storage Permissions in vSphere with Tanzu 243
- Create Storage Policies for vSphere with Tanzu 244
- Change Storage Settings on the Supervisor Cluster 246
- Change Storage Settings on a Namespace 247
- Display Storage Classes in a vSphere Namespace or Tanzu Kubernetes Cluster 247
- Provision a Dynamic Persistent Volume for a Stateful Application 249
- Provision a Static Persistent Volume in a Tanzu Kubernetes Cluster 251
- Creating ReadWriteMany Persistent Volumes in vSphere with Tanzu 253
- Volume Expansion in vSphere with Tanzu 255
 - Expand a Persistent Volume in Offline Mode 256
 - Expand a Persistent Volume in Online Mode 258
- Monitor Persistent Volumes in the vSphere Client 260
- Monitor Volume Health in a vSphere Namespace or Tanzu Kubernetes Cluster 262
- Using vSAN Data Persistence Platform with Modern Stateful Services 264
 - Tag Storage Devices for vSAN Direct 269
 - Set Up vSAN Direct for vSphere with Tanzu 275
 - Enable Stateful Services in vSphere with Tanzu 277
 - Monitor Stateful Services in vSphere with Tanzu 279
 - Check Storage Policies Available for Stateful Services 281
 - Create vSAN SNA Storage Policy 281
 - Create vSAN Direct Storage Policy 282

11 Deploying Workloads to vSphere Pods 284

- Get and Use the Supervisor Cluster Context 284
- Deploy an Application to a vSphere Pod on a vSphere Namespace 285
- Deploy an Application to a vSphere Pod Using the Embedded Harbor Registry 286
- Scale a vSphere Pod Application 287
- Deploy a Confidential vSphere Pod 288

12 Deploying and Managing Virtual Machines in vSphere with Tanzu 292

- Create a VM Class in vSphere with Tanzu 296
 - Attributes of VM Classes in vSphere with Tanzu 298
- Add PCI Devices to a VM Class in vSphere with Tanzu 299
- Edit or Delete a VM Class in vSphere with Tanzu 301
- Associate a VM Class with a Namespace in vSphere with Tanzu 302
- Manage VM Classes on a Namespace in vSphere with Tanzu 304

View VM Resources Available on a Namespace in vSphere with Tanzu	304
Deploy a Virtual Machine in vSphere with Tanzu	307
Install the NVIDIA Guest Driver in a VM in vSphere with Tanzu	310
Monitor Virtual Machines Available in vSphere with Tanzu	312

13 Provisioning and Operating TKGS Clusters 314

Workflow for Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API	314
Virtual Machine Classes for Tanzu Kubernetes Clusters	320
Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API	323
Requirements for Using the TKGS v1alpha2 API	323
TKGS v1alpha2 API for Provisioning Tanzu Kubernetes Clusters	325
Example YAML for Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API	330
Updating a Tanzu Kubernetes Release After the Cluster Spec Is Converted to the TKGS v1alpha2 API	332
Configuring a Tanzu Kubernetes Cluster with a Routable Pod Network Using the v1alpha2 API	338
Configuration Parameters for the TKGS v1alpha2 API	340
Examples for Configuring the TKGS Instance Using the v1alpha2 API	346
Scale a Tanzu Kubernetes Cluster Using the TKGS v1alpha2 API	351
Provisioning Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API	359
Workflow for Provisioning Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API	360
Configuration Parameters for Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API	364
Examples for Provisioning Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API	373
Configuration Parameters for the Tanzu Kubernetes Grid Service v1alpha1 API	382
Examples for Configuring the Tanzu Kubernetes Grid Service v1alpha1 API	386
Scale a Tanzu Kubernetes Cluster Using the Tanzu Kubernetes Grid Service v1alpha1 API	391
Delete a Tanzu Kubernetes Cluster	398
Specify a Default Text Editor for Kubectl	399
Operating Tanzu Kubernetes Clusters	401
Monitor Tanzu Kubernetes Cluster Status Using kubectl	401
Check Tanzu Kubernetes Cluster Readiness	402
View the Full Resource Hierarchy for a Tanzu Kubernetes Cluster	407
View Tanzu Kubernetes Cluster Lifecycle Status	408
Use Tanzu Kubernetes Cluster Operational Commands	409
Use Tanzu Kubernetes Cluster Networking Commands	411
Get Tanzu Kubernetes Cluster Secrets	414
Check Tanzu Kubernetes Machine Health	415
Check Tanzu Kubernetes Cluster Health	417

Monitor Tanzu Kubernetes Cluster Status Using the vSphere Client 419

14 Deploying Workloads and Packages on TKGS Clusters 420

- Deploy Workloads on Tanzu Kubernetes Clusters 420
 - Deploy a Test Workload to a Tanzu Kubernetes Cluster 420
 - Install and Run Octant 421
 - Tanzu Kubernetes Service Load Balancer Example 422
 - Tanzu Kubernetes Service Load Balancer with Static IP Address Example 424
 - Tanzu Kubernetes Ingress Example Using Nginx 426
 - Tanzu Kubernetes Storage Class Example 429
 - Tanzu Kubernetes Persistent Volume Claim Examples 430
 - Tanzu Kubernetes Guestbook Tutorial 432
 - Guestbook Example YAML Files 435
 - Using Pod Security Policies with Tanzu Kubernetes Clusters 439
 - Example Role Bindings for Pod Security Policy 441
 - Example Role for Pod Security Policy 443
- Install Packages on Tanzu Kubernetes Clusters 444
- Deploy AI/ML Workloads on Tanzu Kubernetes Clusters 445
 - About Deploying AI/ML Workloads on TKGS Clusters 445
 - vSphere Administrator Workflow for Deploying AI/ML Workloads on TKGS Clusters (vGPU) 446
 - Cluster Operator Workflow for Deploying AI/ML Workloads on TKGS Clusters 459
 - vSphere Administrator Addendum for Deploying AI/ML Workloads on TKGS Clusters (vGPU and Dynamic DirectPath IO) 467
 - Cluster Operator Addendum for Deploying AI/ML Workloads on TKGS Clusters (DLS) 468

15 Using a Container Registry for vSphere with Tanzu Workloads 471

- Enable the Embedded Harbor Registry on the Supervisor Cluster 472
- Log In to the Embedded Harbor Registry Console 473
- Download and Install the Embedded Harbor Registry Certificate 473
- Configure a Docker Client with the Embedded Harbor Registry Certificate 474
- Install the vSphere Docker Credential Helper and Connect to the Registry 476
- Push Images to the Embedded Harbor Registry 478
- Purge Images from the Embedded Harbor Registry 480
- Use the Embedded Harbor Registry with Tanzu Kubernetes Clusters 481
- Use an External Container Registry with Tanzu Kubernetes Clusters 484

16 Working with vSphere Lifecycle Manager 490

- Requirements 490
- Enable vSphere with Tanzu on a Cluster Managed by vSphere Lifecycle Manager 491
- Upgrade a Supervisor Cluster 491
- Add Hosts to a Supervisor Cluster 492

Remove Hosts from a Supervisor Cluster 493

Disable a Supervisor Cluster 493

17 Updating the vSphere with Tanzu Environment 495

About vSphere with Tanzu Updates 495

Network Topology Upgrade 499

Upgrade the NSX-T Network Topology 502

Upgrade vSphere Distributed Switch 503

Update the Supervisor Cluster by Performing a vSphere Namespaces Update 504

Supervisor Cluster Auto Upgrade 505

Update the vSphere Plugin for kubectl 506

Verify Tanzu Kubernetes Cluster Compatibility for Update 506

Update Tanzu Kubernetes Clusters 507

Update a Tanzu Kubernetes Cluster by Upgrading the Tanzu Kubernetes Release Version 509

Update a Tanzu Kubernetes Cluster by Changing the VirtualMachineClass 511

Update a Tanzu Kubernetes Cluster by Changing the Storage Class 514

Update Tanzu Kubernetes Clusters Using the Patch Method 516

18 Backing Up and Restoring vSphere with Tanzu 519

Considerations for Backing Up and Restoring vSphere with Tanzu 519

Install and Configure the Velero Plugin for vSphere on the Supervisor Cluster 521

Backup and Restore vSphere Pods Using the Velero Plugin for vSphere 533

Install and Configure the Velero Plugin for vSphere on a Tanzu Kubernetes Cluster 536

Backup and Restore Tanzu Kubernetes Cluster Workloads Using the Velero Plugin for vSphere 540

Install and Configure Standalone Velero and Restic on a Tanzu Kubernetes Cluster 541

Backup and Restore Tanzu Kubernetes Cluster Workloads Using Standalone Velero and Restic 546

Backup and Restore vCenter Server 554

Backup and Restore NSX-T Data Center 555

19 Troubleshooting vSphere with Tanzu 556

Storage Best Practices and Troubleshooting 556

Use Anti-Affinity Rules for Control Plane VMs on Non-vSAN Datastores 556

Storage Policy Removed from vSphere Continues to Appear as Kubernetes Storage Class 557

Use External Storage with vSAN Direct 558

Troubleshooting Networking 560

Register vCenter Server with NSX Manager 560

Unable to Change NSX Appliance Password 560

Troubleshooting Failed Workflows and Unstable NSX Edges 561

Collect Support Bundles for Troubleshooting NSX-T 561

Collect Log Files for NSX-T	562
Restart the WCP Service If the NSX-T Management Certificate, Thumbprint, or IP Address Changes	562
VDS Required for Host Transport Node Traffic	563
Troubleshooting the NSX Advanced Load Balancer	564
Collect Support Bundles for Troubleshooting	564
Troubleshooting Network Topology Upgrade	565
Upgrade Precheck Fails Due to Insufficient Edge Load Balancer Capacity	565
Supervisor Cluster Workload Namespaces Skipped During Upgrade	565
Load Balancer Service Skipped During Upgrade	566
Troubleshooting Tanzu Kubernetes Clusters	566
Collect a Support Bundle for Tanzu Kubernetes Clusters	566
Troubleshoot vCenter Single Sign-On Connection Errors	566
Troubleshoot Subscribed Content Library Errors	567
Troubleshoot Local Content Library Errors	567
Troubleshoot Cluster Provisioning Errors	568
Troubleshoot Workload Deployment Errors	568
Troubleshoot Virtual Machine Class Errors	569
Restart a Failed Tanzu Kubernetes Cluster Update Job	569
Troubleshooting Workload Management	570
Collect the Support Bundle for Workload Management	570
Tail the Workload Management Log File	571
Troubleshoot Workload Management Enablement Cluster Compatibility Errors	571
Shut Down and Start Up the vSphere with Tanzu Workload Domain	573

vSphere with Tanzu Configuration and Management

vSphere with Tanzu Configuration and Management provides information about configuring and managing vSphere with Tanzu by using the vSphere Client. It also provides information about using kubectl to connect to namespaces running on vSphere with Tanzu and run Kubernetes workloads on designated namespaces.

vSphere with Tanzu Configuration and Management provides an overview of the platform architecture as well as considerations and best practices for setting up storage, compute, and networking that meet the specific requirements of vSphere with Tanzu. It provides instructions for enabling vSphere with Tanzu on existing vSphere clusters, creating and managing namespaces, and monitoring Tanzu Kubernetes clusters that are created by using the VMware Tanzu™ Kubernetes Grid™ Service.

This information also provides guidelines about establishing a session with the vSphere with Tanzu Kubernetes control plane through kubectl, running a sample application, and creating Tanzu Kubernetes clusters by using the VMware Tanzu™ Kubernetes Grid™ Service.

At VMware, we value inclusion. To foster this principle within our customer, partner, and internal community, we create content using inclusive language.

Intended Audience

vSphere with Tanzu Configuration and Management is intended for vSphere administrators who want to enable vSphere with Tanzu in vSphere, configure and provide namespaces to DevOps teams, as well as manage and monitor Kubernetes workloads in vSphere. vSphere administrators who want to use vSphere with Tanzu should have basic knowledge of containers and Kubernetes.

This information is also intended for DevOps engineers who want to establish a session with the vSphere with Tanzu control plane, run Kubernetes workloads, and deploy Kubernetes clusters by using the VMware Tanzu™ Kubernetes Grid™ Service. In addition, developers who are deploying applications on the platform can refer to the examples for guidance.

Updated Information

1

vSphere with Tanzu Configuration and Management is updated regularly with new information and fixes as needed.

This table provides the update history of the *vSphere with Tanzu Configuration and Management*.

Revision	Description
28 MAR 2024	■ Updated the link for collecting a TKG cluster support bundle. See Collect a Support Bundle for Tanzu Kubernetes Clusters .
18 MAR 2024	■ Updated the Replace the VIP Certificate to Securely Connect to the Supervisor Cluster API Endpoint topic with a note about importing the entire certificate chain. ■ Update the Install and Configure the Velero Plugin for vSphere on the Supervisor Cluster with a warning about using Data Manager. ■ Updated Replace the VIP Certificate to Securely Connect to the Supervisor Cluster API Endpoint topic with a warning about importing the entire certificate chain.
12 JAN 2024	Updated the Supervisor auto upgrade topic with a note and examples. See Supervisor Cluster Auto Upgrade .
21 DEC 2023	Updated Use an External Container Registry with Tanzu Kubernetes Clusters with an explanation about triggering a rolling update to apply changes made to the service specification.
13 DEC 2023	Updated NSX requirements with a note about preparing all ESXi hosts participating in the vSphere Cluster as NSX transport nodes. See System Requirements for Setting Up vSphere with Tanzu with NSX .
6 DEC 2023	Updated TKG cluster rolling update documentation to state that multiple nodepools are updated simultaneously. See About vSphere with Tanzu Updates .
20 Nov 2023	Updated the documentation to indicate that Multi NSX is not supported on the Supervisor Cluster. See Add a Compute Manager .
14 NOV 2023	Updated the documentation to state that a valid config key must consist of alphanumeric characters, '-', '_' or '.' (e.g., 'key.name', or 'KEY_NAME', or 'key-name').
3 NOV 2023	Fixed a typo.
5 OCT 2023	Fixed a typo.
3 OCT 2023	Updated the documentation link for installing Tanzu packages on TKG clusters. See Install Packages on Tanzu Kubernetes Clusters .
19 SEPT 2023	Added note about the deprecation of Ingress API version <code>networking.k8s.io/v1beta1</code> as of v1.22. See Tanzu Kubernetes Ingress Example Using Nginx .

Revision	Description
10 AUG 2023	Minor edit.
7 AUG 2023	Removed documentation for ExternalTrafficPolicy and LoadBalancerSourceRanges for Service type: LoadBalancer with NSX. These features are not supported.
28 JUL 2023	Updated the package installation to link to the vSphere 8 documentation for Tanzu Package installation on TKG clusters. Install Packages on Tanzu Kubernetes Clusters .
21 JUL 2023	Fixed a typo.
14 JUL 2023	Added a requirement about overriding Tier-0 networking settings on a vSphere Namespace. See Create and Configure a vSphere Namespace .
07 JUL 2023	Added information on the HA configuration options for the Tier-0 gateway. See Create a Tier-0 Gateway .
30 JUN 2023	Added instructions for registering an instance of Tanzu Mission Control Self-Managed with Supervisor. See Registering Tanzu Mission Control Self-Managed with Supervisor Cluster .
25 JUN 2023	Updated the TKG cluster provisioning example. See Example YAML for Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API .
21 JUN 2023	■ Clarified statement that you cannot expand volumes created as a part of a StatefulSet when you use the StatefulSet definition. See Volume Expansion in vSphere with Tanzu. ■ Updated Enable Workload Management with vSphere Networking to remove Pod Ephemeral Disks and Container Image Cache as storage policy options. These options do not appear because VDS networking does not support vSphere Pods.
19 JUN 2023	Updated the steps to install the Velero plug-in in an air-gapped environment. See Install and Configure the Velero Plugin for vSphere on the Supervisor Cluster .
08 JUN 2023	Removed the single-cluster architecture for a Supervisor Clusters running with NSX networking. See System Requirements for Setting Up vSphere with Tanzu with NSX .
30 MAY 2023	Added GENEVE encapsulation support requirement for NSX deployments. See System Requirements for Setting Up vSphere with Tanzu with NSX .
24 MAY 2023	■ Added a note the that the DNS Server must be different for the Management and Workload Networks. See System Requirements for Setting Up vSphere with Tanzu with vSphere Networking and NSX Advanced Load Balancer. ■ Added a note that failure to reserve the floating IP addresses by its DHCP Unique Identifier might result in unrecoverable loss of connectivity to the Supervisor Control plane. See Enable Workload Management with vSphere Networking.
16 MAY 2023	Added a note that only Supervisor Clusters configured with the NSX networking stack support Supervisor Services. See Chapter 8 Managing Supervisor Services with vSphere with Tanzu .
11 MAY 2023	■ Update the TKG cluster provisioning examples with the latest TKRs. See Workflow for Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API. ■ Added notes stating that the v1alpha1 API is deprecated and that you should be using the v1alpha2 API for TKGS clusters on vSphere 7. See Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API.
04 MAY 2023	Minor revisions.
27 MAR 2023	Updated the Install and Configure the NSX Advanced Load Balancer section to include support for NSX Advanced Load Balancer version 22.1.3.

Revision	Description
07 FEB 2023	■ Replaced specific Velero versions with a link to the compatibility chart. See Install and Configure the Velero Plugin for vSphere on the Supervisor Cluster. ■ Fixed formatting errors in Provision a Static Persistent Volume in a Tanzu Kubernetes Cluster.
03 FEB 2023	■ Updated documentation for configuring certificates to the L4 load balancer that is used for the Kubernetes API server. See vSphere with Tanzu Authentication and Replace the VIP Certificate to Securely Connect to the Supervisor Cluster API Endpoint. ■ Update the documentation about updating and upgrading Supervisor Clusters with a the support for vSphere Lifecycle Manager. See Requirements and About vSphere with Tanzu Updates.
31 JAN 2023	■ Updated the documentation for configuring vSphere Networking and NSX Advanced Load Balancer section to reorder the topics. Moved the optional Deploy a Controller Cluster step after the Configure the Controller. ■ Fixed the IP address for the Controller VM in Configure the Controller.
20 JAN 2023	■ Updated the backup and restore documentation to add steps for installing the required Velero Plugin configmap on TKG clusters and Supervisor Clusters. See Install and Configure the Velero Plugin for vSphere on a Tanzu Kubernetes Cluster and Install and Configure the Velero Plugin for vSphere on the Supervisor Cluster. ■ Updated considerations to keep in mind while configuring vSphere with Tanzu with the HAProxy load balancer. See Topologies for Deploying the HAProxy Load Balancer.
13 JAN 2023	Reverted previous change to the <code>TkgServiceConfiguration</code> v1alpha2 API topic.
06 JAN 2023	■ Updated the topics for editing the <code>TkgServiceConfiguration</code> to indicate that starting with the v1alpha2 API, manual propagation is no longer required. See Configuration Parameters for the TKG v1alpha2 API and Configuration Parameters for the Tanzu Kubernetes Grid Service v1alpha1 API. ■ Miscellaneous minor edits.
04 JAN 2023	■ Updated the documentation for configuring vSphere Networking and NSX Advanced Load Balancer section to clarify that vSphere with Tanzu only supports a single VIP network, Default-Cloud and Default-Group Service Engine Group. ■ Topology for Supervisor Cluster with vSphere Networking and NSX Advanced Load Balancer ■ Configure the Controller ■ Configure a Virtual IP Network ■ Configure a Service Engine Group ■ Added a step to select Allow Basic Authentication. See Assign a Certificate to the Controller.
08 DEC 2022	Added information about network security policy. See Add Security Policies to an NSX Supervisor Cluster Namespace .
17 OCT 2022	Fixed spelling mistake.

Revision	Description
12 OCT 2022	■ Added link for TKG 1.6 Package installation. See Install Packages on Tanzu Kubernetes Clusters. ■ Updated the TKGS v1alpha2 API specification with the correct datatype (string) for nodeDrainTimeout. See TKGS v1alpha2 API for Provisioning Tanzu Kubernetes Clusters. ■ Updated the Guestbook application source YAML files. See Guestbook Example YAML Files.
15 SEPT 2022	Minor bug fixes.
29 JUL 2022	Clarified the statement about encryption of secrets in vSphere with Tanzu Security .
07 JUL 2022	Added a link to the VMware interoperability matrix for checking the compatibility between vCenter Server and NSX-T. See About vSphere with Tanzu Updates .
28 JUN 2022	Updated the vSphere with Tanzu shut down and start up topic with a link to the latest procedure. See Shut Down and Start Up the vSphere with Tanzu Workload Domain .
24 JUN 2022	Updated the vSphere Namespace creation procedure. See Create and Configure a vSphere Namespace .
03 JUN 2022	■ Updated the External DNS extension documentation. See Install Packages on Tanzu Kubernetes Clusters. ■ Fixed typos.
24 MAY 2022	Updated Creating ReadWriteMany Persistent Volumes in vSphere with Tanzu with a support statement for ReadWriteMany with Tanzu Kubernetes clusters.
20 MAY 2022	Updated the Guestbook example YAML by specifying redis:v6.0.5 for the redis-leader deployment. See Guestbook Example YAML Files .
13 MAY 2022	■ Updated the documentation for installing the Velero Plugin for vSphere. See Install and Configure the Velero Plugin for vSphere on the Supervisor Cluster. ■ Fixed minor typo.
06 MAY 2022	■ Added note that scaling TKGS cluster worker node volumes deletes existing volume data. See Scale Node Volumes. ■ Updated the SVC-TMC integration topic. See Integrate the Tanzu Kubernetes Grid Service on the Supervisor Cluster with Tanzu Mission Control. ■ Updated the TKGS cluster provisioning workflow. See Workflow for Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API. ■ Updated the TKGS networking concepts. See Tanzu Kubernetes Grid Service Cluster Networking. ■ Fixed typos and made minor edits.
21 APR 2022	Minor revisions.
18 APR 2022	■ Strengthened the statement recommending the use of the guaranteed VM class for production clusters to avoid resource contention and potential cluster downtime. See Virtual Machine Classes for Tanzu Kubernetes Clusters and View VM Resources Available on a Namespace in vSphere with Tanzu. ■ Added note about being aware of setting limits on a vSphere Namespace where Tanzu Kubernetes clusters are provisioned with best effort VM classes. See Configure Object Limitations on a vSphere Namespace.

Revision	Description
15 APR 2022	■ Added a note to the TKGS provisioning documentation that states that a cluster provisioned with 0 worker nodes/nodepools is not assigned any load balancer services. ■ Updated the local content library documentation. See Create, Secure, and Synchronize a Local Content Library for Tanzu Kubernetes releases. ■ Added more information about the Sysadmin login authentication key. See Deploy the Controller. ■ Added more information about Subject Alternate Name (SAN). See Assign a Certificate to the Controller. ■ Clarified that vsan-direct and vsan-sna storage classes can be used only by applications on the Supervisor Cluster, but not inside a Tanzu Kubernetes cluster. See Enable Stateful Services in vSphere with Tanzu. ■ Added information about how to configure remote streaming of the Supervisor Cluster control plane logs. See Streaming Logs of the Supervisor Cluster Control Plane to a Remote rsyslog.
28 MAR 2022	Added information about configuration an HTTP proxy in the vSphere with Tanzu environment. See Configuring HTTP Proxy Settings in vSphere with Tanzu for use with Tanzu Mission Control .
18 MAR 2022	Fixed minor typos.
04 MAR 2022	Added troubleshooting topic for local content libraries. See Troubleshoot Local Content Library Errors .
28 FEB 2022	■ Added link to install the TKG 1.5 Packages. See Install Packages on Tanzu Kubernetes Clusters. ■ Fixed the control plane VM names in the diagrams. See Topologies for a Supervisor Cluster with NSX. ■ Fixed minor typos.
18 FEB 2022	■ Updated Configure the Controller with a step for configuration options when DHCP is not enabled. ■ Updated Assign a Certificate to the Controller with a step for uploading a pre-created valid certificate. ■ Updated the system requirements to indicate that IPv6 is not supported. See Chapter 4 Networking for vSphere with Tanzu. ■ Updated the networking requirements for TKGS clusters if customizing the <code>pod CIDR blocks</code> setting in the cluster spec. See Requirements for Using the TKGS v1alpha2 API and TKGS v1alpha2 API for Provisioning Tanzu Kubernetes Clusters. ■ Updated the list of example networking commands for TKGS clusters. See Use Tanzu Kubernetes Cluster Networking Commands.
11 FEB 2022	■ Updated Enable Workload Management with vSphere Networking to include a link to the procedure for creating and configuring a vSphere Namespace as a next step after enablement. ■ Updated Create and Configure a vSphere Namespace to include a link for logging into the Supervisor Cluster as a next step.
08 FEB 2022	Updated the system requirements for installing the HA Proxy load balancer with a note that the workload network must be on a different subnet than the management network. See System Requirements for Setting Up vSphere with Tanzu with vSphere Networking and HAProxy Load Balancer .

Revision	Description
04 FEB 2022	■ Updated Creating ReadWriteMany Persistent Volumes in vSphere with Tanzu to remove outdated guidelines for networking. ■ Updated the Workflow for Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API with additional context and examples. ■ Updated the Supervisor Cluster Networking topic to fix a typo. ■ Updated the Update Tanzu Kubernetes Clusters Using the Patch Method topic with a warning not to use the <code>kubectl patch</code> method to update a cluster spec to conform with the v1alpha2 TKGS API. ■ Updated the Updating a Tanzu Kubernetes Release After the Cluster Spec Is Converted to the TKGS v1alpha2 API with an explicit mention that the <code>kubectl edit</code> method should be use to perform this operation.
28 JAN 2022	■ Updated the Kubernetes CLI Downloads topic with screenshots to better assist user navigation. See Download and Install the Kubernetes CLI Tools for vSphere. ■ Updated the scaling Tanzu Kubernetes clusters topic with information about adding or changing cluster node volumes. See Scale a Tanzu Kubernetes Cluster Using the TKGS v1alpha2 API. ■ Updated the Download TKG Extensions 1.3.1 topic to indicate that you must select the 1.3.1 version to view the extensions manifest for download. See Install Packages on Tanzu Kubernetes Clusters. ■ Updated the requirements for vSphere Pod CIDR range when configuring vSphere Networking and NSX Advanced Load Balancer for vSphere with Tanzu. See System Requirements for Setting Up vSphere with Tanzu with vSphere Networking and NSX Advanced Load Balancer.
17 DEC 2021	Updated the TKG 1.3.1 Extension prerequisite topic to describe how to edit the Kapp Controller configuration to add a proxy server. See Install Packages on Tanzu Kubernetes Clusters .
10 DEC 2021	■ Refreshed the Update Tanzu Kubernetes Clusters topics to support the Tanzu Kubernetes Grid Service Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API. See also Updating a Tanzu Kubernetes Release After the Cluster Spec Is Converted to the TKGS v1alpha2 API. ■ Updated the documentation for TKG Extensions to address customer feedback. See Install Packages on Tanzu Kubernetes Clusters. ■ Updated the TKGS Proxy configuration screenshot and requirements. See Configuration Parameters for the TKGS v1alpha2 API.
24 NOV 2021	Updated the documentation for configuring vSphere Networking and NSX Advanced Load Balancer for vSphere with Tanzu. See Configuring vSphere Networking and NSX Advanced Load Balancer for vSphere with Tanzu
05 NOV 2021	Added a link for installing TKG 1.4 Packages on Tanzu Kubernetes clusters provisioned by the Tanzu Kubernetes Grid Service. See Install Packages on Tanzu Kubernetes Clusters .

Revision	Description
29 OCT 2021	■ Updated the documentation for deploying vGPU workloads on TKGS clusters. See Deploy AI/ML Workloads on Tanzu Kubernetes Clusters. ■ Updated the Velero Plugin for vSphere installation documentation. See Install and Configure the Velero Plugin for vSphere on the Supervisor Cluster. ■ Updated RBAC examples. See Example Role Bindings for Pod Security Policy. ■ Updated the Supervisor Cluster networking. See Supervisor Cluster Networking. ■ Added a caution to the networking and enabling workload management prerequisite topics that DRS should not be disabled on the Supervisor Cluster and that disabling DRS leads to breaking of your clusters.
21 OCT 2021	■ Added documentation for deploying AI/ML workloads on vGPU-enabled TKGS clusters. See Deploy AI/ML Workloads on Tanzu Kubernetes Clusters. ■ Updated Add PCI Devices to a VM Class in vSphere with Tanzu with information about support of PCI devices in passthrough mode. ■ Moved the listing of Tanzu Kubernetes releases to dedicated Release Notes. Refer to these release notes for all Tanzu Kubernetes release information. ■ Fixed typos and minor doc bugs.
08 OCT 2021	■ Updated the Tanzu Kubernetes release version. See Verify Tanzu Kubernetes Cluster Compatibility for Update and Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API. ■ Added the procedure for installing the Velero Plugin for vSphere in an air-gapped environment. For more information, see Install and Configure the Velero Plugin for vSphere on the Supervisor Cluster. ■ Updated information about Supervisor Cluster backup and restore. For more information, see Considerations for Backing Up and Restoring vSphere with Tanzu. ■ Fixed typos.
05 OCT 2021	Initial release.

vSphere with Tanzu Concepts

2

By using vSphere with Tanzu you can turn a vSphere cluster to a platform for running Kubernetes workloads in dedicated resource pools. Once enabled on a vSphere cluster, vSphere with Tanzu creates a Kubernetes control plane directly in the hypervisor layer. You can then run Kubernetes containers by deploying vSphere Pods, or you can create upstream Kubernetes clusters through the VMware Tanzu™ Kubernetes Grid™ Service and run your applications inside these clusters.

Read the following topics next:

- [What Is vSphere with Tanzu?](#)
- [What Is a vSphere Pod?](#)
- [What Is a Tanzu Kubernetes Cluster?](#)
- [When to Use vSphere Pods and Tanzu Kubernetes Clusters](#)
- [Using Virtual Machines in vSphere with Tanzu](#)
- [vSphere with Tanzu User Roles and Workflows](#)
- [How Does vSphere with Tanzu Change the vSphere Environment?](#)
- [Licensing for vSphere with Tanzu](#)

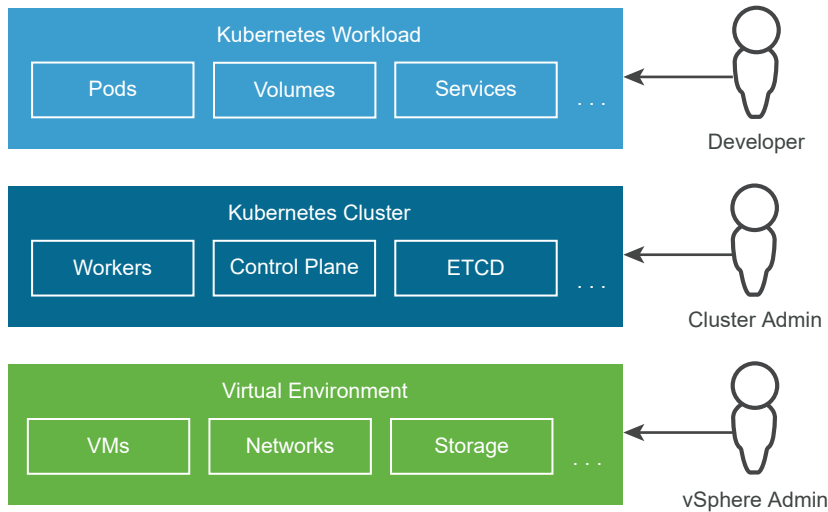
What Is vSphere with Tanzu?

You can use vSphere with Tanzu to transform vSphere to a platform for running Kubernetes workloads natively on the hypervisor layer. When enabled on a vSphere cluster, vSphere with Tanzu provides the capability to run Kubernetes workloads directly on ESXi hosts and to create upstream Kubernetes clusters within dedicated resource pools.

The Challenges of Today's Application Stack

Today's distributed systems are constructed of multiple microservices usually running a large number of Kubernetes pods and VMs. A typical stack that is not based on vSphere with Tanzu consists of an underlying virtual environment, with Kubernetes infrastructure that is deployed inside VMs, and respectively Kubernetes pods also running in these VMs. Three separate roles operate each part of the stack, which are application developers, Kubernetes cluster administrators, and vSphere administrators.

Figure 2-1. Today's Application Stack



The different roles do not have visibility or control over each other's environments:

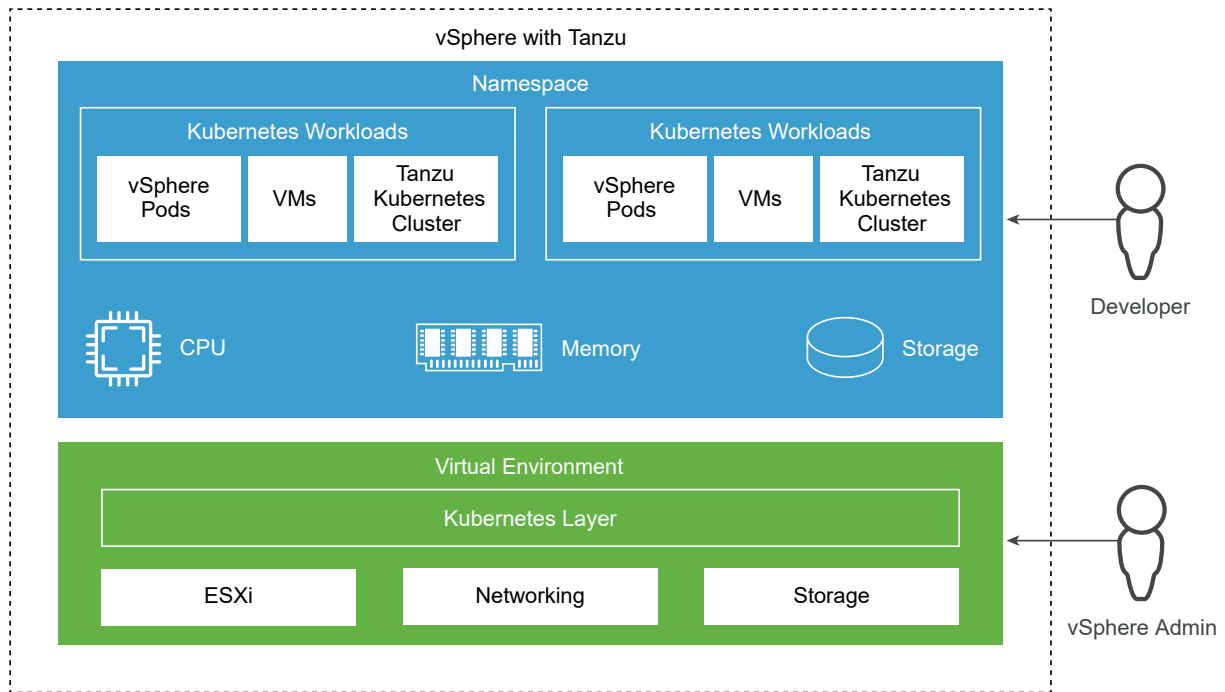
- As an application developer, you can run Kubernetes pods, and deploy and manage Kubernetes based applications. You do not have visibility over the entire stack that is running hundreds of applications.
- As a DevOps engineer, you only have control over the Kubernetes infrastructure, without the tools to manage or monitor the virtual environment and resolve any resource-related and other problems.
- As a vSphere administrator, you have full control over the underlying virtual environment, but you do not have visibility over the Kubernetes infrastructure, the placement of the different Kubernetes objects in the virtual environment, and how they consume resources.

Operations on the full stack can be challenging, because they require communication between all three roles. The lack of integration between the different layers of the stack can also introduce challenges. For example, the Kubernetes scheduler does not have visibility over the vCenter Server inventory and it cannot place pods intelligently.

How Does vSphere with Tanzu Help?

vSphere with Tanzu creates a Kubernetes control plane directly on the hypervisor layer. As a vSphere administrator, you enable existing vSphere clusters for **Workload Management**, thus creating a Kubernetes layer within the ESXi hosts that are part of the cluster. A cluster enabled with **Workload Management** is called a Supervisor Cluster.

Figure 2-2. vSphere with Tanzu



Having a Kubernetes control plane on the hypervisor layer enables the following capabilities in vSphere:

- As a vSphere administrator, you can create namespaces on the Supervisor Cluster, called vSphere Namespaces, and configure them with specified amount of memory, CPU, and storage. You provide vSphere Namespaces to DevOps engineers.
- As a DevOps engineer, you can run workloads consisting of Kubernetes containers on the same platform with shared resource pools within a vSphere Namespace. In vSphere with Tanzu, containers run inside a special type of VM called vSphere Pod. You can also deploy regular VMs.
- As a DevOps engineer, you can create and manage multiple Kubernetes clusters inside a namespace and manage their lifecycle by using the Tanzu Kubernetes Grid Service. Kubernetes clusters created by using the Tanzu Kubernetes Grid Service are called Tanzu Kubernetes clusters.
- As a vSphere administrator, you can manage and monitor vSphere Pods, VMs, and Tanzu Kubernetes clusters by using the vSphere Client.
- As a vSphere administrator, you have full visibility over vSphere Pods, VMs, and Tanzu Kubernetes clusters running within different namespaces, their placement in the environment, and how they consume resources.

Having Kubernetes running on the hypervisor layer also eases the collaboration between vSphere administrators and DevOps teams, because both roles are working with the same objects.

What Is a Workload?

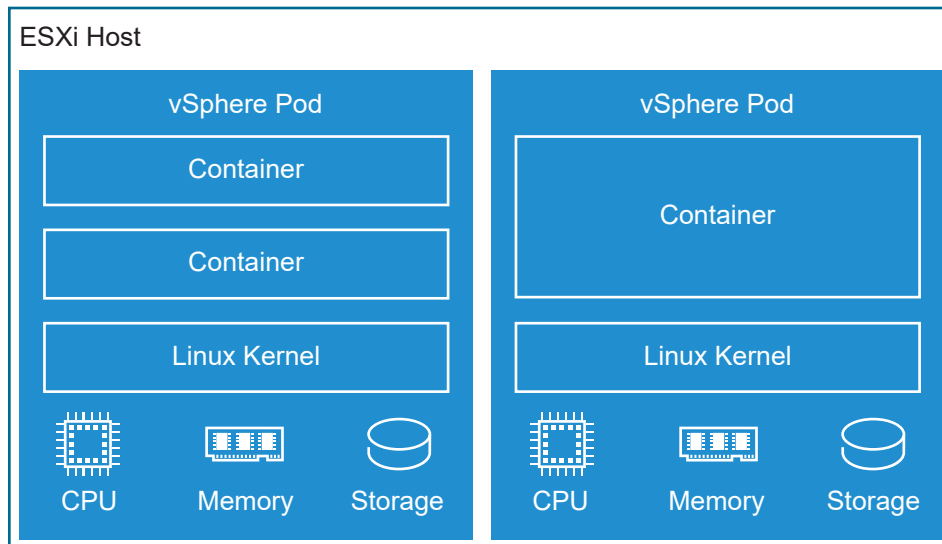
In vSphere with Tanzu, workloads are applications deployed in one of the following ways:

- Applications that consist of containers running inside vSphere Pods, regular VMs, or both.
- Tanzu Kubernetes clusters deployed by using the VMware Tanzu™ Kubernetes Grid™ Service.
- Applications that run inside the Tanzu Kubernetes clusters that are deployed by using the VMware Tanzu™ Kubernetes Grid™ Service.

What Is a vSphere Pod?

vSphere with Tanzu introduces a new construct that is called vSphere Pod, which is the equivalent of a Kubernetes pod. A vSphere Pod is a VM with a small footprint that runs one or more Linux containers. Each vSphere Pod is sized precisely for the workload that it accommodates and has explicit resource reservations for that workload. It allocates the exact amount of storage, memory, and CPU resources required for the workload to run. vSphere Pods are only supported with Supervisor Clusters that are configured with NSX-T Data Center as the networking stack.

Figure 2-3. vSphere Pods



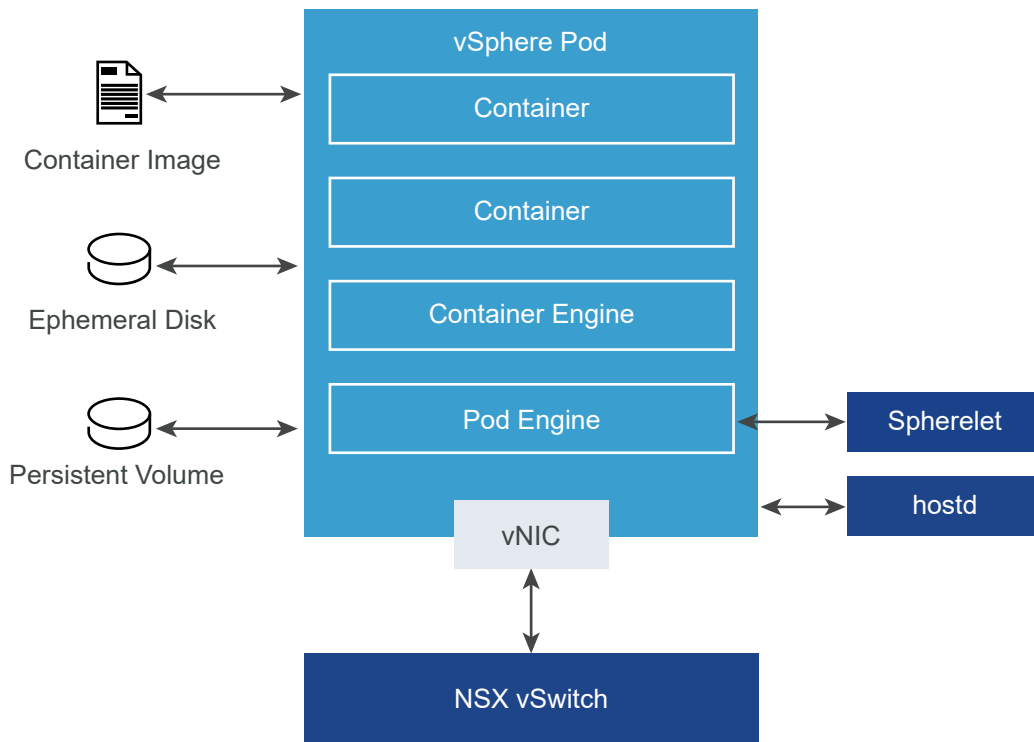
vSphere Pods are objects in vCenter Server, and therefore enable the following capabilities for workloads:

- Strong isolation. A vSphere Pod is isolated in the same manner as a virtual machine. Each vSphere Pod has its own unique Linux kernel that is based on the kernel used in Photon OS. Rather than many containers sharing a kernel, as in a bare metal configuration, in a vSphere Pod, each container has a unique Linux kernel

- **Resource Management.** vSphere DRS handles the placement of vSphere Pods on the Supervisor Cluster.
- **High performance.** vSphere Pods get the same level of resource isolation as VMs, eliminating noisy neighbor problems while maintaining the fast start-up time and low overhead of containers.
- **Diagnostics.** As a vSphere administrator you can use all the monitoring and introspection tools that are available with vSphere on workloads.

vSphere Pods are Open Container Initiative (OCI) compatible and can run containers from any operating system as long as these containers are also OCI compatible.

Figure 2-4. vSphere Pod Networking and Storage



vSphere Pods use three types of storage depending on the objects that are stored, that are ephemeral VMDKs, persistent volume VMDKs, and containers image VMDKs. As a vSphere administrator, you configure storage policies for placement of container image cache, ephemeral VMDKs, and control plane VMs on the Supervisor Cluster level. On a vSphere Namespace level, you configure storage policies for placement of persistent volumes and for placement of the VMs of Tanzu Kubernetes clusters. See [Chapter 10 Using Persistent Storage in vSphere with Tanzu](#) for details about the storage requirements and concepts with vSphere with Tanzu.

For networking, vSphere Pods and the VMs of the Tanzu Kubernetes clusters created through the Tanzu Kubernetes Grid Service use the topology provided by NSX-T Data Center. For details, see [Supervisor Cluster Networking](#).

vSphere Pods are only supported on Supervisor Clusters that use NSX-T Data Center as their networking stack. They are not supported on clusters that are configured with the vSphere networking stack.

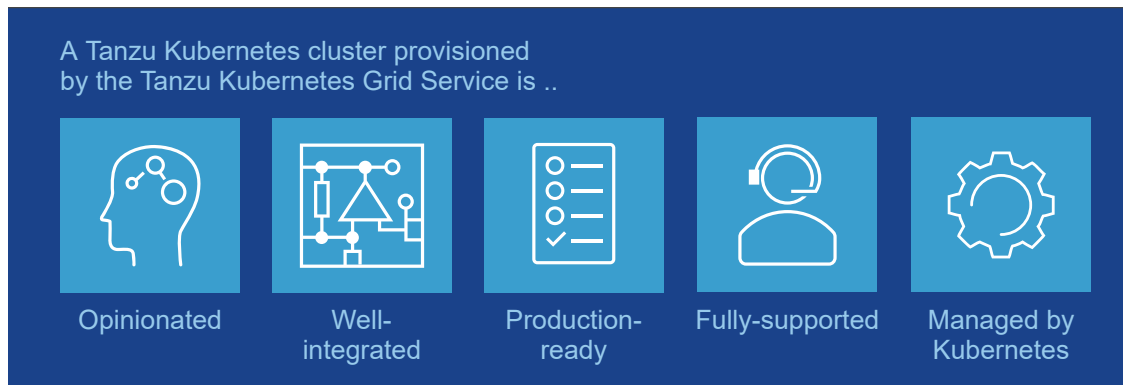
What Is a Tanzu Kubernetes Cluster?

A Tanzu Kubernetes cluster is a full distribution of the open-source Kubernetes container orchestration platform that is built, signed, and supported by VMware. You can provision and operate Tanzu Kubernetes clusters on the Supervisor Cluster by using the Tanzu Kubernetes Grid Service. A Supervisor Cluster is a vSphere cluster that is enabled with vSphere with Tanzu.

Key Characteristics of Tanzu Kubernetes Clusters Created by the Tanzu Kubernetes Grid Service

A Tanzu Kubernetes cluster that is provisioned by the Tanzu Kubernetes Grid Service has the following characteristics:

- Opinionated Installation of Kubernetes
- Integrated with the vSphere Infrastructure
- Production Ready
- Fully Supported by VMware
- Managed by Kubernetes



Note VMware markets a suite of Kubernetes-focused products under the Tanzu brand. The Tanzu Kubernetes clusters that you create by using the Tanzu Kubernetes Grid Service are components of the Tanzu edition license. For more information about the other Kubernetes-focused products that VMware markets under Tanzu, see the [VMware Tanzu](#) documentation. For information about licensing in vSphere with Tanzu, see [Licensing for vSphere with Tanzu](#).

Opinionated Installation of Kubernetes

A Tanzu Kubernetes cluster is an opinionated installation of Kubernetes.

The Tanzu Kubernetes Grid Service provides well-thought-out defaults that are optimized for vSphere to provision Tanzu Kubernetes clusters. By using the Tanzu Kubernetes Grid Service, you can reduce the amount of time and effort that you typically spend for deploying and running an enterprise-grade Kubernetes cluster.

For more information, see [Tanzu Kubernetes Grid Service Architecture](#).

Integrated with the vSphere Infrastructure

A Tanzu Kubernetes cluster is integrated with the underlying vSphere infrastructure, which is optimized for running Kubernetes.

A Tanzu Kubernetes cluster is integrated with the vSphere SDDC stack, including storage, networking, and authentication. In addition, a Tanzu Kubernetes cluster is built on a Supervisor Cluster that maps to a vCenter Server cluster. Because of the tight integration, running a Tanzu Kubernetes cluster is a unified product experience.

For more information, see [vSphere with Tanzu Architecture](#).

Production Ready

A Tanzu Kubernetes cluster is tuned for running production workloads.

The Tanzu Kubernetes Grid Service provisions production-ready Tanzu Kubernetes clusters. You can run production workloads without the need to perform any additional configuration. In addition, you can ensure availability and allow for rolling Kubernetes software upgrades and run different versions of Kubernetes in separate clusters.

For more information, see [Chapter 13 Provisioning and Operating TKGS Clusters](#).

Fully Supported by VMware

A Tanzu Kubernetes cluster is supported by VMware.

Tanzu Kubernetes clusters use the open source, Linux-based Photon OS from VMware, are deployed on vSphere infrastructure, and run on ESXi hosts. If you experience problems with any layer of the stack, from the hypervisor to the Kubernetes cluster, VMware is the only vendor you need to contact.

For more information, contact [VMware Support](#).

Managed by Kubernetes

A Tanzu Kubernetes cluster is managed by Kubernetes.

Tanzu Kubernetes clusters are built on top of the Supervisor Cluster, which is itself a Kubernetes cluster. A Tanzu Kubernetes cluster is defined in the vSphere Namespace using a custom resource. You provision Tanzu Kubernetes clusters in a self-service way using familiar kubectl commands. There is consistency across the toolchain, whether you are provisioning a cluster or deploying workloads, you use the same commands, familiar YAML, and common workflows.

For more information, see [Tanzu Kubernetes Cluster Tenancy Model](#).

When to Use vSphere Pods and Tanzu Kubernetes Clusters

Using vSphere Pods or Tanzu Kubernetes clusters provisioned by the Tanzu Kubernetes Grid Service depends on your goals related to deploying and managing Kubernetes workloads on the Supervisor Cluster.

Use vSphere Pods if you are a vSphere administrator or DevOps engineer who wants to:

- Run containers without needing to customize a Kubernetes cluster.
- Create containerized applications with strong resource and security isolation.
- Deploy vSphere Pods directly on ESXi hosts.

Use Tanzu Kubernetes clusters provisioned by the Tanzu Kubernetes Grid Service if you are a DevOps engineer or developer who wants to:

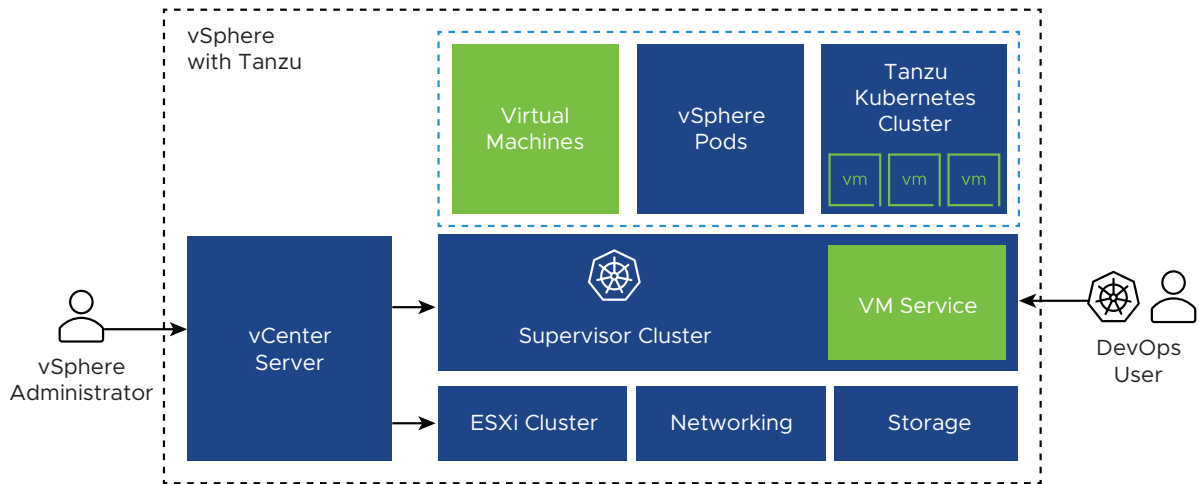
- Run containerized applications on open-source, community-aligned Kubernetes software.
- Have control over the Kubernetes cluster, including root level access to the control plane and worker nodes.
- Stay current with Kubernetes versions without requiring infrastructure upgrades.
- Use a CI/CD pipeline to provision short-lived Kubernetes clusters.
- Customize the Kubernetes cluster, for example, install custom resource definitions, Operators, and helm charts.
- Create Kubernetes namespaces using the `kubectl` CLI.
- Manage cluster-level access control and configure `PodSecurityPolicies`.
- Create services of type `NodePort`.
- Use `HostPath` volumes.
- Run privileged pods.

Using Virtual Machines in vSphere with Tanzu

vSphere with Tanzu offers a VM Service functionality that enables DevOps engineers to deploy and run VMs, in addition to containers, in a common, shared Kubernetes environment. Both, containers and VMs, share the same vSphere Namespace resources and can be managed through a single vSphere with Tanzu interface.

The VM Service addresses the needs of DevOps teams that use Kubernetes, but have existing VM-based workloads that cannot be easily containerized. It also helps users reduce the overhead of managing a non-Kubernetes platform alongside a container platform. When running containers and VMs on a Kubernetes platform, DevOps teams can consolidate their workload footprint to just one platform.

Note In addition to stand-alone VMs, the VM Service manages the VMs that make up Tanzu Kubernetes clusters. For information about the clusters, see [Tanzu Kubernetes Grid Service Architecture](#) and [Chapter 13 Provisioning and Operating TKGS Clusters](#).



Each VM deployed through the VM Service functions as a complete machine running all the components, including its own operating system, on top of the vSphere with Tanzu infrastructure. The VM has access to networking and storage that a Supervisor Cluster provides, and is managed using the standard Kubernetes `kubectl` command. The VM runs as a fully isolated system that is immune to interference from other VMs or workloads in the Kubernetes environment.

When to Use Virtual Machines on a Kubernetes Platform?

Generally, a decision to run workloads in a container or in a VM depends on your business needs and goals. Among the reasons to use VMs appear the following:

- Your applications cannot be containerized.
- You have specific hardware requirements for your project.
- Applications are designed for a custom kernel or custom operating system.
- Applications are better suited to running in a VM.
- You want to have a consistent Kubernetes experience and avoid overhead. Rather than running separate sets of infrastructure for your non-Kubernetes and container platforms, you can consolidate these stacks and manage them with a familiar `kubectl` command.

For information about deploying and managing virtual machines, see [Chapter 12 Deploying and Managing Virtual Machines in vSphere with Tanzu](#).

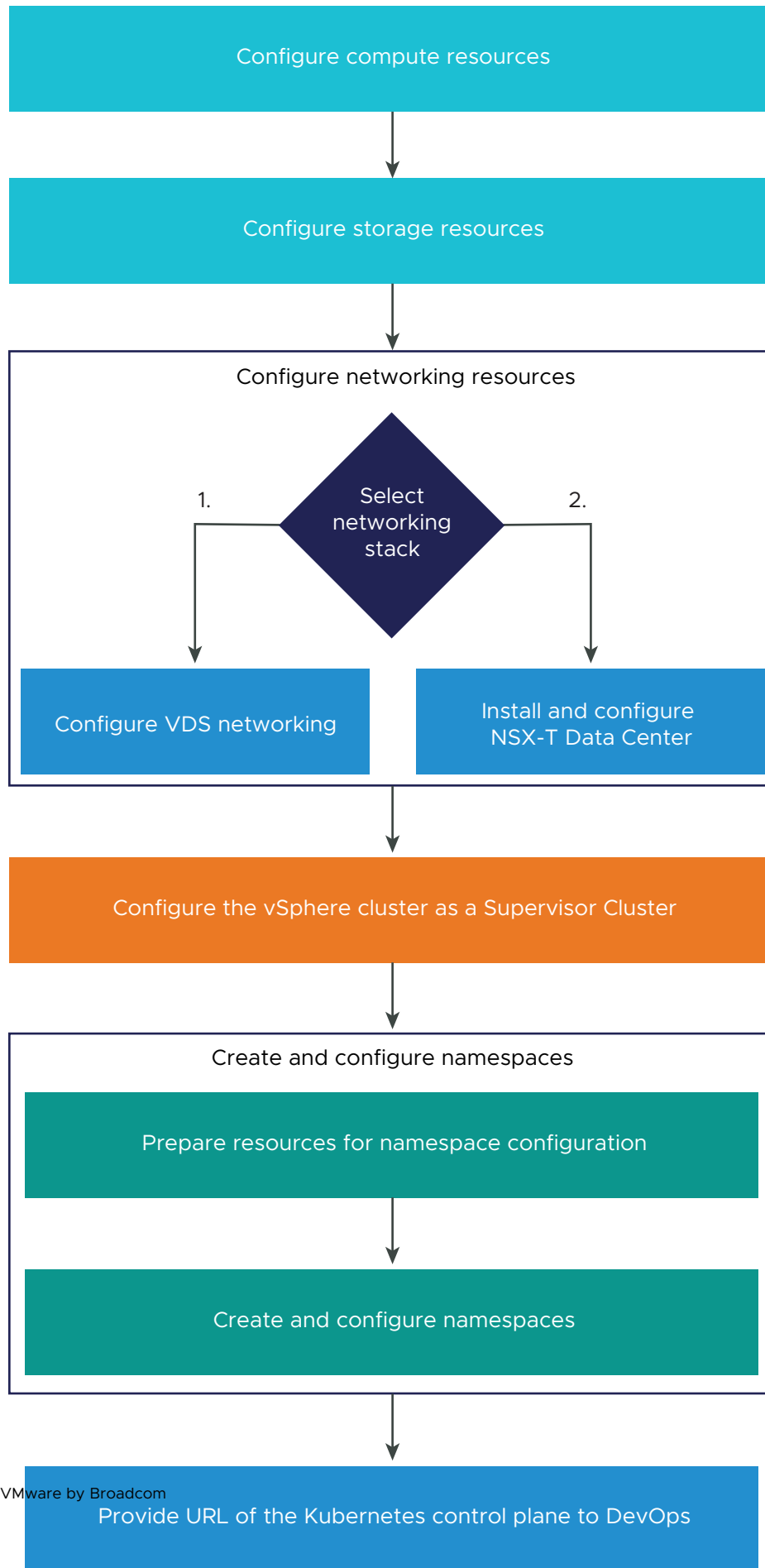
vSphere with Tanzu User Roles and Workflows

The vSphere with Tanzu platform involves two roles, the vSphere administrator and the DevOps engineer. Both roles interact with the platform through different interfaces and can have users or user groups defined for them in vCenter Single Sign-On with associated permissions. The workflows for the vSphere administrator and DevOps engineer roles are distinct and determined by the specific area of expertise these roles require.

User Roles and Workflows

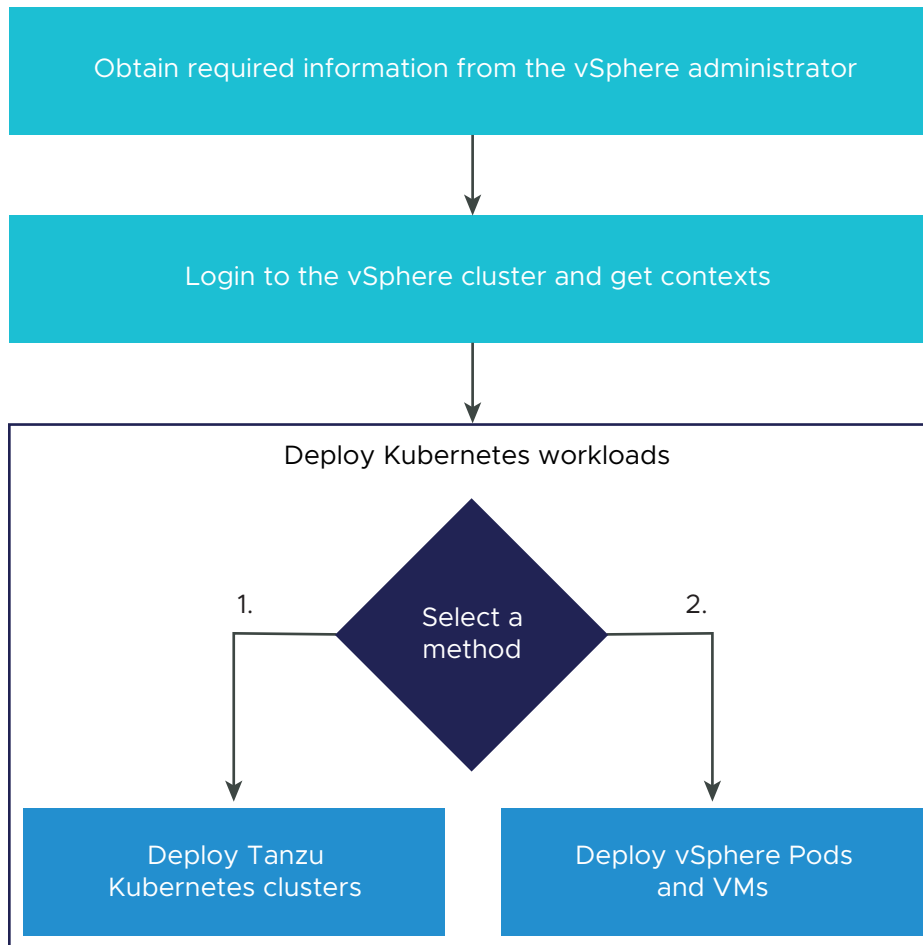
As a vSphere administrator, the primary interface through which you interact with the vSphere with Tanzu platform is the vSphere Client. At a high level, your responsibilities involve configuring a Supervisor Cluster and namespaces, where DevOps engineers can deploy Kubernetes workloads. You should have excellent knowledge about the vSphere and NSX-T technologies, and basic understanding about Kubernetes.

Figure 2-5. vSphere Administrator High Level Workflow



As a DevOps engineer, you might be a Kubernetes developer and an application owner, a Kubernetes administrator, or combine functions of both. As a DevOps engineer, you use `kubectl` commands to deploy vSphere Pods, VMs, and Tanzu Kubernetes clusters on existing namespaces on the Supervisor Cluster. Typically, as a DevOps engineer, you do not need to be an expert on vSphere and NSX-T, but have basic understanding about these technologies and the vSphere with Tanzu platform to interact with the vSphere administrators more efficiently.

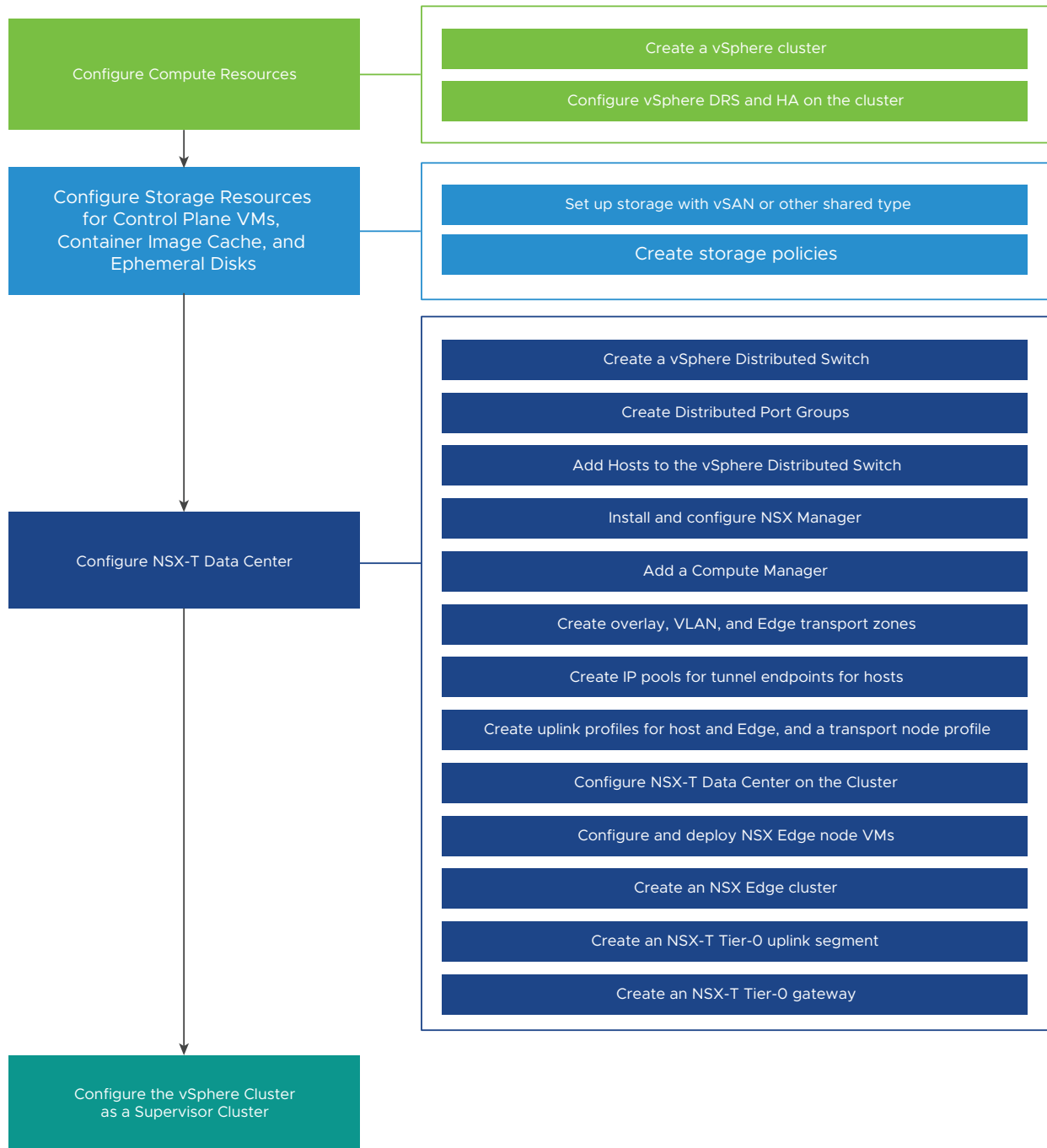
Figure 2-6. DevOps Engineer High Level Workflow



Supervisor Cluster with NSX-T Data Center Workflow

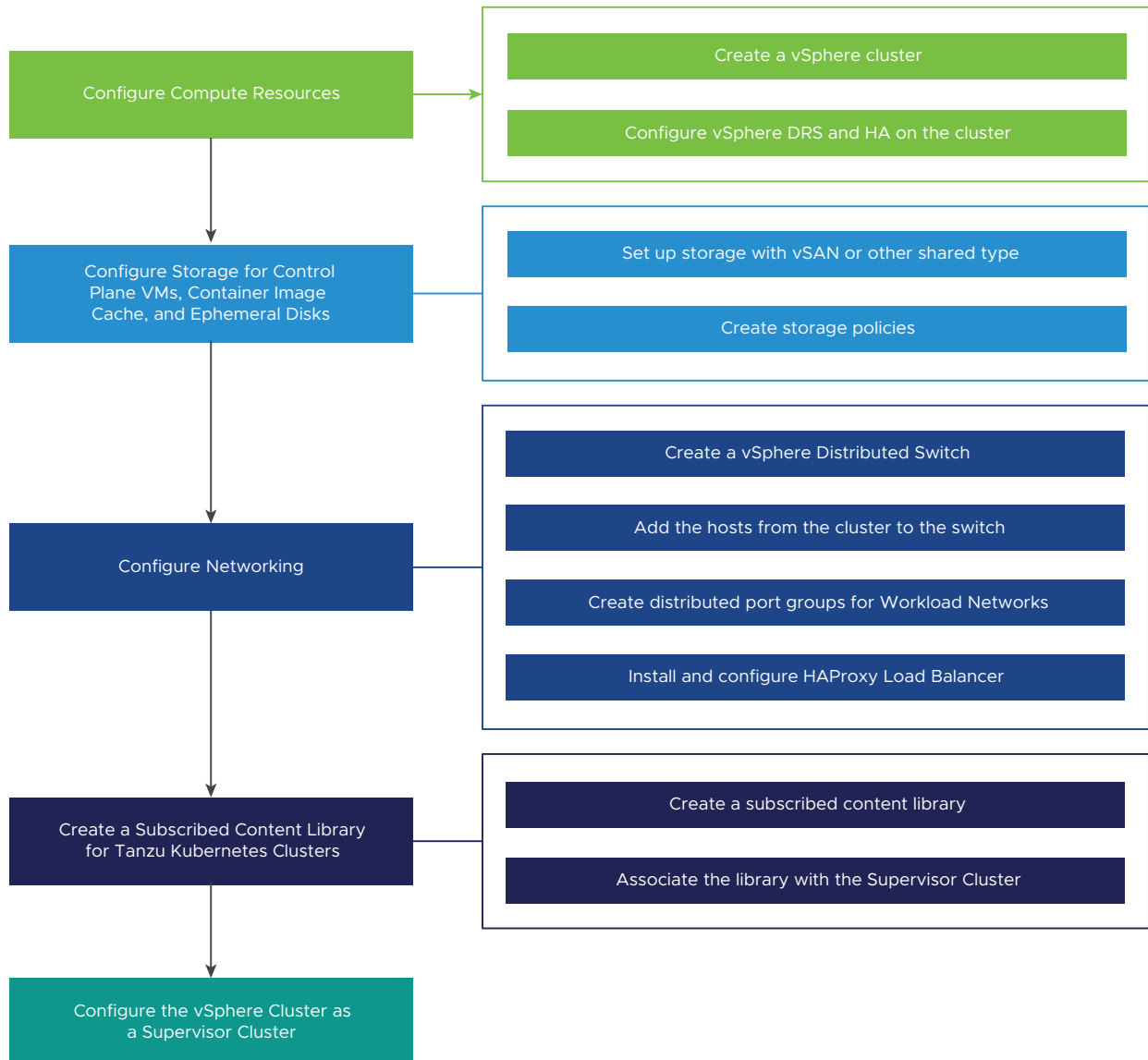
As a vSphere administrator, you configure the vSphere with Tanzu platform with the necessary compute, storage, and networking components. You can use NSX-T Data Center as a the networking stack for Supervisor Cluster. For more information about the system requirements, see [System Requirements for Setting Up vSphere with Tanzu with NSX](#).

Figure 2-7. Supervisor Cluster with NSX-Data Center Networking Workflow



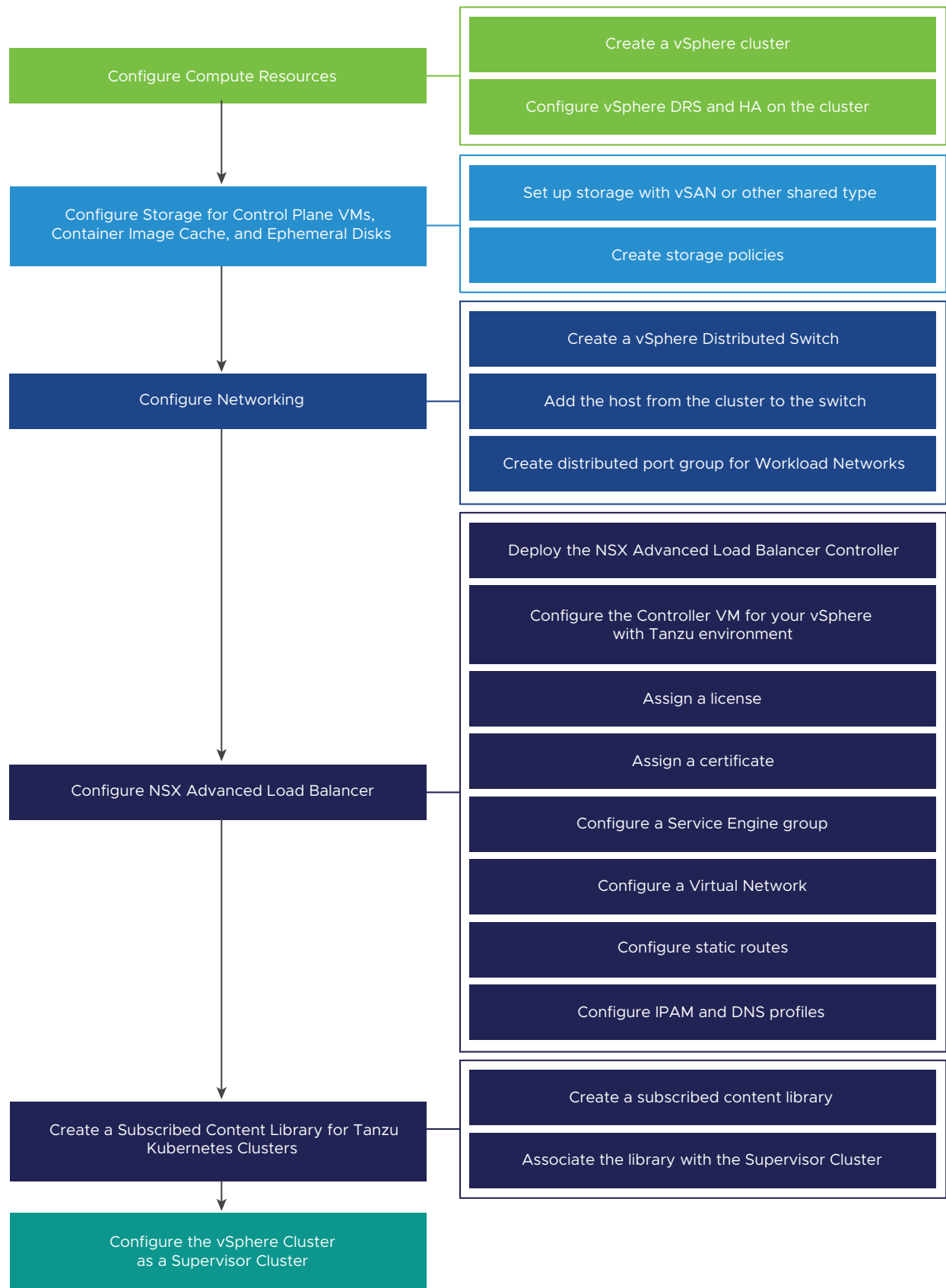
Supervisor Cluster with vSphere Networking Stack Workflow

As a vSphere administrator, you can configure a vSphere cluster as a Supervisor Cluster with the vSphere networking stack. For more information about the system requirements, see [System Requirements for Setting Up vSphere with Tanzu with vSphere Networking and HAProxy Load Balancer](#).

Figure 2-8. Supervisor Cluster with vSphere Networking Stack Configuration Workflow

Supervisor Cluster with vSphere Networking and NSX Advanced Load Balancer Workflow

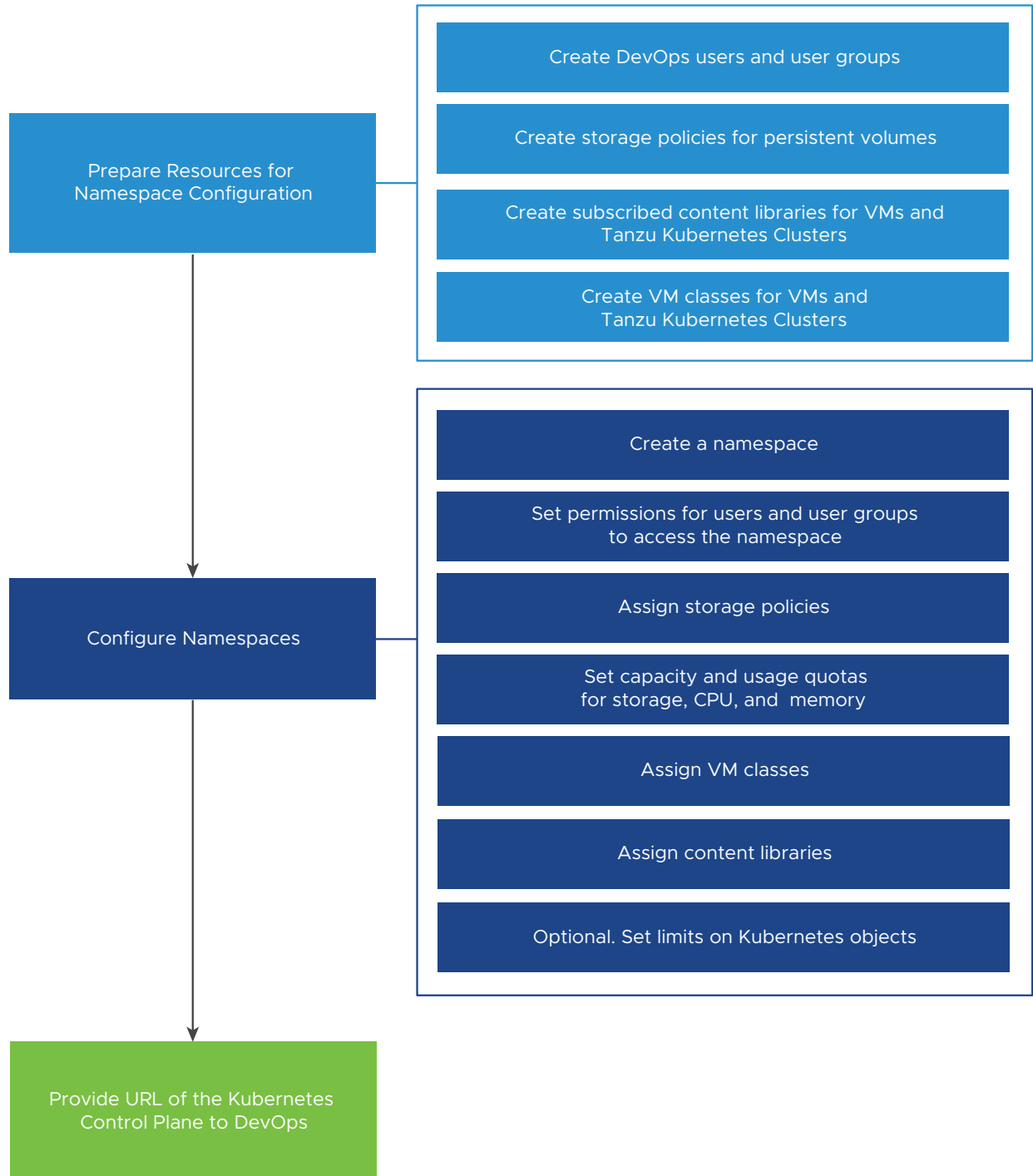
The diagram shows the workflow for configuring vSphere Networking and NSX Advanced Load Balancer for vSphere with Tanzu. For more information, see [Install and Configure the NSX Advanced Load Balancer](#).



Namespace Creation and Configuration Workflow

As a vSphere administrator, you create and configure namespaces on the Supervisor Cluster. You must gather specific resource requirements from DevOps engineers about the applications and workloads they want to run and configure the namespaces accordingly. For more information see, [Chapter 7 Configuring and Managing vSphere Namespaces](#).

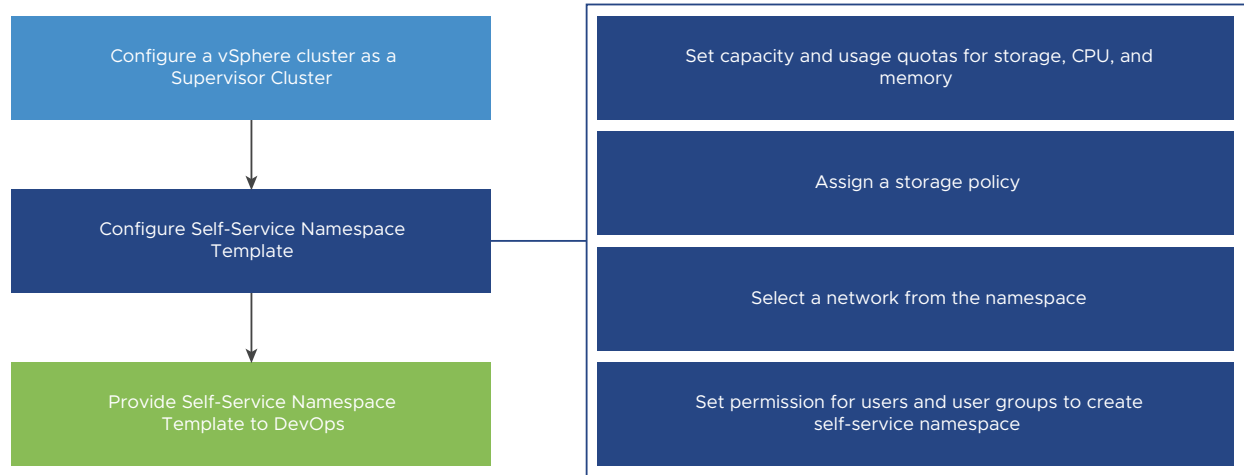
Figure 2-9. Namespace Configuration Workflow



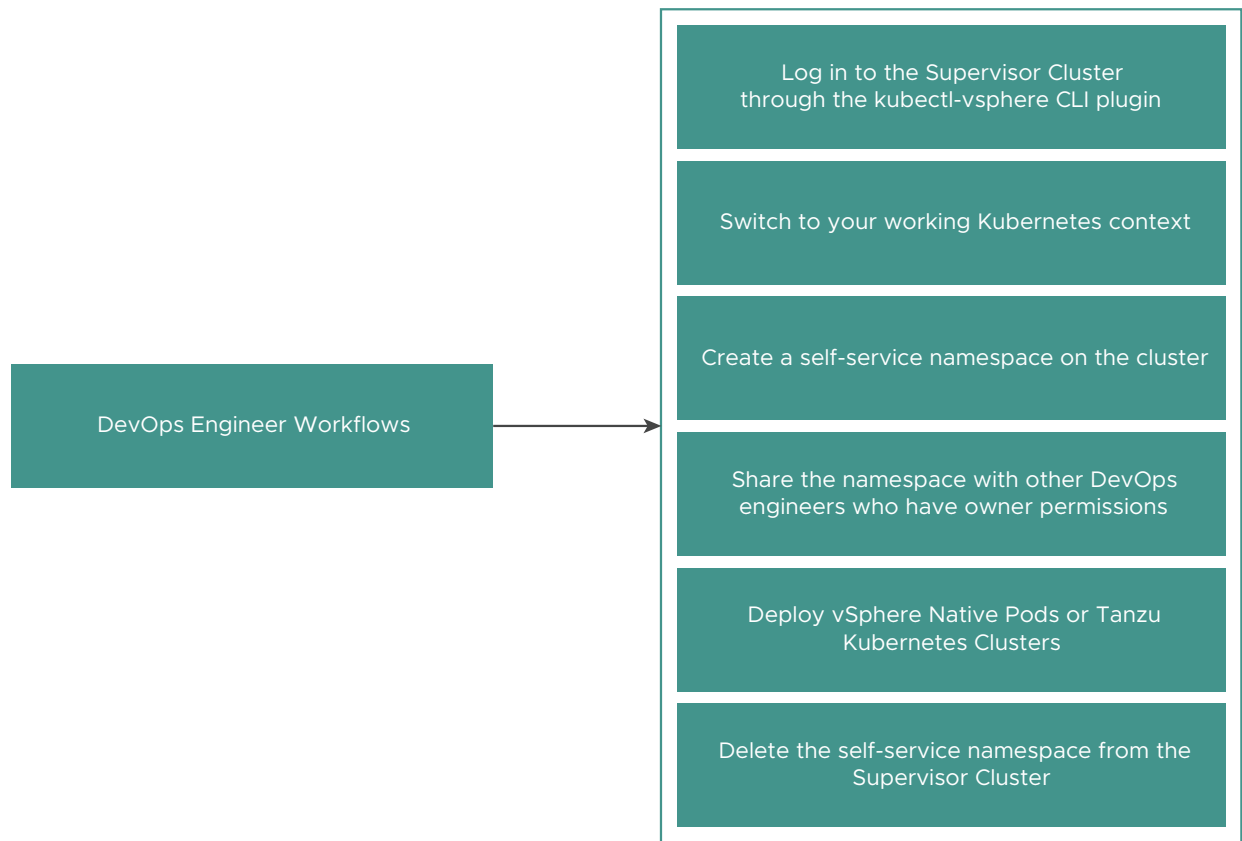
Self-Service Namespace Creation and Configuration Workflow

As a vSphere administrator, you can create a Supervisor Namespace, set CPU, memory, and storage limits to the namespace, assign permissions, and provision or activate the namespace service on a cluster as a template.

Figure 2-10. Self-service Namespace Template Provisioning Workflow



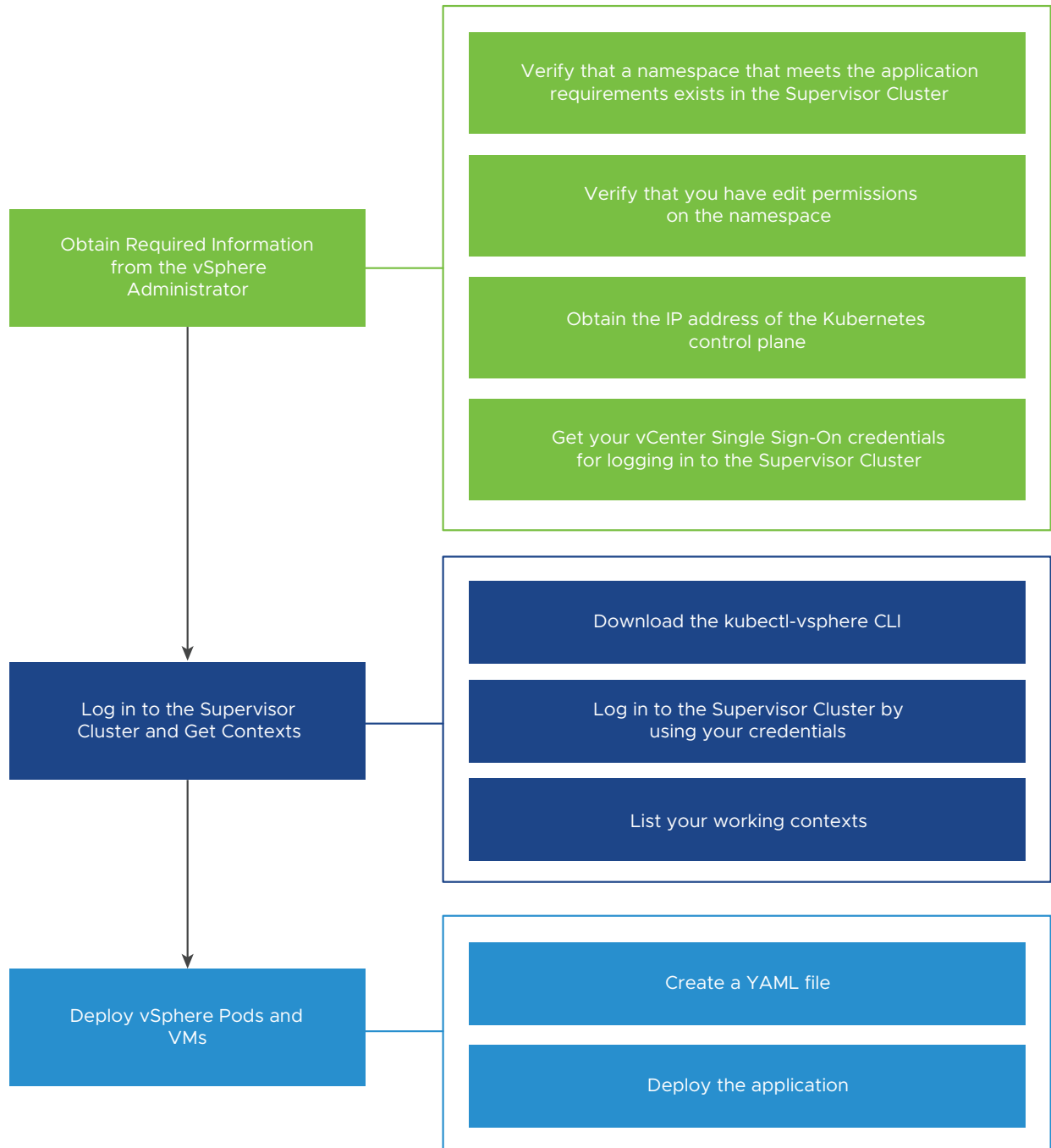
As a DevOps engineer, you can create a Supervisor Namespace in a self-service manner and deploy workloads within it. You can share it with other DevOps engineers or delete it when it is no longer required.

Figure 2-11. Self-service Namespace Creation Workflow

vSphere Pod and VM Provisioning Workflow

As a DevOps engineer, you can deploy vSphere Pods and VMs within the resources boundaries of a namespace that is running on a Supervisor Cluster. For more information, see [Chapter 11 Deploying Workloads to vSphere Pods](#) and [Chapter 12 Deploying and Managing Virtual Machines in vSphere with Tanzu](#).

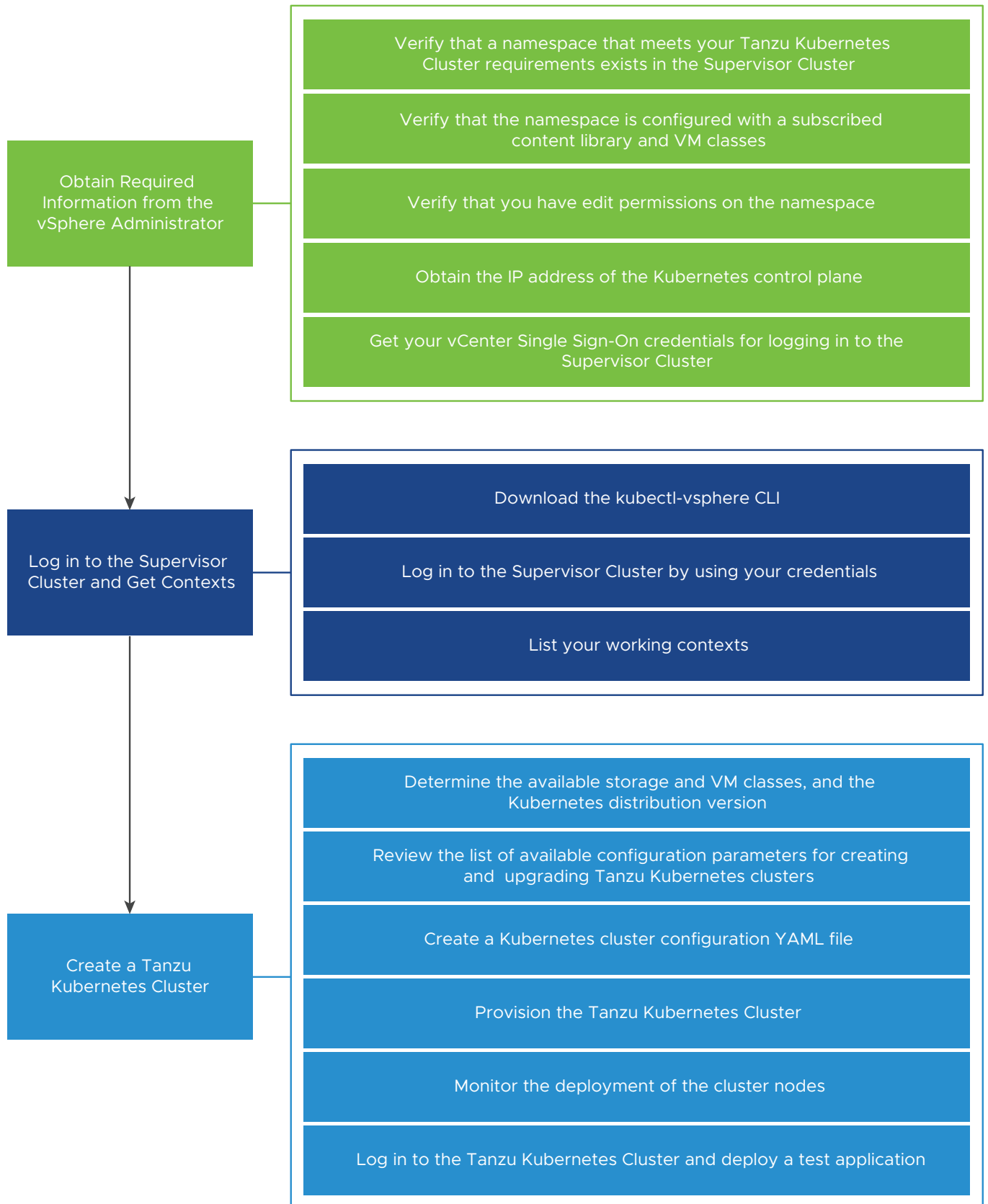
Figure 2-12. vSphere Pods and VM Provisioning Workflow



Tanzu Kubernetes Cluster Provisioning Workflow

As a DevOps engineer, you create and configure Tanzu Kubernetes clusters on a namespace created and configured by your vSphere administrator. For more information, see [Workflow for Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API](#).

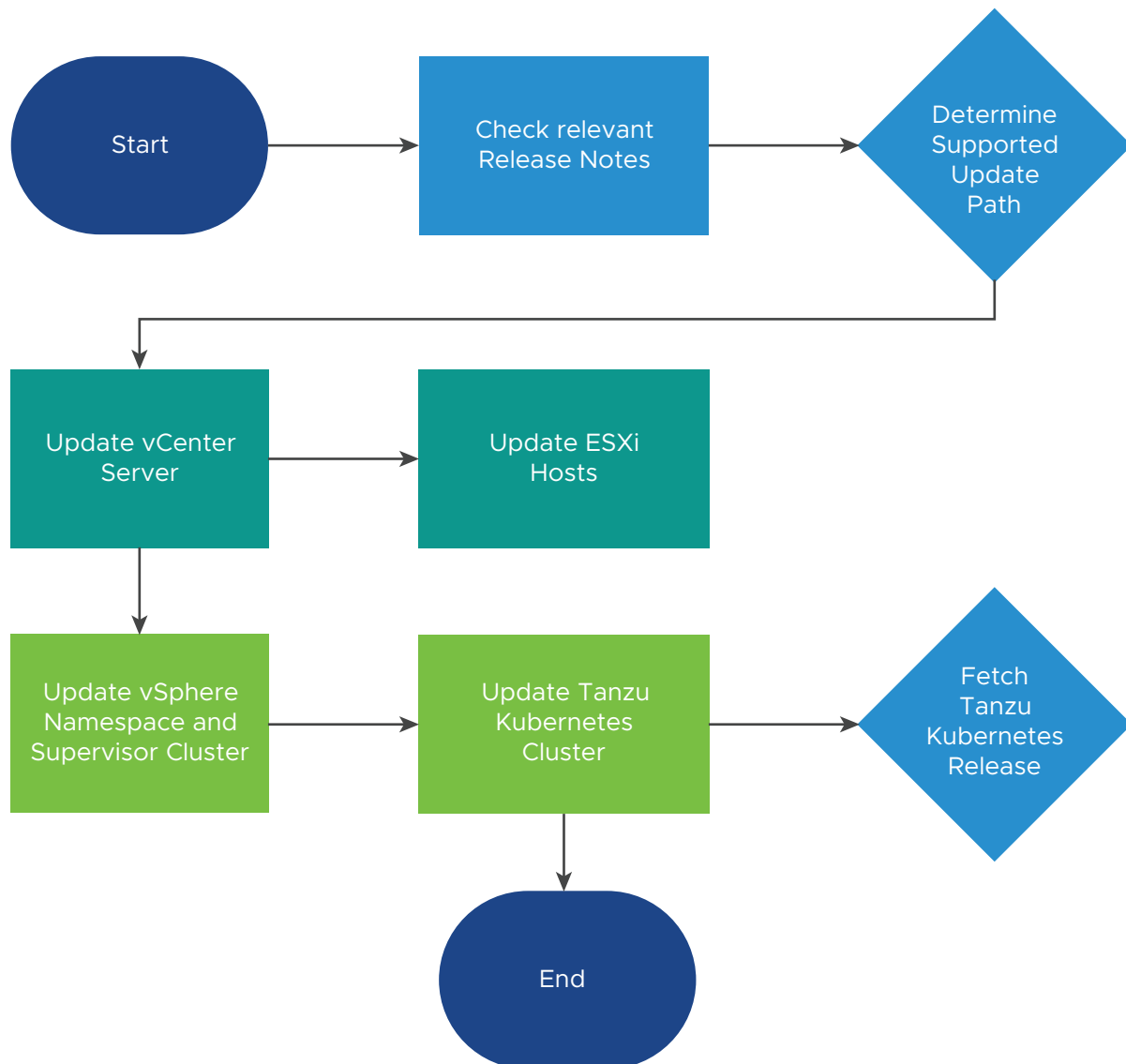
Figure 2-13. Tanzu Kubernetes Cluster Provisioning Workflow



vSphere with Tanzu Update Workflow

The diagram shows the workflow for updating the vSphere with Tanzu environment, including the Supervisor Cluster and Tanzu Kubernetes clusters. For more information, see [Chapter 17 Updating the vSphere with Tanzu Environment](#).

Figure 2-14. vSphere with Tanzu Update Workflow



How Does vSphere with Tanzu Change the vSphere Environment?

When a vSphere cluster is configured for Kubernetes workloads, thus becoming a Supervisor Cluster, it adds objects to the vCenter Server inventory, such as namespaces, vSphere Pods, and Tanzu Kubernetes clusters provisioned using the Tanzu Kubernetes Grid Service.

Under each Supervisor Cluster, you can view:

- Namespaces that represent logical applications running in the cluster.
- Resource pools for each namespace on the Supervisor Cluster.

Within every namespace, you can view:

- vSphere Pods.
- Kubernetes clusters created through the Tanzu Kubernetes Grid Service.
- Kubernetes control plane VMs.
- Networking and storage resources.
- User permissions for that namespace.

Licensing for vSphere with Tanzu

Once you configure a vSphere cluster for vSphere with Tanzu and it becomes a Supervisor Cluster, you must assign the cluster a Tanzu edition license before the 60 day evaluation period expires.

About the Tanzu Licenses

A Tanzu license enables the Workload Management functionality in vSphere 7.0 Update 1 and later. It is applicable to Supervisor Clusters that are configured with the vSphere networking stack or with NSX-T Data Center. For Supervisor Clusters running on vSphere 7.0, you need the VMware vSphere 7 Enterprise Plus with Add-on for Kubernetes license assigned to each host from the Supervisor Cluster.

As a vSphere administrator, when you assign a Tanzu license to a Supervisor Cluster, you can create and configure namespaces and provide access to these namespaces to DevOps engineers. As a DevOps engineer, you can deploy Tanzu Kubernetes clusters and vSphere Pods inside the namespaces to which you have access. If the Supervisor Cluster is configured with the vSphere networking stack, you can only deploy Tanzu Kubernetes clusters in them.

Licensing a Supervisor Cluster

After you enable **Workload Management** on a vSphere cluster, which deploys a Supervisor Cluster, you can use the full set of capabilities of the cluster within a 60 day evaluation period. You must assign a Tanzu license to the Supervisor Cluster before the 60 day evaluation period expires.

If you configure NSX-T Data Center as the networking stack for the Supervisor Cluster, you must assign an NSX-T Data Center Advanced or higher license to NSX Manager. If you configure the Supervisor Cluster with the vSphere networking stack with NSX Advanced Load Balancer, you need an appropriate license for the load balancer depending on your Tanzu license edition.

If your environment runs on top of vSphere 7.0 and you upgrade the Supervisor Cluster to vSphere 7.0 Update 1 or later, the cluster enters evaluation mode after the upgrade completes. The VMware vSphere 7 Enterprise Plus with Add-on for Kubernetes license that is assigned to the hosts acts as a regular vSphere Enterprise 7 Plus license, it does not enable any vSphere with Tanzu functionality. In that case, you must assign the Supervisor Cluster a Tanzu edition license before the 60 day evaluation period expires.

Tanzu License Expiration

- vSphere 7.0 Update 3. Starting from vSphere 7.0 Update 3, when a Tanzu edition license expires, you can continue using the full set of capabilities of vSphere with Tanzu until you procure new licenses. However, you cannot assign the expired license on new Supervisor Clusters. You must assign a valid Tanzu edition license to newly-created Supervisor Clusters before their 60 day evaluation period expires.
- vSphere 7.0 Update 2 and Update 1. When a Tanzu edition license expires in an environment that runs on vSphere Update 2 or Update 1, as a vSphere administrator you cannot create new namespaces or update the Kubernetes version of the Supervisor Cluster. As a DevOps engineer, you cannot deploy new workloads. You cannot change the configuration of the existing Tanzu Kubernetes clusters such as adding new nodes.

You can still deploy workloads on Tanzu Kubernetes clusters and all existing workloads continue to run as expected. All Kubernetes workloads that are already deployed continue their normal operation.

Tanzu License Compliance

A Tanzu license key has per CPU capacity with up to 32 cores per CPU, similarly to the ESXi host licenses. When you assign a Tanzu license to a Supervisor Cluster, the amount of capacity consumed is determined by the number of CPUs on the hosts from the cluster and the number of cores in each CPU. You can assign a Tanzu edition license key to multiple Supervisor Clusters at a time, but you cannot assign multiple license keys to one cluster.

- vSphere 7.0 Update 3. Starting from vSphere 7.0 Update 3, if you expand a Supervisor Cluster by adding new hosts for example, and the license key that you have assigned to the cluster runs out of capacity, you can continue using the same license key. To remain EULA compliant however, you must procure a new license key with sufficient capacity to cover all the CPUs and cores in the Supervisor Cluster.
- vSphere 7.0 Update 2 and Update 1. If your vSphere with Tanzu environment runs on vSphere 7.0 Update 2 and Update 1, the total number of CPUs in a Supervisor Cluster must not exceed the amount of CPU capacity of the Tanzu edition license that is assigned to the cluster.

Evaluation Period Expiration

When the evaluation period for a Supervisor Cluster expires, as a vSphere administrator you cannot create new namespaces or update the Kubernetes version of the Supervisor Cluster. As a DevOps engineer, you cannot deploy new workloads and you cannot make changes to the configuration of the existing Tanzu Kubernetes clusters such as adding new nodes.

You can still deploy workloads on Tanzu Kubernetes clusters and all existing workloads continue to run as expected. All Kubernetes workloads that are already deployed continue their normal operation.

The evaluation period expiration behavior is valid for both vSphere 7.0 Update 2 and Update 3.

vSphere with Tanzu Architecture and Components

3

A cluster enabled with vSphere with Tanzu is called a Supervisor Cluster. The cluster is at the base of the vSphere with Tanzu providing the necessary components and resources for running workloads that include vSphere Pods, VMs, and Tanzu Kubernetes clusters.

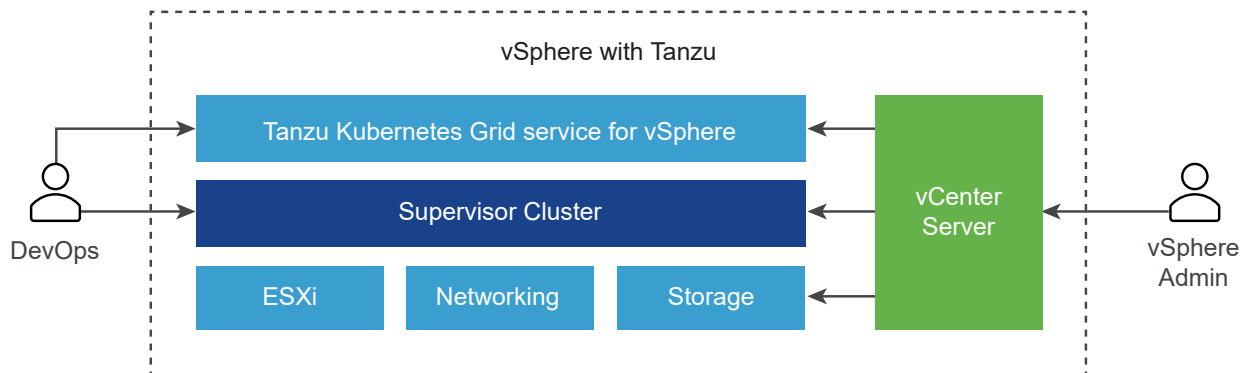
Read the following topics next:

- [vSphere with Tanzu Architecture](#)
- [Tanzu Kubernetes Grid Service Architecture](#)
- [Tanzu Kubernetes Cluster Tenancy Model](#)
- [vSphere with Tanzu Authentication](#)
- [vSphere with Tanzu Networking](#)
- [vSphere with Tanzu Security](#)
- [vSphere with Tanzu Storage](#)

vSphere with Tanzu Architecture

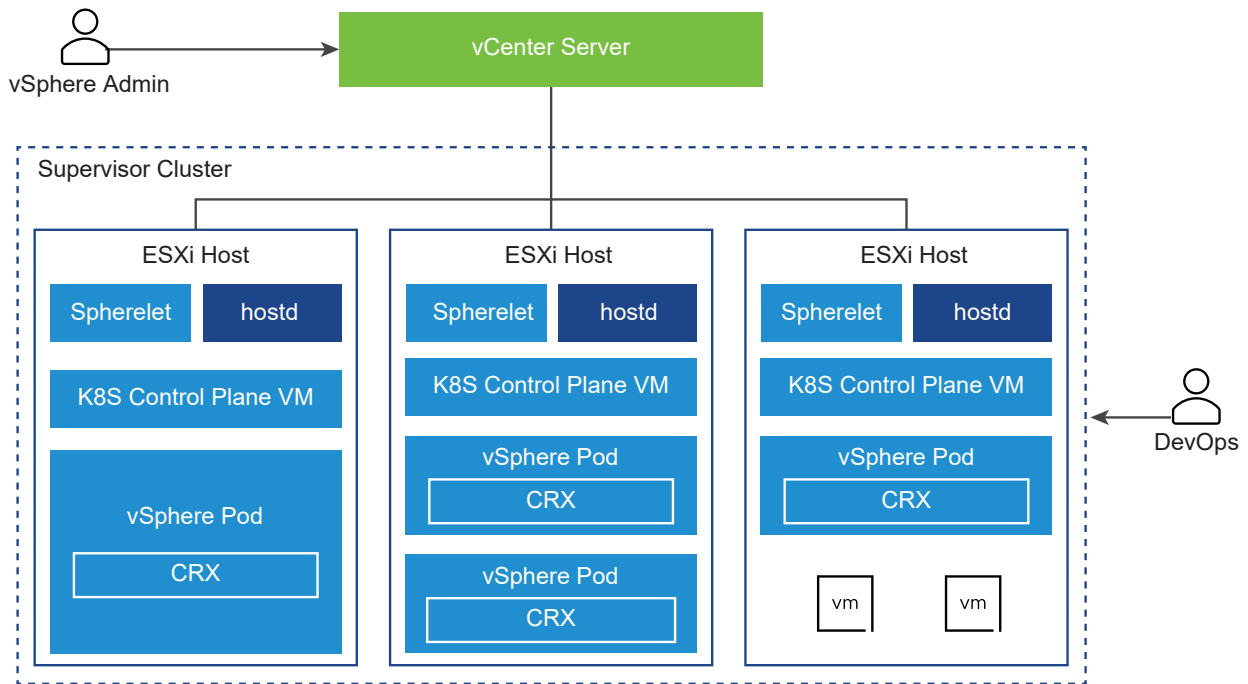
When vSphere with Tanzu is enabled on a vSphere cluster, it creates a Kubernetes control plane inside the hypervisor layer. This layer contains specific objects that enable the capability to run Kubernetes workloads within ESXi.

Figure 3-1. Supervisor Cluster General Architecture



A cluster that is enabled for vSphere with Tanzu is called a Supervisor Cluster. It runs on top of an SDDC layer that consists of ESXi for compute, NSX-T Data Center or vSphere networking, and vSAN or another shared storage solution. Shared storage is used for persistent volumes for vSphere Pods, VMs running inside the Supervisor Cluster, and pods in a Tanzu Kubernetes cluster. After a Supervisor Cluster is created, as a vSphere administrator you can create namespaces within the Supervisor Cluster that are called vSphere Namespaces. As a DevOps engineer, you can run workloads consisting of containers running inside vSphere Pods and create Tanzu Kubernetes clusters.

Figure 3-2. Supervisor Cluster Architecture



- **Kubernetes control plane VM.** Three Kubernetes control plane VMs in total are created on the hosts that are part of the Supervisor Cluster. The three control plane VMs are load balanced as each one of them has its own IP address. Additionally, a floating IP address is assigned to one of the VMs. vSphere DRS determines the exact placement of the control plane VMs on the ESXi hosts and migrates them when needed. vSphere DRS is also integrated with the Kubernetes Scheduler on the control plane VMs, so that DRS determines the placement of vSphere Pods. When as a DevOps engineer you schedule a vSphere Pod, the request goes through the regular Kubernetes workflow then to DRS, which makes the final placement decision.
- **Spherelet.** An additional process called Spherelet is created on each host. It is a kubelet that is ported natively to ESXi and allows the ESXi host to become part of the Kubernetes cluster.
- **Container Runtime Executive (CRX).** CRX is similar to a VM from the perspective of Hostd and vCenter Server. CRX includes a paravirtualized Linux kernel that works together with the

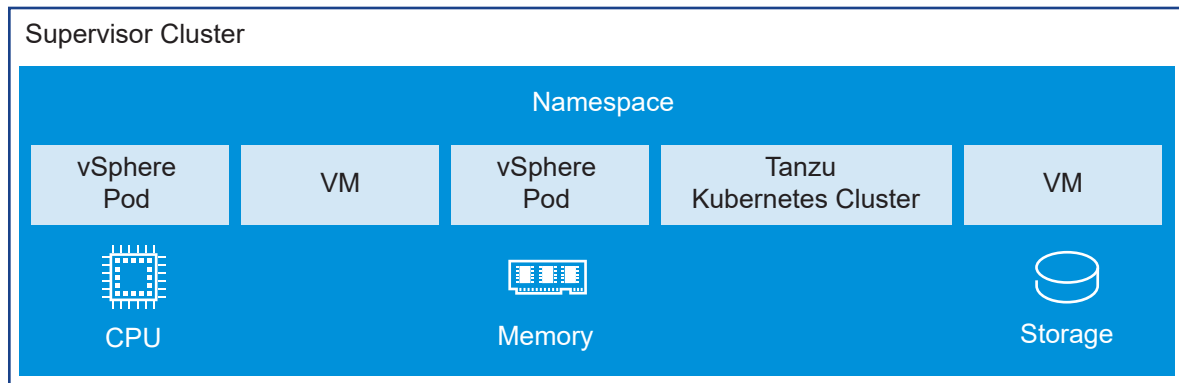
hypervisor. CRX uses the same hardware virtualization techniques as VMs and it has a VM boundary around it. A direct boot technique is used, which allows the Linux guest of CRX to initiate the main init process without passing through kernel initialization. This allows vSphere Pods to boot nearly as fast as containers.

- The Cluster API and VMware Tanzu™ Kubernetes Grid™ Service are modules that run on the Supervisor Cluster and enable the provisioning and management of Tanzu Kubernetes clusters. The Virtual Machine Service module is responsible for deploying and running stand-alone VMs and VMs that make up Tanzu Kubernetes clusters.

vSphere Namespace

A vSphere Namespace sets the resource boundaries where vSphere Pods and Tanzu Kubernetes clusters created by using the Tanzu Kubernetes Grid Service can run. When initially created, the namespace has unlimited resources within the Supervisor Cluster. As a vSphere administrator, you can set limits for CPU, memory, storage, as well as the number of Kubernetes objects that can run within the namespace. A resource pool is created per each namespace in vSphere. Storage limitations are represented as storage quotas in Kubernetes.

Figure 3-3. vSphere Namespace



To provide access to namespaces to DevOps engineer, as a vSphere administrator you assign permission to users or user groups available within an identity source that is associated with vCenter Single Sign-On.

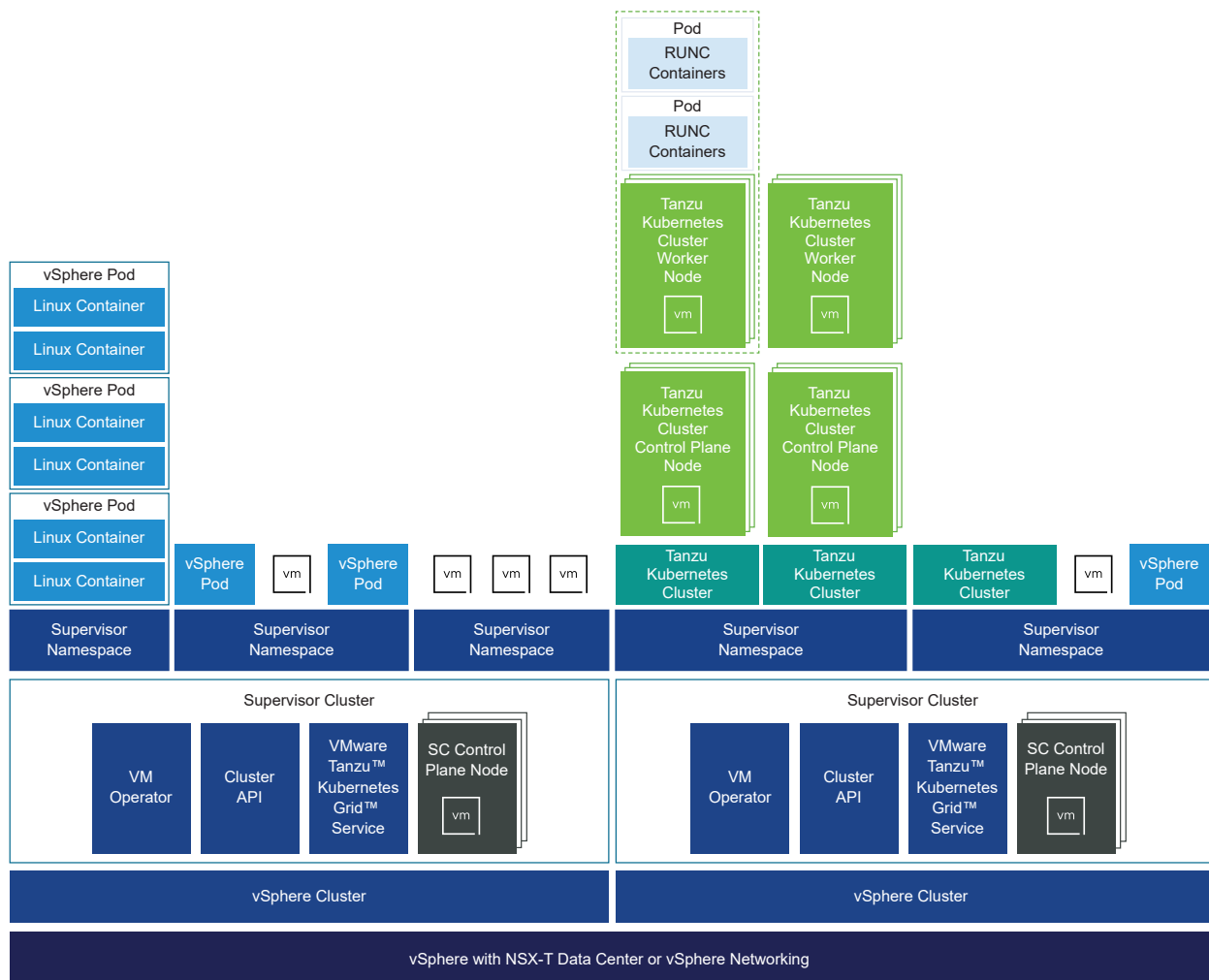
After a namespace is created and configured with resource and object limits as well as with permissions and storage policies, as a DevOps engineer you can access the namespace to run Kubernetes workloads and create Tanzu Kubernetes clusters by using the Tanzu Kubernetes Grid Service.

Tanzu Kubernetes Clusters

A Tanzu Kubernetes cluster is a full distribution of the open-source Kubernetes software that is packaged, signed, and supported by VMware. In the context of vSphere with Tanzu, you can use the Tanzu Kubernetes Grid Service to provision Tanzu Kubernetes clusters on the Supervisor Cluster. You can invoke the Tanzu Kubernetes Grid Service API declaratively by using `kubectl` and a YAML definition.

A Tanzu Kubernetes cluster resides in a vSphere Namespace. You can deploy workloads and services to Tanzu Kubernetes clusters the same way and by using the same tools as you would with standard Kubernetes clusters.

Figure 3-4. vSphere with Tanzu Architecture for Tanzu Kubernetes Clusters



Supervisor Cluster Configured with the vSphere Networking Stack

A Supervisor Cluster that is configured with the vSphere networking stack only supports running Tanzu Kubernetes clusters created by using the Tanzu Kubernetes Grid Service. The cluster also supports the vSphere Network Service and the Storage Service.

A Supervisor Cluster that is configured with the vSphere networking stack does not support vSphere Pods. Therefore, the Spherelet component is not available in such Supervisor Cluster and Kubernetes pods run inside Tanzu Kubernetes clusters only. A Supervisor Cluster that is configured with the vSphere networking stack also does not support the Harbor Registry, because the service is only used with vSphere Pods.

A vSphere Namespace created on a cluster that is configured with the vSphere networking stack also does not support running vSphere Pods, but only Tanzu Kubernetes clusters.

Tanzu Kubernetes Grid Service Architecture

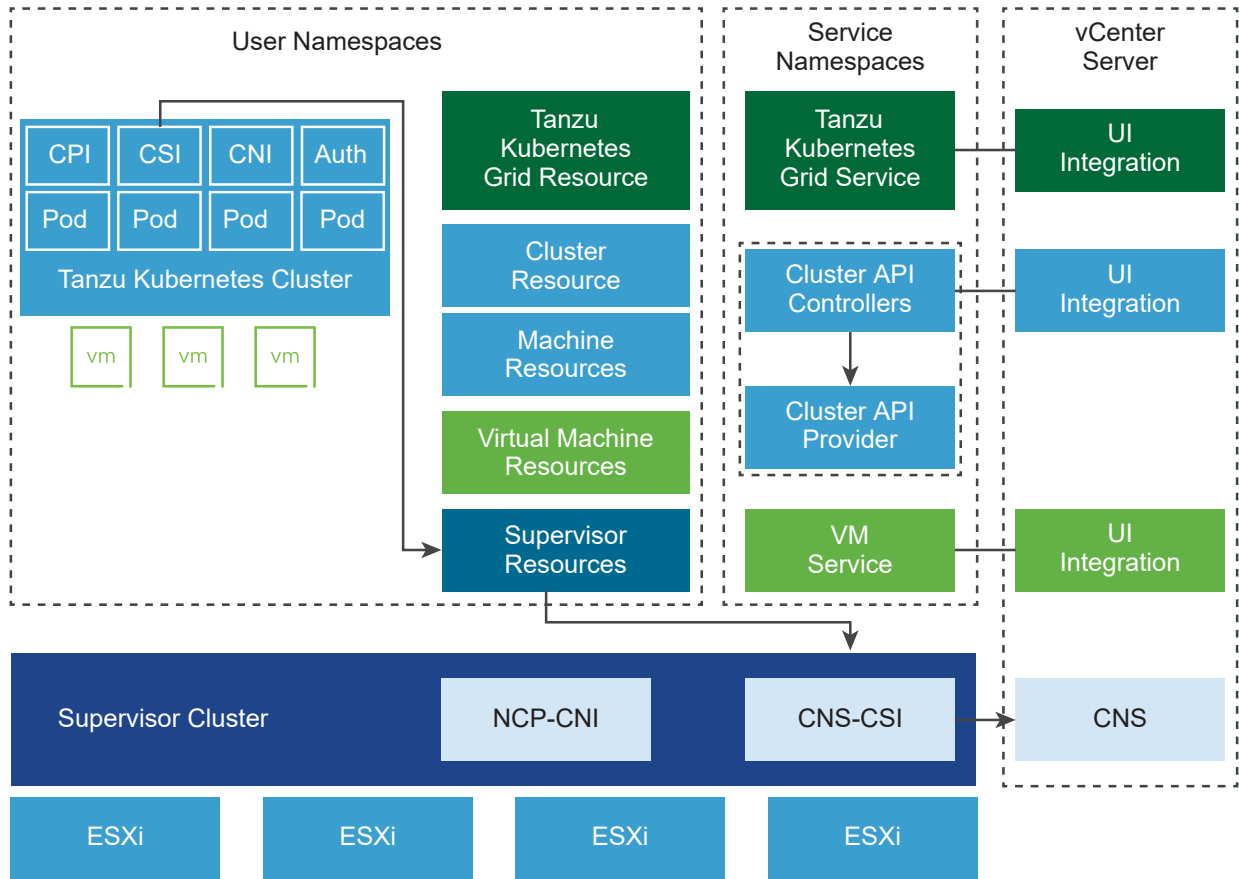
The Tanzu Kubernetes Grid Service provides self-service lifecycle management of Tanzu Kubernetes clusters. You use the Tanzu Kubernetes Grid Service to create and manage Tanzu Kubernetes clusters in a declarative manner that is familiar to Kubernetes operators and developers.

Tanzu Kubernetes Grid Service Components

The Tanzu Kubernetes Grid Service exposes three layers of controllers to manage the lifecycle of a Tanzu Kubernetes cluster.

- The Tanzu Kubernetes Grid Service provisions clusters that include the components necessary to integrate with the underlying vSphere Namespace resources. These components include a Cloud Provider Plugin that integrates with the Supervisor Cluster. In addition, a Tanzu Kubernetes cluster passes requests for persistent volumes to the Supervisor Cluster, which is integrated with VMware Cloud Native Storage (CNS). See [Chapter 10 Using Persistent Storage in vSphere with Tanzu](#).
- The Cluster API provides declarative, Kubernetes-style APIs for cluster creation, configuration, and management. The inputs to Cluster API include a resource describing the cluster, a set of resources describing the virtual machines that make up the cluster, and a set of resources describing cluster add-ons.
- The Virtual Machine Service provides a declarative, Kubernetes-style API for management of VMs and associated vSphere resources. The Virtual Machine Service introduces the concept of a virtual machine class that represents an abstract reusable hardware configuration. The functionality provided by the Virtual Machine Service is used to manage the lifecycle of the control plane and worker node VMs hosting a Tanzu Kubernetes cluster.

Figure 3-5. Tanzu Kubernetes Grid Service Architecture and Components



Tanzu Kubernetes Cluster Components

The components that run in a Tanzu Kubernetes cluster span four areas: Authentication and authorization, storage integration, pod networking, and load balancing.

- **Authentication webhook:** A webhook running as a pod inside the cluster to validate user authentication tokens.
- **Container Storage Interface Plugin:** A Paravirtual CSI plug-in that integrates with CNS through the Supervisor Cluster.
- **Container Network Interface Plug-in:** A CNI plugin that provides pod networking.
- **Cloud Provider Implementation:** Supports creating Kubernetes load balancer services.

Tanzu Kubernetes Grid Service API

You use the Tanzu Kubernetes Grid Service API to provision and manage Tanzu Kubernetes clusters. It is a declarative API that you invoke using `kubectl` and YAML.

With a declarative API, instead of making imperative commands to the system, you specify the desired state of the Tanzu Kubernetes cluster: how many nodes, available storage, VM sizes, Kubernetes software version. The Tanzu Kubernetes Grid Service does the work to provision a cluster that matches the desired state.

To call the Tanzu Kubernetes Grid Service API, you invoke `kubectl` using a YAML file, which in turn invokes the API. After the cluster is created, you update the YAML to update the cluster.

Tanzu Kubernetes Grid Service Interfaces

vSphere administrators use the vSphere Client to configure the vSphere Namespace and grant permissions. They can also monitor the resources used by cluster components and to view relevant information from those resources in the vSphere inventory.

DevOps engineers use the vSphere Plugin for `kubectl` to connect to the vSphere Namespace with their vCenter Single Sign-On credentials. After connecting, DevOps engineers use `kubectl` to provision Tanzu Kubernetes clusters.

Developers can connect to a provisioned cluster using the vSphere Plugin for `kubectl` and their vCenter Single Sign-On credentials. Alternatively, if the cluster administrator configured a supported Kubernetes authentication provider, developers can connect using `kubectl`. To deploy workloads in Kubernetes and interact with the cluster environment, developers use `kubectl`.

Tanzu Kubernetes Grid Service Demo

Watch the following video to learn how you can use the Tanzu Kubernetes Grid Service to create and operate Tanzu Kubernetes clusters: [vSphere 7 with Kubernetes - Tanzu Kubernetes cluster - Technical Overview](#).

Tanzu Kubernetes Cluster Tenancy Model

The Supervisor Cluster is the management plane for Tanzu Kubernetes clusters provisioned by the Tanzu Kubernetes Grid Service. The tenancy model is enforced using a vSphere Namespace where Tanzu Kubernetes clusters reside.

Supervisor Cluster

The Supervisor Cluster provides the management layer on which Tanzu Kubernetes clusters are built. The Tanzu Kubernetes Grid Service is a custom controller manager with a set of controllers that is part of the Supervisor Cluster. The purpose of the Tanzu Kubernetes Grid Service is to provision Tanzu Kubernetes clusters.

While there is a one-to-one relationship between the Supervisor Cluster and the vSphere cluster, there is a one-to-many relationship between the Supervisor Cluster and Tanzu Kubernetes clusters. You can provision multiple Tanzu Kubernetes clusters within a single Supervisor Cluster. The workload management functionality provided by the Supervisor Cluster gives you control over the cluster configuration and lifecycle, while allowing you to maintain concurrency with upstream Kubernetes.

For more information, see [Chapter 5 Configuring and Managing a Supervisor Cluster](#).

vSphere Namespace

You deploy one or more Tanzu Kubernetes clusters to a vSphere Namespace. Resource quotas and storage policy are applied to a vSphere Namespace and inherited by the Tanzu Kubernetes clusters deployed there.

When you provision a Tanzu Kubernetes cluster, a resource pool and VM folder are created in the vSphere Namespace. The Tanzu Kubernetes cluster control plane and worker node VMs are placed within this resource pool and VM folder. Using the vSphere Client, you can view this hierarchy by selecting the **Hosts and Clusters** perspective, and also by selecting the **VMs and Templates** view.

For more information, see [Chapter 7 Configuring and Managing vSphere Namespaces](#).

Content Library

A vSphere Content Library provides the virtual machine template used to create the Tanzu Kubernetes cluster nodes. For each Supervisor Cluster where you intend to deploy a Tanzu Kubernetes cluster, you must define a Subscribed Content Library object that sources the OVA used by the Tanzu Kubernetes Grid Service to build cluster nodes. The same Subscribed Content Library can be configured for multiple Supervisor Clusters. There is no relationship between the Subscribed Content Library and the vSphere Namespace. The Subscribed Content Library downloads the latest templates directly from VMware. You upload the OVA templates you want to use to a Local Content Library.

For more information, see [Creating and Managing Content Libraries for Tanzu Kubernetes releases](#).

vSphere with Tanzu Authentication

As a vSphere administrator, you need privileges to configure a Supervisor Cluster and to manage namespaces. You define permissions on namespaces to determine which DevOps engineers can access them. As a DevOps engineer, you authenticate with the Supervisor Cluster by using your vCenter Single Sign-On credentials, and can access only the namespaces for which you have permissions.

Note In vSphere 7.x, x509 certificate chains are not supported with the L4 load balancer endpoint used for the Kubernetes API server. Certificate chains that involve intermediate CAs (for example, if you bring your own CA chain) might experience certificate verification errors.

Permissions for vSphere Administrators

As a vSphere administrator, you need permissions on vSphere clusters to configure them as Supervisor Clusters as well as to create and manage namespaces. You must have at least one of the following privileges associated with your user account on a vSphere cluster:

- **Modify namespace configuration.** Allows you to create and configure namespaces on a Supervisor Cluster.
- **Modify cluster-wide configuration.** Allows you to configure a vSphere cluster as a Supervisor Cluster.

Setting Permissions for DevOps Engineers

As a vSphere administrator, you grant view, edit, or owner permissions to user accounts on namespace level. The user accounts must be available in an identity source that is connected to vCenter Single Sign-On. One user account can have access to multiple namespaces. Users which are members of the Administrators groups have access to all the namespaces on the Supervisor Cluster.

After you configure a namespace with permissions, resource quotas, and storage, you provide the URL of the Kubernetes control plane to DevOps engineers, who can use it to log in to the control plane. Once logged in, DevOps engineers can access all the namespaces for which they have permissions across all of the Supervisor Clusters that belong to a vCenter Server system. When vCenter Server systems are in Enhanced Linked Mode, DevOps engineers can access all namespaces for which they have permissions across all the Supervisor Clusters available in the Linked Mode group. The IP address of the Kubernetes control plane is a virtual IP generated by NSX-T or a load balancer in use with the VDS networking stack to serve as an access point to the Kubernetes control plane.

DevOps engineers with owner permissions can deploy workloads. They can share the namespace with other DevOps engineers or groups and delete it when it is no longer required. When DevOps engineers share the namespace, they can assign view, edit, or owner permissions to other DevOps engineers and groups.

Authentication with the Supervisor Cluster

As DevOps engineer, you use the Kubernetes CLI Tools for vSphere to authenticate to the Supervisor Cluster by using your vCenter Single Sign-On credentials and the Kubernetes control plane IP address. For more information, see [Connect to the Supervisor Cluster as a vCenter Single Sign-On User](#).

When you log in to the Supervisor Cluster, an authentication proxy redirects the request to vCenter Single Sign-On. The vSphere kubectl plug-in establishes a session with vCenter Server and obtains an authentication token from vCenter Single Sign-On. It also fetches a list of namespaces to which you have access, and populates the configuration with these namespaces. The list of namespaces is updated on the next login, if there are changes to the permissions of your user account.

The account that you use to login to the Supervisor Cluster provides you with access only to the namespaces that are assigned to you. You cannot login to vCenter Server with that account. To login to vCenter Server, you will need explicit permissions.

Note The session to kubectl lasts for 10 hours. After the session expires, you must authenticate with the Supervisor Cluster again. At logout, the token is deleted from the configuration file of your user account, but remains valid until the session ends.

Authentication with Tanzu Kubernetes Clusters

Tanzu Kubernetes cluster users, including DevOps engineers, developers, and administrators, can authenticate with a cluster in various ways. For more information, see [Authenticating with Tanzu Kubernetes Clusters](#).

Note Tanzu Kubernetes clusters require user and system accounts to have pod security policy to deploy pods and resources to a cluster. For more information, see [Using Pod Security Policies with Tanzu Kubernetes Clusters](#).

vSphere with Tanzu Networking

A Supervisor Cluster can either use the vSphere networking stack or VMware NSX-T™ Data Center to provide connectivity to Kubernetes control plane VMs, services, and workloads. The networking used for Tanzu Kubernetes clusters provisioned by the Tanzu Kubernetes Grid Service is a combination of the fabric that underlies the vSphere with Tanzu infrastructure and open-source software that provides networking for cluster pods, services, and ingress.

For more information, see [Chapter 4 Networking for vSphere with Tanzu](#)

vSphere with Tanzu Security

vSphere with Tanzu leverages vSphere security features and provisions Tanzu Kubernetes clusters that are secure by default.

vSphere with Tanzu is an add-on module to vSphere that is able to leverage the security features that are built into vCenter Server and ESXi. For more information, see the [vSphere Security](#) documentation.

The Supervisor Cluster encrypts all secrets stored in the database (etcd). The secrets are encrypted via a local decryption key file, which is provided at boot by vCenter Server. The decryption key is stored in memory (tempfs) on the Supervisor Cluster nodes and on disk in encrypted form within vCenter Server database. The key is available in clear text to the root users of each system. Secrets held within the database of each workload cluster are stored in clear text. All etcd connections are authenticated with certificates that are generated at installation and rotated during upgrades. Manual rotation or updating of the certificates is currently not possible.

Starting from vSphere 7.0 Update 2, you can run confidential vSphere Pods in a Supervisor Cluster on AMD systems. You can create confidential vSphere Pods by adding Secure Encrypted Virtualization-Encrypted State (SEV-ES) as a security enhancement. For more information, see [Deploy a Confidential vSphere Pod](#).

A Tanzu Kubernetes cluster is secure by default. Restrictive PodSecurityPolicy (PSP) is available for any Tanzu Kubernetes cluster provisioned by the Tanzu Kubernetes Grid Service. If developers need to run privileged pods or root containers, at a minimum a cluster administrator must create a RoleBinding that grants user access to the default privileged PSP. For more information, see [Using Pod Security Policies with Tanzu Kubernetes Clusters](#).

A Tanzu Kubernetes cluster does not have infrastructure credentials. The credentials that are stored within a Tanzu Kubernetes cluster are only sufficient to access the vSphere Namespace where the Tanzu Kubernetes cluster has tenancy. As a result, there is no privilege escalation avenue for cluster operators or users.

The authentication tokens used to access Tanzu Kubernetes clusters are scoped such that the tokens cannot be used to access the Supervisor Cluster. This prevents cluster operators, or individuals who might try to compromise a cluster, from using their root-level access to capture a vSphere administrator's token when they log in to a Tanzu Kubernetes cluster.

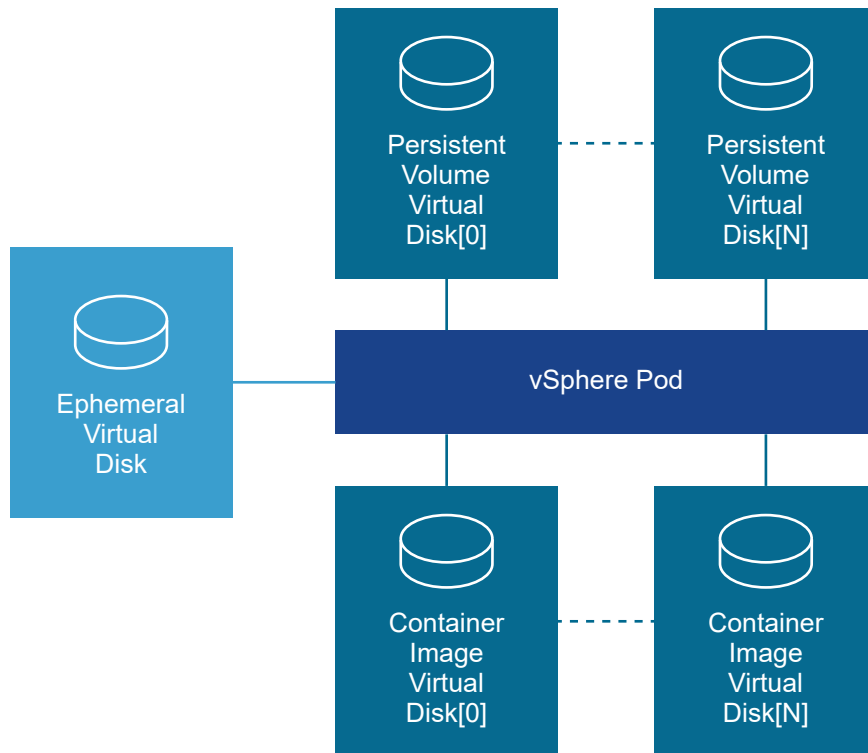
vSphere with Tanzu Storage

vSphere with Tanzu uses storage policies to integrate with shared datastores available in your environment, including VMFS, NFS, vSAN, or vVols datastores. The policies represent datastores and manage the storage placement of such objects as control plane VMs, pod ephemeral disks, container images, and persistent storage volumes. If you use Tanzu Kubernetes clusters, the storage policies also dictate how the Tanzu Kubernetes cluster nodes are deployed.

Before you enable vSphere with Tanzu, create storage policies to be used by the Supervisor Cluster and namespaces.

Depending on your vSphere storage environment and the needs of DevOps, you can create several storage policies to represent different classes of storage.

For example, if a vSphere Pod mounts all three types of virtual disks and your vSphere storage environment has three classes of datastores, Bronze, Silver, and Gold, you can create storage policies for all datastores. You can then use the Bronze datastore for ephemeral and container image virtual disks, and use the Silver and Gold datastores for persistent volume virtual disks.



For general information about storage policies, see the [Storage Policy Based Management](#) chapter in the *vSphere Storage* documentation. For information about creating storage policies, see [Create Storage Policies for vSphere with Tanzu](#).

Ephemeral Virtual Disks

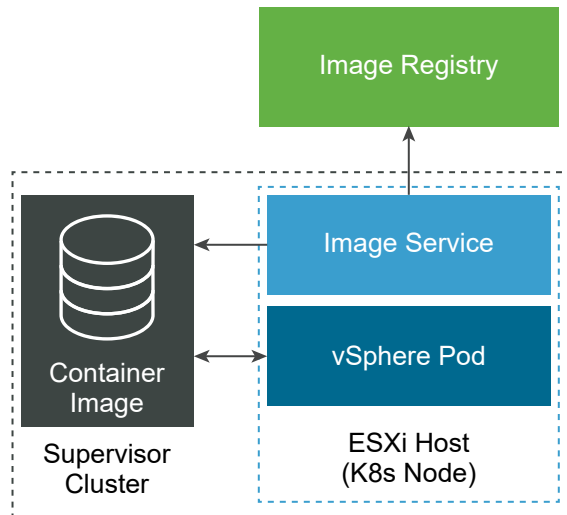
A vSphere Pod and a pod that runs in a Tanzu Kubernetes cluster requires ephemeral storage to store such Kubernetes objects as logs, `emptyDir` volumes, and `ConfigMaps` during its operations. This ephemeral, or transient, storage lasts as long as the pod continues to exist. Ephemeral data persists across container restarts, but once the pod reaches the end of its life, the ephemeral virtual disk disappears.

Each pod has one ephemeral virtual disk. A vSphere administrator uses a storage policy to define the datastore location for all ephemeral virtual disks when configuring storage for the Supervisor Cluster.

Container Image Virtual Disks

Containers inside the pod use images that contain the software to be run. The pod mounts images used by its containers as image virtual disks. When the pod completes its life cycle, the image virtual disks are detached from the pod.

Image Service, an ESXi component, is responsible for pulling container images from the image registry and transforming them into virtual disks to run inside the pod.



ESXi can cache images that are downloaded for the containers running in the pod. Subsequent pods that use the same image pull it from the local cache rather than the external container registry.

As with ephemeral disks, the vSphere administrator specifies the datastore location for the image cache at the Supervisor Cluster level. See [Chapter 5 Configuring and Managing a Supervisor Cluster](#) and [Change Storage Settings on the Supervisor Cluster](#).

For information about working with the container images, see [Chapter 15 Using a Container Registry for vSphere with Tanzu Workloads](#).

Persistent Storage Virtual Disks

Certain Kubernetes workloads require persistent storage to store data permanently. To provision persistent storage for Kubernetes workloads, vSphere with Tanzu integrates with Cloud Native Storage (CNS), a vCenter Server component that manages persistent volumes.

Persistent storage can be used by vSphere Pods, Tanzu Kubernetes clusters, and VMs. To make persistent storage available to the DevOps team, the vSphere administrators create VM storage policies that describe different storage requirements and classes of services. They can then assign the storage policies to a vSphere Namespace. See [Create and Configure a vSphere Namespace](#) and [Change Storage Settings on a Namespace](#).

For more information and for specifics on how persistent storage is used by the Supervisor Cluster and the Tanzu Kubernetes clusters, see [Chapter 10 Using Persistent Storage in vSphere with Tanzu](#) and [Chapter 13 Provisioning and Operating TKGS Clusters](#).

Networking for vSphere with Tanzu

4

A Supervisor Cluster can either use the vSphere networking stack or VMware NSX-T™ Data Center to provide connectivity to Kubernetes control plane VMs, services, and workloads. The networking used for Tanzu Kubernetes clusters provisioned by the Tanzu Kubernetes Grid Service is a combination of the fabric that underlies the vSphere with Tanzu infrastructure and open-source software that provides networking for cluster pods, services, and ingress.

Read the following topics next:

- [Supervisor Cluster Networking](#)
- [Tanzu Kubernetes Grid Service Cluster Networking](#)
- [Configuring NSX for vSphere with Tanzu](#)
- [Configuring vSphere Networking and NSX Advanced Load Balancer for vSphere with Tanzu](#)
- [Configuring vSphere Networking and HAProxy Load Balancer for vSphere with Tanzu](#)

Supervisor Cluster Networking

In a vSphere with Tanzu environment, a Supervisor Cluster can either use the vSphere networking stack or VMware NSX® to provide connectivity to Kubernetes control plane VMs, services, and workloads. When a Supervisor Cluster is configured with the vSphere networking stack, all hosts from the cluster are connected to a vSphere Distributed Switch that provides connectivity to Kubernetes workloads and control plane VMs. A Supervisor Cluster that uses the vSphere networking stack requires a load balancer on the vCenter Server management network to provide connectivity to DevOps users and external services. A Supervisor Cluster that is configured with VMware NSX®, uses the software-based networks of the solution as well as an NSX Edge load balancer to provide connectivity to external services and DevOps users.

Supervisor Cluster Networking with NSX

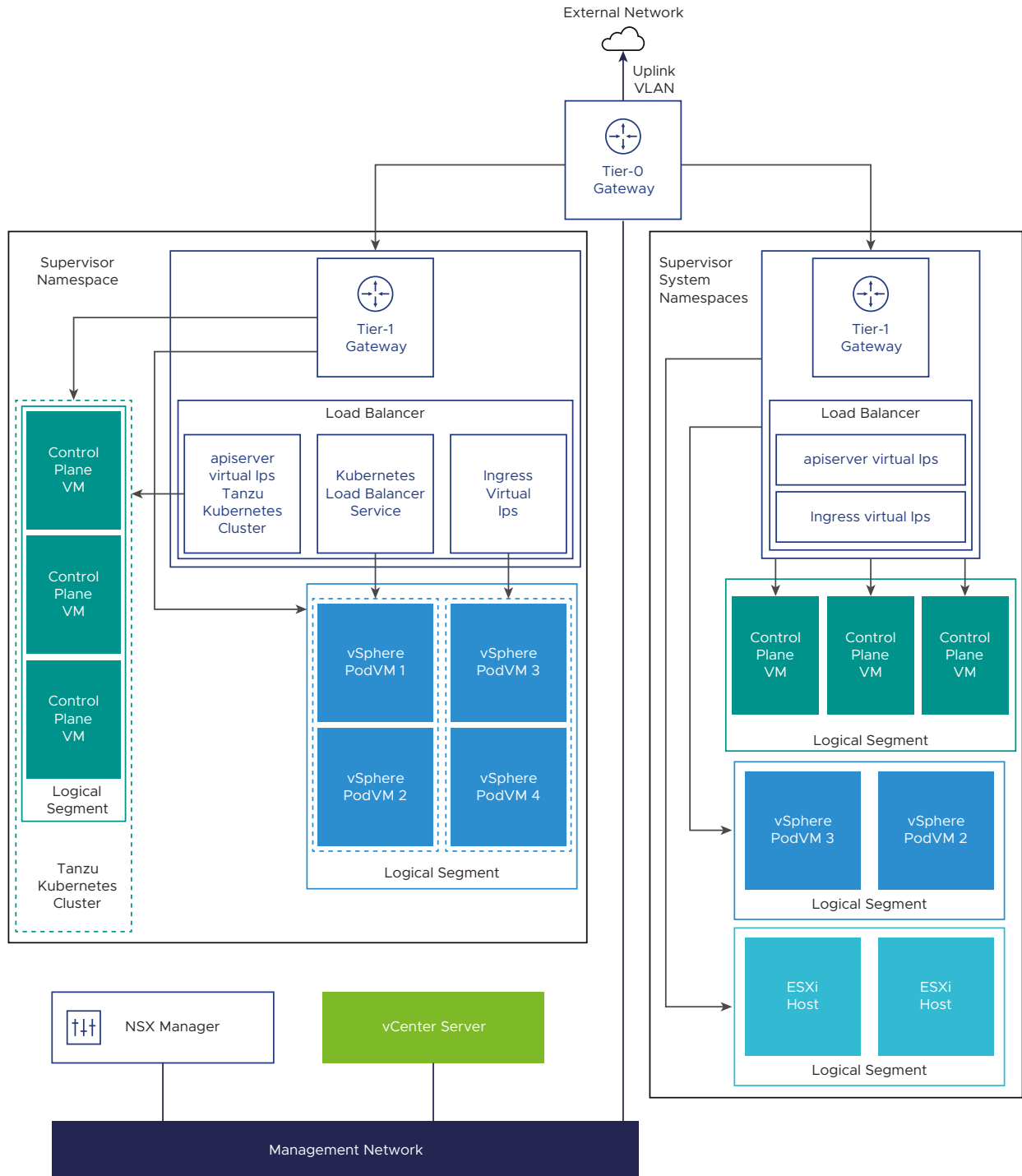
VMware NSX® provides network connectivity to the objects inside the Supervisor Cluster and external networks. Connectivity to the ESXi hosts comprising the cluster is handled by the standard vSphere networks.

You can also configure the Supervisor Cluster networking manually by using an existing NSX deployment or by deploying a new instance of NSX.

See the [VMware Product Interoperability Matrix](#) for information about compatibility of vCenter Server and VMware NSX®.

This section describes the networking topology when you install and configure vSphere with Tanzu Version 7.0 Update 2. For information on the upgrade when you upgrade from vSphere with Tanzu Version 7.0 Update 1 to Version 7.0 Update 2, see [Network Topology Upgrade](#) .

Figure 4-1. Supervisor Cluster Networking



- **NSX Container Plugin (NCP)** provides integration between NSX and Kubernetes. The main component of NCP runs in a container and communicates with NSX Manager and with the Kubernetes control plane. NCP monitors changes to containers and other resources and manages networking resources such as logical ports, segments, routers, and security groups for the containers by calling the NSX API.

The NCP creates one shared tier-1 gateway for system namespaces and a tier-1 gateway and load balancer for each namespace, by default. The tier-1 gateway is connected to the tier-0 gateway and a default segment.

System namespaces are namespaces that are used by the core components that are integral to functioning of the supervisor cluster and Tanzu Kubernetes. The shared network resources that include the tier-1 gateway, load balancer, and SNAT IP are grouped in a system namespace.

- NSX Edge provides connectivity from external networks to Supervisor Cluster objects. The NSX Edge cluster has a load balancer that provides a redundancy to the Kubernetes API servers residing on the control plane VMs and any application that must be published and be accessible from outside the Supervisor Cluster.
- A tier-0 gateway is associated with the NSX Edge cluster to provide routing to the external network. The uplink interface uses either the dynamic routing protocol, BGP, or static routing.
- Each vSphere Namespace has a separate network and set of networking resources shared by applications inside the namespace such as, tier-1 gateway, load balancer service, and SNAT IP address.
- Workloads running in Sphere Pods, regular VMs, or Tanzu Kubernetes clusters, that are in the same namespace, share a same SNAT IP for North-South connectivity.
- Workloads running in Sphere Pods or Tanzu Kubernetes clusters will have the same isolation rule that is implemented by the default firewall.
- A separate SNAT IP is not required for each Kubernetes namespace. East west connectivity between namespaces will be no SNAT.
- The segments for each namespace reside on the vSphere Distributed Switch (VDS) functioning in Standard mode that is associated with the NSX Edge cluster. The segment provides an overlay network to the Supervisor Cluster.
- Supervisor clusters have separate segments within the shared tier-1 gateway. For each Tanzu Kubernetes cluster, segments are defined within the tier-1 gateway of the namespace.
- The Spherelet processes on each ESXi hosts communicate with vCenter Server through an interface on the Management Network.

To learn more about Supervisor Cluster networking, watch the video [vSphere 7 with Kubernetes Network Service - Part 1 - The Supervisor Cluster](#).

Networking Configuration Methods with NSX

The Supervisor Cluster uses an opinionated networking configuration. Two methods exist to configure the Supervisor Cluster networking with NSX that result in deploying the same networking model:

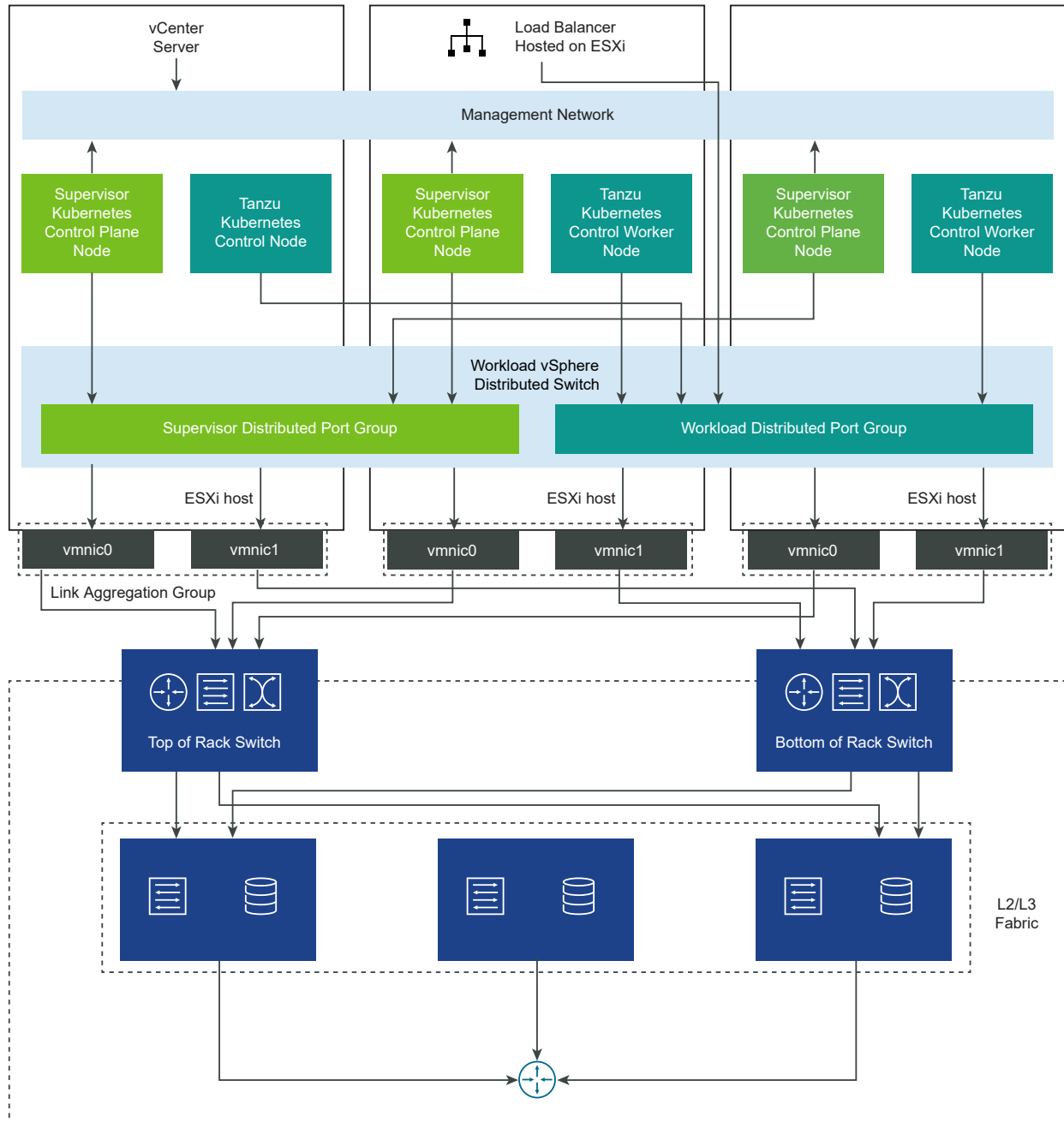
- The simplest way to configure the Supervisor Cluster networking is by using the VMware Cloud Foundation SDDC Manager. For more information, see the VMware Cloud Foundation SDDC Manager documentation. For information, see [Working with Workload Management](#).

- You can also configure the Supervisor Cluster networking manually by using an existing NSX deployment or by deploying a new instance of NSX. See [Install and Configure NSX for vSphere with Tanzu](#) for more information.

Supervisor Cluster Networking with vSphere Distributed Switch

A Supervisor Cluster that is backed by a vSphere Distributed Switch uses distributed port groups as Workload Networks for namespaces.

Figure 4-2. Namespace Networking with vSphere Distributed Switch



Depending on the topology that you implement for the Supervisor Cluster, you can use one or more distributed port groups as Workload Networks. The network that provides connectivity to the Kubernetes Control Plane VMs is called Primary Workload Network. You can assign this network to all the namespaces on the Supervisor Cluster, or you can use different networks for each namespace. The Tanzu Kubernetes clusters connect to the Workload Network that is assigned to the namespace where the clusters reside.

A Supervisor Cluster that is backed by a vSphere Distributed Switch uses a load balancer for providing connectivity to DevOps users and external services. You can use the NSX Advanced Load Balancer or the HAProxy load balancer.

For more information, see [Configuring vSphere Networking and NSX Advanced Load Balancer for vSphere with Tanzu](#) and [Install and Configure the HAProxy Load Balancer](#).

Tanzu Kubernetes Grid Service Cluster Networking

A Tanzu Kubernetes Grid Service cluster provisioned by the Tanzu Kubernetes Grid Service supports two CNI options: Antrea (default) and Calico. Both are open-source software that provide networking for cluster pods, services, and ingress.

Tanzu Kubernetes Grid Service clusters provisioned by the Tanzu Kubernetes Grid Service support the following [Container Network Interface](#) (CNI) options:

- [Antrea](#)
- [Calico](#)

Antrea is the default CNI for new Tanzu Kubernetes Grid Service clusters. If you are using Antrea, you do not have to specify it as the CNI during cluster provisioning. To use Calico as the CNI you have two options:

- Specify the CNI directly in the cluster YAML. See [Examples for Provisioning Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API](#).
- Change the default CNI. See [Examples for Configuring the Tanzu Kubernetes Grid Service v1alpha1 API](#).

Note The use of Antrea as the default CNI requires a minimum version of the OVA file for Tanzu Kubernetes Grid Service clusters. See [Verify Tanzu Kubernetes Cluster Compatibility for Update](#).

The table summarizes Tanzu Kubernetes Grid Service cluster networking features and their implementation.

Table 4-1. Tanzu Kubernetes Grid Service Cluster Networking

Endpoint	Provider	Description
Pod connectivity	Antrea or Calico	Container network interface for pods. Antrea uses Open vSwitch. Calico uses the Linux bridge with BGP.
Service type: ClusterIP	Antrea or Calico	Default Kubernetes service type that is only accessible from within the cluster.
Service type: NodePort	Antrea or Calico	Allows external access through a port opened on each worker node by the Kubernetes network proxy.
Service type: LoadBalancer	NSX-T load balancer, NSX Advanced Load Balancer, HAProxy	For NSX-T, one virtual server per service type definition. For NSX Advanced Load Balancer, refer to that section of this documentation. Note Some load balancing features may not be available with HAProxy, such as support for static IPs.
Cluster ingress	Third-party ingress controller	Routing for inbound pod traffic; you can use any third-party ingress controller , such as Contour .
Network policy	Antrea or Calico	Controls what traffic is allowed to and from selected pods and network endpoints. Antrea uses Open vSwitch. Calico uses Linux IP tables.

Configuring NSX for vSphere with Tanzu

vSphere with Tanzu requires specific networking configuration to enable connectivity to the Supervisor Clusters, vSphere Namespaces, and all objects that run inside the namespaces, such as vSphere Pods, VMs, and Tanzu Kubernetes clusters. As a vSphere administrator, install and configure NSX for vSphere with Tanzu.

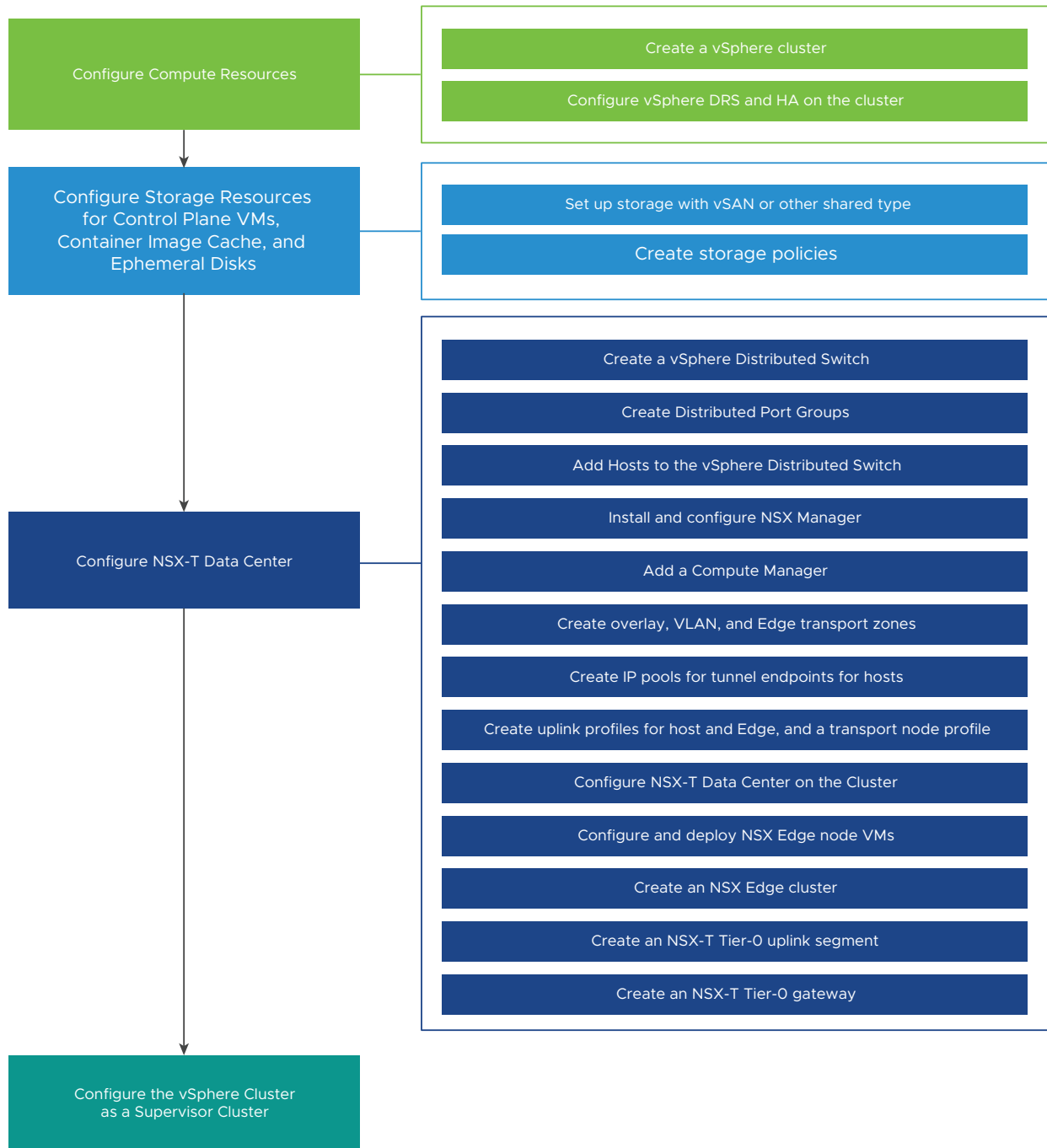
The Supervisor Cluster uses an opinionated networking configuration. Two methods exist to configure the Supervisor Cluster networking that result in deploying the same networking model:

- The simplest way to configure the Supervisor Cluster networking is by using the VMware Cloud Foundation SDDC Manager. For more information, see the VMware Cloud Foundation SDDC Manager documentation. For more information, see [Working with Workload Management](#).
- You can also configure the Supervisor Cluster networking manually by using an existing NSX deployment or by deploying a new instance of NSX.

Supervisor Cluster with NSX-T Data Center Workflow

As a vSphere administrator, you can configure a vSphere cluster as a Supervisor Cluster with the vSphere networking stack.

Figure 4-3. Supervisor Cluster with NSX-Data Center Networking Workflow



What to read next

Procedure

1 System Requirements for Setting Up vSphere with Tanzu with NSX

Review the system requirements for configuring vSphere with Tanzu on a vSphere cluster by using the NSX networking stack.

2 Topologies for a Supervisor Cluster with NSX

You can apply different topologies to your cluster depending on the needs of your Kubernetes workloads and the underlying networking infrastructure.

3 Best Practice Considerations for Configuring the Supervisor Cluster with NSX

Consider these best practices when you configure a vSphere cluster as a Supervisor Cluster using NSX.

4 Install and Configure NSX for vSphere with Tanzu

vSphere with Tanzu requires specific networking configuration to enable connectivity to the Supervisor Clusters, vSphere Namespaces, and all objects that run inside the namespaces, such as vSphere Pods, VMs, and Tanzu Kubernetes clusters. As a vSphere administrator, install and configure NSX for vSphere with Tanzu.

System Requirements for Setting Up vSphere with Tanzu with NSX

Review the system requirements for configuring vSphere with Tanzu on a vSphere cluster by using the NSX networking stack.

Configuration Limits for vSphere with Tanzu Clusters

VMware provides configuration limits in the [VMware Configuration Maximums](#) tool.

For configuration limits specific to vSphere with Tanzu, including Supervisor Clusters and Tanzu Kubernetes clusters, select **vSphere > vSphere 7.0 > vSphere with Kubernetes > VMware Tanzu Kubernetes Grid Service for vSphere** and click **View Limits**, or follow [this link](#).

Minimum Compute Requirements for the Management and Edge Cluster

You can deploy vSphere with Tanzu in two clusters, one cluster for the Management and Edge functions, and another one dedicated to Workload Management.

System	Minimum Deployment Size	CPU	Memory	Storage
vCenter Server 7.0	Small	2	16 GB	290 GB
ESXi hosts 7.0	2 ESXi hosts	8	64 GB per host	Not applicable
NSX Manager	Medium	6	24 GB	300 GB
NSX Edge 1	Large	8	32 GB	200 GB
NSX Edge 2	Large	8	32 GB	200 GB

Note Verify that all ESXi hosts participating in the vSphere Cluster on which you plan to configure vSphere with Tanzu are prepared as NSX transport nodes. For more information, see <https://kb.vmware.com/s/article/95820> and [Prepare ESXi Cluster Hosts as Transport Nodes](#) in the NSX documentation.

Minimum Compute Requirements for the Workload Domain Cluster

System	Minimum Deployment Size	CPU	Memory	Storage
ESXi hosts 7.0	3 ESXi hosts with 1 static IP per host. If you are using vSAN: 3 ESXi hosts with at least 2 physical NICs per host is the minimum; however, 4 ESXi hosts are recommended for resiliency during patching and upgrading. The hosts must be joined in a cluster with vSphere DRS and HA enabled. vSphere DRS must be in Fully Automated mode. Caution Do not disable vSphere DRS after you configure the Supervisor Cluster. Having DRS enabled at all times is a mandatory prerequisite for running workloads on the Supervisor Cluster. Disabling DRS leads to breaking your Tanzu Kubernetes clusters.	8	64 GB per host	Not applicable
Kubernetes control plane VMs	3	4	16 GB	16 GB

Note Verify that all ESXi hosts participating in the vSphere Cluster on which you plan to configure vSphere with Tanzu are prepared as NSX transport nodes. For more information, see <https://kb.vmware.com/s/article/95820> and [Prepare ESXi Cluster Hosts as Transport Nodes](#) in the NSX documentation.

Networking Requirements

No matter of the topology that you implement for Kubernetes workload management in vSphere, your deployment must meet the networking requirements listed in the table below.

Note You cannot create IPv6 clusters with a vSphere 7 Supervisor Cluster, or register IPv6 clusters with Tanzu Mission Control.

Component	Minimum Quantity	Required Configuration
Physical NIC	At least 2 physical NICs per host if vSAN is used	To use Antrea CNI and for optimal NSX performance, each physical NIC on each participating ESXi host must support GENEVE encapsulation and have it enabled.
Static IPs for Kubernetes control plane VMs	Block of 5	A block of 5 consecutive static IP addresses to be assigned to the Kubernetes control plane VMs in the Supervisor Cluster.
Management traffic network	1	A Management Network that is routable to the ESXi hosts, vCenter Server, and a DHCP server. The network must be able to access a container registry and have Internet connectivity if the container registry is on the external network. The container registry must be resolvable through DNS, and the Egress setting described below must be able to reach it.
NTP and DNS Server	1	A DNS server and NTP server that can be used for the vCenter Server. Note Configure NTP on all ESXi hosts, vCenter Server systems, and NSX Manager instances.
DHCP Server	1	Optional. Configure a DHCP server to automatically acquire IP addresses for the management. The DHCP server must support Client Identifiers and provide compatible DNS servers, DNS search domains, and an NTP server.
Image Registry	1	Access to a registry for service.

Component	Minimum Quantity	Required Configuration
Management Network Subnet	1	The subnet used for management traffic between ESXi hosts and vCenter Server, NSX Appliances, and the Kubernetes control plane. The size of the subnet must be the following: ■ One IP address per host VMkernel adapter. ■ One IP address for the vCenter Server Appliance. ■ One or four IP addresses for NSX Manager. Four when performing NSX Manager clustering of 3 nodes and 1 virtual IP (VIP). ■ 5 IP addresses for the Kubernetes control plane. 1 for each of the 3 nodes, 1 for virtual IP, 1 for rolling cluster upgrade. Note The Management Network and the Workload Network must be on different subnets. Assigning the same subnet to the Management and the Workload networks is not supported and can lead to system errors and problems.
Management Network VLAN	1	The VLAN ID of the Management Network subnet.
VLANs	3	These VLAN IPs are the IP addresses for the tunnel endpoints (TEPs). The ESXi host TEPs and the Edge TEPs must be routable. VLAN IP addresses are required for the following: ■ ESXi Host VTEP ■ Edge VTEP using the static IP ■ Tier 0 gateway and uplink for transport node. Note The ESXi Host VTEP and the Edge VTEP must have MTU size greater than 1600. ESXi hosts and NSX-T Edge nodes act as tunnel end points, and a Tunnel End Point (TEP) IP is assigned to each host and Edge node. As the TEP IPs for ESXi hosts create an overlay tunnel with TEP IPs on the Edge nodes, the VLAN IPs should be routable. An additional VLAN is required to provide North-South connectivity to Tier-0 gateway. IP pools can be shared across clusters. However, host overlay IP pool/VLAN must not be shared with Edge overlay IP pool/VLAN. Note If host TEP and Edge TEP are using different physical NICs, they can use the same VLAN.

Component	Minimum Quantity	Required Configuration
Tier-0 Uplink IP	/24 Private IP addresses	The IP subnet used for the Tie-0 uplink. The requirements for the IP address of the Tier-0 uplink are as follows: ■ 1 IP, if you do not use Edge redundancy. ■ 4 IPs, if you use BGP and Edge redundancy, 2 IP addresses per Edge. ■ 3 IPs, if you use static routes and Edge redundancy. The Edge Management IP, subnet, gateway, Uplink IP, subnet, gateway must be unique.
Physical Network MTU	1600	The MTU size must be 1600 or greater on any network that carries overlay traffic.
vSphere Pod CIDR range	/23 Private IP addresses	A private CIDR range that provides IP addresses for vSphere Pods. These addresses are also used for the Tanzu Kubernetes cluster nodes. You must specify a unique vSphere Pod CIDR range for each cluster. Note The vSphere Pod CIDR range and the CIDR range for the Kubernetes service addresses must not overlap.
Kubernetes services CIDR range	/16 Private IP addresses	A private CIDR range to assign IP addresses to Kubernetes services. You must specify a unique Kubernetes services CIDR range for each Supervisor Cluster.
Egress CIDR range	/27 Static IP Addresses	A private CIDR annotation to determine the egress IP for Kubernetes services. Only one egress IP address is assigned for each namespace in the Supervisor Cluster. The egress IP is the address that external entities use to communicate with the services in the namespace. The number of egress IP addresses limits the number of egress policies the Supervisor Cluster can have. The minimum is a CIDR of /27 or more. For example, 10.174.4.96/27 Note Egress IP addresses and ingress IP addresses must not overlap.
Ingress CIDR	/27 Static IP Addresses	A private CIDR range to be used for IP addresses of ingresses. Ingress lets you apply traffic policies to requests entering the Supervisor Cluster from external networks. The number of ingress IP addresses limits the number of ingresses the cluster can have. The minimum is a CIDR of /27 or more. Note Egress IP addresses and ingress IP addresses must not overlap.

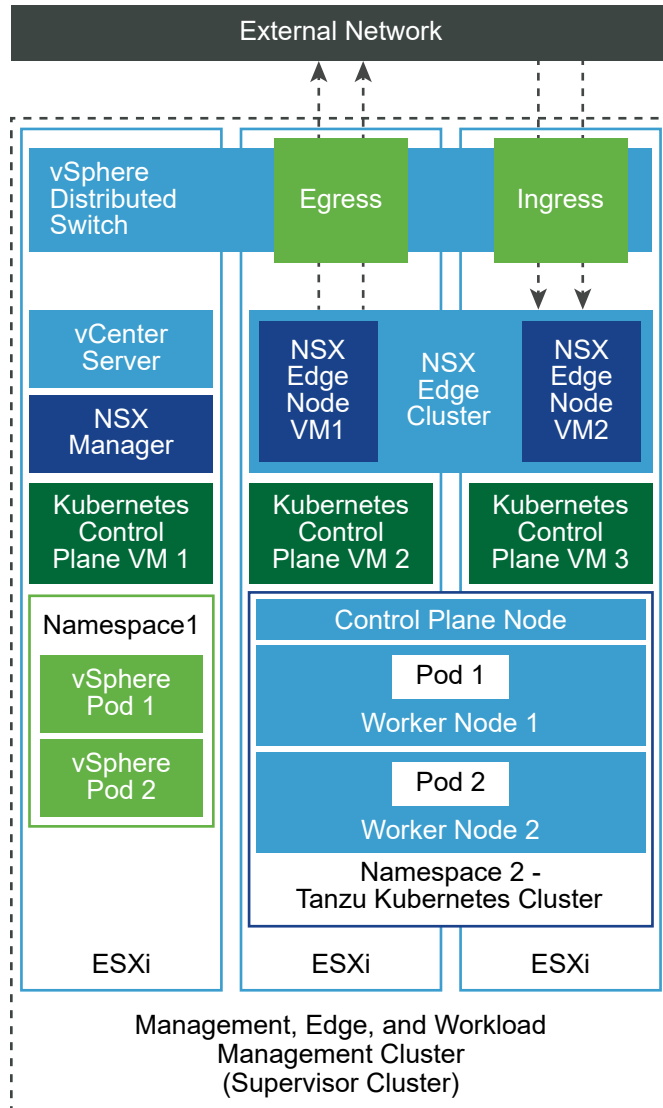
Topologies for a Supervisor Cluster with NSX

You can apply different topologies to your cluster depending on the needs of your Kubernetes workloads and the underlying networking infrastructure.

Topology for a Management, Edge, and Workload Domain Cluster

You can deploy vSphere with Tanzu with combined management, Edge, and workload management functions on a single vSphere cluster.

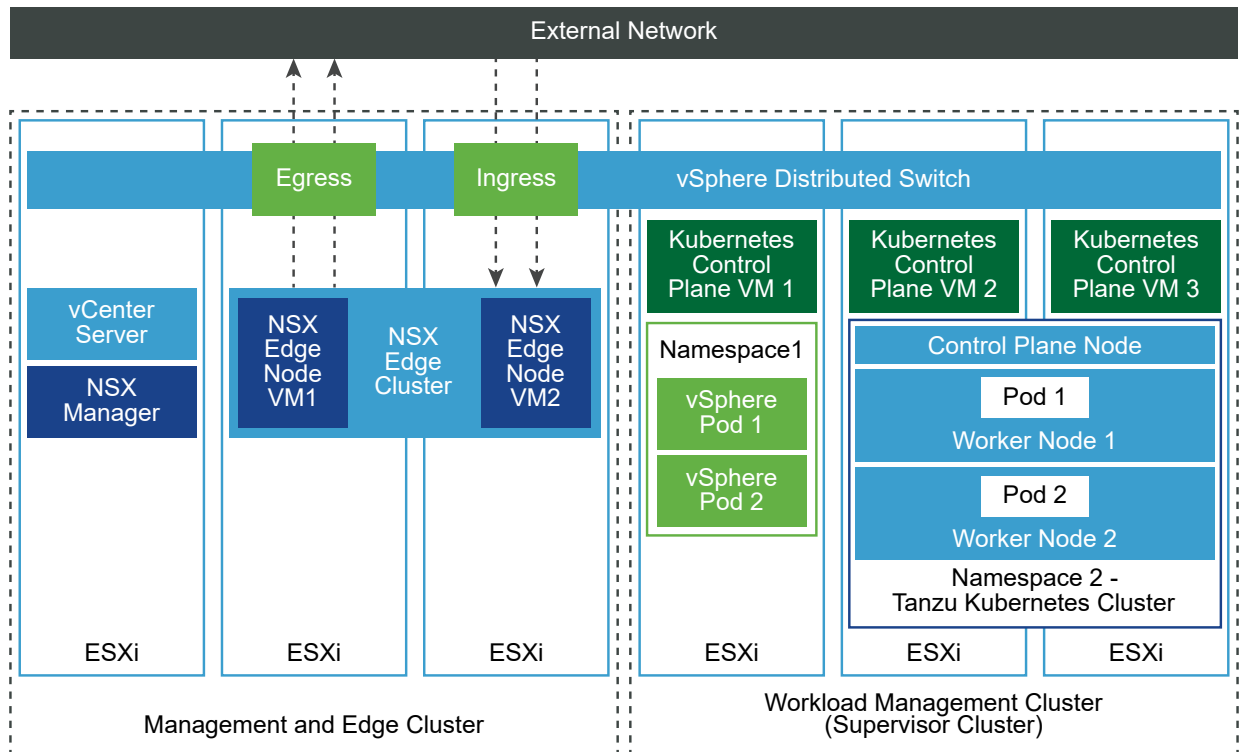
Figure 4-4. Management, Edge, and Workload Management Cluster



Topology with Separate Management and Edge Cluster and Workload Management Cluster

You can deploy vSphere with Tanzu in two clusters, one cluster for the Management and Edge functions, and another one dedicated to Workload Management.

Figure 4-5. Management and Edge and Workload Management Clusters



Best Practice Considerations for Configuring the Supervisor Cluster with NSX

Consider these best practices when you configure a vSphere cluster as a Supervisor Cluster using NSX.

- Use vSAN datastore for the NSX Edges.
- If you use a vSAN datastore, ensure that the vSAN environment is properly sized for the workloads. The vSAN setup requires more memory as the kernel memory is used for vSAN. This reduces the memory available to the NSX Edge VMs. Use the vSAN calculator for appropriate sizing. For more information, see [vSAN ReadyNode Sizer](#).
- If you use NFS datastore, verify that it is shared across all hosts in the cluster. Create a unique NFS datastore for each NSX Edge node.
- Configure a dedicated resource pool for each NSX Edge cluster. Do not share the resource pool with other VMs.
- When you configure ESXi host overlay, use VLANs in the range 1-4094.
- When you configure Edge overlay, use VLANs in the range 1-4094.

Install and Configure NSX for vSphere with Tanzu

vSphere with Tanzu requires specific networking configuration to enable connectivity to the Supervisor Clusters, vSphere Namespaces, and all objects that run inside the namespaces, such

as vSphere Pods, VMs, and Tanzu Kubernetes clusters. As a vSphere administrator, install and configure NSX for vSphere with Tanzu.

This section describes how to configure the Supervisor Cluster networking by deploying a new NSX instance, but the procedures are applicable against an existing NSX deployment as well. This section also provides background to understand what VMware Cloud Foundation SDDC Manager is doing when it sets up the Supervisor Cluster workload domain.

The procedures for installing and configuring NSX, and the configuration settings in this guide are relevant and specific to vSphere with Tanzu. For more information on all available configuration settings for NSX, see the NSX documentation.

Check the VMware Interoperability matrix <https://interopmatrix.vmware.com/Interoperability> to determine compatibility for vCenter Server and NSX.

Prerequisites

- Verify that your environment meets the system requirements for configuring a vSphere cluster as a Supervisor Cluster. For information about requirements, see [System Requirements for Setting Up vSphere with Tanzu with NSX](#).
- Assign the VMware vSphere 7 Enterprise Plus with Add-on for Kubernetes license to all ESXi hosts that will be part of the Supervisor Cluster.
- Create storage policies for the placement of control plane VMs, pod ephemeral disks, and container images.
- Configure shared storage for the cluster. Shared storage is required for vSphere DRS, HA, and storing persistent volumes of containers.
- Verify that DRS and HA is enabled on the vSphere cluster, and DRS is in the fully automated mode.
- Verify that you have the **Modify cluster-wide configuration** privilege on the cluster.

Create a vSphere Distributed Switch

To handle the networking configuration for all hosts in the Supervisor Cluster, create a vSphere Distributed Switch.

Procedure

- 1 In the vSphere Client, navigate to a data center.
- 2 In the navigator, right-click the data center and select **Distributed Switch > New Distributed Switch**.
- 3 Enter a name for the new distributed switch.
For example, `DSwitch`.
- 4 In **Select Version**, enter a version for the distributed switch.
Select **7.0.0 - ESXi 7.0 and later**.

- 5 In **Configure Settings**, enter the number of uplinks ports.
Enter a value of 2.
- 6 Review the settings and click **Finish**.
- 7 Right-click the distributed switch you created and select **Settings > Edit settings**.
- 8 On the **Advanced** tab, enter a value of more than 1600 as the MTU (Bytes) value and click **OK**.
The MTU size must be 1600 or greater on any network that carries overlay traffic.
For example, 9000.

What to do next

Add distributed port groups. See [Create Distributed Port Groups](#).

Create Distributed Port Groups

Create distributed port groups for each NSX Edge node uplink, Edge node TEP, management network, and shared storage.

Prerequisites

Verify that you have created a vSphere Distributed Switch.

Procedure

- 1 In the vSphere Client, navigate to a data center.
- 2 In the navigator, right-click the distributed switch and select **Distributed Port Group > New Distributed Port Group**.
- 3 Create a port group for the NSX Edge uplink.
For example, `DPortGroup-EDGE-UPLINK`.
- 4 Configure **VLAN Type** as VLAN Trunking.
- 5 Right-click the distributed switch and from the **Actions** menu, select **Distributed Port Group > Manage Distributed Port Groups**.
- 6 Select **Teaming and failover** and click **Next**.
- 7 Configure active and standby uplinks.
For example, active uplink is `Uplink1` and standby uplink is `Uplink2`.
- 8 Repeat steps 4-7 for the Edge node TEP, management network, and shared storage.
For example, create the following port groups:

Port group	Name	VLAN Type
Edge node TEP	DPortGroup-EDGE-TEP	Configure VLAN Type as VLAN Trunking. Configure the active uplink as Uplink2 and the standby uplink as Uplink1. Note The VLAN used for the Edge nodes TEP must be different than the VLAN used for ESXi TEP.
Management	DPortGroup-MGMT	Configure VLAN Type as VLAN and enter the VLAN ID of the management network. For example, 1060.
Shared storage or VSAN	DPortGroup-VSAN	Configure VLAN Type as VLAN and enter the VLAN ID. For example, 3082.

9 (Optional) Create port groups for the following components:

- vSphere vMotion
- VM traffic

What to do next

Add hosts to the vSphere Distributed Switch. See [Add Hosts to the vSphere Distributed Switch](#).

Add Hosts to the vSphere Distributed Switch

To manage the networking of your environment by using the vSphere Distributed Switch, you must associate hosts with the switch. Connect the physical NICs, VMkernel adapters, and virtual machine network adapters of the hosts to the distributed switch.

Prerequisites

- Verify that enough uplinks are available on the distributed switch to assign to the physical NICs that you want to connect to the switch.
- Verify that at least one distributed port group is available on the distributed switch.
- Verify that the distributed port group has active uplinks configured in its teaming and failover policy.

Procedure

- 1 In the vSphere Client, select **Networking** and navigate to the distributed switch.
- 2 From the **Actions** menu, select **Add and Manage Hosts**.
- 3 On the **Select task** page, select **Add hosts**, and click **Next**.

- 4 On the **Select hosts** page, click **New hosts**, select the hosts in your data center, click **OK**, and then click **Next**.
- 5 On the **Manage physical adapters** page, configure physical NICs on the distributed switch.
 - a From the **On other switches/unclaimed** list, select a physical NIC.
If you select physical NICs that are already connected to other switches, they are migrated to the current distributed switch.
 - b Click **Assign uplink**.
 - c Select an uplink.
 - d To assign the uplink to all the hosts in the cluster, select **Apply this uplink assignment to the rest of the hosts**.
 - e Click **OK**.
For example, assign Uplink 1 to vmnic0 and Uplink 2 to vmnic1.
- 6 Click **Next**.
- 7 On the **Manage VMkernel adapters** page, configure VMkernel adapters.
 - a Select a VMkernel adapter and click **Assign port group**.
 - b Select a distributed port group.
For example, **DPortGroup**.
 - c To apply the port group to all hosts in the cluster, select **Apply this port group assignment to the rest of the hosts**.
 - d Click **OK**.
- 8 Click **Next**.
- 9 (Optional) On the **Migrate VM networking** page, select the **Migrate virtual machine networking** check box to configure virtual machine networking.
 - a To connect all network adapters of a virtual machine to a distributed port group, select the virtual machine, or select an individual network adapter to connect only that adapter.
 - b Click **Assign port group**.
 - c Select a distributed port group from the list and click **OK**.
 - d Click **Next**.

What to do next

Deploy and configure NSX Manager. See [Deploy and Configure NSX Manager](#)

Deploy and Configure NSX Manager

You can use the vSphere Client to deploy the NSX Manager to the vSphere cluster and use it with vSphere with Tanzu.

To deploy the NSX Manager using the OVA file, perform the steps in this procedure.

For information about deploying the NSX Manager through the user interface or CLI, see the *NSX-T Data Center Installation Guide*.

Prerequisites

- Verify that your environment meets the networking requirements. See [System Requirements for Setting Up vSphere with Tanzu with NSX](#) for more details.
- Verify that the required ports are open. For information about port and protocols, see the *NSX Installation Guide*.

Procedure

- 1 Locate the NSX OVA file on the VMware download portal.
Either copy the download URL or download the OVA file.
- 2 Right-click and select **Deploy OVF template** to start the installation wizard.
- 3 In the **Select an OVF template** tab, enter the download OVA URL or navigate to the OVA file.
- 4 In the **Select a name and folder** tab, enter a name for the NSX Manager virtual machine (VM).
- 5 In the **Select a compute resource** tab, select the vSphere cluster on which to deploy the NSX Manager.
- 6 Click **Next** to review details.
- 7 In the **Configuration tab**, select the NSX deployment size.
The recommended minimum deployment size is Medium.
- 8 In the **Select storage** tab, select the shared storage for deployment.
- 9 Enable thin provisioning by selecting the **Thin Provision** in **Select virtual disk format**.
The virtual disks are thick provisioned by default.
- 10 In the **Select networks** tab, select the management port group or destination network for the NSX Manager in **Destination Network**.
For example, DPortGroup-MGMT.
- 11 In the **Customize template** tab, enter the system root, CLI admin, and audit passwords for the NSX Manager. Your passwords must comply with the password strength restrictions.
 - At least 12 characters.
 - At least one lower-case letter.
 - At least one upper-case letter.
 - At least one digit.
 - At least one special character.
 - At least five different characters.

- Default password complexity rules are enforced by the Linux PAM module.
- 12 Enter the default IPv4 gateway, management network IPv4, management network netmask, DNS server, domain search list, and NTP IP address.
- 13 Enable SSH and allow root SSH login to the NSX Manager command line.
By default, the SSH options are disabled for security reasons.
- 14 Verify that your custom OVF template specification is accurate, and click **Finish** to initiate the installation.
- 15 After the NSX Manager boots, log in to the CLI as admin and run the `get interface eth0` command to verify that the IP address was applied as expected.
- 16 Enter the `get services` command to verify that all the services are running.

Deploy NSX Manager Nodes to Form a Cluster

An NSX Manager cluster provides high availability. You can deploy NSX Manager nodes using the user interface only on ESXi hosts managed by vCenter Server. To create an NSX Manager cluster, deploy two additional nodes to form a cluster of three nodes total. When you deploy a new node from the UI, the node connects to the first deployed node to form a cluster. All the repository details and the password of the first deployed node are synchronized with the newly deployed node.

Prerequisites

- Verify that an NSX Manager node is installed.
- Verify that a compute manager is configured.
- Verify that the required ports are open.
- Verify that a datastore is configured on the ESXi host.
- Verify that you have the IP address and gateway, DNS server IP addresses, domain search list, and the NTP server IP address for the NSX Manager to use.
- Verify that you have a target VM port group network. Place the NSX appliances on a management VM network.

Procedure

- 1 From a browser, log in with admin privileges to the NSX Manager at `https://<manager-ip-address>`.
- 2 To deploy an appliance, select **System > Appliances > Add NSX Appliance**.
- 3 Enter the appliance details.

Option	Description
Hostname	Enter the host name or FQDN to use for the node.
Management IP/Netmask	Enter an IP address to be assigned to the node.

Option	Description
Management Gateway	Enter a gateway IP address to be used by the node.
DNS servers	Enter the list of DNS server IP addresses to be used by the node.
NTP server	Enter the list of NTP server IP addresses
Node Size	Select Medium (6 vCPU, 24 GB RAM, 300 GB storage) form factor from the options.

4 Enter the appliance configuration details

Option	Description
Compute Manager	Select the vCenter Server that you configured as compute manager.
Compute Cluster	Select the cluster that the node must join.
Datastore	Select a datastore for the node files.
Virtual Disk Format	Select Thin Provision format.
Network	Click Select Network to select the management network for the node.

5 Enter the access and credentials details.

Option	Description
Enable SSH	Toggle the button to allow SSH login to the new node.
Enable Root Access	Toggle the button to allow root access to the new node.
System Root Credentials	Set and confirm the root password for the new node. Your password must comply with the password strength restrictions. ■ At least 12 characters. ■ At least one lower-case letter. ■ At least one upper-case letter. ■ At least one digit. ■ At least one special character. ■ At least five different characters. ■ Default password complexity rules are enforced by the Linux PAM module.
Admin CLI Credentials and Audit CLI Credentials	Select the Same as root password check box to use the same password that you configured for root, or deselect the check box and set a different password.

6 Click **Install Appliance**.

The new node is deployed. You can track the deployment process in the **System> > Appliances** page. Do not add additional nodes until the installation is finished and the cluster is stable.

7 Wait for the deployment, cluster formation, and repository synchronization to finish.

The joining and cluster stabilizing process might take from 10 to 15 minutes. Verify that the status for every cluster service group is **UP** before making any other cluster changes.

- 8 After the node boots, log in to the CLI as admin and run the `get interface eth0` command to verify that the IP address was applied as expected.
- 9 If your cluster has only two nodes, add another appliance. Select **System > Appliances > Add NSX Appliance** and repeat the configuration steps.

Add a License

Add a license using the NSX Manager.

Prerequisites

Obtain an NSX Advanced or higher license.

Procedure

- 1 Log in to the NSX Manager.
- 2 Select **System > Licenses > Add**.
- 3 Enter the license key.
- 4 Click **Add**.

Add a Compute Manager

A compute manager is an application that manages resources such as hosts and virtual machines. Configure the vCenter Server that is associated with the NSX as a compute manager in the NSX Manager.

For more information see the *NSX Administration Guide*.

Procedure

- 1 Log in to the NSX Manager.
- 2 Select **System > Fabric > Compute Managers > Add**
- 3 Enter the compute manager details.

Option	Description
Name and Description	Enter the name and description of the vCenter Server.
Type	The default type is VMware vCenter.
Multi NSX	Leave this option unselected. Multi NSX option allows you to register the same vCenter Server with multiple NSX Managers. This option is not supported on Supervisor Cluster and vSphere Lifecycle Manager clusters.
FQDN or IP Address	Enter the FQDN or the IP address of the vCenter Server.
HTTPS Port of Reverse Proxy	The default port is 443. If you use another port, verify that the port is open on all the NSX Manager appliances. Set the reverse proxy port to register the compute manager in NSX.

Option	Description
User name and Password	Enter the vCenter Server login credentials.
SHA-256 Thumbprint	Enter the vCenter Server SHA-256 thumbprint algorithm value.

You can leave the defaults for the other settings.

If you left the thumbprint value blank, you are prompted to accept the server provided thumbprint. After you accept the thumbprint, it takes a few seconds for NSX to discover and register the vCenter resources.

- 4 Select **Enable Trust** to allow vCenter Server to communicate with NSX .
- 5 If you did not provide a thumbprint value for NSX Manager, the system identifies the thumbprint and displays it.
- 6 Click **Add** to accept the thumbprint.

Results

After some time, the compute manager is registered with vCenter Server and the connection status changes to **Up**. If the FQDN/PNID of vCenter Server changes, you must re-register it with the NSX Manager. For more information, see [Register vCenter Server with NSX Manager](#).

Note After the vCenter Server is successfully registered, do not power off and delete the NSX Manager VM without deleting the compute manager first. Otherwise, when you deploy a new NSX Manager, you will not be able to register the same vCenter Server again. You will get an error stating that the vCenter Server is already registered with another NSX Manager.

You can click the compute manager name to view the details, edit the compute manager, or to manage tags that apply to the compute manager.

Create Transport Zones

Transport zones indicate which hosts and VMs can use a particular network. A transport zone can span one or more host clusters.

As a vSphere administrator, you use the default transport zones or create the following ones:

- An overlay transport zone that is used by the Supervisor Cluster Control Plane VMs.
- A VLAN transport zone for the NSX Edge nodes to use for uplinks to the physical network.

Procedure

- 1 Log in to the NSX Manager.
- 2 Select **System > Fabric > Transport Zones > Add**.
- 3 Enter a name for the transport zone and optionally a description.
- 4 Select a traffic type.

You can select **Overlay** or **VLAN**.

The following transport zones exist by default:

- A VLAN transport zone with name `nsx-vlan-transportzone`.
- An overlay transport zone with name `nsx-overlay-transportzone`.

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

The segments attached to the transport zones use these named teaming policies. If the segments do not find a matching named teaming policy, then the default uplink teaming policy is used.

Results

The new transport zone appears on the **Transport Zones** page.

Create an IP Pool for Host Tunnel Endpoint IP Addresses

Create IP pools for the ESXi host tunnel endpoints (TEPs) and the Edge nodes. TEPs are the source and destination IP addresses used in the external IP header to identify the ESXi hosts that originate and end the NSX encapsulation of overlay frames. You can use DHCP or manually configured IP pools for TEP IP addresses.

Procedure

- 1 Log in to the NSX Manager.
- 2 Select **Networking > IP Address Pools > Add IP Address Pool**.
- 3 Enter the following IP pool details.

Option	Description
Name and Description	Enter the IP pool name and optional description. For example, <code>ESXI-TEP-IP-POOL</code> .
IP Ranges	Enter the IP allocation range. For example, <code>10.197.79.158 - 10.197.79.160</code>
Gateway	Enter the gateway IP address. For example, <code>10.197.79.253</code> .
CIDR	Enter the network address in a CIDR notation. For example, <code>10.197.79.0/24</code> .

- 4 Click **Add** and **Apply**.
- 5 Repeat steps 2 - 4 to create an IP pool for the Edge nodes.
For example, `EDGE-TEP-IP-POOL`.
- 6 Verify that the TEP IP pools you created are listed in the **IP Pool** page.

Create a Host Uplink Profile

A host uplink profile defines policies for the uplinks from the ESXi hosts to NSX segments.

Procedure

- 1 Log in to the NSX Manager.

- 2 Select **System > Fabric > Profiles > Uplink Profiles > Add**.

- 3 Enter an uplink profile name, and optionally, an uplink profile description.

For example, `ESXI-UPLINK-PROFILE`.

- 4 In the **Teaming** section, click **Add** to add a naming teaming policy, and configure a **Failover Order** policy.

A list of active uplinks is specified, and each interface on the transport node is pinned to one active uplink. This configuration allows use of several active uplinks at the same time.

- 5 Configure active and standby uplinks.

For example, configure `uplink-1` as the active uplink and `uplink-2` as the standby uplink.

- 6 Enter a transport VLAN value.

The transport VLAN set in the uplink profile tags overlay traffic and the VLAN ID is used by the tunnel endpoint (TEP).

For example, `1060`.

- 7 Enter the MTU value.

The default value for uplink profile MTU is 1600.

Note The value must be at least 1600 but not higher than the MTU value on the physical switches and the vSphere Distributed Switch.

Create an Edge Uplink Profile

Create an uplink profile with the failover order teaming policy with one active uplink for edge virtual machine overlay traffic.

Procedure

- 1 Log in to the NSX Manager.

- 2 Select **System > Fabric > Profiles > Uplink Profiles > Add**.

- 3 Enter an uplink profile name, and optionally, add an uplink profile description.

For example, `EDGE-UPLINK-PROFILE`.

- 4 In the **Teaming** section, click **Add** to add a naming teaming policy, and configure a **Failover** policy.

A list of active uplinks is listed, and each interface on the transport node is pinned to one active uplink. This configuration allows use of several active uplinks at the same time.

- 5 Configure an active uplinks.

For example, configure `uplink-1` as active uplink.

- 6 View the uplinks in the **Uplink Profile** page.

Create a Transport Node Profile

A transport node profile defines how NSX is installed and configured on the hosts in a particular cluster the profile is attached to.

Prerequisites

Verify that you have created an overlay transport zone.

Procedure

- 1 Log in to the NSX Manager.
- 2 Select **System > Fabric > Profiles > Transport Node Profiles > Add**.
- 3 Enter a name for the transport node profile and optionally a description.
For example, `HOST-TRANSPORT-NODE-PROFILE`.
- 4 In the **New Node Switch** section, Select **Type** as `VDS`.
- 5 Select **Mode** as `Standard`.
- 6 Select the vCenter Server and the Distributed Switch names from the list.
For example, `DSwitch`
- 7 Select the overlay transport zone created previously.
For example, `NSX-OVERLAY-TRANSPORTZONE`.
- 8 Select the host uplink profile created previously.
For example, `ESXI-UPLINK-PROFILE`.
- 9 Select **Use IP Pool** from the **IP Assignment** list.
- 10 Select the host TEP pool created previously.
For example, `ESXI-TEP-IP-POOL`.
- 11 In the **Teaming Policy Switch Mapping**, click the edit icon and map the uplinks defined in the NSX uplink profile with the vSphere Distributed Switch uplinks.
For example, map `uplink-1 (active)` to `Uplink 1` and `uplink-2 (standby)` to `Uplink 2`.
- 12 Click **Add**.
- 13 Verify that the profile that you created is listed in the **Transport Node Profiles** page.

Configure NSX on the Cluster

To install NSX and prepare the overlay TEPs, apply the transport node profile to the vSphere cluster.

Prerequisites

Verify that you have created a transport node profile.

Procedure

- 1 Log in to the NSX Manager.
- 2 Select **System > Fabric > Nodes > Host Transport Nodes**.
- 3 From the **Managed By** drop-down menu, select an existing vCenter Server.
The page lists the available vSphere clusters.
- 4 Select the compute cluster on which you want to configure NSX.
- 5 Click **Configure NSX**.
- 6 Select the transport node profile created previously and click **Apply**.
For example, `HOST-TRANSPORT-NODE-PROFILE`.
- 7 From the **Host Transport Node** page, verify that the NSX configuration state is `Success` and NSX Manager connectivity status of hosts in the cluster is `Up`.

Results

The transport node profile created previously is applied to the vSphere cluster to install NSX and prepare the overlay TEPs.

Configure and Deploy an NSX Edge Transport Node

You can add an NSX Edge virtual machine (VM) to the NSX fabric and proceed to configure it as an NSX Edge transport node VM.

Prerequisites

Verify that you have created transport zones, edge uplink profile, and edge TEP IP pool.

Procedure

- 1 Log in to the NSX Manager.
- 2 Select **System > Fabric > Nodes > Edge Transport Nodes > Add Edge VM**.
- 3 In **Name and Description**, enter a name for the NSX Edge.
For example, `nsx-edge-1`
- 4 Enter the host name or FQDN from vCenter Server.
For example, `nsx-edge-1.lab.com`.
- 5 Select `Large` form factor.

6 In **Credentials**, enter the CLI and the root passwords for the NSX Edge. Your passwords must comply with the password strength restrictions.

- At least 12 characters.
- At least one lower-case letter.
- At least one upper-case letter.
- At least one digit.
- At least one special character.
- At least five different characters.
- Default password complexity rules are enforced by the Linux PAM module.

7 Enable **Allow SSH Login** for CLI and Root credentials.

8 In **Configure Deployment**, configure the following properties:

Option	Description
Compute Manager	Select the compute manager from the drop-down menu. For example, select <code>vCenter</code> .
Cluster	Select the cluster from drop-down menu. For example, select <code>Compute-Cluster</code> .
Datastore	Select the shared datastore from the list. For example, <code>vsanDatastore</code> .

9 Configure the node settings.

Option	Description
IP Assignment	Select Static. Enter the values for: ■ Management IP: Enter the IP address on the same VLAN as the vCenter Server management network. For example, <code>10.197.79.146/24</code>. ■ Default gateway: The default gateway of the management network. For example, <code>10.197.79.253</code>.
Management Interface	Click Select interface , and select the vSphere Distributed Switch port group on the same VLAN as the management network from the drop-down menu that you created previously. For example, <code>DPortGroup-MGMT</code> .

10 In **Configure NSX**, click **Add Switch** to configure the switch properties.

11 Use the default name for the **Edge Switch Name**.

For example, `nvds1`.

- 12 Select the transport zone to which the transport node belongs.

Select the overlay transport zones created previously.

For example, `nsx-overlay-transportzone`.

- 13 Select the edge uplink profile created previously.

For example, `EDGE-UPLINK-PROFILE`.

- 14 Select **Use IP Pool** in **IP Assignment**.

- 15 Select the edge TEP IP pool created previously.

For example, `EDGE-TEP-IP-POOL`.

- 16 In the **Teaming Policy Switch Mapping** section, the uplink to the edge uplink profiles created previously.

For example, for `Uplink1`, select `DPortGroup-EDGE-TEP`.

- 17 Repeat steps 10-16, to add a new switch.

For example, configure the following values:

Property	Value
Edge Switch Name	<code>nvds2</code>
Transport Zone	<code>nsx-vlan-transportzone</code>
Edge uplink profile	<code>EDGE-UPLINK-PROFILE</code>
Teaming Policy Switch Mapping	<code>DPortGroup-EDGE-UPLINK</code>

- 18 Click **Finish**.

- 19 Repeat steps 2–18 for a second NSX Edge VM.

- 20 View the connection status on the **Edge Transport Nodes** page.

Create an NSX Edge Cluster

To ensure that at least one NSX Edge is always available, create an NSX Edge cluster.

Procedure

- 1 Log in to the NSX Manager.
- 2 Select **System > Fabric > Nodes > Edge Clusters > Add**.
- 3 Enter the NSX Edge cluster name.
For example, `EDGE-CLUSTER`.
- 4 Select the default NSX Edge cluster profile from the drop-down menu.
Select **nsx-default-edge-high-availability-profile**.
- 5 In **Member Type** drop-down menu, select the **Edge Node**.

- 6 From the **Available** column, select the NSX Edge VMs previously created, and click the right-arrow to move them to the **Selected** column.
- 7 For example, `nsx-edge-1` and `nsx-edge-2`.
- 8 Click **Save**.

Create a Tier-0 Uplink Segment

The tier-0 uplink segment provides the North-South connectivity from NSX to the physical infrastructure.

Prerequisites

Verify that you have created a Tier-0 gateway.

Procedure

- 1 Log in to the NSX Manager.
- 2 Select **Networking > Segments > Add Segment**.
- 3 Enter a name for the segment.
For example, `TIER-0-LS-UPLINK`.
- 4 Select the transport zone previously created.
For example, select `nsx-vlan-transportzone`.
- 5 Toggle the **Admin Status** to enable it.
- 6 Enter a VLAN ID of the Tier-0 gateway.
For example, `1089`.
- 7 Click **Save**.

Create a Tier-0 Gateway

The tier-0 gateway is the NSX logical router that provides the North-South connectivity for the NSX logical networking to the physical infrastructure. vSphere with Tanzu supports multiple tier-0 gateways on multiple NSX Edge clusters in the same transport zone.

A tier-0 gateway has downlink connections to tier-1 gateways and external connections to physical networks.

You can configure the HA (high availability) mode of a tier-0 gateway to be active-active or active-standby. The following services are only supported in active-standby mode:

- NAT
- Load balancing
- Stateful firewall
- VPN

Proxy ARP is automatically enabled on a tier-0 gateway when a NAT rule or a load balancer VIP uses an IP address from the subnet of the tier-0 gateway external interface. By enabling proxy-ARP, hosts on the overlay segments and hosts on a VLAN segment can exchange network traffic together without implementing any change in the physical networking fabric.

Before NSX 3.2, proxy ARP is supported on a tier-0 gateway in only an active-standby configuration. Starting in NSX 3.2, proxy ARP is also supported on a tier-0 gateway in an active-active configuration.

For more information, see the *NSX Administration Guide*.

Prerequisites

Verify that you have created an NSX Edge cluster.

Procedure

- 1 Log in to the NSX Manager.
- 2 Select **Networking > Tier-0 Gateways**.
- 3 Click **Add Tier-0 Gateway**.
- 4 Enter a name for the tier-0 gateway.

For example, `Tier-0_VWT`.

- 5 Select an active-standby HA mode.

In active-standby mode, the elected active member processes all traffic. If the active member fails, a new member is elected to be active.

- 6 Select the NSX Edge cluster previously created.

For example, select `EDGE-CLUSTER`.

- 7 Click **Save**.

The tier-0 gateway is created.

- 8 Select **Yes** to continue with the configuration.

- 9 Configure interfaces.

a Expand **Interfaces** and click **Set**.

b Click **Add Interface**.

c Enter a name.

For example, enter the name `TIER-0_VWT-UPLINK1`.

d Select **Type** as **External**.

- e Enter an IP address from the Edge Logical Router – Uplink VLAN. The IP address must be different from the management IP address configured for the NSX Edge VMs previously created.

For example, 10.197.154.1/24.

- f In **Connected To**, select the tier-0 uplink segment previously created.

For example, TIER-0-LS-UPLINK

- g Select an NSX Edge node from the list.

For example, nsx-edge-1.

- h Click **Save**.

- i Repeat steps a - h for the second interface.

For example, create a second uplink TIER-0_VWT-UPLINK2 with IP address 10.197.154.2/24 connected to nsx-edge-2 Edge node.

- j Click **Close**.

10 To configure high availability, click **Set in **HA VIP Configuration**.**

- a Click **ADD HA VIP CONFIGURATION**.

- b Enter the IP address.

For example, 10.197.154.3/24

- c Select the interfaces.

For example, TIER-0_VWT-UPLINK1 and TIER-0_VWT-UPLINK2

- d Click **Add** and **Apply**.

11 To configure routing, click **Routing.**

- a Click **Set** in Static Routes.

- b Click **ADD STATIC ROUTE**.

- c Enter a name.

For example, DEFAULT-STATIC-ROUTE.

- d Enter 0.0.0.0/0 for network IP address.

- e To configure next hops, click **Set Next Hops** and then **Add Next Hop**.

- f Enter the IP address of the next hop router. Typically, this is the default gateway of the management network VLAN from the NSX Edge logical router uplink VLAN.

For example, 10.197.154.253.

- g Click **Add** and **Apply** and **SAVE**.

- h Click **Close**.

- 12 To verify connectivity, make sure that an external device in the physical architecture can ping the uplinks that you configured.

What to do next

Configure a Supervisor Cluster. See [Enable Workload Management with NSX Networking](#)

Configuring vSphere Networking and NSX Advanced Load Balancer for vSphere with Tanzu

vSphere with Tanzu supports the NSX Advanced Load Balancer, also known as Avi Load Balancer, Enterprise Edition. You can install and configure the NSX Advanced Load Balancer 22.1.X in your vSphere with Tanzu environment only if you are using vSphere Distributed Switch networking for the Supervisor Cluster .

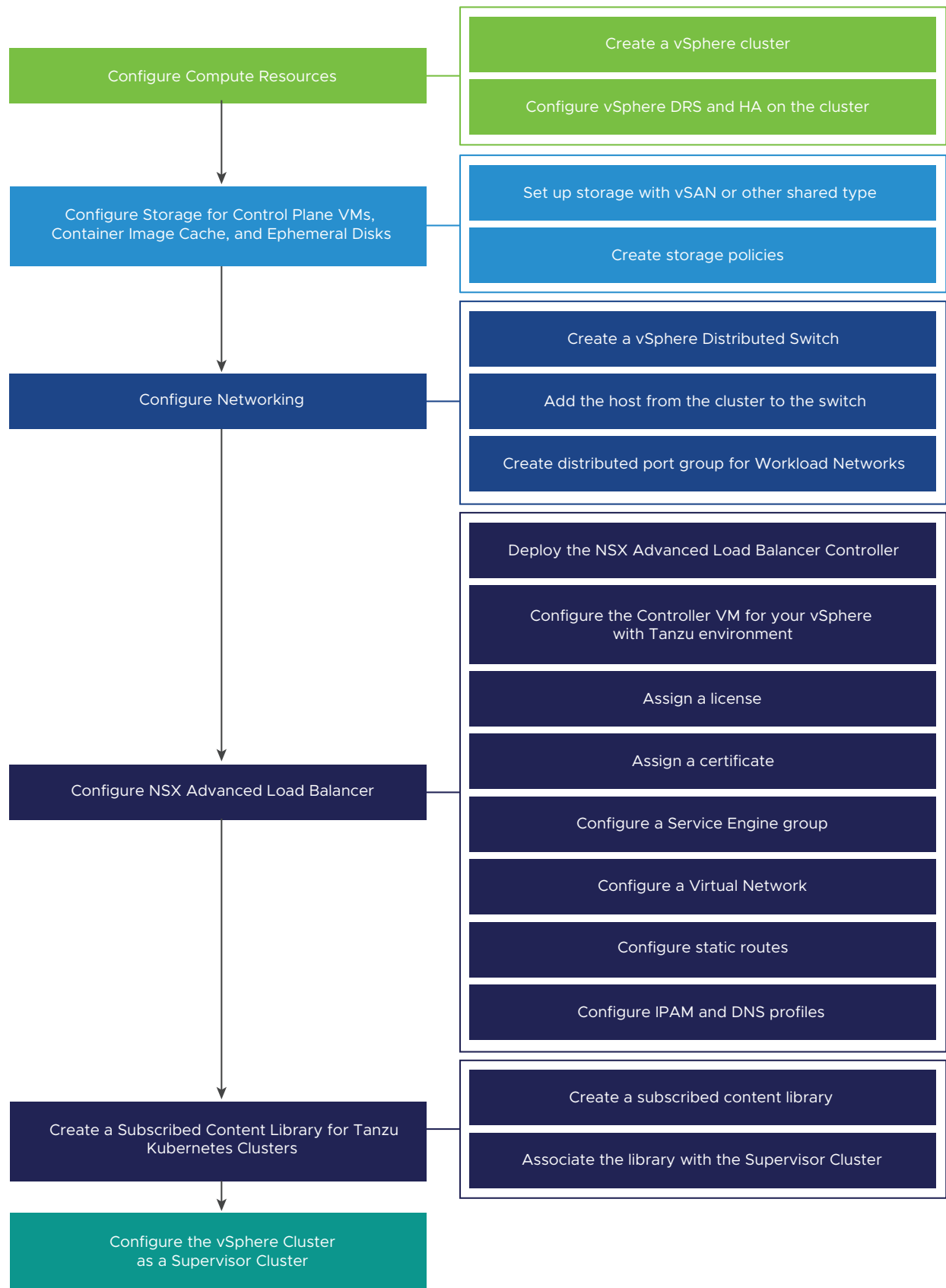
How the Load Balancer Works with Tanzu Kubernetes Clusters

The NSX Advanced Load Balancer provides dynamically scaling load balancing endpoints for Tanzu Kubernetes clusters provisioned by the Tanzu Kubernetes Grid Service. You install and configure the Avi Controller VM. Once you have configured the Controller, it automatically provisions load balancing endpoints for you. For example, when you provision a Tanzu Kubernetes cluster, the Controller creates a virtual service and deploys a Service Engine VM to host that service. This virtual service provides load balancing for the Kubernetes control plane.

When you create a Kubernetes service of type load balancer for that cluster, the Controller automatically creates a virtual service and deploys it to the Service Engine. The first Service Engine is created only after the first virtual service is configured. Any subsequent virtual services that are configured use the existing Service Engine. You can deploy multiple virtual services on a Service Engine VM.

Supervisor Cluster with vSphere Networking and NSX Advanced Load Balancer Workflow

This diagram shows the workflow for configuring vSphere Networking and NSX Advanced Load Balancer for vSphere with Tanzu.



What to read next

Procedure

1 [NSX Advanced Load Balancer Components](#)

The components of the NSX Advanced Load Balancer, also known as Avi Load Balancer, include the Avi Controller cluster, Service Engines (data plane) VMs and the Avi Kubernetes Operator (AKO).

2 [System Requirements for Setting Up vSphere with Tanzu with vSphere Networking and NSX Advanced Load Balancer](#)

To configure vSphere with Tanzu with the NSX Advanced Load Balancer, also known as the Avi Load Balancer, your environment must meet certain requirements. vSphere with Tanzu supports multiple topologies for Avi networking: a single VDS network for the Avi Service Engine and load balancer services, and a VDS for the Avi management plane and another VDS for the NSX Advanced Load Balancer.

3 [Topology for Supervisor Cluster with vSphere Networking and NSX Advanced Load Balancer](#)

The Avi Controller is always deployed to the Management Network where it can interface with the vCenter Server, ESXi hosts, and Supervisor Cluster control plane nodes. The Service Engines are deployed with interfaces to the Management Network and the Data Network.

4 [Install and Configure the NSX Advanced Load Balancer](#)

If you are using vSphere Distributed Switch (VDS) networking, you can install and configure the NSX Advanced Load Balancer 22.1.X in your vSphere with Tanzu environment.

NSX Advanced Load Balancer Components

The components of the NSX Advanced Load Balancer, also known as Avi Load Balancer, include the Avi Controller cluster, Service Engines (data plane) VMs and the Avi Kubernetes Operator (AKO).

Controller

The Avi Controller, also called the Controller, interacts with the vCenter Server to automate the load balancing for the Tanzu Kubernetes clusters. It is responsible for provisioning service engines, coordinating resources across service engines, and aggregating service engine metrics and logging. The Controller provides a Web interface, command-line interface, and API for user operation and programmatic integration.

After you deploy and configure the Controller VM in vSphere, see [Deploy a Controller Cluster](#) for information on how to set up the control plane cluster for HA.

Service Engine

The Avi Service Engine, also called the Service Engine, is the data plane virtual machine. A Service Engine runs one or more virtual services. A Service Engine is managed by the controller. The controller provisions Service Engines to host virtual services.

The Service Engine has two types of network interfaces:

- The first network interface, `vnic0` of the VM, connects to the Management Network where it can connect to the Avi Controller.
- The remaining interfaces, `vnic1` – 8, connect to the Data Network where virtual services run.

The Service Engine interfaces automatically connect to correct VDS port groups. Unused interfaces are connected to a port group called `Avi Internal`, which is created automatically, and reserved for future use. Each Service Engine can support up to 1000 virtual services.

A virtual service provides layer 4 and layer 7 load balancing services for Tanzu Kubernetes cluster workloads. A virtual service is configured with one virtual IP and multiple ports. When a virtual service is deployed, the Controller automatically selects an ESX server, spins up a Service Engine, and connects it to the correct networks (port groups).

The first Service Engine is created only after the first virtual service is configured. Any subsequent virtual services that are configured use the existing Service Engine.

Each virtual server exposes a layer 4 load balancer with a distinct IP address of type load balancer for a Tanzu Kubernetes cluster. The IP address assigned to each virtual server is selected from the IP address block given to the Controller when you configure it.

AVI includes native IPAM and external IPAM provider support. In vSphere, AVI native IPAM is leveraged.

Avi Kubernetes Operator

The Avi Kubernetes operator (AKO) watches Kubernetes resources and communicates with the Controller to request the corresponding load balancing resources.

The Avi Kubernetes Operator is installed on the Supervisor Cluster as part of the enablement process.

System Requirements for Setting Up vSphere with Tanzu with vSphere Networking and NSX Advanced Load Balancer

To configure vSphere with Tanzu with the NSX Advanced Load Balancer, also known as the Avi Load Balancer, your environment must meet certain requirements. vSphere with Tanzu supports multiple topologies for Avi networking: a single VDS network for the Avi Service Engine and load balancer services, and a VDS for the Avi management plane and another VDS for the NSX Advanced Load Balancer.

Workload Networks

To configure a Supervisor Cluster with the vSphere Networking stack, you must connect all hosts from the cluster to a vSphere Distributed Switch. Depending on the topology that you implement for the Supervisor Cluster, you create one or more distributed port groups. You designate the port groups as Workload Networks to vSphere Namespaces.

Before you add a host to a Supervisor Cluster, you must add it to all the vSphere Distributed Switches that are part of the cluster.

Workload Networks provide connectivity to the nodes of Tanzu Kubernetes clusters and to Supervisor Cluster control plane VMs. The Workload Network that provides connectivity to Kubernetes control plane VMs is called a Primary Workload Network. Each Supervisor Cluster must have one Primary Workload Network. You must designate one of the distributed port group as the Primary Workload Network to the Supervisor Cluster.

The Kubernetes control plane VMs on the Supervisor Cluster use three IP addresses from the IP address range that is assigned to the Primary Workload Network. Each node of a Tanzu Kubernetes cluster has a separate IP address assigned from the address range of the Workload Network that is configured with the namespace where the Tanzu Kubernetes cluster runs.

Networking Requirements

The NSX Advanced Load Balancer requires two routable subnets:

- **The Management Network.** The Management Network is where the Avi Controller, also called the Controller, resides. The Management Network provides the Controller with connectivity to the vCenter Server, ESXi hosts, and the Supervisor Cluster control plane nodes. This network is where the management interface of Avi Service Engine is placed. This network requires a vSphere Distributed Switch (VDS) and a distributed port group.
- **The Data Network.** The data interface of the Avi Service Engines, also called Service Engines, connect to this network. The load balancer Virtual IPs (VIPs) are assigned from this network. This network requires a vSphere Distributed Switch (VDS) and distributed port groups. You must configure the VDS and port groups before installing the load balancer.

Allocation of IP Addresses

The Controller and the Service Engine are connected to the Management Network. When you install and configure the NSX Advanced Load Balancer, provide a static, routable IP address for each Controller VM.

The Service Engines can use DHCP. If DHCP is unavailable, you can configure a pool of IP addresses for the Service Engines.

For more information, see [Configure Static Routes](#).

Minimum Compute Requirements

The table lists the minimum compute requirements for vSphere Networking with NSX Advanced Load Balancer. Consider separating the management and workload domain as a best practice. The workload domain will be the Supervisor Cluster where you run workloads. The management domain will contain all the management components such as vCenter Server

Caution Do not disable vSphere DRS after you configure the Supervisor Cluster. Having DRS enabled at all times is a mandatory prerequisite for running workloads on the Supervisor Cluster. Disabling DRS leads to breaking your Tanzu Kubernetes clusters.

Table 4-2. Minimum Compute Requirements

System	Minimum Deployment Size	CPU	Memory	Storage
vCenter Server 7.0, 7.0.2, 7.0.3	Small	2	16 GB	290 GB
ESXi hosts 7.0	3 ESXi hosts with 1 static IP per host. If you are using vSAN: 3 ESXi hosts with at least 2 physical NICs is the minimum; however, 4 ESXi hosts are recommended for resiliency during patching and upgrading. The hosts must be joined in a cluster with vSphere DRS and HA enabled. vSphere DRS must be in Fully Automate or Partially Automate mode. Note Make sure that the names of the hosts that join the cluster use lower case letters. Otherwise, the enablement of the cluster for Workload Management might fail.	8	64 GB per host	Not applicable
Kubernetes control plane VMs	3	4	16 GB	16 GB

Table 4-2. Minimum Compute Requirements (continued)

System	Minimum Deployment Size	CPU	Memory	Storage
Avi Controller	Essentials	4	12 GB	128 GB
	Enterprise	8	24 GB	128 GB
Note For smaller deployments, you can deploy the Essentials size controller as a single controller node. You can create an Avi controller cluster, but it does not have any performance benefits and it defeats the purpose of low resource utilization. You can use a remote backup for disaster recovery in this case. This size must only be used with Avi Essentials licensing mode and is limited to 50 virtual services and 10 Service Engines For production environments, it is recommended to install a cluster of 3 Avi Controller VMs. A minimum of 2 Service Engine VMs are required for HA.				
Service Engine	A minimum of 2 Service Engine VMs are required for HA.	1	2 GB	15 GB

Minimum Network Requirements

The table lists the minimum network requirements for vSphere Networking with NSX Advanced Load Balancer.

Note You cannot create IPv6 clusters with a vSphere 7 Supervisor Cluster, or register IPv6 clusters with Tanzu Mission Control. NSX Advanced Load Balancer services do not currently support IPv6.

Table 4-3. Minimum Networking Requirements

Component	Minimum Quantity	Required Configuration
Static IPs for Kubernetes control plane VMs	Block of 5	A block of 5 consecutive static IP addresses to be assigned from the Management Network to the Kubernetes control plane VMs in the Supervisor Cluster.
Management traffic network	1	A Management Network that is routable to the ESXi hosts, vCenter Server, the Supervisor Cluster and load balancer. The network must be able to access an image registry and have Internet connectivity if the image registry is on the external network. The image registry must be resolvable through DNS.
vSphere Distributed Switch 7.0 or later	1	All hosts from the cluster must be connected to a vSphere Distributed Switch.
Workload Networks	1	At least one distributed port group must be created on the vSphere Distributed Switch that you configure as the Primary Workload Network. Depending on the topology of choice, you can use the same distributed port group as the Workload Network of namespaces or create more port groups and configure them as Workload Networks. Workload Networks must meet the following requirements: ■ Workload Networks that are used for Tanzu Kubernetes cluster traffic must be routable between each other and the Supervisor Cluster Primary Workload Network. ■ Routability between any Workload Network with the network that the NSX Advanced Load Balancer uses for virtual IP allocation. ■ No overlapping of IP address ranges across all Workload Networks within a Supervisor Cluster.
NTP and DNS Server	1	DNS server IP is required for the Avi Controller to resolve the vCenter Server and ESXi hostnames correctly. NTP is optional as public NTP servers are used by default. Note Configure NTP on all ESXi hosts and vCenter Server . Important The DNS Server must be different for the Management and Workload Networks. Assigning the same DNS Server to both networks can lead to issues while enabling Workload Management and pulling images. The DNS Server of the Management Network must be reachable from the Management Network and the DNS Server of the Workload Network must be reachable from the Workload.

Table 4-3. Minimum Networking Requirements (continued)

Component	Minimum Quantity	Required Configuration
DHCP Server	1	Optional. Configure a DHCP server to automatically acquire IP addresses for the Management and Workload Networks as well as floating IPs. The DHCP server must support Client Identifiers and provide compatible DNS servers, DNS search domains, and an NTP server. For the Management Network, all the IP addresses, such as control plane VM IPs, a Floating IP, DNS servers, DNS, search domains, and NTP server are acquired automatically from the DHCP server. The DHCP configuration is used by the Supervisor Cluster. Load balancers may require static IP addresses for Management. DHCP Scopes should not overlap these static IP's. DHCP is not used for virtual IPs. (VIPs)
Management Network Subnet	1	The Management Network is where the Avi Controller, also called the Controller, resides. It is also where the Service engine management interface is connected. The Avi Controller must be connected to the vCenter Server and ESXi management IPs from this network Important The Management Network and the Workload Network must be on different subnets. Assigning the same subnet to the Management and the Workload Networks is not supported and can lead to system errors and problems.
Data Network Subnet	1	The data interface of the Avi Service Engines, also called Service Engines, connect to this network. Configure a pool of IP addresses for the Service Engines. The load balancer Virtual IPs (VIPs) are assigned from this network.
Physical Network MTU	1500	The MTU size must be 1500 or greater on any vSphere Distributed Switch port group.
vSphere Pod CIDR range	/24 Private IP addresses	A private CIDR range that provides IP addresses for vSphere Pods.
Avi Controller IPs	1 or 4	If you deploy the Avi controller as a single node, one static IP is required for its management interface. For a 3 node cluster, 4 IP addresses are required. One for each Avi controller VM and one for the cluster VIP. These IPs must be from the Management Network subnet.
VIP IPAM range	-	A private CIDR range to assign IP addresses to Kubernetes services. The IPs must be from the Data Network Subnet. You must specify a unique Kubernetes services CIDR range for each Supervisor Cluster.

Ports and Protocols

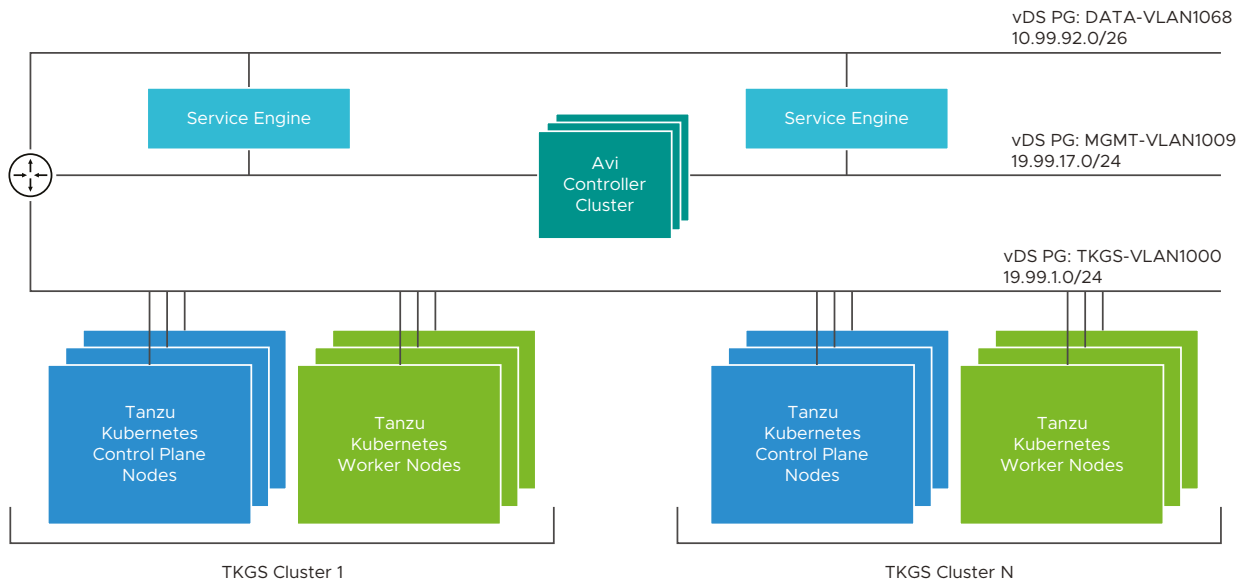
This table lists the protocols and ports required for managing IP connectivity between the NSX Advanced Load Balancer, vCenter and other vSphere with Tanzu components.

Source	Destination	Protocol and Ports
Avi Controller	Avi Controller (in cluster)	TCP 22 (SSH) TCP 443 (HTTPS) TCP 8443 (HTTPS)
Service Engine	Service Engine in HA	TCP 9001 for VMware, LSC and NSX-T cloud
Service Engine	Avi Controller	TCP 22 (SSH) TCP 8443 (HTTPS) UDP 123 (NTP)
Avi Controller	vCenter Server, ESXi, NSX-T Manager	TCP 443 (HTTPS)
Supervisor Control plane nodes (AKO)	Avi Controller	TCP 443 (HTTPS)

For more information about ports and protocols for the NSX Advanced Load Balancer, see <https://ports.esp.vmware.com/home/NSX-Advanced-Load-Balancer>.

Topology for Supervisor Cluster with vSphere Networking and NSX Advanced Load Balancer

The Avi Controller is always deployed to the Management Network where it can interface with the vCenter Server, ESXi hosts, and Supervisor Cluster control plane nodes. The Service Engines are deployed with interfaces to the Management Network and the Data Network.



The Management Network, such as `MGMT-VLAN1009`, is where the Controller resides and management interface of the Services Engines are connected to.

The Data Network, such as `DATA-VLAN1068`, is where the Service Engine interfaces connect for VIP placement. The client traffic reaches the VIP and the Service Engines load balances the traffic to the Workload Network IPs through this network.

The Workload Network, such as `TKGS-VLAN1000`, is where the Tanzu Kubernetes clusters run. The Service Engines do not require interfaces to the Workload Network.

The Service Engines run in a one-arm mode. They route the load balanced traffic to the Workload Network through the router. The Service Engines do not get the default gateway IP from DHCP on the Data Networks. You must configure static routes so that the Service Engines can route traffic to the Workload Networks and Client IP correctly. For more information about configuring static routes, see [Configure Static Routes](#).

This topology allows the Service Engine to be located on a single network. The Service Engine creation and network connections are automated by the Avi Controller.

Install and Configure the NSX Advanced Load Balancer

If you are using vSphere Distributed Switch (VDS) networking, you can install and configure the NSX Advanced Load Balancer 22.1.X in your vSphere with Tanzu environment.

- Verify that your environment meets the requirements to configure vSphere with Tanzu with the NSX Advanced Load Balancer. See [System Requirements for Setting Up vSphere with Tanzu with vSphere Networking and NSX Advanced Load Balancer](#).
- Download the NSX Advanced Load Balancer OVA. VMware provides an NSX Advanced Load Balancer OVA file that you deploy in your vSphere environment where you enable Workload Management. Download the latest version of the OVA file that is supported with vSphere with Tanzu from the [VMware Customer Connect](#) portal.

Note The procedures in this guide are relevant to the NSX Advanced Load Balancer version 22.1.X that is supported with this version of vSphere with Tanzu. There could be later versions of NSX Advanced Load Balancer that are available and the UI workflows might be different.

Create a vSphere Distributed Switch for a Supervisor Cluster for Use with NSX Advanced Load Balancer

To configure a vSphere cluster as a Supervisor Cluster that uses the vSphere networking stack and the NSX Advanced Load Balancer, you must create a vSphere Distributed Switch. Create port groups on the distributed switch that you can configure as Workload Networks to the Supervisor Cluster. The NSX Advanced Load Balancer needs a distributed port group to connect the Avi Service Engine data interfaces. The port group is used to place the application Virtual IPs (VIPs) on the Service Engines.

Prerequisites

Review the system requirements and network topologies for using vSphere networking for the Supervisor Cluster with the NSX Advanced Load Balancer. See [System Requirements for Setting Up vSphere with Tanzu with vSphere Networking and NSX Advanced Load Balancer](#).

Procedure

- 1 In the vSphere Client, navigate to a data center.
- 2 Right click the data center and select **Distributed Switch > New Distributed Switch**.
- 3 Enter a name for the switch, for example **Workload Distributed Switch** and click **Next**.
- 4 Select version 7.0 for the switch and click **Next**.
- 5 In **Port group name**, enter **Primary Workload Network**, click **Next**, and click **Finish**.
- 6 Create distributed port groups for Workload Networks.
 - a Navigate to the newly-created distributed switch.
 - b Right-click the switch and select **Distributed Port Groups > New Distributed Port Group**.
 - c Enter a name for the port group, for example **Workload Network** and click **Next**.
 - d Leave the defaults, click **Next** and click **Finish**.

7 Create a port group for the Data Network.

- a Right-click the distributed switch and select **Distributed port group > New distributed port group**.
- b Enter a name for the port group, for example, **Data Network** and click **Next**.
- c On the **Configure Settings** page, enter the general properties for the new distributed port group and click **Next**.

Property	Description
Port binding	Select when ports are assigned to virtual machines connected to this distributed port group. Select Static binding to assign a port to a virtual machine when the virtual machine connects to the distributed port group.
Port Allocation	Select Elastic port allocation. The default number of ports is eight. When all ports are assigned, a new set of eight ports is created.
Number of ports	Retain the default value.
Network resource pool	From the drop-down menu, assign the new distributed port group to a user-defined network resource pool. If you have not created a network resource pool, this menu is empty.
VLAN	From the drop-down menu, select the type of VLAN traffic filtering and marking: ■ None: Do not use VLAN. Select this option if you are using External Switch Tagging. ■ VLAN: In the VLAN ID text box, enter a value from 1 through 4094 for Virtual Switch Tagging. ■ VLAN trunking: Use this option for Virtual Guest Tagging and to pass VLAN traffic with an ID to the guest OS. Enter a VLAN trunk range. You can set multiple ranges or individual VLANs by using a comma-separated list. For example, 1702–1705, 1848–1849. ■ Private VLAN: Associate the traffic with a private VLAN created on the distributed switch. If you did not create any private VLANs, this menu is empty.
Advanced	Leave this option unselected.

8 On the **Ready to Complete** page, review the configuration and click **Finish**.

Results

The distributed switch is created and distributed port groups appear under the distributed switch. You can now use the port group that you created for the Data Network for the NSX Advanced Load Balancer.

Import the NSX Advanced Load Balancer to a Local Content Library

To store the NSX Advanced Load Balancer OVA image, create a local content library and import the OVA to it.

Creating a local content library involves configuring the library, downloading the OVA files, and importing them to the local content library.

Prerequisites

Verify that you have downloaded the NSX Advanced Load Balancer OVA.

Create a local Content Library. See [Chapter 6 Creating and Managing Content Libraries in vSphere with Tanzu](#).

Procedure

- 1 Log in to the vCenter Server using the vSphere Client.
- 2 Select **Menu > Content Libraries**.
- 3 From the list of **Content Libraries**, click the link for the name of the local content library you created. For example, **NSX ALB**.
- 4 Click **Actions**.
- 5 Select **Import Item**.
- 6 In the **Import Library Item** window, select **Local File**.
- 7 Click **Upload Files**.
- 8 Select the OVA file that you downloaded.
- 9 Click **Import**.
- 10 Reveal the **Recent Tasks** pane at the bottom of the page.
- 11 Monitor the task **Fetch Content of a Library Item** and verify that it is successfully **Completed**.

What to do next

Deploy the NSX Advanced Load Balancer Controller. See [Deploy the Controller](#).

Deploy the Controller

Deploy the Controller VM to the Management Network in your vSphere with Tanzu environment.

Prerequisites

- Verify that you have a Management Network on which to deploy the NSX Advanced Load Balancer. This can be a vSphere Distributed Switch (vDS) or a vSphere Standard Switch (vSS).
- Verify that you have created a vDS switch and portgroup for the Data Network. See [Create a vSphere Distributed Switch for a Supervisor Cluster for Use with NSX Advanced Load Balancer](#).
- Verify that you have completed the prerequisites. See [System Requirements for Setting Up vSphere with Tanzu with vSphere Networking and NSX Advanced Load Balancer](#).

Procedure

- 1 Log in to the vCenter Server by using the vSphere Client.
- 2 Select the vSphere cluster that is designated for management components.
- 3 Create a resource pool named **AVI-LB**.
- 4 Right-click the resource pool and select **Deploy OVF Template**.
- 5 Select **Local File** and click **Upload Files**.
- 6 Browse to and select the `controller-VERSION.ova` file you downloaded as a prerequisite.
- 7 Enter a name and select a folder for the Controller.

Option	Description
Virtual machine name	<code>avi-controller-1</code>
Location for the virtual machine	Datacenter

- 8 Select the **AVI-LB** resource pool as a compute resource.
- 9 Review the configuration details and click **Next**.
- 10 Select a **VM Storage Policy**, such as **vsanDatastore**.
- 11 Select the Management Network, such as **network-1**.
- 12 Customize the configuration as follows and click **Next** when you are done.

Option	Description
Management Interface IP Address	Enter the IP address for the Controller VM.
Management Interface Subnet Mask	Enter the subnet mask.
Default Gateway	Enter the default gateway for the Management Network.
Sysadmin login authentication key	Paste the contents of a private key (optional). This is the private SSH key that you require to SSH into the VM. You can create it using OpenSSH or PuTTY.

- 13 Review the deployment settings.
- 14 Click **Finish** to complete the configuration.
- 15 Use vSphere Client to monitor the provisioning of the Controller VM in the **Tasks** panel.
- 16 Use the vSphere Client to power on the Controller VM after it is deployed.

Deploy a Controller Cluster

Optionally, you can deploy a cluster of three controller nodes. Configuring a cluster is recommended in production environments for HA and disaster recovery. If you are running a single node Avi Controller you must use the Backup and Restore feature.

To run a three node cluster, after you deploy the first Controller VM, deploy and power on two more Controller VMs. You must not run the initial configuration wizard or change the admin password for these controllers. The configuration of the first controller VM is assigned to the two new Controller VMs.

Procedure

- 1 Go to **Administration > Controller**.

- 2 Select **Nodes**.

- 3 Click the edit icon.

- 4 Add a static IP for **Controller Cluster IP**.

This IP address must be from the Management Network.

- 5 In **Cluster Nodes**, configure the two new cluster nodes.

Option	Description
IP	IP address of the controller node.
Name	Name of the node. The name can be the IP address.
Password	Password of the controller node. Leave the password empty.
Public IP	The public IP address of the controller node. Leave this empty.

- 6 Click **Save**.

Note Once you deploy a cluster, you must use the controller cluster IP for any further configuration and not the controller node IP.

Power On the Controller

After you deploy the Controller VM, you can power it on. During the boot up process, the IP address specified during the deployment gets assigned to the VM.

After power on, the first boot process of the Controller VM can take up to 10 minutes.

Prerequisites

Deploy the Controller.

Procedure

- 1 In the vCenter Server, right click the `avi-controller-1` VM that you deployed.

- 2 Select **Power > Power On**.

The VM is assigned the IP address that you specified during deployment.

- 3 To verify if the VM is powered on, access the IP address in a browser.

When the VM comes online, warnings about the TLS certificate and connection appear.

- 4 In the **This Connection Is Not Private** warning, click **Show Details**.

- Click **visit this website** in the window that appears.

You are prompted for user credentials.

Configure the Controller

Configure the Controller VM for your vSphere with Tanzu environment.

To connect the load balancer control plane with the vCenter Server environment, the Controller requires several post-deployment configuration parameters.

Prerequisites

- Verify that your environment meets the system requirements for configuring the NSX Advanced Load Balancer. See [System Requirements for Setting Up vSphere with Tanzu with vSphere Networking and NSX Advanced Load Balancer](#).
- Deploy the Controller. See [Deploy the Controller](#)

Procedure

- Using a browser, navigate to the IP address that you specified when deploying the Controller.
- Create an **Administrator Account**.

Option	Description
Username	The administrator user name for initial configuration. You cannot edit this field.
Password	Enter an administrator password for the Controller VM. The password must be at least 8 characters and contain a combination of numeric, special, uppercase, and lowercase characters.
Confirm Password	Enter the administrator password again.
Email Address (optional)	Enter an administrator email address. It is recommended that you provide an email address for password recovery in a production environment.

- Configure **System Settings**.

Option	Description
Passphrase	Enter a passphrase for the Controller backup. The Controller configuration is automatically backed up to the local disk on a periodic basis. For more information, see Backup and Restore . The passphrase must be at least 8 characters and contain a combination of numeric, special, uppercase, and lowercase characters.
Confirm Passphrase	Enter the backup passphrase again.
DNS Resolver	Enter an IP address for the DNS server you are using in the vSphere with Tanzu environment. For example, 10.14.7.12.
DNS Search Domain	Enter a domain string.

4 (Optional) Configure **Email/SMTP**

Option	Description
SMTP Source	None, Local Host, SMTP Server, or Anonymous Server
From Address	Email address

5 Click **Next**.

6 Configure the multi-tenant settings.

- a Retain the default tenant access.
- b Select **Setup Cloud After** and click **Save**.

Note If you did not select **Setup Cloud After** option before saving, the initial configuration wizard exits. The Cloud configuration window does not automatically launch and you are directed to a Dashboard view on the controller. In this case browse to **Infrastructure > Clouds** and edit the **Default-Cloud** and continue with below steps.

7 Configure **Default-Cloud**.

- a Edit **Default-Cloud** settings.
- b Select **VMware vCenter/vSphere ESX** as the infrastructure type.
vSphere with Tanzu supports only **Default-Cloud**.
- c Select **Yes, Continue**.

The **Default-Cloud** settings page is displayed.

8 Configure the **General** settings.

Option	Description
Name	Optionally enter a name for the Default-Cloud .
Type	The cloud type is VMware vCenter/vSphere ESX .

9 (Optional) Select **DHCP Enabled** if DHCP is available on the vSphere port groups.

Leave the option unselected if you want the Service Engine interfaces to use only static IP addresses. You can configure them individually for each network.

For more information, see [Configure a Virtual IP Network](#).

10 Configure the **Virtual Service Placement** settings.

Option	Description
Prefer Static Routes vs Directly Connected Network for Virtual Service Placement	Select this option to force the Service Engine VM to access the server network by routing it through the default gateway. By default, the Controller directly connects a NIC to the server network and you must force the Service Engine to connect only to the Data Network and route to the Workload Network.
Use Static Routes for Network Resolution of VIP for Virtual Service Placement	Leave this option unselected.

11 Configure the **vCenter/vSphere** credentials.

Option	Description
vCenter Address	Enter the vCenter Server hostname or IP address for the vSphere with Tanzu environment.
Username	Enter the vCenter administrator user name, such as administrator@vsphere.local . To use lesser permissions, create a dedicated role. See VMware User Role for details.
Password	Enter the user password.
Access Permissions	Read: You create and manage the service engine VMs. Write: Controller creates and manages the service engine VMs. You must select Write.

12 Configure the **Data Center** settings.

- Select the vSphere **Data Center** where you want to enable **Workload Management**.
- Select the **Use Content Library** option and select the local content library from the list.

13 Select **SAVE & RELAUNCH** to create the **VMware vCenter/vSphere ESX** cloud with the settings you configured.

14 Configure the **Network** settings and click **Save**.

Option	Description
Management Network	Select the VM Network. This network interface is used by the Service Engines to connect with the Controller.
Service Engine	Leave the Template Service Engine Group empty.
Management Network IP Address Management	Select DHCP Enabled .

15 (Optional) Configure the following network settings only if you do not select **DHCP Enabled**.

Option	Description
IP Subnet	Enter the IP subnet for the Management Network. Note Enter an IP subnet only if DHCP is not available.
Add Static IP Address Pool	Enter one or more IP addresses or IP address range. Note Enter an IP subnet only if DHCP is not available.
Default Gateway	Enter the default gateway for the Management Network. Note Enter an IP subnet only if DHCP is not available.

16 Create an IPAM profile and configure **IPAM/DNS** settings.

IPAM is required to allocate virtual IP addresses when virtual services get created.

- a From the More actions menu of **IPAM Profile**, select **Create**.

The **NEW IPAM/DNS PROFILE** page is displayed.

- b Configure the **IPAM Profile**.

Option	Description
Name	User-defined string, such as ipam-profile
Type	Select AVI Vantage IPAM
Allocate IP in VRF	Deselect this option.
Cloud	Select Default-Cloud from the drop-down list.

- c Click **Add** in the **Usable Network** and select the Virtual IP network that you configured. This network is the primary network.
- d Click **SAVE**.

17 (Optional) Configure NTP settings if you want to use an internal NTP server.

- a Select **Administration > Settings > DNS/NTP**.
- b Delete existing NTP servers if any and enter the IP address for the DNS server you are using.

Results

Once you complete the configuration, you see the Controller **Dashboard**. Select the **Infrastructure > Clouds** and verify that the status of the Controller for **Default-Cloud** is green. Sometimes the status can be yellow for some time till the Avi Controller discovers all the port groups in the vCenter environment, before it turns green.

Add a License

Once you configure the NSX Advanced Load Balancer, you must add a license to it. The Controller boots in evaluation mode that has all the features equivalent to an Enterprise edition license available. You must assign a valid Enterprise Tier license to the Controller before the evaluation period expires.

Prerequisites

Verify that you have the Enterprise Tier license.

Procedure

- 1 In the NSX Advanced Load Balancer Controller dashboard, select **Administration > Licensing**.
- 2 Select **Settings**.
- 3 Select **Enterprise Tier**.
- 4 Click **SAVE**.
- 5 To add the license, select **Upload from Computer**.

After the license file is uploaded, it appears in the Controller license list. The system displays the information about the license, including the start date and the expiration date.

Assign a Certificate to the Controller

The Controller must send a certificate to clients to establish secure communication. This certificate must have a Subject Alternative Name (SAN) that matches the Controller cluster hostname or IP address.

The Controller has a default self-signed certificate. But this certificate does not have the correct SAN. You must replace it with a valid or self-signed certificate that has the correct SAN. You create a self-signed certificate or upload an external certificate.

For more information about certificates, see the [Avi documentation](#).

Procedure

- 1 In the Controller dashboard, select **Templates > Security**.
- 2 Select **SSL/TLS Certificate**.
- 3 To create a certificate, click **Create** and select **Controller Certificate**.
The **New Certificate (SSL/TLS)** window appears.
- 4 Enter a name for the certificate.

- 5 If you do not have a pre-created valid certificate, add a self-signed certificate by selecting **Type** as *Self Signed*.

a Enter the following details:

Option	Description
Common Name	Specify the fully-qualified name of the site. For the site to be considered trusted, this entry must match the hostname that the client entered in the browser.
Subject Alternate Name (SAN)	Enter the cluster IP address or FQDN, or both, of the Avi Controller if it is deployed as a single node. If only the IP address or FQDN is used, it must match the IP address of the Controller VM that you specify during deployment. See Deploy the Controller . Enter the cluster IP or FQDN of the Avi Controller cluster if it is deployed as a cluster of three nodes. For information about deploying a cluster of three controller nodes, see Deploy a Controller Cluster .
Algorithm	Select either EC (elliptic curve cryptography) or RSA. EC is recommended.
Key Size	Select the level of encryption to be used for handshakes: ■ SECP256R1 is used for EC certificates. ■ 2048-bit is recommended for RSA certificates.

b Click **SAVE**.

You need this certificate when you configure the Supervisor Cluster to enable the Workload Management functionality.

- 6 Download the self-signed certificate that you create.

a Select **Security > SSL/TLS Certificates**.

If you do not see the certificate, refresh the page.

b Select the certificate you created and click the download icon.

c In the **Export Certificate** page that appears, click **Copy to clipboard** against the certificate. Do not copy the key.

d Save the copied certificate for later use when you enable workload management.

- 7 If you have a pre-created valid certificate, upload it by selecting **Type** as *Import*.

a In **Certificate**, click **Upload File** and import the certificate.

The SAN field of the certificate you upload must have the cluster IP address or FQDN of the Controller.

Note Make sure that you upload or paste the contents of the certificate only once.

b In **Key (PEM) or PKCS12**, click **Upload File** and import the key.

- c Click **Validate** to validate the certificate and key.
 - d Click **SAVE**.
- 8 To change the portal certificate, perform the following steps.
- a In the Controller dashboard, select **Administration > Settings**.
 - b Click **Edit**.
 - c Select the **Access** tab.
 - d From **SSL/TLS Certificate**, remove the existing default portal certificates.
 - e In the drop-down, select the newly created or uploaded certificate.
 - f Select **Basic Authentication**.
 - g Click **SAVE**.

Configure a Service Engine Group

vSphere with Tanzu uses the **Default-Group** Service Engine Group. Optionally, you can configure the **Default-Group** Service Engines within a group which defines the placement and number of Service Engine VMs within vCenter. You can also configure high availability if the NSX Advanced Load Balancer Controller is in Enterprise mode. vSphere with Tanzu supports only the **Default-Group** Service Engine. You cannot create other Service Engine Groups.

For information on how you can provision excess capacity in the event of a failover, see the [Avi documentation](#).

Procedure

- 1 In the NSX Advanced Load Balancer Controller dashboard, select **Infrastructure > Cloud Resources > Service Engine Group**.

- 2 In the **Service Engine Group** page, click the edit icon in the **Default-Group**.

The **General Settings** tab appears.

vSphere with Tanzu only supports **Default-Cloud**.

- 3 In the **Placement** section, select **High Availability Mode**.

The default option is **N + M (buffer)**. You can keep the default value or select one of the following options:

- **Active/Standby**
- **Active/Active**

- 4 In the **Service Engine** section, you can configure excess capacity for the Service Engine group.

The **Number of Service Engines** option defines the maximum number of Service Engines that may be created within a Service Engine group. Default is **10**.

To configure excess capacity, specify a value in the **Buffer Service Engines**. The value you specify is the number of VMs that are deployed to ensure excess capacity in the event of a failover.

Default is **1**.

- 5 In the **Virtual Service** section, configure the following options.

Option	Description
Virtual Services per Service Engine	The maximum number of virtual services the Controller cluster can place on any one of the Service Engines in the group. Enter a value of 1000 .
Virtual Services Placement Across Service Engines	Select Distributed . Selecting this option maximizes the performance by placing virtual services on newly spun-up Service Engines up to the maximum number of Service Engines specified. Default is Compact .

- 6 You can keep the default values for the other settings.

- 7 Click **Save**.

Configure Static Routes

A default gateway enables the Service Engine to route traffic to the pool servers on the Workload Network. You must configure the Data Network gateway IP as the default gateway. The Service Engines do not get the default gateway IP from DHCP on the Data Networks. You must configure static routes so that the Service Engines can route traffic to the Workload Networks and Client IP correctly.

Procedure

- 1 In the NSX Advanced Load Balancer Controller dashboard, select **Infrastructure > Cloud Resources > VRF Context**.
- 2 Click **Create..**
- 3 In the **General** settings, enter a name for the routing context.
- 4 In the **Static Route** section, click **ADD**.
- 5 In **Gateway Subnet**, enter `172.16.10.0/24`.
- 6 In **Next Hop**, enter the gateway IP address for the Data network.
For example, `192.168.1.1`.
- 7 (Optional) Select **BGP Peering** to configure BGP local and peer details.
For more information, see the [Avi documentation](#).
- 8 Click **Save**.

Configure a Virtual IP Network

Configure a virtual IP (VIP) subnet for the Data Network. You can configure the VIP range to use when a virtual service is placed on the specific VIP network. You can configure DHCP for the Service Engines. Optionally, if DHCP is unavailable, you can configure a pool of IP addresses which will get assigned to the Service Engine interface on that network. vSphere with Tanzu supports only a single VIP network.

Procedure

- 1 In the NSX Advanced Load Balancer Controller dashboard, **Infrastructure > Networks > .**

- 2 Select the cloud from the list.

For example, select **Default-Cloud**.

- 3 Enter a name for the network.

For example, `Data Network`.

- 4 Keep **DHCP Enabled** selected if DHCP is available on the Data Network.

Deselect this option if DHCP is not available.

- 5 Select **Enable IPv6 Auto Configuration**.

The NSX Advanced Load Balancer Controller discovers the network CIDR automatically if a VM is running on the network and it appears with type **Discovered**.

- 6 If the Controller discovers the IP subnet automatically, configure the IP range for the subnet.

- a Edit settings.

- b Enter a **Subnet Prefix**.

- c If DHCP is available for the Service Engine IP address, deselect **Use Static IP Address for VIPs and SE**.

- d Enter one or more IP addresses or IP address ranges.

For example, `10.202.35.1-10.202.35.254`.

Note You can enter an IP address that ends with 0. For example, `192.168.0.0` and ignore any warning that appears.

- e Click **Save**.

- 7 If the Controller does not discover an IP subnet and its type, perform the following steps:

- a Click **Add**.

- b Enter a **Subnet Prefix**.

- c Click **ADD**.

- d If DHCP is available for the Service Engine IP address, deselect **Use Static IP Address for VIPs and SE**.

- e In **IP Address**, enter the CIDR of the network that provides the virtual IP addresses.

For example, `10.202.35.0/22`

- f Enter one or more IP addresses or IP address ranges.

The range must be a subset of the network CIDR in **IP Subnet**. For example, `10.202.35.1-10.202.35.254`.

Note You can enter an IP address that ends with 0. For example, `192.168.0.0` and ignore any warning that appears.

- g Click **Save** to save the subnet configuration.

The **Edit Network Settings** page lists the IP Subnet with type **Configured** and an IP address pool.

- 8 Click **Save** to save the network settings.

Results

The **Network** page lists the configured networks.

Example

The **Primary Workload Network** network displays the discovered network as `10.202.32.0/22` and configured subnets as `10.202.32.0/22 [254/254]`. This indicates that 254 virtual IP addresses come from `10.202.32.0/22`. Note that the summary view does not list the IP range `10.202.35.1-10.202.35.254`.

Test the NSX Advanced Load Balancer

After you have deployed and configured the NSX Advanced Load Balancer control plane, verify its functionality.

Procedure

- 1 In the Avi Controller dashboard, go to **Infrastructure > Clouds**.
- 2 Verify that the status of the Controller for **Default-Cloud** is green.

To troubleshoot problems that you might encounter, see [Troubleshooting the NSX Advanced Load Balancer](#).

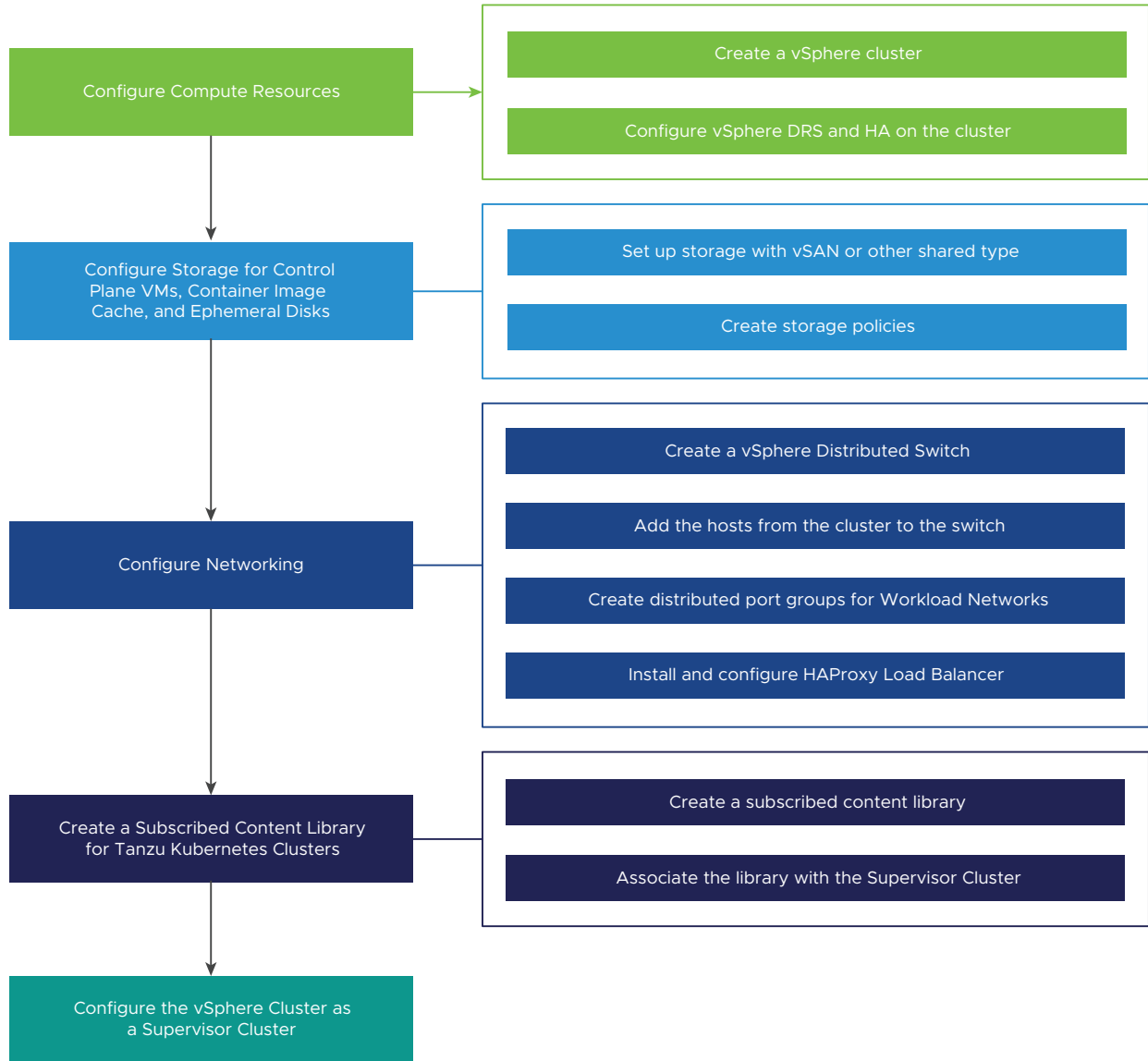
Configuring vSphere Networking and HAProxy Load Balancer for vSphere with Tanzu

If you are using vSphere Distributed Switch networking for your vSphere with Tanzu environment, you can install and configure the open source HAProxy load balancer. VMware provides an implementation of HAProxy that you can deploy from an OVA file.

Supervisor Cluster with vSphere Networking and HAProxy Load Balancer Workflow

This diagram shows the workflow for configuring vSphere networking and HAProxy load balancer for vSphere with Tanzu.

Figure 4-6. Configuring vDS networking with HAProxy workflow



System Requirements for Setting Up vSphere with Tanzu with vSphere Networking and HAProxy Load Balancer

Check out the system requirements for setting up a vSphere cluster as a Supervisor Cluster with the vSphere networking stack and HAProxy load balancer.

Minimum Compute Requirements

Consider separating the management and workload domain as a best practice. The workload domain will be the Supervisor Cluster where you run workloads. The management domain will contain all the management components such as vCenter Server.

System	Minimum Deployment Size	CPU	Memory	Storage
vCenter Server 7.0	Small	2	16 GB	290 GB
ESXi hosts 7.0	Without vSAN: 3 ESXi hosts with 1 static IP per host. With vSAN: 4 ESXi hosts with at least 2 physical NICs. The hosts must be joined in a cluster with vSphere DRS and HA enabled. vSphere DRS must be in Fully Automate or Partially Automate mode. Note Make sure that the names of the hosts that join the cluster use lower case letters. Otherwise, the enablement of the cluster for Workload Management might fail.	8	64 GB per host	Not applicable
Kubernetes control plane VMs	3	4	16 GB	16 GB

Minimum Network Requirements

Note You cannot create IPv6 clusters with a vSphere 7 Supervisor Cluster, or register IPv6 clusters with Tanzu Mission Control.

Component	Minimum Quantity	Required Configuration
Static IPs for Kubernetes control plane VMs	Block of 5	A block of 5 consecutive static IP addresses to be assigned to the Kubernetes control plane VMs in the Supervisor Cluster.
Management traffic network	1	A Management Network that is routable to the ESXi hosts, vCenter Server, the Supervisor Cluster and load balancer. The network must be able to access an image registry and have Internet connectivity if the image registry is on the external network. The image registry must be resolvable through DNS.
vSphere Distributed Switch	1	All hosts from the cluster must be connected to a vSphere Distributed Switch.

Component	Minimum Quantity	Required Configuration
HAProxy load balancer	1	An instance of HAProxy load balancer configured with the vCenter Server instance. ■ If the same the HAProxy instance is serving multiple Supervisor Clusters, it must be able to route traffic to and from all Workload Networks across all Supervisor Clusters. IP ranges across Workload Networks in all Supervisor Clusters that the HAProxy serves must not overlap. ■ The load balancer IP ranges must not reside on the same subnet as the Management network. ■ A dedicated IP range for virtual IPs. The HAProxy VM must be the only owner of this virtual IP range. The range must not overlap with any IP range assigned to any Workload Network owned by any Supervisor Cluster. ■ The network that HAProxy uses to allocate Virtual IPs must be routable to the Workload Networks used across all Supervisor Clusters to which HAProxy is connected.
Workload Networks	1	At least one distributed port group must be created on the vSphere Distributed Switch that you configure as the Primary Workload Network. Depending on the topology of choice, you can use the same distributed port group as the Workload Network of namespaces or create more port groups and configure them as Workload Networks. Workload Networks must meet the following requirements: ■ Workload Networks that are used for Tanzu Kubernetes cluster traffic must be routable between each other and the Supervisor Cluster Primary Workload Network. ■ Routability between any Workload Network with the network that HAProxy uses for virtual IP allocation. ■ No overlapping of IP address ranges across all Workload Networks within a Supervisor Cluster. Important The workload network must be on a different subnet than the management network.
NTP and DNS Server	1	A DNS server and NTP server that can be used with vCenter Server. Note Configure NTP on all ESXi hosts and vCenter Server .

Component	Minimum Quantity	Required Configuration
DHCP Server	1	Optional. Configure a DHCP server to automatically acquire IP addresses for the management and workload networks as well as floating IPs. The DHCP server must support Client Identifiers and provide compatible DNS servers, DNS search domains, and an NTP server. The DHCP configuration is used by the Supervisor Cluster. Load balancers may require static IP addresses for Management. DHCP Scopes should not overlap these static IP's. DHCP is not used for virtual IPs. (VIPs)
Management Network Subnet	1	The subnet used for management traffic between ESXi hosts and vCenter Server, and the Kubernetes control plane. The size of the subnet must be the following: ■ One IP address per host VMkernel adapter. ■ One IP address for the vCenter Server Appliance. ■ 5 IP addresses for the Kubernetes control plane. 1 for each of the 3 nodes, 1 for virtual IP, 1 for rolling cluster upgrade. Note The Management Network and the Workload Network must be on different subnets. Assigning the same subnet to the Management and the Workload networks is not supported and can lead to system errors and problems.
Management Network VLAN	1	The VLAN ID of the Management Network subnet.
Physical Network MTU	1600	The MTU size must be 1600 or greater on any network that carries overlay traffic.
Kubernetes services CIDR range	/16 Private IP addresses	A private CIDR range to assign IP addresses to Kubernetes services. You must specify a unique Kubernetes services CIDR range for each Supervisor Cluster.

Topologies for Deploying the HAProxy Load Balancer

When using vSphere with Tanzu with vDS networking, HAProxy provides load balancing for developers accessing the Tanzu Kubernetes control plane, and for Kubernetes Services of Type Load Balancer. Review the possible topologies that you can implement for the HAProxy load balancer.

Workload Networks on the Supervisor Cluster

To configure a Supervisor Cluster with the vSphere networking stack, you must connect all hosts from the cluster to a vSphere Distributed Switch. Depending on the topology that you implement for the Supervisor Cluster Workload Networks, you create one or more distributed port groups. You designate the port groups as Workload Networks to vSphere Namespaces.

Before you add a host to a Supervisor Cluster, you must add it to all the vSphere Distributed Switches that are part of the cluster.

Workload Networks provide connectivity to the nodes of Tanzu Kubernetes clusters and to Supervisor Cluster control plane VMs. The Workload Network that provides connectivity to Kubernetes control plane VMs is called a Primary Workload Network. Each Supervisor Cluster must have one Primary Workload Network. You must designate one of the distributed port group as the Primary Workload Network to the Supervisor Cluster.

Note Workload Networks are only added when you enable the Supervisor Cluster and cannot be added later.

The Kubernetes control plane VMs on the Supervisor Cluster use three IP addresses from the IP address range that is assigned to the Primary Workload Network. Each node of a Tanzu Kubernetes cluster has a separate IP address assigned from the address range of the Workload Network that is configured with the namespace where the Tanzu Kubernetes cluster runs.

Allocation of IP Ranges

When you plan for the networking topology of the Supervisor Cluster with HA Proxy load balancer, plan for having two types of IP ranges:

- A range for allocating virtual IPs for HAProxy. The IP range that you configure for the virtual servers of HAProxy are reserved by the load balancer appliance. For example, if the virtual IP range is `192.168.1.0/24`, all hosts on that range are not accessible for traffic other than virtual IP traffic.

Note You must not configure a gateway within the HAProxy virtual IP range, because all routes to that gateway will fail.

- An IP range for the nodes of the Supervisor Cluster and Tanzu Kubernetes clusters. Each Kubernetes control plane VM in the Supervisor Cluster has an IP address assigned, which make three IP addresses in total. Each node of a Tanzu Kubernetes cluster also has a separate IP assigned. You must assign a unique IP range to each Workload Network on the Supervisor Cluster that you configure to a namespace.

An example configuration with one /24 network:

- Network: `192.168.120.0/24`
- HAProxy VIPs: `192.168.120.128/25`
- 1 IP address for the HAProxy workload interface: `192.168.120.5`

Depending on the IPs that are free within the first 128 addresses, you can define IP ranges for Workload Networks on the Supervisor Cluster, for example:

- `192.168.120.31-192.168.120.40` for the Primary Workload Network

- 192.168.120.51-192.168.120.60 for another Workload Network

Note The ranges that you define for Workload Networks must not overlap with the HAProxy VIP range.

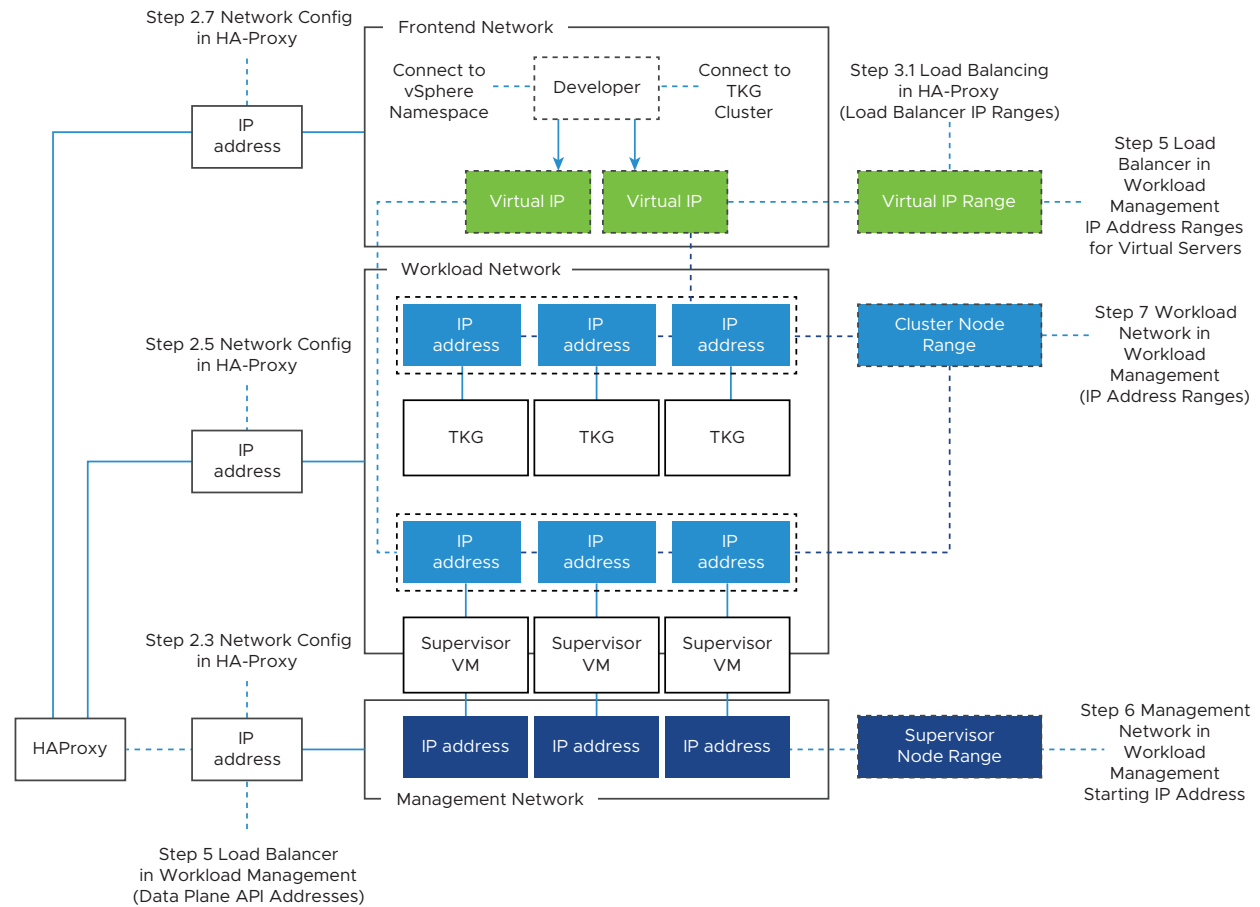
HAProxy Network Topology

There are two network configuration options for deploying HAProxy: **Default** and **Frontend**. The default network has 2 NICs: one for the Management network and one for the Workload network. The Frontend network has 3 NICs: Management network, Workload network, and the Frontend network for clients. The table lists and describes characteristics of each network.

For production installations, it is recommended that you deploy the HAProxy load balancer using the **Frontend Network** configuration. If you deploy the HAProxy load balancer using the **Default** configuration, it is recommended that you assign a /24 IP address block size to the Workload network. For both configuration options, DHCP is not recommended.

Network	Characteristics
Management	The Supervisor Cluster uses the Management network to connect to and program the HAProxy load balancer. ■ The HAProxy Data Plane API endpoint is bound to the network interface connected to the Management network. ■ The Management IP address assigned to the HAProxy control plane VM must be a static IP on the Management network so that the Supervisor Cluster can reliably connect to the load balancer API. ■ The default gateway for the HAProxy VM should be on this network. ■ DNS queries should occur on this network.
Workload	The HAProxy control plane VM uses the Workload network to access the services on the Supervisor Cluster and Tanzu Kubernetes cluster nodes. ■ The HAProxy control plane VM forwards traffic to the Supervisor and Tanzu Kubernetes cluster nodes on this network. ■ If the HAProxy control plane VM is deployed in Default mode (two NICs), the Workload network must provide the logical networks used to access the load balancer services. ■ In the Default configuration, the load balancer virtual IPs and the Kubernetes cluster node IPs will come from this network. They will be defined as separate, non-overlapping ranges within the network. Note The workload network must be on a different subnet than the management network. Refer to the System Requirements for Setting Up vSphere with Tanzu with vSphere Networking and HAProxy Load Balancer.
Frontend (optional)	External clients (such as users or applications) accessing cluster workloads use the Frontend network to access backend load balanced services using virtual IP addresses. ■ The Frontend network is only used when the HAProxy control plane VM is deployed with three NICs. ■ Recommended for production installations. ■ The Frontend network is where you expose the virtual IP address (VIP). HAProxy will balance and forward the traffic to the appropriate backend.

The diagram below illustrates an HAProxy deployment using a **Frontend Network** topology. The diagram indicates where configuration fields are expected during the installation and configuration process.



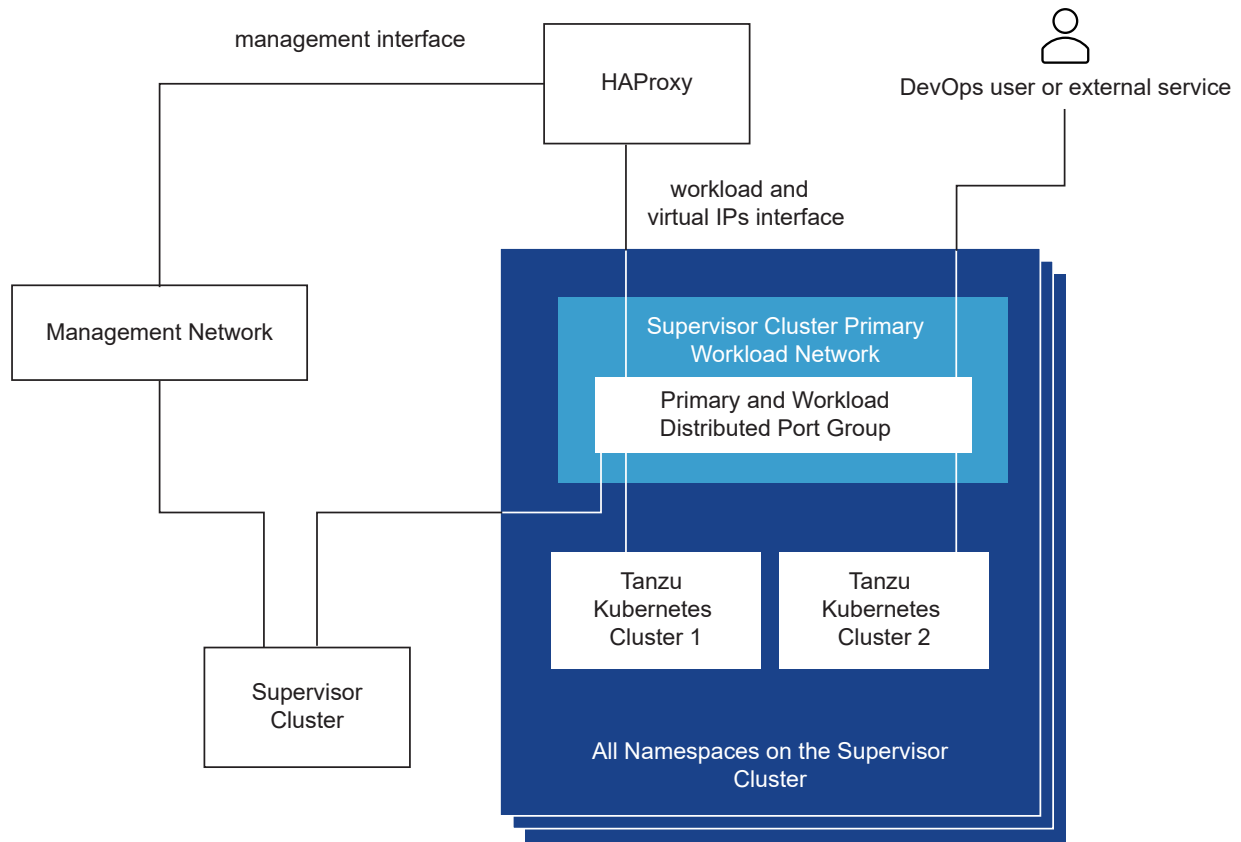
Supervisor Cluster Topology with One Workload Network and HA Proxy with Two Virtual NICs

In this topology, you configure a Supervisor Cluster with one Workload Network for the following components:

- Kubernetes control plane VMs
- The nodes of Tanzu Kubernetes clusters.
- The HAProxy virtual IP range where external services and DevOps users connect. In this configuration, HAProxy is deployed with two virtual NICs (**Default** configuration), one connected to the management network, and a second one connected to the Primary Workload Network. You must plan for allocating Virtual IPs on a separate subnet from the Primary Workload Network.

You designate one port group as the Primary Workload Network to the Supervisor Cluster and then use the same port group as the Workload Network for vSphere Namespaces. Supervisor Cluster, Tanzu Kubernetes clusters, HAProxy, DevOps users, and external services all connect to the same distributed port group that is set as the Primary Workload Network.

Figure 4-7. Supervisor Cluster Backed by One Network



The traffic path for DevOps users or external applications is the following:

- 1 The DevOps user or external service sends traffic to a virtual IP on the Workload Network subnet of the distributed port group.
- 2 HAProxy load balances the virtual IP traffic to either Tanzu Kubernetes node IP or control plane VM IP. HAProxy claims the virtual IP address so that it can load balance the traffic coming on that IP.
- 3 The control plane VM or Tanzu Kubernetes cluster node delivers the traffic to the target pods running inside the Supervisor Cluster or Tanzu Kubernetes cluster respectively.

Supervisor Cluster Topology with an Isolated Workload Network and HA Proxy with Two Virtual NICs

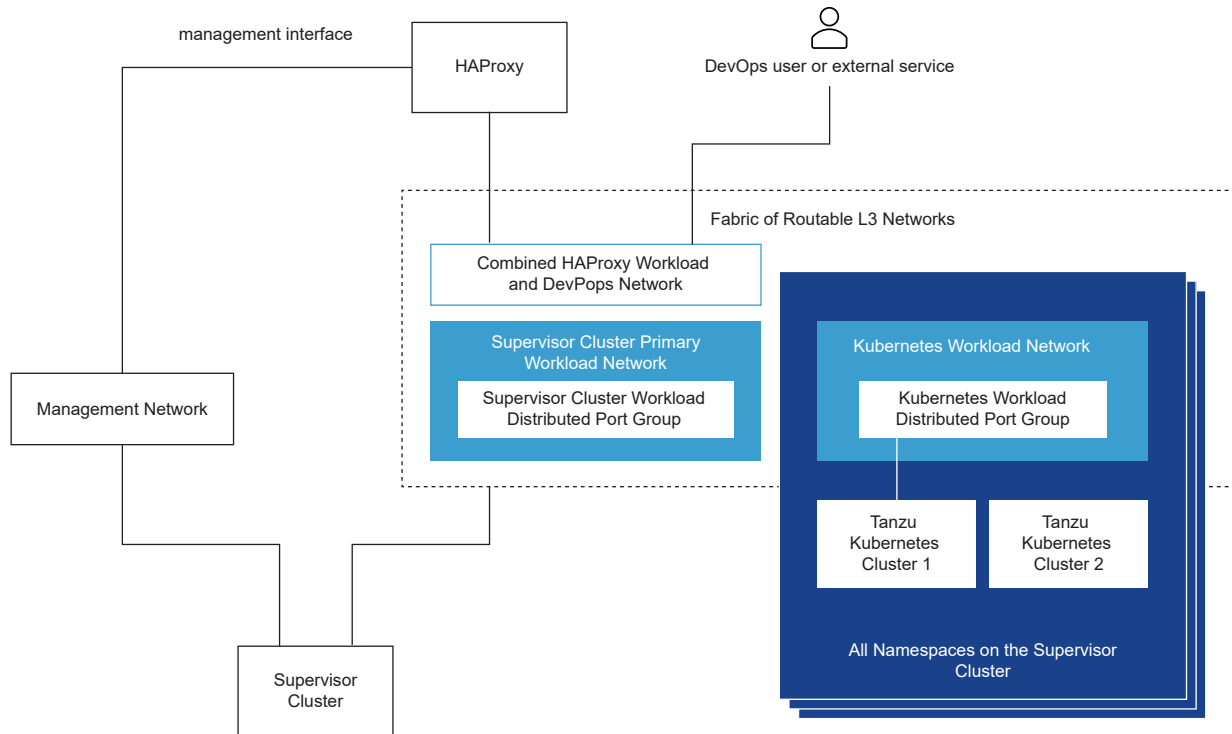
In this topology, you configure networks to the following components:

- Kubernetes control plane VMs. A Primary Workload Network to handle the traffic for Kubernetes control plane VMs.
- Tanzu Kubernetes cluster nodes. A Workload Network, that you assign to all namespaces on the Supervisor Cluster. This network connects the Tanzu Kubernetes cluster nodes.

- HAProxy virtual IPs. In this configuration, the HAProxy VM is deployed with two virtual NICs (**Default** configuration). You can either connect the HAProxy VM to the Primary Workload Network or the Workload Network that you use for namespaces. You can also connect HAProxy to a VM network that already exists in vSphere and is routable to the Primary and Workload networks.

The Supervisor Cluster is connected to the distributed port group backing the Primary Workload Network and Tanzu Kubernetes clusters are connected to a distributed port group backing the Workload Network. The two port groups must be layer 3 routable. You can implement layer 2 isolation through VLANs. Layer 3 traffic filtering is possible through IP firewalls and gateways.

Figure 4-8. Supervisor Cluster with an Isolated Workload Network



The traffic path for DevOps users or external service is the following:

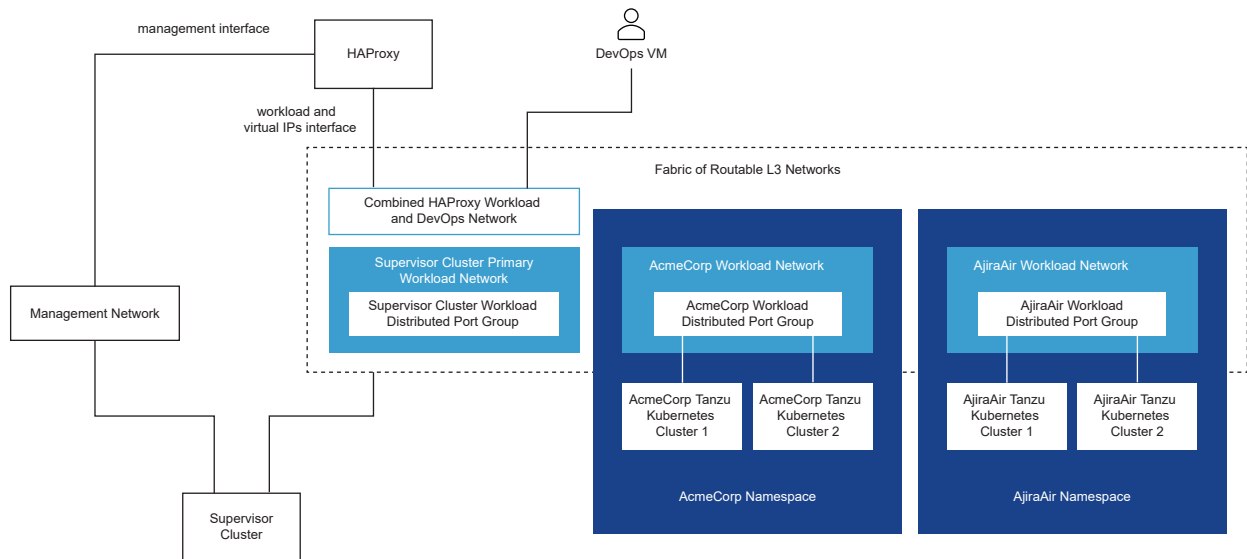
- 1 The DevOps user or external service sends traffic to a virtual IP. Traffic is routed to the network where HAProxy is connected.
- 2 HAProxy load balances the virtual IP traffic to either Tanzu Kubernetes node IP or control plane VM. HAProxy is claiming the virtual IP address so that it can load balance the traffic coming on that IP.
- 3 The control plane VM or Tanzu Kubernetes cluster node delivers the traffic to the target pods running inside the Tanzu Kubernetes cluster.

Supervisor Cluster Topology with Multiple Workload Networks and HA Proxy with Two Virtual NICs

In this topology, you can configure one port group to act as the Primary Workload Network and a dedicated port group to serve as the Workload Network to each namespace. HAProxy is deployed with two virtual NICs (**Default** configuration), and you can either connect it to the Primary Workload Network or to any of the Workload Networks. You can also use an existing VM network that is routable to the Primary and Workload Networks.

The traffic path for the DevOps users and external services in this topology is the same as with the isolated Workload Network topology.

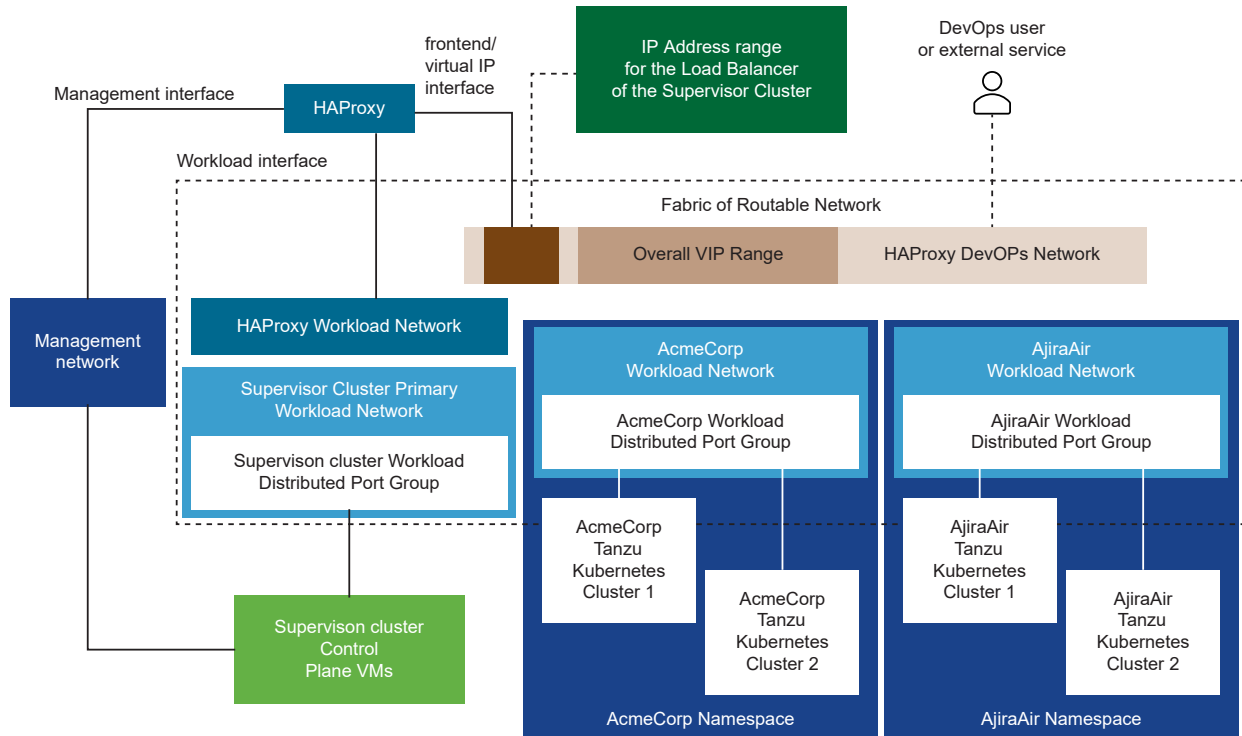
Figure 4-9. Supervisor Cluster Backed by Multiple Isolated Workload Networks



Supervisor Cluster Topology with Multiple Workload Networks and HAProxy with Three Virtual NICs

In this configuration, you deploy the HAProxy VM with three virtual NICs, thus connecting HAProxy to a Frontend network. DevOps users and external services can access HAProxy through virtual IPs on the Frontend network. Deploying HAProxy with three virtual NICs is recommended for production environments.

Figure 4-10. HAProxy Deployed with Three Virtual NICs



Selecting Between the Possible Topologies

Before you select between each of the possible topologies, assess the needs of your environment:

- 1 Do you need layer 2 isolation between Supervisor Cluster and Tanzu Kubernetes clusters?
 - a No: the simplest topology with one Workload Network serving all components.
 - b Yes: the isolated Workload Network topology with a separate Primary and Workload Networks.
- 2 Do you need further layer 2 isolation between your Tanzu Kubernetes clusters?
 - a No: isolated workload network topology with a separate Primary and Workload Networks.
 - b Yes: multiple workload networks topology with a separate Workload Network for each namespace and a dedicated Primary Workload Network.
- 3 Do you want to prevent your DevOps users and external services from directly routing to Kubernetes control plane VMs and Tanzu Kubernetes cluster nodes?
 - a No: two NIC HAProxy configuration.
 - b Yes: three NIC HAProxy configuration. This configuration is recommended for production environments

Considerations for Using the HAProxy Load Balancer with vSphere with Tanzu

Keep in mind the following considerations when configuring vSphere with Tanzu with the HAProxy load balancer.

- A support contract is required with [HAProxy](#) to get technical support for the HAProxy load balancer. VMware GSS cannot provide support for the HAProxy appliance.
- The HAProxy appliance is a singleton with no possibility for a highly available topology. For highly available environments, VMware recommends that you use either a full installation of NSX or the NSX Advanced Load Balancer.
- It is not possible to expand the IP address range used for the front-end at a later date, meaning the network should be sized for all future growth.

Create a vSphere Distributed Switch for a Supervisor Cluster for Use with HAProxy Load Balancer

To configure a vSphere cluster as a Supervisor Cluster that uses the vSphere networking stack and the HA proxy load balancer, you must add the hosts to a vSphere Distributed Switch. You must create port groups on the distributed switch that you will configure as Workload Networks to the Supervisor Cluster.

You can select between different topologies for the Supervisor Cluster depending on the level of isolation that you want to provide to the Kubernetes workloads that will run on the cluster.

Prerequisites

- Review the system requirements for using vSphere networking for the Supervisor Cluster with the HAProxy load balancer. See [System Requirements for Setting Up vSphere with Tanzu with vSphere Networking and HAProxy Load Balancer](#).
- Determine the topology for setting up Workload Networks with HA Proxy on the Supervisor Cluster. See [Topologies for Deploying the HAProxy Load Balancer](#).

Procedure

- 1 In the vSphere Client, navigate to a data center.
- 2 Right click the data center and select **Distributed Switch > New Distributed Switch**.
- 3 Enter a name for the switch, for example **Workload Distributed Switch** and click **Next**.
- 4 Select version 7.0 for the switch and click **Next**.
- 5 In **Port group name**, enter **Primary Workload Network**, click **Next**, and click **Finish**.

A new distributed switch with one port group is created on the data center. You can use this port group as the Primary Workload Network for the Supervisor Cluster that you will create. The Primary Workload Network handles the traffic for Kubernetes control plane VMs.

6 Create distributed port groups for Workload Networks.

The number of port groups that you create depends on the topology that you want to implement for the Supervisor Cluster. For a topology with one isolated Workload Network, create one distributed port group that you will use as a network for all namespaces on the Supervisor Cluster. For a topology with isolated networks for each namespace, create the same number of port groups as the number of namespaces that you will create.

- a Navigate to the newly-created distributed switch.
 - b Right-click the switch and select **Distributed Port Groups > New Distributed Port Group**.
 - c Enter a name for the port group, for example **Workload Network** and click **Next**.
 - d Leave the defaults, click **Next** and click **Finish**.
- 7 Add hosts from the vSphere clusters that you will configure as Supervisor Cluster to the distributed switch.
- a Right-click the distributed switch and select **Add and Manage Hosts**.
 - b Select **Add Hosts**.
 - c Click **New Hosts**, select the hosts from the vSphere cluster that you will configure as a Supervisor Cluster, and click **Next**.
 - d Select a physical NIC from each host and assign it an uplink on the distributed switch.
 - e Click **Next** through the remaining screens of the wizard and click **Finish**.

Results

The hosts are added to the distributed switch. You can now use the port groups that you created on the switch as Workload Networks of the Supervisor Cluster.

Install and Configure the HAProxy Load Balancer

VMware provides an implementation of the open source HAProxy load balancer that you can use in your vSphere with Tanzu environment. If you are using vSphere Distributed Switch (vDS) networking for **Workload Management**, you can install and configure the HAProxy load balancer.

Deploy the HAProxy Load Balancer Control Plane VM

If you want to use the vSphere networking stack for Kubernetes workloads, install the HAProxy control plane VM to provide load balancing services to Tanzu Kubernetes clusters.

Prerequisites

- Verify that your environment meets the compute and networking requirements for deploying HA Proxy. See [System Requirements for Setting Up vSphere with Tanzu with vSphere Networking and HAProxy Load Balancer](#).
- Verify that you have a Management network on a vSphere standard or distributed switch where to deploy the HAProxy load balancer. The Supervisor Cluster communicates with the HAProxy load balancer on that Management network.

- Create a vSphere Distributed Switch and port groups for Workload Networks. The HAProxy load balancer communicates with Supervisor Cluster and Tanzu Kubernetes cluster nodes over the Workload Networks. See [Create a vSphere Distributed Switch for a Supervisor Cluster for Use with HAProxy Load Balancer](#). For information on Workload Networks, see [Workload Networks on the Supervisor Cluster](#)
- Download the latest version of the VMware HAProxy OVA file from the [VMware-HAProxy site](#).
- Select a topology for deploying the HAProxy load balancer and Workload Networks on the Supervisor Cluster. See [Topologies for Deploying the HAProxy Load Balancer](#)

It may be helpful to view a demonstration of how to use vSphere with Tanzu with vDS networking and HAProxy. Check out the video [Getting Started Using vSphere with Tanzu](#).

Procedure

- 1 Log in to the vCenter Server using the vSphere Client.
- 2 Create a new VM from the HAProxy OVA file.

Option	Description
Content Library	If you imported the OVA to a Local Content Library: ■ Go to Menu > Content Library. ■ Select the library where you imported the OVA. ■ Select the <code>vmware-haproxy-vX.X.X</code> template. ■ Right-click and choose New VM from This Template.
Local file	If you downloaded the OVA file to your local host: ■ Select the vCenter cluster where you will enable Workload Management. ■ Right-click and select Deploy OVF Template. ■ Select Local File and click Upload Files. ■ Browse to and select the <code>vmware-haproxy-vX.X.X.ova</code> file.

- 3 Enter a **Virtual machine name**, such as `haproxy`.
- 4 Select the **Datacenter** where you are deploying HAProxy and click **Next**.
- 5 Select the vCenter Cluster where you will enable **Workload Management** and click **Next**.
- 6 Review and confirm the deployment details and click **Next**.
- 7 Accept the License agreements and click **Next**.

- 8 Select a deployment configuration. See [HAProxy Network Topology](#) for details.

Configuration	Description
Default	Select this option to deploy the appliance with 2 NICs: a Management network and a single Workload network.
Frontend Network	Select this option to deploy the appliance with 3 NICs. The frontend subnet is used to isolate cluster nodes from the network used by developers to access the cluster control plane.

- 9 Select the storage policy to use for the VM and click **Next**.
- 10 Select the network interfaces to use for the load balancer and click **Next**.

Source Network	Destination Network
Management	Select the Management network, such as VM Network .
Workload	Select the vDS portgroup configured for Workload Management .
Frontend	Select the vDS portgroup configured for the Frontend subnet. If you did not select Frontend configuration, this setting is ignored during installation, so you can leave the default.

Note The workload network must be on a different subnet than the management network. Refer to the [System Requirements for Setting Up vSphere with Tanzu with vSphere Networking and HAProxy Load Balancer](#).

- 11 Customize the application configuration settings. See [Appliance Configuration Settings](#).
- 12 Provide the network configuration details. See [Network Configuration](#).
- 13 Configure load balancing. See [Load Balancing Settings](#).
- 14 Click **Next** to complete the configuration of the OVA.
- 15 Review the deployment configuration details and click **Finish** to deploy the OVA.
- 16 Monitor the deployment of the VM using the **Tasks** panel.
- 17 When the VM deployment completes, power it on.

What to do next

Once the HAProxy load balancer is successfully deployed and powered on, proceed with enabling **Workload Management**. See [Chapter 5 Configuring and Managing a Supervisor Cluster](#).

Customize the HAProxy Load Balancer

Customize the HAProxy control plane VM, including configuration settings, network settings, and load balancing settings.

Appliance Configuration Settings

The table lists and describes the parameters for HAProxy appliance configuration.

Parameter	Description	Remark or Example
Root Password	Initial password for the root user (6-128 characters).	Subsequent changes of password must be performed in operating system.
Permit Root Login	Option to allow the root user to login to the VM remotely over SSH.	Root login might be needed for troubleshooting, but keep in mind the security implications of allowing it.
TLS Certificate Authority (ca.crt)	To use the self-signed CA certificate, leave this field empty. To use your own CA certificate (ca.crt), paste its contents into this field. You might need to Base64-encode the contents. https://www.base64encode.org/	If you are using the self-signed CA certificate, the public and private keys will be generated from the certificate.
Key (ca.key)	If you are using the self-signed certificate, leave this field empty. If you provided a CA certificate, paste the contents of the certificate private key in this field.	

Network Configuration

The table lists and describes the parameters for HAProxy network configuration.

Parameter	Description	Remark or Example
Host Name	The host name (or FQDN) to assign to the HAProxy control plane VM	Default value: <code>haproxy.local</code>
DNS	A comma-separated list of DNS server IP addresses.	Default values: <code>1.1.1.1, 1.0.0.1</code> Example value: <code>10.8.8.8</code>
Management IP	The static IP address of the HAProxy control plane VM on the Management network.	A valid IPv4 address with the prefix length of the network, for example: <code>192.168.0.2/24</code> .
Management Gateway	The IP address of the gateway for the Management network.	For example: <code>192.168.0.1</code>
Workload IP	The static IP address of the HAProxy control plane VM on the Workload network. This IP address must be outside of the load balancer IP address range.	A valid IPv4 address with the prefix length of the network, for example: <code>192.168.10.2/24</code> .
Workload Gateway	The IP address of the gateway for the Workload network.	For example: <code>192.168.10.1</code> If you select Frontend configuration, you must enter a gateway. The deployment will not be successful if Frontend is selected and no gateway is specified.

Parameter	Description	Remark or Example
Frontend IP	The static IP address of the HAProxy appliance on the Frontend network. This value is only used when the Frontend deployment model is selected.	A valid IPv4 address with the prefix length of the network, for example: 192.168.100.2/24
Frontend Gateway	The IP address of the gateway for the Frontend network. This value is only used when the Frontend deployment model is selected.	For example: 192.168.100.1

Load Balancing Settings

The table lists and describes the parameters for HAProxy load balancer configuration.

Parameter	Description	Example or Remark
Load Balancer IP Range(s)	In this field you specify a range of IPv4 addresses using CIDR format. The value must be a valid CIDR range or the installation will fail. HAProxy reserves the IP addresses for virtual IPs (VIPs). Once assigned, each VIP address is allocated and HAProxy replies to requests on that address. The CIDR range you specify here must not overlap with the IPs you assign for the Virtual Servers when you enable Workload Management in the vCenter Server using the vSphere Client. Note The load balancer IP range must reside on a different subnet than the Management network. It's not supported to have the load balancer IP range on the same subnet as the Management network.	For example, the network CIDR 192.168.100.0/24 gives the load balancer 256 virtual IP addresses with range 192.168.100.0 - 192.168.100.255. For example, the network CIDR 192.168.100.0/25 gives the load balancer 128 virtual IP addresses with range 192.168.100.0 - 192.168.100.127.
Dataplane API Management Port	The port on the HAProxy VM on which the load balancer's API service listens.	A valid port. Port 22 is reserved for SSH. The default value is 5556.

Parameter	Description	Example or Remark
HAProxy User ID	Load balancer API user name	The username clients use to authenticate to the load balancer's API service. Note You need this username when you enable the Supervisor Cluster.
HAProxy Password	Load balancer API password	The password clients use to authenticate to the load balancer's API service. Note You need this password when you enable the Supervisor Cluster.

Configuring and Managing a Supervisor Cluster

5

As a vSphere Administrator, you enable a vSphere cluster for Workload Management by creating a Supervisor Cluster. You can select between creating the Supervisor Cluster with the vSphere networking stack or with NSX-T Data Center as the networking solution. A cluster configured with NSX-T Data center supports running a vSphere Pod and a Tanzu Kubernetes cluster created through the VMware Tanzu™ Kubernetes Grid™ Service. A Supervisor Cluster that is configured with the vSphere networking stack only supports Tanzu Kubernetes clusters.

After you enable the Supervisor Cluster, you can use the vSphere Client to manage and monitor the cluster.

Read the following topics next:

- [Prerequisites for Configuring vSphere with Tanzu on a vSphere Cluster](#)
- [Enable Workload Management with vSphere Networking](#)
- [Enable Workload Management with NSX Networking](#)
- [Assign the Tanzu Edition License to a Supervisor Cluster](#)
- [Replace the VIP Certificate to Securely Connect to the Supervisor Cluster API Endpoint](#)
- [Integrate the Tanzu Kubernetes Grid Service on the Supervisor Cluster with Tanzu Mission Control](#)
- [Set the Default CNI for Tanzu Kubernetes Clusters](#)
- [Add Workload Networks to a Supervisor Cluster Configured with VDS Networking](#)
- [Change the Control Plane Size of a Supervisor Cluster](#)
- [Change the Management Network Settings on a Supervisor Cluster](#)
- [Change the Workload Network Settings on a Supervisor Cluster Configured with VDS Networking](#)
- [Change Workload Network Settings on a Supervisor Cluster Configured with NSX-T Data Center](#)
- [Resolving Errors Health Statuses on Supervisor Cluster During Initial Configuration Or Upgrade](#)
- [Configuring HTTP Proxy Settings in vSphere with Tanzu for use with Tanzu Mission Control](#)

- [Streaming Logs of the Supervisor Cluster Control Plane to a Remote rsyslog](#)
- [Registering Tanzu Mission Control Self-Managed with Supervisor Cluster](#)

Prerequisites for Configuring vSphere with Tanzu on a vSphere Cluster

Check out the prerequisites for enabling vSphere with Tanzu in your vSphere environment. To run container-based workloads natively on vSphere, as a vSphere administrator you enable **Workload Management** on a vSphere cluster. The result is a Kubernetes management cluster known as a Supervisor Cluster where you run vSphere Pods, provision Tanzu Kubernetes clusters, and VMs.

Create and Configure a vSphere Cluster

A vSphere cluster is a collection of ESXi hosts managed by a vCenter Server system. The Supervisor Cluster runs on a vSphere cluster. Create a vSphere cluster that meets the following requirements so that you can enable **Workload Management** on it:

- Create and configure a vSphere cluster with at least three ESXi hosts. If you are using vSAN, it is recommended but not required to use four ESXi hosts. See [Creating and Configuring Clusters](#).
- Configure the cluster with shared storage such as vSAN. Shared storage is required for vSphere HA, DRS, and for storing persistent container volumes. See [Creating a vSAN Cluster](#).
- If you plan to use persistent volumes in ReadWriteMany mode, enable File Services on the vSAN cluster. See [Creating ReadWriteMany Persistent Volumes in vSphere with Tanzu](#).
- Enable the cluster with vSphere HA. See [Creating and Using vSphere HA Clusters](#).
- Enable the cluster with vSphere DRS in fully-automated mode. See [Creating a DRS Cluster](#).
- Verify that your user account has the **Modify cluster-wide configuration** on the vSphere cluster so that you can enable the **Workload Management** functionality.
- If you want to use vSphere Lifecycle Manager images with the Supervisor Cluster, switch the vSphere cluster where you want to activate **Workload Management** to use vSphere Lifecycle Manager images before activating **Workload Management**. You can manage the lifecycle of a Supervisor Cluster with either vSphere Lifecycle Manager baselines or vSphere Lifecycle Manager images. However, you cannot convert a Supervisor Cluster that uses vSphere Lifecycle Manager baselines to a Supervisor Cluster that uses vSphere Lifecycle Manager images. Therefore, switching the vSphere cluster to using vSphere Lifecycle Manager images before you activate **Workload Management** is required.

Caution Do not disable vSphere DRS after you configure the Supervisor Cluster. Having DRS enabled at all times is a mandatory prerequisite for running workloads on the Supervisor Cluster. Disabling DRS leads to breaking your Tanzu Kubernetes clusters.

Choose and Configure the Networking Stack

To enable **Workload Management** on a vSphere cluster, you must configure the networking stack to be used for the Supervisor Cluster. You have two options: NSX or vSphere Distributed Switch (vDS) networking with a load balancer. You can configure the NSX Advanced Load Balancer or the HAProxy load balancer.

The table lists the high-level differences between the two supported networking stacks. For more information about the architectural differences, see [Supervisor Cluster Configured with the vSphere Networking Stack](#).

Functionality	NSX-T Networking	vDS Networking
vSphere Pods	Yes	No
Tanzu Kubernetes clusters	Yes	Yes
Embedded Harbor Registry	Yes	No
Load Balancing	Yes	Yes, by installing and configuring the NSX Advanced Load Balancer or the HAProxy load balancer.

To use NSX networking for the Supervisor Cluster:

- Review the system requirements and topologies for NSX-T networking. See [System Requirements for Setting Up vSphere with Tanzu with NSX](#).
- Install and configure NSX-T Data Center for vSphere with Tanzu. See [Install and Configure NSX for vSphere with Tanzu](#).

To use vSphere vDS networking with the NSX Advanced Load Balancer for the Supervisor Cluster:

- Review the NSX Advanced Load Balancer requirements. See [System Requirements for Setting Up vSphere with Tanzu with vSphere Networking and NSX Advanced Load Balancer](#).
- Create a vSphere Distributed Switch (vDS) and add all ESXi hosts from the cluster to the vDS and create port groups for Workload Networks. See [Create a vSphere Distributed Switch for a Supervisor Cluster for Use with NSX Advanced Load Balancer](#).
- Deploy and configure the NSX Advanced Load Balancer. See [Deploy the Controller](#).

Note vSphere with Tanzu supports the NSX Advanced Load Balancer with vSphere 7 U2 and later.

To use vSphere vDS networking with HAProxy load balancing for the Supervisor Cluster:

- Review the system requirements and network topologies for vSphere networking with an external load balancer. See [System Requirements for Setting Up vSphere with Tanzu with vSphere Networking and HAProxy Load Balancer](#) and [Topologies for Deploying the HAProxy Load Balancer](#).

- Create a vSphere Distributed Switch (vDS) and add all ESXi hosts from the cluster to the vDS and create port groups for Workload Networks. See [Create a vSphere Distributed Switch for a Supervisor Cluster for Use with HAProxy Load Balancer](#).
- Install and configure HAProxy load balancer instance that is routable to the vSphere Distributed Switch that is connected to the hosts from the vSphere cluster. The HAProxy load balancer supports the network connectivity to workloads from client networks and to load balance traffic between Tanzu Kubernetes clusters. See [Install and Configure the HAProxy Load Balancer](#).

Note vSphere with Tanzu supports the HAProxy load balancer with vSphere 7 U1 and later.

Create Storage Policy

You must create storage policies that will determine the datastore placement of the Kubernetes control plane VMs, containers, and images. You can create storage policies associated with different storage classes.

Before enabling **Workload Management** on a vSphere cluster, create a storage policy for the placement of Kubernetes control plane VMs. See [Create Storage Policies for vSphere with Tanzu](#).

Create a Content Library

To provision Tanzu Kubernetes clusters and VMs, you need a **Content Library** created in the vCenter Server that manages the vSphere cluster where the Supervisor Cluster runs.

The **Content Library** provides the system with the distributions of Tanzu Kubernetes releases in the form of OVA templates. When you provision a Tanzu Kubernetes cluster, the OVA template for the selected version is used to create the Kubernetes cluster nodes.

You can create a **Subscribed Content Library** to automatically pull the latest released images, or you can create a **Local Content Library** and manually upload the images, which may be required for air-gapped provisioning of Tanzu Kubernetes clusters.

See [Creating and Managing Content Libraries for Tanzu Kubernetes releases](#).

Watch vSphere with Tanzu Demos

Although not a hard requirement, before you embark it may be helpful to watch some demonstrations of vSphere with Tanzu, including setting up the vSphere environment in preparation for deploying the Supervisor Cluster, enabling **Workload Management**, and provisioning Tanzu Kubernetes clusters. If this sounds useful, check out the series of [vSphere with Tanzu Deep Dive](#) videos on the VMware vSphere channel. You can also check the [vSphere Tanzu Quick Bytes](#) series of short videos for configuring **Workload Management** with vDS networking and HAProxy Load Balancer.

Enable Workload Management with vSphere Networking

As a vSphere administrator, you can enable the **Workload Management** platform on a vSphere cluster by configuring the vSphere networking stack to provide connectivity to workloads. A Supervisor Cluster that is configured with vSphere networking supports the deployment of Tanzu Kubernetes clusters created by using the Tanzu Kubernetes Grid Service. It does not support running vSphere Pod or using the embedded Harbor Registry.

Caution Do not disable vSphere DRS after you configure the Supervisor Cluster. Having DRS enabled at all times is a mandatory prerequisite for running workloads on the Supervisor Cluster. Disabling DRS leads to breaking your Tanzu Kubernetes clusters.

Prerequisites

- Complete the prerequisites for configuring a vSphere cluster as a Supervisor Cluster. See [Prerequisites for Configuring vSphere with Tanzu on a vSphere Cluster](#).

Procedure

- 1 From the home menu, select **Workload Management**.
- 2 Select a licensing option for the Supervisor Cluster.
 - If you have a valid Tanzu Edition license, click **Add License** to add the license key to the license inventory of vSphere.
 - If you do not have a Tanzu edition license yet, enter the contact details so that you can receive communication from VMware and click **Get Started**.

The evaluation period of a Supervisor Cluster lasts for 60 days. Within that period, you must assign a valid Tanzu Edition license to the cluster. If you added a Tanzu Edition license key, you can assign that key within the 60 day evaluation period once you complete the Supervisor Cluster setup.

- 3 On the **Workload Management** screen, click **Get Started** again.
- 4 Select a vCenter Server system, select **vCenter Server Network**, and click **Next**.
- 5 Select a cluster from the list of compatible clusters.
- 6 From the **Control Plane Size** page, select the size for the Kubernetes control plane VMs that will be created on each host from the cluster.

The amount of resources that you allocate to control plane VMs determines the number of Kubernetes workloads that the Supervisor Cluster can manage.

- 7 On the **Load Balancer** screen, select the load balancer you want to use. You can select NSX Advanced Load Balancer or HAProxy.

- Enter the following settings for NSX Advanced Load Balancer:

Option	Description
Name	Enter a name for the NSX Advanced Load Balancer.
Avi Controller IP	The IP address of the NSX Advanced Load Balancer Controller. The default port is 443.
User name	The user name that is configured with the NSX Advanced Load Balancer. You use this username to access the Controller.
Password	The password for the user name.
Server Certificate Authority	The certificate used by the Controller. You can provide the certificate that you assigned during the configuration. For more information, see Assign a Certificate to the Controller .

- Enter the following settings for HAProxy:

Option	Description
Name	A user-friendly name for the load balancer.
Data plane API Address(es)	The IP address and port of the HAProxy Data Plane API. This component controls the HAProxy server and runs inside the HAProxy VM. This is the management network IP address of the HAProxy appliance.
User name	The user name that is configured with the HAProxy OVA file. You use this name to authenticate with the HAProxy Data Plane API.
Password	The password for the user name.

Option	Description
IP Address Ranges for Virtual Servers	Range of IP addresses that is used in the Workload Network by Tanzu Kubernetes clusters. This IP range comes from the list of IPs that were defined in the CIDR you configured during the HAProxy appliance deployment. Typically this will be the entire range specified in the HAProxy deployment, but it can also be a subset of that CIDR because you may create multiple Supervisor Clusters and use IPs from that CIDR range. This range must not overlap with the IP range defined for the Workload Network in this wizard. The range must also not overlap with any DHCP scope on this Workload Network.
Server Certificate Authority	The certificate in PEM format that is signed or is a trusted root of the server certificate that the Data Plane API presents. ■ Option 1: If root access is enabled, SSH to the HAProxy VM as root and copy <code>/etc/haproxy/ca.crt</code> to the Server Certificate Authority. Do not use escape lines in the <code>\n</code> format. ■ Option 2: Right-click the HAProxy VM and select Edit Settings. Copy the CA cert from the appropriate field and convert it from Base64 using an conversion tool such as https://www.base64decode.org/. ■ Option 3: Run the following PowerCLI script. Replace the variables <code>\$vc</code>, <code>\$vc_user</code>, and <code>\$vc_password</code> with appropriate values. <pre> \$vc = "10.21.32.43" \$vc_user = "administrator@vsphere.local" \$vc_password = "PASSWORD" Connect-VIServer -User \$vc_user -Password \$vc_password -Server \$vc \$VMname = "haproxy-demo" \$AdvancedSettingName = "guestinfo.dataplaneapi.cacert" \$Base64cert = get-vm \$VMname Get- AdvancedSetting -Name \$AdvancedSettingName while ([string]::IsNullOrEmpty(\$Base64cert.V alue)) { Write-Host "Waiting for CA Cert Generation... This may take a under 5-10 minutes as the VM needs to boot and generate the CA Cert (if you haven't provided one already)." \$Base64cert = get-vm \$VMname Get-AdvancedSetting -Name \$AdvancedSettingName Start-sleep -seconds 2 } </pre>

Option	Description
	<pre>Write-Host "CA Cert Found... Converting from BASE64" \$cert = [Text.Encoding]::Utf8.GetString([Conve rt]::FromBase64String(\$Base64cert.Valu e)) Write-Host \$cert</pre>

- 8 On the **Management Network** screen, configure the parameters for the network that will be used for Kubernetes control plane VMs.

a Select a **Network Mode**.

- **DHCP Network.** In this mode, all the IP addresses for the management network, such as control plane VM IP addresses, a Floating IP, DNS servers, DNS, search domains, and NTP server are acquired automatically from a DHCP server. To obtain floating IP addresses, the DHCP server must be configured to support client identifiers.

Note In DHCP mode, all control plane VMs use stable DHCP client identifiers to acquire IP addresses. These client identifiers can be used to setup static IP assignment for the IP addresses of the control plane VMs on the DHCP server to ensure they do not change. Changing the IP addresses of control plane VMs and floating IP addresses is not supported. Failure to reserve the floating IP addresses by its DHCP Unique Identifier might result in unrecoverable loss of connectivity to the Supervisor Control plane.

- **Static.** Manually enter all networking settings for the management network.

b Configure the settings for the management network.

If you have selected the DHCP network mode, but you want to override the settings acquired from DHCP, click **Additional Settings** and enter new values. If you have selected the static network mode, fill in the values for the management network settings manually.

Option	Description
Network	Select a network that has a VMkernel adapter configured for the management traffic.
Starting Control IP address	Enter an IP address that determines the starting point for reserving five consecutive IP addresses for the Kubernetes control plane VMs as follows: ■ An IP address for each of the Kubernetes control plane VMs. ■ A floating IP address for one of the Kubernetes control plane VMs to serve as an interface to the management network. The control plane VM that has the floating IP address assigned acts as a leading VM for all three Kubernetes control plane VMs. The floating IP moves to the control plane node that is the etcd leader in this Kubernetes cluster. This improves availability in the case of a network partition event. ■ An IP address to serve as a buffer in case a Kubernetes control plane VM fails and a new control plane VM is being brought up to replace it.
Subnet Mask	Only applicable for static IP configuration. Enter the subnet mask for the management network. For example, 255.255.255.0
DNS Servers	Enter the addresses of the DNS servers that you use in your environment. If the vCenter Server system is registered with an FQDN, you must enter the IP addresses of the DNS servers that you use with the vSphere environment so that the FQDN is resolvable in the Supervisor Cluster.

Option	Description
DNS Search Domains	Enter domain names that DNS searches inside the Kubernetes control plane nodes, such as <code>corp.local</code> , so that the DNS server can resolve them.
NTP	Enter the addresses of the NTP servers that you use in your environment, if any.

- 9 In the **Workload Network** page, enter the settings for the network that will handle the networking traffic for Kubernetes workloads running on the Supervisor Cluster.

Note If you select using a DHCP server to provide the networking settings for Workload Networks, you will not be able to create any new Workload Networks once you complete the Supervisor Cluster configuration.

- a Select a network mode.
 - **DHCP Network.** In this network mode, all networking settings for Workload Networks are acquired through DHCP.
 - **Static.** Manually configure Workload Network settings.
- b Select the port group that will serve as the Primary Workload Network to the Supervisor Cluster

The primary network handles the traffic for the Kubernetes control plane VMs and Kubernetes workload traffic.

Depending on your networking topology, you can later assign a different port group to serve as the network to each namespace. This way, you can provide layer 2 isolation between the namespaces in the Supervisor Cluster. Namespaces that do not have a different port group assigned as their network use the primary network. Tanzu Kubernetes clusters use only the network that is assigned to the namespace where they are deployed or they use the primary network if there is no explicit network assigned to that namespace

- c Configure the settings for workload networks.

If you have selected the DHCP network mode, all the values under the **Additional Settings** section are automatically filled in from the DHCP server. If You want to override these values, click **Additional Settings** and enter new values. If you have selected the **Static** network mode, fill in all the settings manually.

Option	Description
Internal Network for Kubernetes Services	Enter a CIDR notation that determines the range of IP addresses for Tanzu Kubernetes clusters and services that run inside the clusters.
Network Name	Enter the network name.

Option	Description
DNS Server	Enter the IP addresses of the DNS servers that you use with your environment, if any. For example, 10.142.7.1. When you enter IP address of the DNS server, a static route is added on each control plane VM. This indicates that the traffic to the DNS servers goes through the workload network. If the DNS servers that you specify are shared between the management network and workload network, the DNS lookups on the control plane VMs are routed through the workload network after initial setup.
Gateway	Enter the gateway for the primary network.
Subnet Mask IP	Enter the subnet mask IP address.
IP Address Ranges	Enter an IP range for allocating IP address of Kubernetes control plane VMs and workloads. This address range connects the Supervisor Cluster nodes and, in the case of a single Workload Network, also connects the Tanzu Kubernetes cluster nodes. This IP range must not overlap with the load balancer VIP range when using the Default configuration for HAProxy.

10 On the **Storage** page, configure storage and file volume support.

- a Select a storage policy for the Supervisor Cluster.

The storage policy you select ensures that the control plane VMs of the Supervisor Cluster are placed on the datastore referenced in the storage policy.

Option	Description
Control Plane Node	Select the storage policy for placement of the control plane VMs.

- b (Optional) Activate file volume support.

This option is required if you plan to deploy ReadWriteMany persistent volumes on a cluster. See [Creating ReadWriteMany Persistent Volumes in vSphere with Tanzu](#).

11 On the **Tanzu Kubernetes Grid** page, click **Add** and select the subscribed content library that contains the VM images for deploying the nodes of Tanzu Kubernetes clusters.

12 Review your settings and click **Finish**.

Results

A task runs on vCenter Server that creates the Supervisor Cluster. Once the task completes, three Kubernetes control plane VMs are created on the hosts that are part of the vSphere cluster.

What to do next

Create and configure vSphere Namespaces on the Supervisor Cluster. See [Create and Configure a vSphere Namespace](#).

Enable Workload Management with NSX Networking

As a vSphere administrator, you can configure a vSphere cluster as a Supervisor Cluster that uses the NSX networking stack to provide connectivity to Kubernetes workloads.

Prerequisites

- Verify that your environment meets the prerequisites for configuring a vSphere cluster as a Supervisor Cluster. For information about requirements, see [Prerequisites for Configuring vSphere with Tanzu on a vSphere Cluster](#).

Caution Do not disable vSphere DRS after you configure the Supervisor Cluster. Having DRS enabled at all times is a mandatory prerequisite for running workloads on the Supervisor Cluster. Disabling DRS leads to breaking your Tanzu Kubernetes clusters.

Procedure

- 1 From the vSphere Client home menu, select **Workload Management**.
- 2 Click **Get Started**.
- 3 Select the vCenter Server system that you want to configure.
- 4 Select the **NSX** networking stack.
- 5 Click **Next**.
- 6 Select **Select a Cluster > Datacenter**.
- 7 Select a cluster from the list of compatible clusters and click **Next**.
- 8 From the **Control Plane Size** page, select the select the sizing for the control plane VMs.

The size of the control plane VMs determines the amount of workloads that you can run on the Supervisor Cluster.

Refer to the [VMware Configuration Maximums](#) site for guidance.
- 9 Click **Next**.

10 On the **Management Network** screen, configure the parameters for the network that will be used for Kubernetes control plane VMs.

a Select a **Network Mode**.

- **DHCP Network.** In this mode, all the IP addresses for the management network, such as control plane VM IPs, DNS servers, DNS, search domains, and NTP server are acquired automatically from a DHCP.
- **Static.** Manually enter all networking settings for the management network.

b Configure the settings for the management network.

If you have selected the DHCP network mode, but you want to override the settings acquired from DHCP, click **Additional Settings** and enter new values. If you have selected the static network mode, fill in the values for the management network settings manually.

Option	Description
Network	Select a network that has a VMkernel adapter configured for the management traffic.
Starting Control IP address	Enter an IP address that determines the starting point for reserving five consecutive IP addresses for the Kubernetes control plane VMs as follows: ■ An IP address for each of the Kubernetes control plane VMs. ■ A floating IP address for one of the Kubernetes control plane VMs to serve as an interface to the management network. The control plane VM that has the floating IP address assigned acts as a leading VM for all three Kubernetes control plane VMs. The floating IP moves to the control plane node that is the etcd leader in this Kubernetes cluster, which is the Supervisor Cluster. This improves availability in the case of a network partition event. ■ An IP address to serve as a buffer in case a Kubernetes control plane VM fails and a new control plane VM is being brought up to replace it.
Subnet Mask	Only applicable for static IP configuration. Enter the subnet mask for the management network. For example, 255.255.255.0
DNS Servers	Enter the addresses of the DNS servers that you use in your environment. If the vCenter Server system is registered with an FQDN, you must enter the IP addresses of the DNS servers that you use with the vSphere environment so that the FQDN is resolvable in the Supervisor Cluster.
DNS Search Domains	Enter domain names that DNS searches inside the Kubernetes control plane nodes, such as <code>corp.local</code> , so that the DNS server can resolve them.
NTP	Enter the addresses of the NTP servers that you use in your environment, if any.

11 In the **Workload Network** pane, configure settings for the networks for namespaces.

The namespace network settings provide connectivity to vSphere Pods and namespaces running in the Supervisor Cluster. By default, the namespace will use cluster level network configuration.

Option	Description
vSphere Distributed Switch	Select the vSphere Distributed Switch that handles overlay networking for the Supervisor Cluster. For example, select <code>DSwitch</code> .
DNS Server	Enter the IP addresses of the DNS servers that you use with your environment, if any. For example, <code>10.142.7.1</code> .
API Server Endpoint FQDN	Optionally, enter the FQDN of the API server endpoint.
Edge Cluster	Select the NSX Edge cluster that has the tier-0 gateway that you want to use for namespace networking. For example, select <code>EDGE-CLUSTER</code> .
Tier-0 Gateway	Select the tier-0 gateway to associate with the cluster tier-1 gateway.
NAT Mode	The NAT mode is selected by default. If you deselect the option, all the workloads such as the vSphere Pods, VMs, and Tanzu Kubernetes clusters Node IP addresses are directly accessible from outside the tier-0 gateway and you do not have to configure the egress CIDRs. Note If you deselect NAT mode, File Volume storage is not supported.
Namespace Network	Enter one or more IP CIDRs to create subnets/segments and assign IP addresses to workloads.
Namespace Subnet Prefix	Enter the subnet prefix that specifies the size of the subnet reserved for namespaces segments. Default is 28.
Services CIDRs	Enter a CIDR annotation to determine the IP range for Kubernetes services. You can use the default value.
Ingress CIDRs	Enter a CIDR annotation that determines the ingress IP range for the Kubernetes services. This range is used for services of type load balancer and ingress.
Egress CIDRs	Enter a CIDR annotation that determines the egress IP for Kubernetes services. Only one egress IP address is assigned for each namespace in the Supervisor Cluster. The egress IP is the IP address that the vSphere Pods in the particular namespace use to communicate outside of NSX.

12 Click **Next**.

13 On the **Storage** page, configure storage and file volume support.

- a Select storage policies for the Supervisor Cluster.

The storage policy you select for each of the following objects ensures that the object is placed on the datastore referenced in the storage policy. You can use the same or different storage policies for the objects.

Option	Description
Control Plane Node	Select the storage policy for placement of the control plane VMs.
Pod Ephemeral Disks	Select the storage policy for placement of the vSphere Pods.
Container Image Cache	Select the storage policy for placement of the cache of container images.

- b (Optional) Activate file volume support.

This option is required if you plan to deploy ReadWriteMany persistent volumes on a cluster. See [Creating ReadWriteMany Persistent Volumes in vSphere with Tanzu](#).

14 In the **Ready to Complete** section, review the settings and **Finish**.

The cluster is enabled with vSphere with Tanzu and you can create vSphere Namespaces to provide to DevOps engineers. Kubernetes control plane nodes are created on the hosts that are part of the cluster and the Spherelet process.

What to do next

Create and configure a vSphere Namespace on the Supervisor Cluster. See [Create and Configure a vSphere Namespace](#)

Assign the Tanzu Edition License to a Supervisor Cluster

If you are using a Supervisor Cluster in evaluation mode, you must assign the cluster a Tanzu edition license before the 60 day evaluation period expires.

Check out [Licensing for vSphere with Tanzu](#) for information about how the Tanzu license works.

Procedure

- 1 In the vSphere Client, navigate to the Supervisor Cluster.
- 2 Select **Configure** and under **Licensing** select **Supervisor Cluster**.
- 3 Select **Assign License**.
- 4 In the **Assign License** dialog, click **New License**.
- 5 Enter a valid license key and click **OK**.

Replace the VIP Certificate to Securely Connect to the Supervisor Cluster API Endpoint

As a vSphere administrator, you can replace the certificate for the virtual IP address (VIP) to securely connect to the Supervisor Cluster API endpoint with a certificate signed by a CA that your hosts already trust. The certificate authenticates the Kubernetes control plane to DevOps engineers, both during login and subsequent interactions with the Supervisor Cluster.

Prerequisites

Verify that you have access to a CA that can sign CSRs. For DevOps engineers, the CA must be installed on their system as a trusted root.

Note In vSphere 7.x, x509 certificate chains are not supported with the L4 load balancer endpoint used for the Kubernetes API server. Certificate chains that involve intermediate CAs (for example, if you bring your own CA chain) might experience certificate verification errors.

Procedure

- 1 In the vSphere Client, navigate to the Supervisor Cluster.
- 2 Click **Configure** then under **Namespaces** select **Certificates**.
- 3 In the **Workload platform MTG** pane, select **Actions > Generate CSR**.
- 4 Provide the details for the certificate.

Note If you use an identity provider service, you must also include the entire certificate chain. The chain is not required for standard HTTPS traffic, however.

- 5 Once the CSR is generated, click **Copy**.
- 6 Sign the certificate with a CA.
- 7 From the **Workload platform MTG** pane, select **Actions > Replace Certificate**.
- 8 Upload the signed certificate file and click **Replace Certificate**.
- 9 Validate the certificate on the IP address of the Kubernetes control plane.

For example, you can open the Kubernetes CLI Tools for vSphere download page and confirm that the certificate is replaced successfully by using the browser. On a Linux or Unix system you can also use `echo | openssl s_client -connect https://ip:6443`.

Integrate the Tanzu Kubernetes Grid Service on the Supervisor Cluster with Tanzu Mission Control

You can integrate Tanzu Kubernetes Grid Service that runs on the Supervisor Cluster with Tanzu Mission Control. Doing so allows you to provision and manage Tanzu Kubernetes clusters using Tanzu Mission Control.

For more information on Tanzu Mission Control, see [Managing the Lifecycle of Tanzu Kubernetes Clusters](#). To watch a demonstration, check out the video [Tanzu Mission Control Integrated with Tanzu Kubernetes Grid Service](#).

View the Tanzu Mission Control Namespace on the Supervisor Cluster

vSphere with Tanzu v7.0.1 U1 and later ships with a vSphere Namespace for Tanzu Mission Control. This namespace exists on the Supervisor Cluster where you install the Tanzu Mission Control agent. Once the agent is installed, you can provision and manage Tanzu Kubernetes clusters using the Tanzu Mission Control web interface.

- 1 Using the vSphere Plugin for kubectl, authenticate with the Supervisor Cluster. See [Connect to the Supervisor Cluster as a vCenter Single Sign-On User](#).
- 2 Switch context to the Supervisor Cluster, for example:

```
kubectl config use-context 10.199.95.59
```

- 3 Run the following command to list the namespaces.

```
kubectl get ns
```

- 4 The vSphere Namespace provided for Tanzu Mission Control is identified as `svc-tmc-cXX` (where XX is a number).
- 5 Install the Tanzu Mission Control agent in this namespace. See [Install the Tanzu Mission Control Agent on the Supervisor Cluster](#).

Install the Tanzu Mission Control Agent on the Supervisor Cluster

To integrate the Tanzu Kubernetes Grid Service with Tanzu Mission Control, install the agent on the Supervisor Cluster.

Note The following procedure requires vSphere 7.0 U3 with the Supervisor Cluster version 1.21.0 or later.

- 1 Using the Tanzu Mission Control web interface, register the Supervisor Cluster with Tanzu Mission Control. See [Register a Management Cluster with Tanzu Mission Control](#).
- 2 Using the Tanzu Mission Control web interface, obtain the Registration URL by navigating to **Administration > Management Clusters**.
- 3 Open a firewall port in your vSphere with Tanzu environment for the port required by Tanzu Mission Control (typically 443). See [Outbound Connections Made by the Cluster Agent Extensions](#).
- 4 Log in to your vSphere with Tanzu environment using the vSphere Client.
- 5 Select the vCenter cluster where **Workload Management** is enabled.

- 6 Select the **Configure** tab.
- 7 Select **TKG Service > Tanzu Mission Control**.
- 8 Provide registration URL in the **Registration URL** field.
- 9 Click **Register**.

The screenshot shows the 'compute-cluster' interface with the 'Configure' tab selected. The left sidebar lists various configuration options, with 'Namespaces' and 'TKG Service' expanded. Under 'TKG Service', 'Tanzu Mission Control' is selected. The main panel is titled 'Tanzu Mission Control Registration' and contains the instruction: 'Add a URL token here to automatically connect all of your Tanzu Kubernetes clusters to Tanzu Mission Control.' Below this is a text input field labeled 'Registration URL' with an information icon. The field contains the URL: `https://myorg.tmc.cloud.vmware.com/installer?id=121f2verylongstring23e&source=registration`. At the bottom right of the panel are two buttons: 'REGISTER' and 'CANCEL'.

Uninstall the Tanzu Mission Control Agent

To uninstall the Tanzu Mission Control agent from the Supervisor Cluster, see [Manually Remove the Cluster Agent from a Supervisor Cluster in vSphere with Tanzu](#).

Set the Default CNI for Tanzu Kubernetes Clusters

As a vSphere administrator, you can set the default container network interface (CNI) for Tanzu Kubernetes clusters.

Default CNI

The Tanzu Kubernetes Grid Service supports two CNI options for Tanzu Kubernetes clusters: [Antrea](#) and [Calico](#).

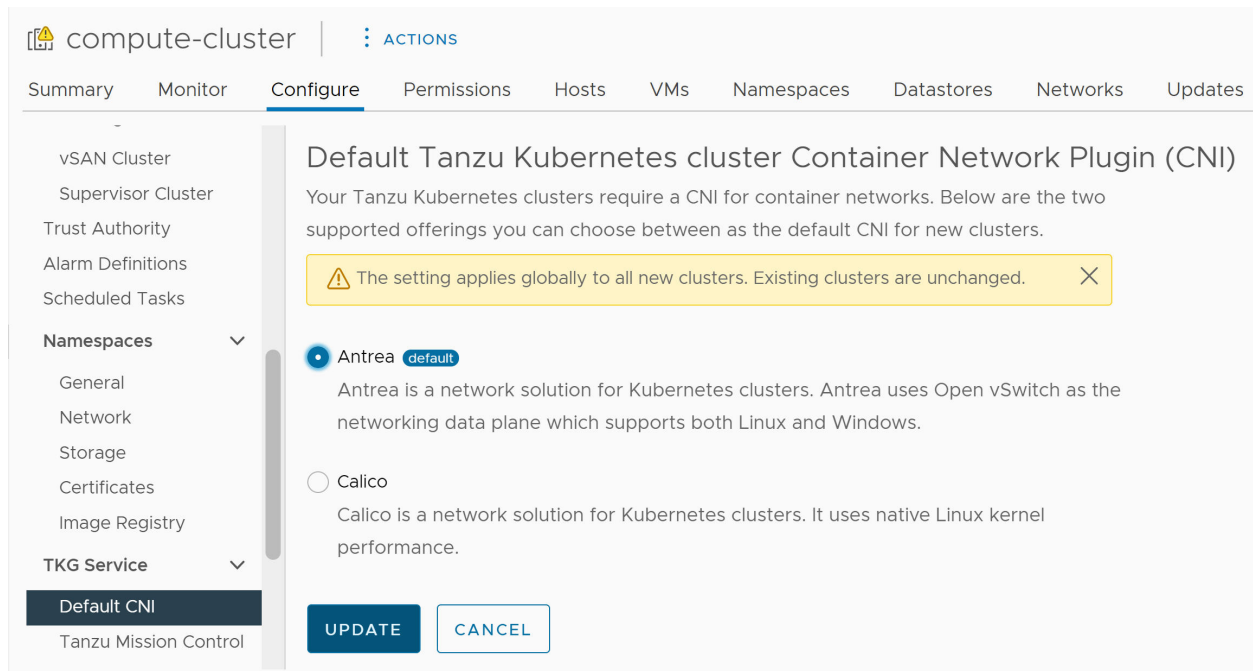
The system-defined default CNI is Antrea. For more information on the default CNI setting, see [Configuration Parameters for the Tanzu Kubernetes Grid Service v1alpha1 API](#).

You can change the default CNI using the vSphere Client. To set the default CNI, complete the following procedure.

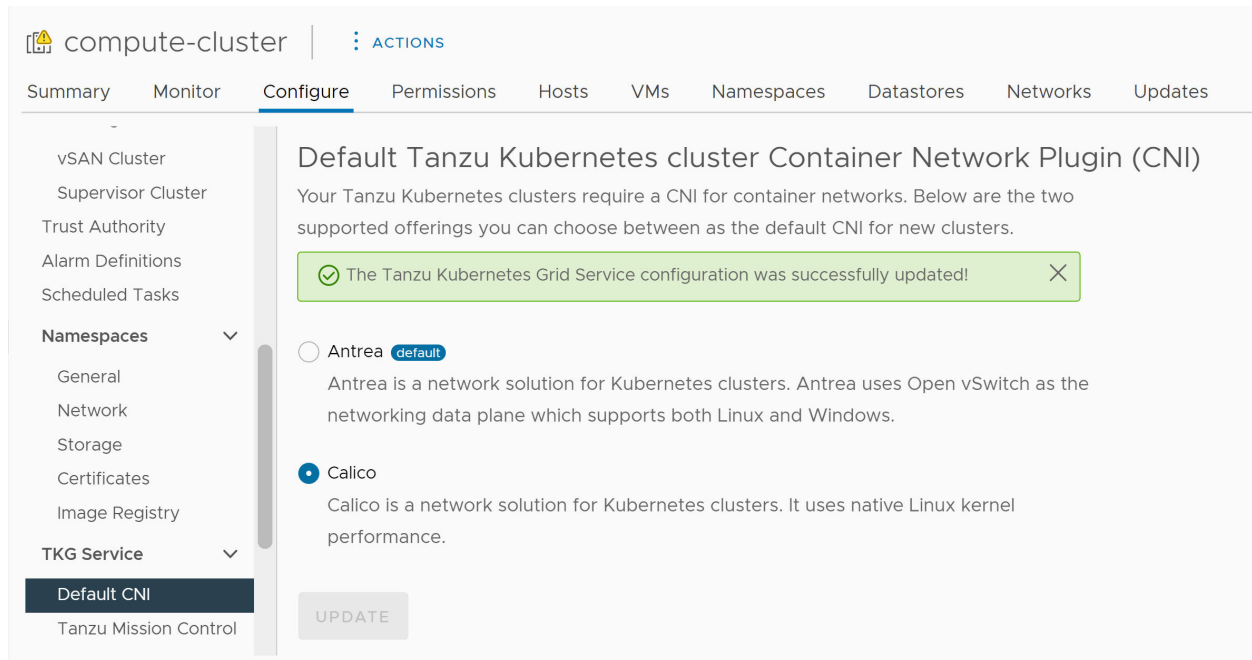
Caution Changing the default CNI is a global operation. The newly set default applies to all new clusters created by the service. Existing clusters are unchanged.

- 1 Log in to your vSphere with Tanzu environment using the vSphere Client.
- 2 Select the vCenter cluster where Workload Management is enabled.
- 3 Select the **Configure** tab.
- 4 Select **TKG Service > Default CNI**.
- 5 Choose the default CNI for new clusters.
- 6 Click **Update**.

The following image shows the default CNI selection.



The following image shows changing the CNI selection from Antrea to Calico.



Add Workload Networks to a Supervisor Cluster Configured with VDS Networking

For a Supervisor Cluster that is configured with the vSphere networking stack, you can provide Layer 2 isolation for your Kubernetes workloads by creating Workload Networks and assigning them to namespaces. Workload Networks provide connectivity to Tanzu Kubernetes clusters in the namespace and are backed by distributed port groups on the switch that is connected to the hosts in the Supervisor Cluster.

For more information on the topologies that you can implement for the Supervisor Cluster, see [Topology for Supervisor Cluster with vSphere Networking and NSX Advanced Load Balancer](#) or [Topologies for Deploying the HAProxy Load Balancer](#).

Note If you have configured the Supervisor Cluster with a DHCP server providing networking settings for Workload Networks, you cannot create new Workload Networks post Supervisor Cluster configuration.

Prerequisites

- Create a distributed port group that will back the Workload Network.
- Verify that the IP range that you will assign to the Workload Network is unique within all Supervisor Clusters available in your environment.

Procedure

- 1 In the vSphere Client, navigate to the Supervisor Cluster.
- 2 Select **Configure**.

3 Under **Supervisor Cluster**, select **Network**.

4 Select **Workload Network** and click **Add**.

Option	Description
Port Group	Select the distributed port group to be associated with this Workload Network. The vSphere Distributed Switch (VDS) that is configured for the Supervisor Cluster networking contains the port groups from which you can select.
Network Name	The network name that identifies the Workload Network when assigned to namespaces. This value is automatically populated from the name of the port group that you select, but you can change it as appropriate.
IP Address Ranges	Enter an IP range for allocating IP addresses of Tanzu Kubernetes cluster nodes. . The IP range must be in the subnet indicated by the subnet mask. Note You must use a unique IP address ranges for each Workload Network. Do not configure the same IP address ranges for multiple networks.
Subnet Mask	Enter the IP address of the subnet mask for the network on the port group.
Gateway	Enter the default gateway for the network on the port group. The gateway must be in the subnet indicated by the subnet mask. Note Do not use the gateway that is assigned to the HAProxy loadbalancer.

5 Click **Add**.

What to do next

Assign the newly-created Workload Network to vSphere Namespaces.

Change the Control Plane Size of a Supervisor Cluster

Checkout how to change the size of the Kubernetes control plane VMs of a Supervisor Cluster in your vSphere with Tanzu environment.

Prerequisites

- Verify that you have the **Modify cluster-wide configuration** privilege on the cluster.

Procedure

- 1 In the vSphere Client, navigate to the Supervisor Cluster.
- 2 Select **Configure** and click **General**.
- 3 Expand **Control Plane Size**, click **Edit** and select new control plane size from the drop-down menu.
- 4 Click **Save**.

You can only scale up the control plane size.

Change the Management Network Settings on a Supervisor Cluster

Learn how to update the DNS and NTP settings on the Supervisor Cluster management network on your vSphere with Tanzu environment.

Prerequisites

- Verify that you have the **Modify cluster-wide configuration** privilege on the cluster.

Procedure

- 1 In the vSphere Client, navigate to the Supervisor Cluster.
- 2 Select **Configure**.
- 3 Under **Supervisor Cluster**, select **Network**.
- 4 Click **Management Network**.
- 5 Edit the DNS and NTP settings.

Option	Description
DNS Server(s)	Enter the addresses of the DNS servers that you use in your environment. If the vCenter Server system is registered with an FQDN, you must enter the IP addresses of the DNS servers that you use with the vSphere environment so that the FQDN is resolvable in the Supervisor Cluster.
DNS Search Domain(s)	Enter domain names that DNS searches inside the Kubernetes control plane nodes, such as <code>corp.local</code> , so that the DNS server can resolve them.
NTP Server(s)	Enter the addresses of the NTP servers that you use in your environment, if any.

Change the Workload Network Settings on a Supervisor Cluster Configured with VDS Networking

Checkout how to change the NTP and DNS server settings for the Workload Networks of a Supervisor Cluster configured with the VDS networking stack. The DNS servers that you configure for Workload Networks are external DNS servers exposed to Kubernetes workloads and they resolve default domain names that are hosted outside of the Supervisor Cluster.

Prerequisites

- Verify that you have the **Modify cluster-wide configuration** privilege on the cluster.

Procedure

- 1 In the vSphere Client, navigate to the Supervisor Cluster.
- 2 Select **Configure**.
- 3 Under **Supervisor Cluster**, select **Network**.

4 Select **Workload Network**.**5** Edit the DNS server settings.

Enter the addresses of DNS Servers that can resolve the domain names of the vSphere management components, such as vCenter Server .

For example, **10.142.7.1**.

When you enter the IP address of the DNS server, a static route is added on each control plane VM. This indicates that the traffic to the DNS servers goes through the workload network.

If the DNS servers that you specify are shared between the management network and workload network, the DNS lookups on the control plane VMs are routed through the Workload Network after initial setup.

6 Edit the NTP settings as needed.

Change Workload Network Settings on a Supervisor Cluster Configured with NSX-T Data Center

Learn how to change the networking settings for DNS server, namespace networks, ingress and egress of a Supervisor Cluster configured for NSX-T Data Center as the networking stack.

Prerequisites

- Verify that you have the **Modify cluster-wide configuration** privilege on the cluster.

Procedure

- 1 In the vSphere Client, navigate to the Supervisor Cluster.
- 2 Select **Configure**.
- 3 Under **Supervisor Cluster**, select **Network**.
- 4 Select **Workload Network**.

5 Change networking settings as needed.

Option	Description
DNS Server(s)	Enter the addresses of DNS Servers that can resolve the domain names of the vSphere management components, such as vCenter Server . For example, 10.142.7.1. When you enter IP address of the DNS server, a static route is added on each control plane VM. This indicates that the traffic to the DNS servers go through the workload network. If the DNS servers that you specify are shared between the management network and workload network, the DNS lookups on the control plane VMs are routed through the workload network after initial setup.
Namespace Network	Enter a CIDR annotation to change the IP range for Kubernetes workloads that are attached to the namespace segments of the Supervisor Cluster. If NAT Mode is not configured, then this IP CIDR range must be a routable IP Address.
Ingress	Enter a CIDR annotation to change the ingress IP range for the Kubernetes services. This range is used for services of type load balancer and ingress. For Tanzu Kubernetes clusters, publishing services through ServiceType loadbalancer will also get the IP addresses from this IP CIDR block. Note You can only add CIDRs to ingress and workload network fields, but you cannot edit or remove existing ones.
Egress	Enter a CIDR annotation for allocating IP addresses for SNAT (Source Network Address Translation) for traffic exiting the Supervisor Cluster to access external services. Only one egress IP address is assigned for each namespace in the Supervisor Cluster. The egress IP is the IP address that the vSphere Pods in the particular namespace use to communicate outside of NSX-T Data Center.

Resolving Errors Health Statuses on Supervisor Cluster During Initial Configuration Or Upgrade

After you initially configure a vSphere cluster as a Supervisor Cluster or you upgrade or edit the settings of an existing Supervisor Cluster, all the settings that you have specified are validated and applied to the cluster until the configuration completes. Health checks are performed on the entered parameters that might detect errors in the configuration resulting in an error health status of the Supervisor Cluster. You must resolve these error health statuses so that the configuration or upgrade of the Supervisor Cluster can resume.

You view the health status of the Supervisor Cluster in the vSphere Client, under **Workload Management > Supervisor Clusters** The status of the cluster configuration is displayed in the **Config Status** column.

Table 5-1. vCenter Server Connection Errors

Error Message	Cause	Solution
Unable to resolve the vCenter Primary Network Identifier <FQDN> with the configured management DNS server(s) on control plane VM <VM name>. Validate that the management DNS servers <server name> can resolve <network name>.	■ At least one management DNS server is reachable. ■ At least one management DNS is statically supplied. ■ The management DNS servers do not have any host name lookup for the vCenter Server PNID. ■ The vCenter Server PNID is a domain name, not a static IP address.	■ Add a host entry for the vCenter Server PNID to the management DNS servers. ■ Verify the configured DNS servers are correct.
Unable to resolve the vCenter Primary Network Identifier <network name> with the DNS server(s) acquired via DHCP on the management network of the control plane VM <VM name>. Validate that the management DNS servers can resolve <network name>.	■ The management DNS servers supplied by the DHCP server (at least one) are reachable. ■ The management DNS servers are statically supplied. ■ The management DNS servers do not have any host name lookup for the vCenter Server PNID. ■ The management DNS servers do not have any host name lookup for the vCenter Server PNID. ■ The vCenter Server PNID is a domain name, not a static IP address.	■ Add a host entry for the vCenter Server PNID to the management DNS servers supplied by the configured DHCP server. ■ Verify the DNS servers supplied by the DHCP server are correct.
Unable to resolve the host <host name> on control plane VM <VM name>, as there are no configured management DNS servers.	■ The vCenter Server PNID is a domain name, not a static IP address. ■ There are no DNS servers configured.	Configure a management DNS server.
Unable to resolve the host <host name> on control plane VM <VM name>. The hostname ends with the '.local' top level domain, which requires 'local' to be included in the management DNS search domains.	The vCenter Server PNID contains .local as a top-level domain (TLD), but the configured search domains do not include local .	Add local to the management DNS search domains.

Table 5-1. vCenter Server Connection Errors (continued)

Error Message	Cause	Solution
Unable to connect to the management DNS servers <server name> from control plane VM <VM name>. The connection was attempted over the workload network.	■ The management DNS servers are unable to be connected to vCenter Server. ■ The provided <code>worker_dns</code> values wholly contain the provided management DNS values. This means that traffic is routed via the workload network, as the Supervisor Cluster must pick one network interface to direct static traffic to these IPs.	■ Check the Workload Network to verify that it can route to the configured management DNS servers. ■ Verify there are no conflicting IP addresses that might trigger alternate routing between the DNS servers and some other servers on the Workload Network. ■ Verify the configured DNS server is, in fact, a DNS server, and is hosting its DNS port on port 53. ■ Verify the workload DNS servers are configured to allow connections from the IPs of the control plane VMs (the Workload Network provided IPs). ■ Verify that there are no typos in the management DNS servers' addresses. ■ Verify search domains don't include an unnecessary '~' that could be resolving the host name incorrectly.
Unable to connect to the management DNS servers <server name> from the control plane VM <VM name>.	Unable to connect to the DNS servers.	■ Check the management network to verify that routes to the management DNS servers exist. ■ Verify there are no conflicting IP addresses that may trigger alternate routing between the DNS servers and other servers. ■ Verify the configured DNS server is, in fact, a DNS server, and is hosting its DNS port on port 53. ■ Verify the management DNS servers are configured to allow connections from the IPs of the control plane VMs. ■ Verify that there are no typos in the management DNS servers' addresses. ■ Verify that search domains do not include an unnecessary '~' that could be resolving the host name incorrectly.

Table 5-1. vCenter Server Connection Errors (continued)

Error Message	Cause	Solution
Unable to connect to <component name> <component address> from control plane VM <vm name>. Error: <i>error message text</i>	■ A generic network failure occurred. ■ Error occurred while connecting to actual connecting to vCenter Server.	■ Validate that the host name or IP address of the configured components, such as vCenter Server, HAProxy, NSX Manager, or NSX Advanced Load Balancer are correct. ■ Validate any external network settings such as conflicting IPs, firewall rules, and others, on the management network.
The control plane VM <VM name> was unable to validate the vCenter <vCenter Server name> certificate. The vCenter server certificate is invalid.	The certificate provided by vCenter Server is in invalid format, and therefore is untrusted.	■ Restart <code>wcp svc</code> to verify that the Trusted Roots bundle in the control plane VMs are up-to-date with the latest vCenter Server root certificates. ■ Verify that the vCenter Server certificate is actually a valid certificate.
The control plane VM <VM name> does not trust the vCenter <vCenter Server name> certificate.	■ The <code>vmca.pem</code> certificate presented by vCenter Server is different from what is configured to the control plane VMs. ■ The trusted root certificates were replaced in the vCenter Server appliance, but <code>wcp svc</code> wasn't restarted.	■ Restart <code>wcp svc</code> to verify that the Trusted Roots bundle in the control plane VMs are up-to-date with the latest vCenter Server certificate roots.

Table 5-2. NSX Manager Connection Errors

The control plane VM <VM name> was unable to validate the NSX Server<NSX server name> certificate. The thumbprint returned by the server <NSX-T address> doesn't match the expected client certificate thumbprint registered in vCenter <vCenter Server name>	The SSL thumbprints registered to the Supervisor Cluster don't match the SHA-1 hash of the certificate presented by the NSX manager.	■ Re-enable trust on the NSX manager between NSX and the vCenter Server instance. ■ Restart <code>wcp svc</code> on vCenter Server.
Unable to connect to <component name> <component address> from control plane VM <vm name>. Error: <i>error message text</i>	A generic network failure occurred.	■ Validate any external network settings, conflicting IPs, firewall rules, and others, on the management network for the NSX manager. ■ Verify the NSX manager IP in the NSX extension is correct. ■ Verify that the NSX manager is running.

Table 5-3. Load Balancer Errors

The control plane VM <i><vm name></i> does not trust the load balancer's (<i><load balancer></i> - <i><load balancer endpoint></i>) certificate.	The certificate the load balancer presents is different from the certifier that is configured to the control plane VMs.	Verify that you have configured the correct Management TLS certificate to the load balancer.
The control plane VM <i><vm name></i> was unable to validate the load balancer's (<i><load balancer></i> - <i><load balancer endpoint></i>) certificate. The certificate is invalid.	The certificate the load balancer presents is in an invalid format, or expired.	Correct the server certificate of the configured load balancer.
The control plane VM <i><vm name></i> was unable to authenticate to the load balancer (<i><load balancer></i> - <i><load balancer endpoint></i>) with the username <i><user name></i> and the supplied password.	The user name or password of the load balancer are incorrect.	Verify the if the user name and password configured to the load balancer are correct.
An HTTP error occurred when attempting to connect to the load balancer (<i><load balancer></i> - <i><load balancer endpoint></i>) from the control plane VM <i><vm name></i> .	The control plane VMs can connect to the load balancer endpoint, but the endpoint does not return a successful (200) http response	Verify that the load balancer is healthy and accepting requests.
Unable to connect to <i><load balancer></i> (<i><load balancer endpoint></i>) from control plane VM <i><vm name></i> . Error: <i><error text></i>	■ A generic network failure occurred. ■ Typically, it means the load balancer is not working, or some firewall blocks the connection.	■ Validate that the load balancer endpoint is accessible ■ Validate no firewalls are blocking the connection to the load balancer.

Configuring HTTP Proxy Settings in vSphere with Tanzu for use with Tanzu Mission Control

Check out how to configure HTTP proxy settings for Supervisor Clusters and Tanzu Kubernetes Grid Service clusters. Learn what is the workflow for configuring HTTP proxy for Supervisor Clusters and Tanzu Kubernetes Grid Service clusters when you register them with Tanzu Mission Control. You use an HTTP proxy for image pulling and container traffic for on-premises Supervisor Clusters that you register as management clusters in Tanzu Mission Control.

Note You cannot use an HTTP Proxy to pull images for use with vSphere Pods or Supervisor Services. You can only use and HTTP proxy to register on-premises Supervisor Clusters as management clusters in Tanzu Mission Control.

Workflow for Configuring HTTP Proxy Settings on Supervisor Clusters and Tanzu Kubernetes Clusters to Use with Tanzu Mission Control

To configure an HTTP proxy on Supervisor Clusters that you want to register as management clusters with Tanzu Mission Control, follow the steps:

- 1 In vSphere, configure HTTP proxy on Supervisor Clusters by either inheriting the HTTP proxy settings from vCenter Server, or configuring proxy settings on individual Supervisor Clusters through the [Namespace Management Clusters APIs](#) or DCLI command line.
- 2 In Tanzu Mission Control, create proxy configuration object by using the proxy settings you configured to the Supervisor Clusters in vSphere with Tanzu. See [Create a Proxy Configuration Object for a Tanzu Kubernetes Grid Service Cluster Running in vSphere with Tanzu](#).
- 3 In Tanzu Mission Control, use this proxy configuration object when you register the Supervisor Clusters as a Management Cluster. See [Register a Management Cluster with Tanzu Mission Control](#) and [Complete the Registration of a Supervisor Cluster in vSphere with Tanzu](#).

To configure an HTTP proxy to Tanzu Kubernetes Grid Service clusters that you provision or add as workload clusters in Tanzu Mission Control:

- 1 Create a proxy configuration object with the proxy settings that you want to use with Tanzu Kubernetes clusters. See [Create a Proxy Configuration Object for a Tanzu Kubernetes Grid Service Cluster Running in vSphere with Tanzu](#).
- 2 Use that proxy configuration object when you provision or add Tanzu Kubernetes clusters as workload clusters. See [Provision a Cluster in vSphere with Tanzu](#) and [Add a Workload Cluster into Tanzu Mission Control Management](#)

Configuring HTTP Proxy to Tanzu Kubernetes Grid Service Clusters in vSphere with Tanzu

Use one of the following methods to configure a proxy to your Tanzu Kubernetes clusters in vSphere with Tanzu:

- Configure proxy settings to individual Tanzu Kubernetes Grid Service clusters. See [Configuration Parameters for Provisioning Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha2 API](#). For an example configuration YAML, see [Example YAML for Provisioning a Custom Tanzu Kubernetes Cluster Using the Tanzu Kubernetes Grid Service v1alpha2 API](#).
- Create a global proxy configuration that will be applied to all Tanzu Kubernetes clusters. See [Configuration Parameters for the Tanzu Kubernetes Grid Service v1alpha2 API](#).

Note If you use Tanzu Mission Control to manage your Tanzu Kubernetes clusters, you do not have to configure proxy settings through the cluster YAML file in vSphere with Tanzu. You can configure proxy settings when you add the Tanzu Kubernetes Grid Service clusters as workload clusters to Tanzu Mission Control.

Configuring Proxy Settings on Newly-Created vSphere 7.0 Update 3 and Later Supervisor Clusters

For newly-created Supervisor Clusters on a vSphere 7.0 Update 3 and later environment, HTTP proxy settings are inherited from vCenter Server. No matter if you create the Supervisor Clusters before or after you configure HTTP proxy settings on vCenter Server, the settings are inherited by the clusters.

See [Configure the DNS, IP Address, and Proxy Settings](#) to learn how to configure the HTTP proxy settings on vCenter Server.

You can also override the inherited HTTP proxy configuration on individual Supervisor Clusters through the Cluster Management API or DCLI.

Since inheriting the vCenter Server proxy settings is the default configuration for newly-created vSphere 7.0.3 Supervisor Clusters, you can also use the Cluster Management API or DCLI to not inherit any HTTP proxy settings in case the Supervisor Clusters don't require a proxy, but vCenter Server still does.

Configuring Proxy Settings on Supervisor Clusters Upgraded to vSphere 7.0 Update 3 and Later

If you have upgraded your Supervisor Clusters to vSphere 7.0 Update 3 and later, the HTTP proxy settings of vCenter Server are not automatically inherited. In that case, you configure proxy settings Supervisor Clusters by using the `vcenter/namespace-management/clusters` API or DCLI command line.

Using the Cluster Management API to Configure HTTP Proxy on Supervisor Clusters

You configure the Supervisor Cluster proxy settings through the `vcenter/namespace-management/clusters` API. The API provides three options for proxy configuration on the Supervisor Cluster:

API Setting	Newly-Created vSphere 7.0.3 and Later Supervisor Clusters	Supervisor Clusters Upgraded to vSphere 7.0.3 and Later
VC_INHERITED	This is the default setting for new Supervisor Clusters and you don't have to use the API to configure the Supervisor Clusters proxy settings. You can just configure proxy settings on vCenter Server through its management interface.	Use this setting to push the HTTP proxy configuration to Supervisor Clusters upgraded to vSphere 7.0.3 and later.
CLUSTER_CONFIGURED	Use this setting to override the HTTP proxy configuration inherited from vCenter Server in one of the following cases: ■ A Supervisor Cluster resides on a different subnet than vCenter Server and a different proxy server is required. ■ The proxy server uses custom CA bundles.	Use this setting to configure HTTP proxy to individual Supervisor Clusters upgraded to vSphere 7.0.3 and later in one of the following cases: ■ You cannot use the vCenter Server proxy because the Supervisor Cluster resides on a different subnet than vCenter Server and a different proxy server is required. ■ The proxy server uses custom CA bundles.
NONE	Use this setting when the Supervisor Cluster has direct connectivity to the internet while vCenter Server requires a proxy. The NONE settings prevents the proxy settings of vCenter Server to be inherited by Supervisor Clusters.	

To set an HTTP proxy to a Supervisor Cluster or modify the existing settings, use the following commands in an SSH session with vCenter Server :

```
vc_address=<IP address>
cluster_id=domain-c<number>
session_id=$(curl -ksX POST --user '<SSO user name>:<password>' https://$vc_address/api/session | xargs -t)
curl -k -X PATCH -H "vmware-api-session-id: $session_id" -H "Content-Type: application/json" -d '{ "cluster_proxy_config": { "proxy_settings_source": "CLUSTER_CONFIGURED", "http_proxy_config": "<proxy_url>" } }' https://$vc_address/api/vcenter/namespace-management/clusters/$cluster_id
```

You only need to pass the `domain_c<number>` from the full cluster ID which. For example, take `domain-c50` from the following cluster ID: `ClusterComputeResource:domain-c50:5bbb510f-759f-4e43-96bd-97fd703b4edb`.

When using the `VC_INHERITED` or `NONE` settings, omit `"http_proxy_config:<proxy_url>"` from the command.

To use a custom CA bundle, add a `"tlsRootCaBundle": "<TLS_certificate>"` to the command by providing the TLS CA certificate in plain text.

For HTTPS proxy settings use the following command:

```
curl -k -X PATCH -H "vmware-api-session-id: $session_id"
-H "Content-Type: application/json" -d '{ "cluster_proxy_config":
{ "proxy_settings_source": "CLUSTER_CONFIGURED", "https_proxy_config": "<proxy_url>" } }'
https://$vc_address/api/vcenter/namespace-management/clusters/$cluster_id
```

Using DCLI to Configure HTTP Proxy Settings on Supervisor Clusters

You can use the following DCLI command to configure HTTP proxy settings to Supervisor Clusters by using the `CLUSTER_CONFIGURED` setting.

```
<dcli> namespacemanagement clusters update --cluster domain-c57 --cluster-proxy-config-http-
proxy-config <proxy URL> --cluster-proxy-config-https-proxy-config <proxy URL> --cluster-
proxy-config-proxy-settings-source CLUSTER_CONFIGURED
```

Streaming Logs of the Supervisor Cluster Control Plane to a Remote rsyslog

Checkout how to configure the streaming of logs from the Supervisor Cluster control plane VMs to a remote rsyslog receiver so that to avoid losing valuable logging data.

Logs generated by the components in the Supervisor Cluster control plane VMs are stored locally in the file systems of the VMs. When a large amount of logs is accumulated, the logs are rotated at high rate, which leads to losing valuable messages that might help with identifying the root cause of different issues. vCenter Server and the Supervisor Cluster control plane VMs support streaming their local logs to a remote rsyslog receiver. This feature helps capture logs for the following services and components:

- On vCenter Server: Workload Control Plane service, ESX Agent Manager service, Certificate Authority service, and all other services running in vCenter Server.
- Supervisor Cluster control plane components and Supervisor Cluster embedded services, such as the VM service, and Tanzu Kubernetes Grid Service.

You can configure the vCenter Server appliance to collect and stream local log data to a remote rsyslog receiver. Once this configuration is applied to vCenter Server, the rsyslog sender running inside vCenter Server starts sending logs generated by services inside that vCenter Server system.

Supervisor Cluster uses the same mechanism as vCenter Server to offload local logs to reduce configuration management overhead. The Workload Control Plane service monitors the vCenter Server rsyslog configuration by polling logs periodically. If the Workload Control Plane service detects that the remote vCenter Server rsyslog configuration is not empty, the service propagates this configuration to each control plane VM in all Supervisor Clusters. This can generate a very large amount of rsyslog message traffic that can overwhelm the remote rsyslog receiver. Therefore, the receiver machine must have sufficient storage capacity to sustain large amounts of rsyslog messages.

Removing the rsyslog configuration from vCenter Server stops rsyslog messages from vCenter Server. The Workload Control Plane service detects the change and propagates it to each control plane VM in every Supervisor Cluster, eventually stopping the control plane VM streams too.

Configuration Steps

Take the following steps to configure rsyslog streaming for Supervisor Cluster control plane VMs:

- 1 Configure an rsyslog receiver by provisioning a machine that:
 - Runs the rsyslog service in receiver mode. See the [Receiving massive amounts of messages with high performance](#) example from the rsyslog documentation.
 - Has sufficient storage space to accommodate large amounts of log data.
 - Has network connectivity to receive data from vCenter Server and the Supervisor Cluster control plane VMs.
- 2 Log in to the vCenter Server appliance management interface at `https://<vcenter server address>:5480` as an root.
- 3 Configure vCenter Server to stream to rsyslog receiver through the vCenter Server appliance management interface. See [Forward vCenter Server Log Files to Remote Syslog Server](#).

It might takes a few minutes for the rsyslog configuration of vCenter Server to be applied to the Supervisor Cluster control plane VMs. The Workload Control Plane service on the vCenter Server appliance polls the appliance configuration every 5 minutes and propagates it to all available Supervisor Clusters. The amount of time needed for the propagation to complete depends on the number of Supervisor Clusters in your environment. In case some of the control plane VMs on the Supervisor Clusters are unhealthy or performing some other operation, the Workload Control Plane service retries applying the rsyslog configuration until it succeeds.

Inspecting Logs of the Control Plane VM Components

The rsyslog of the Supervisor Cluster control plane VMs embeds tags in the log messages that indicate the source component of these log messages.

Log tags	Description
<code>vns-control-plane-pods <pod_name>/<instance_number>.log</code>	Logs originating from Kubernetes pods in control plane VMs. For example: <code>vns-control-plane-pods etcd/0.log</code> or <code>vns-control-plane-pods nsx-ncp/573.log</code>
<code>vns-control-plane-imc</code>	Initial configuration logs from control plane VMs.
<code>vns-control-plane-bootstrap</code>	Bootstrap logs from control plane deployment of Kubernetes nodes.
<code>vns-control-plane-upgrade-logs</code>	Logs from control plane node patches and minor version upgrades.

Log tags	Description
vns-control-plane-svchost-logs	Control plane VM system level service host or agent logs.
vns-control-plane-update-controller	Control plane desired state synchronizer and realizer log.
vns-control-plane-compact-etcd-logs	Logs for keeping control plane etcd service storage compaction.

Registering Tanzu Mission Control Self-Managed with Supervisor Cluster

To register Tanzu Mission Control Self-Managed with Supervisor Cluster, create and apply a custom resource definition for the TMC Agent.

About Tanzu Mission Control Self-Managed

For more information about Tanzu Mission Control Self-Managed, including how to install and configure, refer to the documentation [Installing and Running VMware Tanzu Mission Control Self-Managed](#).

Register Tanzu Mission Control Self-Managed with Supervisor Cluster

To integrate Tanzu Mission Control Self-Managed with Supervisor Cluster, create a custom resource definition with that references the TMC Agent. Supervisor Cluster includes a Kubernetes namespace for TMC where the agent is installed.

Complete the following procedure.

- 1 Install Tanzu Mission Control Self-Managed as described in the documentation. See [Installing and Running VMware Tanzu Mission Control Self-Managed](#).
- 2 Using a web browser, access the Tanzu Mission Control Self-Managed local deployment.
- 3 Export the root CA certificate for the Tanzu Mission Control Self-Managed installation.
 - If you are using a well-known CA, click the lock icon to the left of the address bar in the browser and view the certificate. If you are using a private CA, click the not secure button and view the certificate.
 - On the certificate dialog pop-up, select the `Details` tab and then the `Export` button to download a copy of the CA cert.
 - Open the ca cert file with the text editor of your choice to access the CA cert content
- 4 Using the vSphere Plugin for kubectl, authenticate with Supervisor Cluster.

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME
```

- 5 Run the following command to list the available Kubectl contexts.

```
kubectl config get-contexts
```

- 6 Switch context to the target vSphere Namespace where the TKG cluster running Tanzu Mission Control Self-Managed is provisioned.

```
kubectl config use-context VSPHERE-NAMESPACE
```

- 7 Run the following command to list the Kubernetes namespaces.

```
kubectl get ns
```

- 8 The Kubernetes namespace on Supervisor Cluster for TMC is named `svc-tmc-cXXXX` (where XXXX is a number). For example, `svc-tmc-c1208`. Verify that this Kubernetes namespace exists and is active.
- 9 Use a text editor to create the custom resource definition named `agentconfig.yaml`. This file includes the TMC namespace, the hostname of your TMC Self-Managed deployment, and the CA cert contents.
 - Enter the name of the Kubernetes Namespace for TMC in the `namespace` field.
 - Enter the CA certificates in the `caCerts` fields.
 - Enter the TMC host name in the `allowedHostNames` field.

```
apiVersion: "installers.tmc.cloud.vmware.com/v1alpha1"
kind: "AgentConfig"
metadata:
  name: "tmc-agent-config"
  namespace: "<namespace>"
spec:
  caCerts: |-
    -----BEGIN CERTIFICATE-----
    Certificate1
    -----END CERTIFICATE-----
    -----BEGIN CERTIFICATE-----
    Certificate2
    -----END CERTIFICATE-----
  allowedHostNames:
    - "google.com"
```

- 10 Apply the `AgentConfig` yaml file that you created.

```
kubectl apply -f agentconfig.yaml
```

- 11 Complete the registration process using the web interface for your Tanzu Mission Control Self-Managed installation. Refer to the Tanzu Mission Control [documentation](#) for instructions.

Creating and Managing Content Libraries in vSphere with Tanzu

6

vSphere with Tanzu objects, such as stand-alone VMs and Tanzu Kubernetes clusters, use content libraries as centralized repositories for templates, images, distributions, and other files related to their deployment.

Read the following topics next:

- [Creating and Managing Content Libraries for Tanzu Kubernetes releases](#)
- [Creating and Managing Content Libraries for Stand-Alone VMs in vSphere with Tanzu](#)

Creating and Managing Content Libraries for Tanzu Kubernetes releases

VMware Tanzu distributes Kubernetes software versions as Tanzu Kubernetes releases. To consume these releases, you configure a vSphere Content Library and synchronize the available releases. You can do so using a subscription-based model, or on-demand. If you want to provision Tanzu Kubernetes in an internet restricted environment, you can create a local library and manually import the releases.

About Tanzu Kubernetes release Distributions

A Tanzu Kubernetes release provides the Kubernetes software distribution signed and supported by VMware for use with Tanzu Kubernetes clusters. You obtain and manage Tanzu Kubernetes releases using a vSphere Content Library.

Each Tanzu Kubernetes release is distributed as an OVA template. The Tanzu Kubernetes Grid Service uses the OVA template to construct the virtual machine nodes for Tanzu Kubernetes clusters.

For a list of Tanzu Kubernetes releases and compatibility with the Supervisor Cluster, refer to the [Tanzu Kubernetes releases Release Notes](#).

A Tanzu Kubernetes release is supported on Photon OS and Ubuntu. The disk size of the virtual machine built from the OVA template is fixed. You specify the CPU and RAM resources when you provision a Tanzu Kubernetes cluster. See [Virtual Machine Classes for Tanzu Kubernetes Clusters](#).

The Tanzu Kubernetes Grid Service pulls the Tanzu Kubernetes release OVA templates from a [vSphere Content Library](#). You can use a subscribed content library for automating the process, or a local content library for internet-restricted environments. See [Create, Secure, and Synchronize a Subscribed Content Library for Tanzu Kubernetes releases](#) and [Create, Secure, and Synchronize a Local Content Library for Tanzu Kubernetes releases](#).

The size of the content library can grow over time as new Tanzu Kubernetes releases are distributed. If the underlying storage runs out of space, you can migrate to a new content library. See [Migrate Tanzu Kubernetes Clusters to a New Content Library](#).

After you have created the content library and synchronized it, configure each vSphere Namespace where you plan to provision Tanzu Kubernetes clusters. This includes associating the content library and virtual machine classes with the vSphere Namespace. At this point you can log in to the vSphere Namespace and verify that the Tanzu Kubernetes releases are available. See [Configure a vSphere Namespace for Tanzu Kubernetes releases](#).

Create, Secure, and Synchronize a Subscribed Content Library for Tanzu Kubernetes releases

To store Tanzu Kubernetes release for use with Tanzu Kubernetes clusters, create a subscribed content library on the vCenter Server where vSphere with Tanzu is enabled.

A subscribed content library originates from a published content library. After the subscription is created, the system synchronizes it with the published library. You can choose the synchronization mode: immediate or on demand. For more information, see [Managing a Subscribed Library](#).

Prerequisites

Review [About Tanzu Kubernetes release Distributions](#).

The following vSphere privileges are required to create a content library:

- **Content library.Create local library** or **Content library.Create subscribed library** on the vCenter Server instance where you want to create the library.
- **Datastore.Allocate space** on the destination datastore.

Procedure

- 1 Log in to the vCenter Server using the vSphere Client.
- 2 Select **Menu > Content Libraries**.
- 3 Click **Create**.

The **New Content Library** wizard opens.

- 4 Specify the **Name and location** of the content library and click **Next** when you are done.

Field	Description
Name	Enter a descriptive name, such as TanzuKubernetesRelease-subscriber .
Notes	Include a description, such as On-demand subscription library for Tanzu Kubernetes releases
vCenter Server	Select the vCenter Server instance where vSphere with Tanzu is enabled.

- 5 Configure the content library subscription at the **Configure content library** page and click **Next** when you are done.

- a Select the **Subscribed content library** option.

Note To use a Local Content Library, see [Create, Secure, and Synchronize a Local Content Library for Tanzu Kubernetes releases](#).

- b Enter the **Subscription URL** address of the publisher:

<https://wp-content.vmware.com/v2/latest/lib.json>

- c For the **Download content** option, select one of the following:

Option	Description
Immediately	The subscription process synchronizes both the library metadata and images. If items are deleted from the published library, their contents remain in the subscribed library storage, and you have to manually delete them.
When needed	The subscription process synchronizes only the library metadata. The Tanzu Kubernetes Grid Service downloads the images when published. When you no longer need the item, you can delete the item contents to free storage space. To save storage this option is recommended.

- 6 When prompted, accept the SSL certificate thumbprint.

The SSL certificate thumbprint is stored on your system until you delete the subscribed content library from the inventory.

- 7 Configure the OVF security policy at the **Apply security policy** page and click **Next** when you are done.

- a Select **Apply Security Policy**

- b Select **OVF default policy**

When you select this option, the system verifies the OVF signing certificate during the synchronization process. An OVF template that does not pass certificate validation is marked with the **Verification Failed** tag. The the template metadata is kept, but the OVF files cannot be synchronized.

Note Currently the **OVF default policy** is the only supported security policy.

- 8 At the **Add storage** page, select a datastore as a storage location for the content library contents and click **Next**.
- 9 On the **Ready to complete** page, review the details and click **Finish**.
- 10 At the **Content Libraries** page, select the new content library you created.
- 11 Confirm or complete the synchronization of the library contents.

Synchronization Option	Description
Immediately	If you chose to download all content immediately, confirm that the library is synchronized. To view the synchronized library contents, select Templates > OVF & OVA Templates.
When needed	If you chose to synchronize the library on demand, you have two options: ■ Use Actions > Synchronize to synchronize the entire library ■ Right-click an item and select Synchronize to synchronize only it. To view the synchronized library contents, select Templates > OVF & OVA Templates.

- 12 If you chose the **When needed** option, download the OVF templates you want to use.
 If you chose the **When needed** option, you see that the image files are not stored locally, only the metadata is stored. To download the template files, select the item, right-click and select **Synchronize item**.
- 13 To update the subscribed content library settings, choose **Actions > Edit Settings**.

Setting	Value
Subscription URL	https://wp-content.vmware.com/v2/latest/lib.json
Authentication	Not enabled
Library content	Download when needed
Security policy	OVF default policy

Edit Settings | tkgs-tkr



Automatic synchronization	☒ Enable automatic synchronization with the external content library
Subscription URL	https://wp-content.vmware.com/v2/latest/lib.json
Authentication	☐ Enable user authentication for access to this content library
Library content	☐ Download all library content immediately ☒ Download library content only when needed Save storage space by storing only metadata for the items. To use a content library item, synchronize the item or the whole library.

Applying security policy enforces strict validation while importing. It will result in re-synding of all OVF library items.

Security policy	☒ Apply Security Policy
	OVF default policy

CANCEL

OK

What to do next

Configure each vSphere Namespace where you will provision Tanzu Kubernetes clusters by associating the content library and virtual machine classes with the namespace. See [Configure a vSphere Namespace for Tanzu Kubernetes releases](#).

Create, Secure, and Synchronize a Local Content Library for Tanzu Kubernetes releases

To provision a Tanzu Kubernetes cluster in an internet restricted ("air-gapped") environment, create a local content library and manually import each Tanzu Kubernetes release.

Creating a local content library involves configuring the library, downloading the OVA files, and importing them to the local content library.

Prerequisites

Review [About Tanzu Kubernetes release Distributions](#).

The following privileges are required to create a subscribed content library:

- **Content library.Create local library** or **Content library.Create subscribed library** on the vCenter Server instance where you want to create the library.
- **Datastore.Allocate space** on the destination datastore.

Procedure

- 1 Log in to the vCenter Server using the vSphere Client.

- 2 Click **Menu**.
- 3 Click **Content Library**.
- 4 Click **Create**.

The system displays the **New Content Library** wizard.

- 5 Specify the **Name and location** of the content library and click **Next** when you are done.

Field	Description
Name	Enter a descriptive name, such as TanzuKubernetesRelease-local .
Notes	Include a description, such as Local library for Tanzu Kubernetes releases
vCenter Server	Select the vCenter Server instance where vSphere with Tanzu is enabled.

- 6 At the **Configure content library** page, select the **Local content library** option and click **Next**.

As described below, for local content libraries you manually import the OVF templates you want to use.

Note To use a Subscribed Content Library, see [Create, Secure, and Synchronize a Subscribed Content Library for Tanzu Kubernetes releases](#).

- 7 Configure the OVF security policy at the **Apply security policy** page and click **Next** when you are done.

- a Select **Apply Security Policy**
- b Select **OVF default policy**

When you select this option, the system verifies the OVF signing certificate during the synchronization process. An OVF template that does not pass certificate validation is marked with the **Verification Failed** tag. The the template metadata is kept, but the OVF files cannot be synchronized.

Note Currently the **OVF default policy** is the only supported security policy.

- 8 At the **Add storage** page, select a datastore as a storage location for the content library contents and click **Next**.
- 9 On the **Ready to complete** page, review the details and click **Finish**.
- 10 At the **Content Libraries** page, select the new content library you created.

- 11 Download the OVA files for each Tanzu Kubernetes release you want import to the local content library.

- a Using a browser, navigate to the following URL:

<https://wp-content.vmware.com/v2/latest/>

- b Click the directory for the image you want. Typically this directory is the latest or most recent version of the Kubernetes distribution.

For example:

```
ob-18186591-photon-3-k8s-v1.20.7---vmware.1-tkg.1.7fb9067
```

Note The distribution name is needed to import the files to the local content library, so you might want to copy it to a file or keep the browser open until you complete the procedure.

- c For each of the following files, right-click and select **Save link as**.

- photon-ova-disk1.vmdk
- photon-ova.cert
- photon-ova.mf
- photon-ova.ovf

Index of /26113/v2/latest/ob-18900476-photon-3-k8s-v1.21.6--

Name	Last modified	Size
[DIR] Parent Directory	01-Jan-1970 00:00	-
[FILE] item.json	04-Mar-2022 05:59	1k
[FILE] photon-ova-disk1.vmdk	04-Mar-2022 05:54	-
[FILE] photon-ova.cert	04-Mar-2022 05:54	-
[FILE] photon-ova.mf	04-Mar-2022 05:54	-
[FILE] photon-ova.ovf	04-Mar-2022 05:54	-

- d Verify that each file successfully downloads to your local file system.

Note If the certificate and manifest file are not available in the source directory during the import process, the imported library item will not be usable. This means that for a local content library configured with a security policy, all four required files must be in the local directory from where the ovf and vmdk are imported. In addition to the ovf and vmdk files, you also need to download the cert and manifest files and put all four files in the same source directory.

- 12 Import the OVA files to the local content library.

- a Select **Menu > Content Libraries > .**
 - b From the list of **Content Libraries**, click the link for the name of the local content library you created.
 - c Click **Actions**.
 - d Select **Import Item**.

- e In the **Import Library Item** window, select **Local File**.
- f Click **Upload Files**.
- g Select both files `photon-ova.ovf` and `photon-ova-disk1.vmdk`.

You see the message `2 files ready to import`. Each file is listed with a green check mark beside its name.

- h Change the **Destination Item** name to be the Photon image version plus the Kubernetes version from the directory where you downloaded the files.

For example:

```
photon-3-k8s-v1.20.7---vmware.1-tkg.1.7fb9067
```

- i Click **Import**.

Import Library Item | tkgs-tkr-local

If the certificate or manifest file are not available at source during the import process, the imported library item will not be usable.

Source

Source file

☐ URL

Enter URL

☒ Local file

UPLOAD FILES

REMOVE FILES

Source file details

2 files ready to import

✓ photon-ova.ovf

✓ photon-ova-disk1.vmdk

Destination

Item name

photon-3-k8s-v1.20.7---vmware.1-tkg.1.7fb9067

Notes

Content Library

tkgs-tkr-local

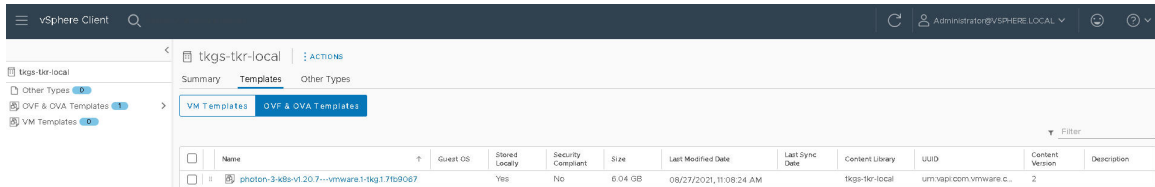
CANCEL

IMPORT

13 Verify that the local content library is populated with the Tanzu Kubernetes release.

- a Reveal the **Recent Tasks** pane at the bottom of the page.
- b Monitor the task **Fetch Content of a Library Item** and verify that it is successfully **Completed**.

- c In the local content library, select **Templates > OVF & OVA Templates**.
- d Verify that the Tanzu Kubernetes release metadata is listed and its content is stored locally.



What to do next

Configure each vSphere Namespace where you will provision Tanzu Kubernetes clusters by associating the content library and virtual machine classes with the namespace. See [Configure a vSphere Namespace for Tanzu Kubernetes releases](#).

Migrate Tanzu Kubernetes Clusters to a New Content Library

If the Subscribed Content Library reaches capacity, you can migrate a Tanzu Kubernetes cluster to use a new library with additional storage capacity.

When the vSphere administrator creates a Subscribed Content Library, the administrator specifies a datastore for storing library contents, in this case OVA files. Over time as more Kubernetes versions are distributed, the Subscribed Content Library will expand in size as OVA files are added for each update. While there is no explicit capacity on the Subscribed Content Library, it is limited by its datastore capacity.

If the Subscribed Content Library reaches capacity, you might see the message `Internal error occurred: get library items failed for`. In this case you can migrate the Tanzu Kubernetes cluster to a new Subscribed Content Library to increase storage capacity. The migration is done by a vSphere administrator using the vSphere Client.

Procedure

- 1 Create a new Subscribed Content Library with sufficient capacity for the target cluster. See [Create, Secure, and Synchronize a Subscribed Content Library for Tanzu Kubernetes releases](#).
- 2 Log in to the vCenter Server using the vSphere Client.
- 3 Select **Menu > Hosts and Clusters**.
- 4 Select the vSphere Cluster object where the Supervisor Cluster containing the Tanzu Kubernetes cluster is provisioned.
- 5 Select the **Configure** tab.
- 6 Select the **Namespaces > General >** option in the navigation panel.
- 7 Click **Edit** beside the **Content Library** section in the main panel.

- 8 Select the new Content Library you created and click **OK**.

This action triggers the update to the cluster configuration.

Note After you modify the Content Library, it might take up to 10 minutes for the Tanzu Kubernetes cluster to pick up the change from the Content Source.

Import the HAProxy OVA to a Local Content Library

If you are using vDS networking, as a convenience you can import the HAProxy OVA file to a Content Library. Importing the HAProxy OVA to a local Content Library can be used for air-gapped deployments.

Prerequisites

Create a local Content Library. See [Create, Secure, and Synchronize a Local Content Library for Tanzu Kubernetes releases](#).

Download the latest version of the VMware HAProxy OVA file from the [VMware-HAProxy site](#).

Procedure

- 1 Log in to the vCenter Server using the vSphere Client.
- 2 Select **Menu > Content Libraries > .**
- 3 From the list of **Content Libraries**, click the link for the name of the Local Content Library you created, such as **HAProxy**.
- 4 Click **Actions**.
- 5 Select **Import Item**.
- 6 In the **Import Library Item** window, select **Local File**.
- 7 Click **Upload Files**.
- 8 Select the file `vmware-haproxy-vX.X.X.ova`.

You see the message `1 file ready to import`. The file is listed with a green check mark beside its name.
- 9 Click **Import**.
- 10 Reveal the **Recent Tasks** pane at the bottom of the page.
- 11 Monitor the task **Fetch Content of a Library Item** and verify that it is successfully **Completed**.

What to do next

Deploy the HAProxy control plane VM. See [Deploy the HAProxy Load Balancer Control Plane VM](#).

Creating and Managing Content Libraries for Stand-Alone VMs in vSphere with Tanzu

A stand-alone VM in the vSphere with Tanzu environment requires an access to VM images, or templates, that contain software configurations, including operating systems, applications, and data. To provide an access to images, configure a VM content library and associate it with the namespace where the VMs are deployed.

What to read next

Procedure

1 [Create a Content Library for Stand-Alone VMs in vSphere with Tanzu](#)

To deploy virtual machines in the vSphere with Tanzu environment, DevOps users must have access to VM templates and images. As a vSphere administrator, create a content library to store and manage VM templates.

2 [Populate a Content Library with VM Images for Stand-Alone VMs in vSphere with Tanzu](#)

As a vSphere administrator, populate a content library with VM templates in OVA or OVF format. Your DevOps engineers can use the templates to provision new stand-alone virtual machines in the vSphere with Tanzu environment.

3 [Associate a VM Content Library with a Namespace in vSphere with Tanzu](#)

As a vSphere administrator, you must give your DevOps users access to a source of VM templates, so that the DevOps engineers can use the templates to provision new stand-alone VMs in the vSphere with Tanzu environment. To give the access, add a content library that contains VM templates to the namespace.

4 [Manage VM Content Libraries on a Namespace in vSphere with Tanzu](#)

As a vSphere administrator, you associate a content library that contains VM templates with a namespace, so that the DevOps engineers can use the templates to provision stand-alone VMs in the vSphere with Tanzu environment. After you associate the library with the namespace, you can remove the library to unpublish it from the Kubernetes namespace. You can also add more libraries.

Create a Content Library for Stand-Alone VMs in vSphere with Tanzu

To deploy virtual machines in the vSphere with Tanzu environment, DevOps users must have access to VM templates and images. As a vSphere administrator, create a content library to store and manage VM templates.

You can create a local content library and populate it with templates and other types of files.

You can also create a subscribed library to use the contents of an already existing published local library.

Starting with vSphere 7.0 Update 3, you can protect the items of a content library by applying an OVF security policy. The OVF security policy enforces strict validation when you deploy or update a content library, import items to a content library, or synchronize templates. To make sure that the templates are signed by a trusted certificate, you can add the OVF signing certificate from a trusted CA to a content library.

For more information about content libraries and VM templates in vSphere, see [Using Content Libraries](#).

Prerequisites

Required privileges:

- **Content library.Create local library** or **Content library.Create subscribed library** on the vCenter Server instance where you want to create the library.
- **Datastore.Allocate space** on the destination datastore.

Procedure

- 1 Navigate to the **VM Service** page.
 - a From the vSphere Client home menu, select **Workload Management**.
 - b Click the **Services** tab and click **Manage** on the **VM Service** pane.
- 2 On the **VM Service** page, click **Content Libraries > Create a content library**.

This action takes you to the content library section in the vSphere Client.

- 3 Click **Create**.

The **New Content Library** wizard opens.

- 4 On the **Name and location** page, enter a name, select a vCenter Server instance for the content library and click **Next**.

Make sure to use an informative name for the content library, so that your DevOps team can easily find and access the library items.

- 5 On the **Configure content library** page, select the type of content library that you want to create and click **Next**.

Option	Description
Local content library	A local content library is accessible only in the vCenter Server instance where you create it by default. - a (Optional) To make the content of the library available to other vCenter Server instances, select Enable publishing. - b (Optional) If you want to require a password for accessing the content library, select Enable authentication and set a password.
Subscribed content library	A subscribed content library originates from a published content library. Use this option to take advantage of existing content libraries. You can synchronize the subscribed library with the published library to see up-to-date content, but you cannot add or remove content from the subscribed library. Only an administrator of the published library can add, modify, and remove contents from the published library. Provide the following information to subscribe to a library: - a In the Subscription URL text box, enter the URL address of the published library. - b If authentication is enabled on the published library, select Enable authentication and enter the publisher password. - c Select a download method for the contents of the subscribed library. ■ If you want to download a local copy of all the items in the published library immediately after subscribing to it, select immediately. ■ If you want to save storage space, select when needed. You download only the metadata for the items in the published library. If you need to use an item, synchronize the item or the entire library to download its content. - d If prompted, accept the SSL certificate thumbprint. The SSL certificate thumbprint is stored on your system until you delete the subscribed content library from the inventory.

- 6 (Optional) On the **Apply security policy** page, select **Apply Security Policy** and select **OVF default policy**.

For the subscribed library, this option appears only if the library supports security policies.

If you select this option, the system performs a strict OVF certificate verification when importing an OVF item to the library from the local host or synchronizing an item. The OVF items that do not pass the certificate validation cannot be imported.

If the item does not pass the validation during synchronization, it is marked with the **Verification Failed** tag. Only the item and metadata will be kept, but not the files in the item.

- 7 On the **Add storage** page, select datastore as a storage location for the content library contents and click **Next**.
- 8 On the **Ready to complete** page, review the details and click **Finish**.

What to do next

After you create the content library, populate the library with VM templates, so that your DevOps engineers can use the templates to provision new virtual machines. See [Populate a Content Library with VM Images for Stand-Alone VMs in vSphere with Tanzu](#).

Populate a Content Library with VM Images for Stand-Alone VMs in vSphere with Tanzu

As a vSphere administrator, populate a content library with VM templates in OVA or OVF format. Your DevOps engineers can use the templates to provision new stand-alone virtual machines in the vSphere with Tanzu environment.

After you create a content library, you can populate it with items in several ways. This topic describes how to add items to a local content library by importing files from your local machine or from a Web server. For other ways to populate the content library, see [Populating Libraries with Content](#).

Prerequisites

- Create a content library for VM provisioning. [Create a Content Library for Stand-Alone VMs in vSphere with Tanzu](#).
- Use only compatible VM images that appear on VMware Cloud Marketplace as OVFs. To find compatible images, search for **VM Service image** on the [VMware Cloud Marketplace](#) web site. See an example of the VM Service image for CentOS at [VM Service Image for CentOS](#).
- If your library is protected by a security policy, make sure that all library items are compliant. If a protected library includes a mix of compliant and non-compliant items, the `kubectl get virtualmachineimages` fails to present VM images to the DevOps engineers.
- Required privilege: **Content library.Add library item** and **Content library.Update files** on the library.

Procedure

- 1 From the vSphere Client home menu, select **Content Libraries**.
- 2 Right-click a local content library and select **Import Item**.

The **Import Library Item** dialog box opens.

- 3 In the **Source** section, select the source of the item.

Option	Description
URL	Enter the path to the Web server where the item is. Note You can import either an <code>.ovf</code> or <code>.ova</code> file. The resulting content library item is of the OVF Template type.
Local File	Click Upload File to navigate to the file that you want to import from your local system. You can use the drop-down menu to filter files in your local system. Note You can import either an <code>.ovf</code> or <code>.ova</code> file. When you import an OVF template, first select the OVF descriptor file (<code>.ovf</code>). Next, you are prompted to select the other files in the OVF template, for example the <code>.vmdk</code> file. The resulting content library item is of the OVF Template type.

vCenter Server reads and validates the manifest and certificate files in the OVF package during importing. A warning is displayed in the **Import Library Item** wizard, if certificate issues exist, for example if vCenter Server detects an expired certificate.

Note vCenter Server does not read signed content, if the OVF package is imported from an `.ovf` file from your local machine.

- 4 In the **Destination** section, enter a name and a description for the item.

- 5 Click **Import**.

Results

The item appears on the **Templates** tab or on the **Other Types** tab.

What to do next

After you create the content library and populate it with VM templates, add the library to the namespace to give your DevOps users access to the content library. See [Associate a VM Content Library with a Namespace in vSphere with Tanzu](#).

Associate a VM Content Library with a Namespace in vSphere with Tanzu

As a vSphere administrator, you must give your DevOps users access to a source of VM templates, so that the DevOps engineers can use the templates to provision new stand-alone VMs in the vSphere with Tanzu environment. To give the access, add a content library that contains VM templates to the namespace.

You can add multiple content libraries to a single namespace. You can add the same content library to different namespaces.

Note This procedure applies only to content libraries for VM Service. Tanzu Kubernetes Grid Service content libraries must be managed from the Tanzu Kubernetes Grid Service pane.

Prerequisites

- Create a content library. See [Create a Content Library for Stand-Alone VMs in vSphere with Tanzu](#).
- Populate the library with VM templates. See [Populate a Content Library with VM Images for Stand-Alone VMs in vSphere with Tanzu](#).
- Required privileges:
 - **Namespaces.Modify cluster-wide configuration**
 - **Namespaces.Modify namespace configuration**

Procedure

- 1 In the vSphere Client, go to the namespace.
 - a From the vSphere Client home menu, select **Workload Management**.
 - b Click the **Namespaces** tab and click the namespace.
- 2 Add a content library.
 - a On the **VM Service** pane, click **Add Content Library**.
 - b Select one or several content libraries and click **OK**.

Results

Contents of the library you added become available in the Kubernetes namespace as VM images and can be used by DevOps to self-service VMs. See [Deploy a Virtual Machine in vSphere with Tanzu](#).

What to do next

After you associate a content library with a namespace, you can add more content libraries or remove the library to unpublish it from the Kubernetes namespace. See [Manage VM Content Libraries on a Namespace in vSphere with Tanzu](#).

Manage VM Content Libraries on a Namespace in vSphere with Tanzu

As a vSphere administrator, you associate a content library that contains VM templates with a namespace, so that the DevOps engineers can use the templates to provision stand-alone VMs in the vSphere with Tanzu environment. After you associate the library with the namespace, you can remove the library to unpublish it from the Kubernetes namespace. You can also add more libraries.

Removing a content library from a namespace does not affect VMs that were previously deployed with the library images.

Note This procedure applies only to content libraries for VM Service. Tanzu Kubernetes Grid Service content libraries must be managed from the Tanzu Kubernetes Grid Service pane.

Prerequisites

- Add a content library to a namespace. [Associate a VM Content Library with a Namespace in vSphere with Tanzu](#).
- Required privileges:
 - **Namespaces.Modify cluster-wide configuration**
 - **Namespaces.Modify namespace configuration**

Procedure

- 1 In the vSphere Client, go to the namespace.
 - a From the vSphere Client home menu, select **Workload Management**.
 - b Click the **Namespaces** tab and click the namespace.
- 2 Add or remove a content library.
 - a On the **VM Service** pane, click **Manage Content Library**.
 - b Perform one of the following operations.

Option	Description
Remove a content library	Deselect the content library and click OK .
Add a content library	Select one or several content libraries and click OK .

What to do next

Contents of the library become available in the Kubernetes namespace as VM images and can be used by DevOps to self-service VMs. See [Deploy a Virtual Machine in vSphere with Tanzu](#).

Configuring and Managing vSphere Namespaces

7

vSphere with Tanzu workloads, including vSphere Pods, VMs, and Tanzu Kubernetes clusters, are deployed to a vSphere Namespace. You define a vSphere Namespace on a Supervisor Cluster and configure it with resource quota and user permissions. Depending on the DevOps needs and workloads they plan to run, you might also assign storage policies, VM classes, and content libraries for fetching the latest Tanzu Kubernetes releases and VM images.

Read the following topics next:

- [Create and Configure a vSphere Namespace](#)
- [Set Resource Limits to a vSphere Namespace](#)
- [Configure Object Limitations on a vSphere Namespace](#)
- [Monitor and Manage Resources in a vSphere Namespace](#)
- [Configure a vSphere Namespace for Tanzu Kubernetes releases](#)
- [Add Security Policies to an NSX Supervisor Cluster Namespace](#)
- [Provision a Self-Service Namespace Template](#)

Create and Configure a vSphere Namespace

As a vSphere administrator, you create a vSphere Namespace on the Supervisor Cluster. You set resources limits to the namespace and permissions so that DevOps engineers can access it. You provide the URL of the Kubernetes control plane to DevOps engineers where they can run Kubernetes workloads on the namespaces for which they have permissions.

Namespaces on Supervisor Clusters configured with the vSphere networking stack and namespaces on clusters configured with NSX have different networking configuration and capabilities.

Namespaces that you create on Supervisor Clusters configured with NSX-T Data Center support the full set of capabilities of the Workload Management platform. They support vSphere Pods, VMs, and Tanzu Kubernetes clusters. The workload networking support for these namespaces is provided by NSX. For more information, see [System Requirements for Setting Up vSphere with Tanzu with NSX](#).

Namespaces that you create on a Supervisor Cluster configured with the vSphere networking stack only support Tanzu Kubernetes clusters and VMs, they do not support vSphere Pods and you cannot use the Harbor Registry with them. The workload networking support for these namespaces is provided by the vSphere Distributed Switch that is connected to the hosts part of the Supervisor Cluster. For more information, see [System Requirements for Setting Up vSphere with Tanzu with vSphere Networking and HAProxy Load Balancer](#).

You can also set resources limits to the namespace, assign permissions, and provision or activate the namespace service on a cluster as a template. As a result, DevOps engineers can create a Supervisor Namespace in a self-service manner and deploy workloads within it. For more information, see [Provision a Self-Service Namespace Template](#).

If you use NSX for your Supervisor Clusters, you have the option to override the networking settings on vSphere Namespace level. Have in mind the following considerations if you select that option:

Table 7-1. vSphere Namespace Network Planning Considerations

Consideration	Description
NSX Installation	To override Supervisor Cluster network settings for a particular vSphere Namespace, the NSX must include an Edge Cluster dedicated for Tier-0 Gateways (routers) and another Edge Cluster dedicated for Tier-1 Gateways. Refer to the NSX installation instructions provided in the guide <i>Installing and Configuring vSphere with Tanzu</i> .
IPAM Required	If you override Supervisor Cluster network settings for a particular vSphere Namespace, the new vSphere Namespace network must specify Ingress, Egress, and Namespace Network subnets that are unique for the Supervisor Cluster and from any other vSphere Namespace network. You will need to manage IP address allocation accordingly.
Supervisor Cluster Routing	The Supervisor Cluster must be able to route directly to the TKG cluster nodes and ingress subnets. When selecting a Tier-0 Gateway for the vSphere Namespace, you have two options for configuring the required routing: ■ Use a Virtual Routing and Forwarding (VRF) Gateway to inherit the configuration from the Supervisor Cluster Tier-0 Gateway ■ Use the Border Gateway Protocol (BGP) to configure routes between the Supervisor Cluster Tier-0 Gateway and the dedicated Tier-0 Gateway Refer to the NSX Tier-0 Gateways documentation for details on these options.

Prerequisites

- Configure a cluster with vSphere with Tanzu.
- Create users or groups for all DevOps engineers who will access the namespace.
- Create storage policies for persistent storage. Storage policies can define different types and classes of storage, for example, gold, silver, and bronze.
- Create VM classes and content libraries for stand-alone VMs.
- Create a content library for Tanzu Kubernetes releases for use with Tanzu Kubernetes clusters. See [Creating and Managing Content Libraries for Tanzu Kubernetes releases](#).

- Required privileges:
 - **Namespaces.Modify cluster-wide configuration**
 - **Namespaces.Modify namespace configuration**

Procedure

- 1 From the vSphere Client home menu, select **Workload Management**.
- 2 Select the **Namespaces** tab.
- 3 Click **Create Namespace**.
- 4 Select the Supervisor Cluster where you want to place the namespace.
- 5 Enter a name for the namespace.
The name must be in a DNS-compliant format.
- 6 From the **Network** drop-down menu, select a Workload Network for the namespace.

Note This step is available only if you create the namespace on a cluster that is configured with the vSphere networking stack.

- 7 If you have configured the NSX networking stack for your cluster, you can select **Override cluster network settings** to override the cluster network settings and configure network settings for the namespace.

Configure the following network settings for the namespace:

Option	Description
NAT Mode	The NAT mode is selected by default. If you deselect this option, all the workloads such as the vSphere Pods, VMs, and Tanzu Kubernetes clusters node IP addresses are directly accessible from outside the tier-0 gateway and you do not have to configure the egress CIDRs. Note Once you enable a namespace mode, you cannot change it.
Tier-0 Gateway	Select the tier-0 gateway to associate with the namespace tier-1 gateway. Selecting a tier-0 gateway overrides the tier-0 gateway you configured while enabling the cluster, so you must configure the CIDR ranges again. Note The Supervisor Cluster must be able to route directly to the TKG cluster nodes and ingress subnets. If you select a VRF gateway that is linked to the tier-0 gateway, the network and subnets are automatically configured. If you have selected the NAT mode, you must configure the subnet, ingress, and egress CIDRs. If you deselect the NAT mode, you must only configure the subnet and ingress CIDRs. Note Once you select a tier-0 gateway, you cannot change it.

Option	Description
Namespace Network CIDR	Enter one or more IP CIDRs to create subnets/segments and assign IP addresses for workloads connected to namespaces. Note Enter the CIDR range if you did not configure it for the cluster. You can configure additional CIDRs after you create the namespace, by editing the namespace network settings.
Namespace Subnet Prefix	Enter the subnet prefix that specifies the size of the subnet reserved for namespaces segments. Default is 28. Note Once you specify the subnet prefix, you cannot change it.
Ingress CIDR	Enter a CIDR annotation that determines the ingress IP range for the virtual IP addresses published by the load balancer service for vSphere Pods or Tanzu Kubernetes clusters. You can configure additional CIDRs after you create the namespace, by editing the namespace network settings.
Egress CIDR	Enter a CIDR annotation that determines the egress IP range for the SNAT IP addresses. You can configure additional CIDRs after you create the namespace, by editing the namespace network settings.
Load balancer Size	Select the size of the load balancer instance on the tier-1 gateway for the namespace.

- 8 Enter a description, and click **Create**.

The namespace is created on the Supervisor Cluster.

- 9 Set permissions so that DevOps engineers can access the namespace.

- a From the **Permissions** pane, select **Add Permissions**.
- b Select an identity source, a user or a group, and a role, and click **OK**.

- 10 Set persistent storage to the namespace.

Storage policies that you assign to the namespace control how persistent volumes and Tanzu Kubernetes cluster nodes are placed within datastores in the vSphere storage environment. The persistent volume claims that correspond to persistent volumes can originate from a vSphere Pod, and VM, or from the Tanzu Kubernetes cluster. For more information, see [Chapter 10 Using Persistent Storage in vSphere with Tanzu](#).

- a From the **Storage** pane, select **Add Storage**.
- b Select a storage policy to control datastore placement of persistent volumes and click **OK**.

After you assign the storage policy, vSphere with Tanzu creates a matching Kubernetes storage class in the vSphere Namespace. If you use VMware Tanzu™ Kubernetes Grid™ Service, the storage class is automatically replicated from the namespace to the Kubernetes cluster. When you assign multiple storage policies to the namespace, a separate storage class is created for each storage policy.

- 11 From the Capacity and Usage pane, select **Edit Limits** and configure resource limitations to the namespace.

Option	Description
CPU	The amount of CPU resources to reserve for the namespace.
Memory	The amount of memory to reserve for the namespace.
Storage	The total amount of storage space to reserve for the namespace.
Storage policies limits	Set the amount of storage dedicated individually to each of the storage policies that you associated with the namespace.

A resource pool for the namespace is created on vCenter Server. The storage limitation determines the overall amount of storage that is available to the namespace whereas storage policies determine the placement of persistent volumes for vSphere Pods on the associated storage classes.

- 12 Set up VM Service for stand-alone VMs.

For information, see [Chapter 12 Deploying and Managing Virtual Machines in vSphere with Tanzu](#).

- 13 Configure the namespace for Tanzu Kubernetes clusters, including the following:

- Associate the Tanzu Kubernetes release content library with the namespace.
- Add the default VM classes to the namespace.

For more information, see [Configure a vSphere Namespace for Tanzu Kubernetes releases](#).

What to do next

Share the Kubernetes Control Plane URL with DevOps engineers as well as the user name they can use to log in to the Supervisor Cluster through the Kubernetes CLI Tools for vSphere. You can grant access to more than one namespace to a DevOps engineer. See [Chapter 9 Connecting to vSphere with Tanzu Clusters](#).

Set Resource Limits to a vSphere Namespace

As a vSphere administrator, you can set resource limits and container defaults on a vSphere Namespace. DevOps engineers can later override the container defaults in pod specifications but without exceeding the total resource limits set to the namespace by the vSphere administrator. Container requests translate to resource reservations in pods..

Prerequisites

- Verify that you have the **Modify namespace configuration** privilege on the Supervisor Cluster.

Procedure

- 1 From the vSphere Client home menu, select **Workload Management**.
- 2 Select a vSphere Namespace, select **Configure**, and click **Resource Limits**.

3 Click **Edit**.

The impact of establishing resource limits for a vSphere Namespace where Tanzu Kubernetes Grid Service clusters are provisioned differs depending on the type of VM class used for cluster nodes. Make sure you are aware of the differences between best effort and guaranteed before setting resource limits. See [Virtual Machine Classes for Tanzu Kubernetes Clusters](#)

Option	Description
CPU	Set a limit to the CPU consumption on the vSphere Namespace.
Memory	Set a limit to the memory consumption on the vSphere Namespace.
Storage	Set a limit on the storage consumption on the vSphere Namespace per storage policy that is used.
Container Defaults	Set the defaults for CPU limits, CPU requests, memory limits, and memory limits for containers in the vSphere Namespace.

Configure Object Limitations on a vSphere Namespace

You can configure limitations for pods running in the vSphere Namespace as well as limitations for various Kubernetes objects. The limitations that you configure for an object depend on the specifics of your applications and the way you want them to consume resources within a vSphere Namespace.

Prerequisites

- Verify that you have the **Modify namespace configuration** privilege on the Supervisor Cluster.

Procedure

- 1 From the vSphere Client home menu, select **Workload Management**.
- 2 Select the vSphere Namespace to which you want to apply object or container restrictions.
- 3 Select **Configure** and select **Object Limits**.
- 4 Click **Edit**.

Option	Description
vSphere Pods	The number of vSphere Pods that can run in the vSphere Namespace.
Deployments	The number of deployments that can run in the vSphere Namespace.
Jobs	The number of jobs that can run in the vSphere Namespace.
Daemon Sets	The number of daemon sets that can run in the vSphere Namespace.
Replica Sets	The number of replica sets in the vSphere Namespace.
Replication Controllers	The number of replication controllers that can run in the vSphere Namespace.
Stateful Sets	The number of StatefulSets that can run in the vSphere Namespace.

Option	Description
Config Maps	The number of ConfigMaps that can run in the vSphere Namespace.
Secrets	The number of secrets that can run in the vSphere Namespace.
Persistent Volume Claims	The persistent volume claims that can exists in the vSphere Namespace.
Services	The services that can exists in the vSphere Namespace.

Monitor and Manage Resources in a vSphere Namespace

You can monitor and manage different aspects of a vSphere Namespace, such as resource consumption for the namespace as well as the number of different Kubernetes objects that exist in a namespace and they states.

Prerequisites

[Create and Configure a vSphere Namespace.](#)

Procedure

- 1 Log in to the vCenter Server using the vSphere Client.
- 2 Navigate to the **Menu > Hosts and Clusters** view.
- 3 Select the vCenter Cluster where you have enabled **Workload Management**.
- 4 Select the **Namespaces** resource pool and expand its contents.

The Supervisor Cluster control plane nodes are located in the Namespaces resource pool. In addition, each vSphere Namespace that is created for this Supervisor Cluster is located in the **Namespaces** resource pool.

- 5 Select the vSphere Namespace object, which is represented as a window icon.

In the **Summary** tab you see the various configuration sections for the vSphere Namespace, including **Status**, **Permissions**, **Storage**, **Capacity and Usage**, and **Tanzu Kubernetes**. From this screen you can manage any of these settings.

The screenshot displays the vSphere Client interface for a vSphere Namespace named 'tkgs-cluster-ns'. The interface is divided into several sections:

- Status:** Shows the namespace is 'Running' and 'Active'. It includes a 'Location' of 'Compute-Cluster' and a 'Link to CLI Tools'.
- Permissions:** Shows 'Can view' and 'Can edit' permissions for the 'Administrator' user.
- Storage:** Shows 'Persistent Volume Claims' and 'Storage Policies' (VWK-Storage-Policy) with a 'No limit' indicator.
- Capacity and Usage:** Shows resource usage for CPU (0 MHz), Memory (594 MB), and Storage (4 GB), all with 'No limit' indicators.
- Tanzu Kubernetes:** Shows '1' Tanzu Kubernetes Cluster, 'Control Plane Nodes' (0 Unhealthy, 3 Healthy), and a 'VIEW ALL' link.

Configure a vSphere Namespace for Tanzu Kubernetes releases

Configure the vSphere Namespace where you plan to provision Tanzu Kubernetes clusters by associating the namespace with the content library for Tanzu Kubernetes releases and with the VM classes you want to use.

Prerequisites

Create a vSphere Namespace. See [Create and Configure a vSphere Namespace](#).

Create a content library for hosting Tanzu Kubernetes releases. See [Create, Secure, and Synchronize a Subscribed Content Library for Tanzu Kubernetes releases](#) or [Create, Secure, and Synchronize a Local Content Library for Tanzu Kubernetes releases](#).

Associate the Content Library with the vSphere Namespace

To associate the content library created for Tanzu Kubernetes releases with a vSphere Namespace, log in to the vCenter Server using the vSphere Client and complete either of the following procedures.

Associate Using the vSphere Inventory Path	Associate Using the Workload Management Path
1 Select Menu > Hosts and Clusters. 2 Select the vSphere Cluster where Workload Management is enabled. 3 Select the Configure tab. 4 Select Namespaces > General. 5 Select Tanzu Kubernetes Grid Service Configuration. 6 Click Edit beside the Content Library label. 7 Select the content library for Tanzu Kubernetes releases. 8 Click OK.	1 Select Menu > Workload Management. 2 Select the Namespaces tab. 3 Select the target vSphere Namespace. 4 Locate the Tanzu Kubernetes Grid Service tile. 5 Click Edit beside the Content Library label. 6 Select the content library for Tanzu Kubernetes releases. 7 Click OK.

Note After you associate the content library with the vSphere Namespace, it can take several minutes for the virtual machine templates to be available for provisioning Tanzu Kubernetes clusters. See [Verify the vSphere Namespace Configuration](#).

Associate the VM Classes with the vSphere Namespace

vSphere with Tanzu provides several default virtual machine classes, and you can create your own. See [Virtual Machine Classes for Tanzu Kubernetes Clusters](#).

To provision Tanzu Kubernetes clusters, you need to associate the virtual machine classes you want to use with each vSphere Namespace where you want to provision Tanzu Kubernetes clusters.

To associate the default VM classes with a vSphere Namespace, log in to the vCenter Server using the vSphere Client and complete the following procedure.

- 1 Select **Menu > Workload Management**.
- 2 Select the **Namespaces** tab.
- 3 Select the target vSphere Namespace where you plan to provision Tanzu Kubernetes clusters.
- 4 Locate the **VM Service** tile.
- 5 Click the **Add VM Class** link.
- 6 Select the VM classes to add.
 - a To add the default VM classes, select the checkbox in the table header on page 1 of the list, navigate to page 2 and select the checkbox in the table header on that page. Verify that all classes are selected.
 - b To create a custom class, click **Create New VM Class**. See [Create a VM Class in vSphere with Tanzu](#).
- 7 Click **OK** to complete the operation.
- 8 Confirm that the classes are added. The **VM Service** tile shows **Manage VM Classes**.

Note The content library referenced in the **VM Service** tile is for use with standalone VMs, not Tanzu Kubernetes releases. See [Creating and Managing Content Libraries for Stand-Alone VMs in vSphere with Tanzu](#).

Verify the vSphere Namespace Configuration

Once you have associated the content library and virtual machine classes with the vSphere Namespace, log in to the Supervisor Cluster and verify that each synchronized Tanzu Kubernetes release is available and each selected VM class is available.

- 1 Install the Kubernetes CLI Tools for vSphere. See [Download and Install the Kubernetes CLI Tools for vSphere](#).
- 2 Log in to the Supervisor Cluster.

```
kubectl vsphere login --server IP-ADDRESS-SUPERVISOR-CLUSTER --vsphere-username VCENTER-SSO-USERNAME
```

- 3 Switch context to the target vSphere Namespace.

```
kubectl config use-context SUPERVISOR-NAMESPACE
```

- 4 List and describe the available Tanzu Kubernetes releases.

```
kubectl get tanzukubernetesreleases
```

```
kubectl describe tanzukubernetesreleases
```

- 5 List the available virtual machine classes.

```
kubectl get virtualmachineclassbindings
```

Once the namespace is configured, you can now provision Tanzu Kubernetes clusters. See [Workflow for Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API](#). If you are using a local content library, you will need to specify an OVA that you uploaded to the library. See [Examples for Provisioning Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API](#).

Add Security Policies to an NSX Supervisor Cluster Namespace

A Supervisor Cluster that uses NSX networking supports network security policies configured through a security policy CRD.

Create a Security Policy

As a DevOps, you can configure the security policy CRD to apply an NSX based security policy to a Supervisor Cluster namespace. The security policy protects traffic for vSphere Pods and VMs. VMs include TKG cluster nodes and other VMs deployed in the Supervisor Cluster.

Prerequisites

Use NSX version 3.2 or later.

Procedure

- 1 Create a security policy CRD.

For the fields to use and CRD examples, see [NSX Operator Security Policy CRD](#) documentation on GitHub.

- 2 Access your namespace in the Kubernetes environment.

See [Get and Use the Supervisor Cluster Context](#).

- 3 Apply the security policy to the namespace.

```
kubectl apply -f policy-name.yaml
```

4 View your security policy.

- a View details for the security policy.

```
kubectl get securitypolicy policy-name
```

- b View a description of your security policy.

```
kubectl describe securitypolicy policy-name
```

Results

You can also use the NSX UI to view details of your policy. For information, see the *VMware NSX Documentation* page.

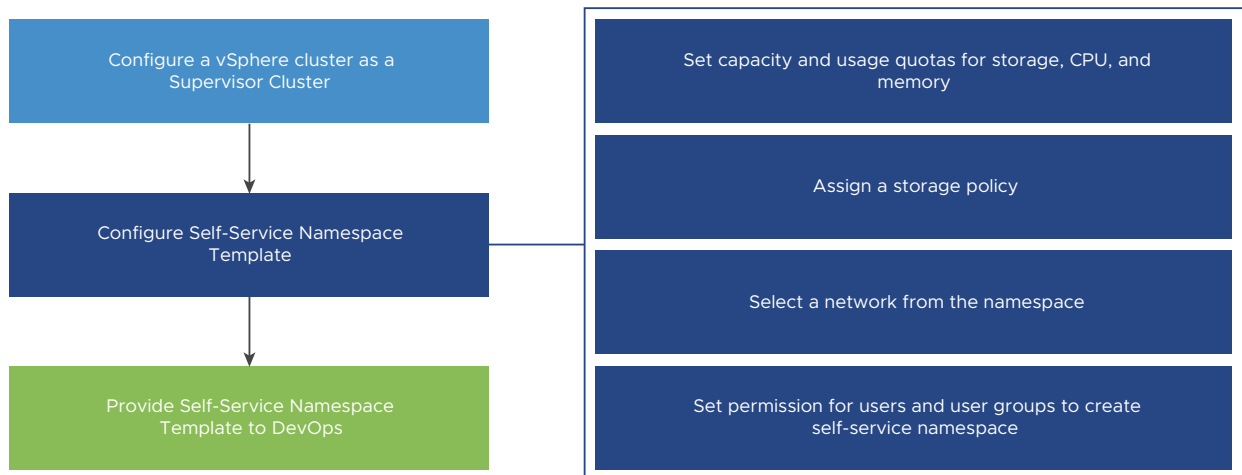
Provision a Self-Service Namespace Template

As a vSphere administrator, you can create a Supervisor Namespace, set CPU, memory, and storage limits to the namespace, assign permissions, and activate the namespace service on a cluster as a template. As a result, DevOps engineers can create a Supervisor Namespace in a self-service manner and deploy workloads within it.

Self-Service Namespace Creation and Configuration Workflow

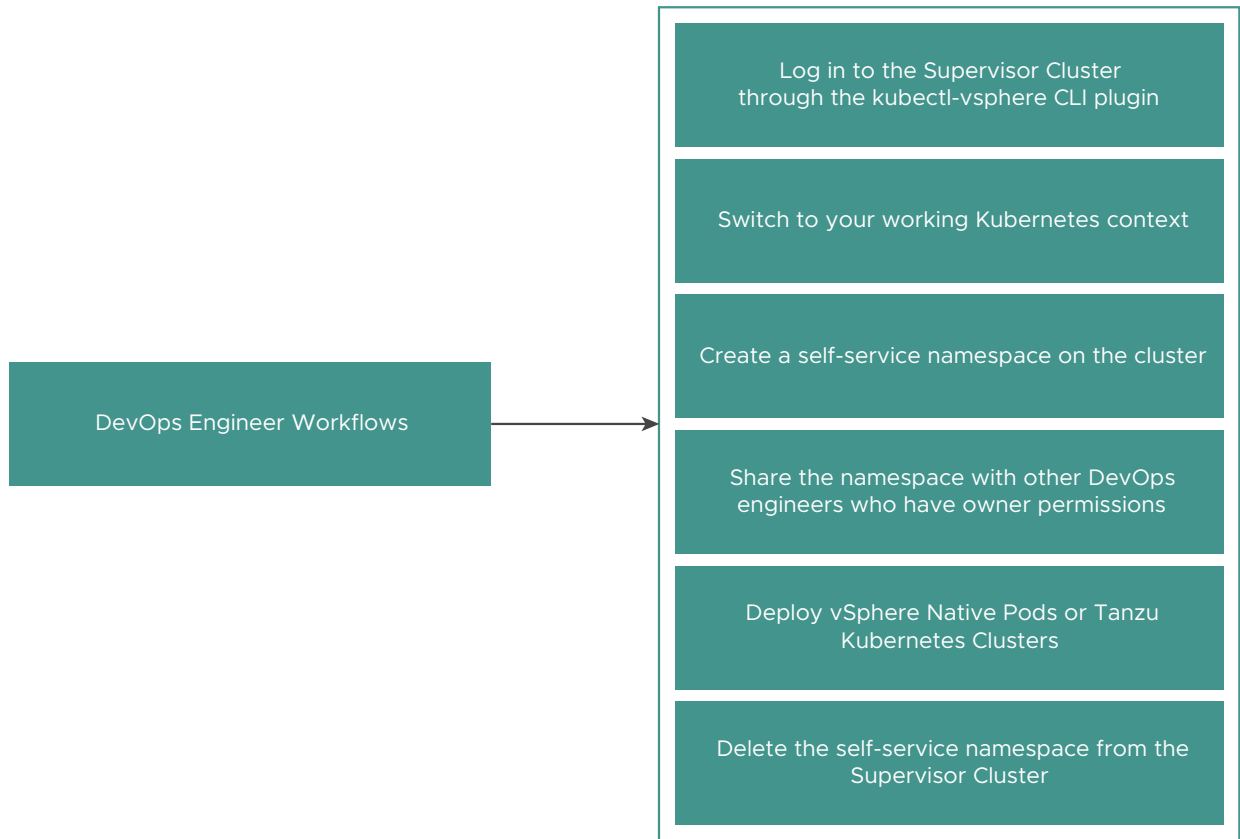
As a vSphere administrator, you can create a Supervisor Namespace, set CPU, memory, and storage limits to the namespace, assign permissions, and provision or activate the namespace service on a cluster as a template.

Figure 7-1. Self-service Namespace Template Provisioning Workflow



As a DevOps engineer, you can create a Supervisor Namespace in a self-service manner and deploy workloads within it. You can share it with other DevOps engineers or delete it when it is no longer required. To share the namespace with other DevOps engineers, contact the vSphere administrator.

Figure 7-2. Self-service Namespace Creation Workflow



Create and Configure a Self-Service Namespace Template

As a vSphere administrator, you can create and configure a Supervisor Namespace as a self-service namespace template. DevOps engineers can then create and delete Supervisor Namespaces using the `kubectl` command line.

Prerequisites

Configure a cluster with vSphere with Tanzu.

Procedure

- 1 In the vSphere Client, select the vSphere cluster where the Supervisor Cluster is enabled.
- 2 In the **Configure** tab, select **Namespaces > General**.
- 3 Select **Namespace Service**.
- 4 Toggle the **Status** switch to enable the feature.

The **Create Namespace Template** page appears.

- 5 In the **Configuration** pane, configure resource limitations to the namespace.

Option	Description
CPU	The amount of CPU resources to reserve for the namespace.
Memory	The amount of memory to reserve for the namespace.
Storage	The total amount of storage space to reserve for the namespace.
Storage Policy	Set the amount of storage dedicated individually to each of the storage policies that you associated with the namespace.
Network	From the Network drop-down menu, select a network for the namespace.

- 6 Click **Next**.
- 7 In the **Permissions** pane, add DevOps engineers and groups to empower them to use the template to create namespaces.
- Select an identity source and a user or a group and click **Next**.
- 8 In the **Review and Confirm** pane, the properties you configure are displayed.
- Review the properties and click **Done**.

Results

A namespace template is configured and is in Active state. As a vSphere administrator, you can edit the template. DevOps engineers can use the template to create namespaces.

Deactivate a Self-Service Namespace

As a vSphere administrator, you can deactivate a self-service namespace on the cluster.

When you deactivate a self-service namespace template, DevOps engineers cannot use the template to create new namespaces on the cluster. They can delete the namespaces that they have already created.

Procedure

- 1 In the vSphere Client, navigate to the Supervisor Cluster.
- 2 In the **Configure** tab, select **General** under **Namespaces**.
- 3 In the **Namespace Self-Service** pane, toggle the **Status** switch to deactivate the template.
- 4 To activate the template again, toggle the **Status** switch.

You can either create another self-service namespace or use the existing one.

Create a Self-Service Namespace

As a DevOps Engineer, you can create a self-service namespace and run workloads within it. Once you create the namespace, you can share it with other DevOps Engineers or delete it when it is no longer required.

Prerequisites

- Verify that a vSphere administrator has created and activated a self-service namespace template on the cluster. See [Create and Configure a Self-Service Namespace Template](#).
- Verify that you are added to the permissions list in the self-service namespace template either individually or as a member of a group.
- Get the IP address of the Supervisor Cluster control plane.

Procedure

- 1 Using the vSphere Plugin for kubectl, authenticate with the Supervisor Cluster. See [Connect to the Supervisor Cluster as a vCenter Single Sign-On User](#).

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME
```

- 2 Switch context to the Supervisor Cluster.

```
kubectl config use-context SUPERVISOR-CLUSTER-IP
```

- 3 Create a self-service namespace on the cluster.

```
kubectl create namespace NAMESPACE NAME
```

For example

```
kubectl create namespace test-ns
```

Note Owner permissions are available to DevOps Engineers after you enable vSphere with Tanzu and upgrade the cluster. If you have only upgraded vCenter Server and not the cluster, the DevOps Engineers will only have edit permissions on the namespaces.

The namespace that you create appears in the cluster. To share the namespace with other DevOps engineers, contact the vSphere administrator.

Create a Self-Service Namespace with Annotations and Labels

DevOps engineers can create self-service namespaces with annotations and labels using the kubectl command line.

DevOps engineers can use a YAML manifest with user-defined annotations and labels.

Procedure**1** Log in to the Supervisor Cluster.

```
kubectl vsphere login --server IP-ADDRESS-SUPERVISOR-CLUSTER --vsphere-username VCENTER-SSO-USERNAME
```

2 Create a namespace YAML manifest file with annotations and labels.

```
kubectl create -f ns-create.yaml
```

For example, create the following `ns-create.yaml` file:

```
apiVersion: v1
kind: Namespace
metadata:
  name: test-ns-yaml
  labels:
    my-label: "my-label-val-yaml"
  annotations:
    my-ann-yaml: "my-ann-val-yaml"
```

3 Apply the YAML manifest.

```
kubectl create -f ns-create.yaml
```

Or

```
kubectl apply -f ns-create.yaml
```

4 Describe the namespace that you created to see the changes.

```
root@localhost [ /tmp ]# kubectl describe ns test-ns-yaml
Name:          test-ns-yaml
Labels:        my-label=my-label-val-yaml
               vSphereClusterID=domain-c50
Annotations:   my-ann-yaml: my-ann-val-yaml
               vmware-system-namespace-owner-count: 1
               vmware-system-resource-pool: resgroup-171
               vmware-system-resource-pool-cpu-limit: 0.4770
               vmware-system-resource-pool-memory-limit: 2000Mi
               vmware-system-self-service-namespace: true
               vmware-system-vm-folder: group-v172
Status:        Active

Resource Quotas
Name:          test-ns-yaml
Resource      Used  Hard
-----
requests.storage 0    5000Mi

Name:          test-ns-yaml-storagequota
Resource      Used  Hard
-----
```

```
namespace-service-storage-profile.storageclass.storage.k8s.io/requests.storage 0
9223372036854775807

No LimitRange resource.
```

Update a Self-Service Namespace Using kubectl annotate and kubectl label

As a DevOps engineer you can update or delete self-service namespace annotations and labels using the `kubectl annotate` and `kubectl label` commands.

Prerequisites

Verify that you have owner permissions on the namespace that you want to update.

Procedure

1 Log in to the Supervisor Cluster.

```
kubectl vsphere login --server IP-ADDRESS-SUPERVISOR-CLUSTER --vsphere-
username VCENTER-SSO-USERNAME
```

2 Describe the namespace that you want to update.

```
root@localhost [ /tmp ]# kubectl describe ns testns
Name:          testns
Labels:        my-label=test-label-2
               vSphereClusterID=domain-c50
Annotations:   my-ann: test-ann-2
               vmware-system-namespace-owner-count: 2
               vmware-system-resource-pool: resgroup-153
               vmware-system-resource-pool-cpu-limit: 0.4770
               vmware-system-resource-pool-memory-limit: 2000Mi
               vmware-system-self-service-namespace: true
               vmware-system-vm-folder: group-v154
Status:        Active

Resource Quotas
Name:          testns
Resource      Used  Hard
-----
requests.storage 0    5000Mi

Name:          testns-storagequota
Resource      Used  Hard
-----
namespace-service-storage-profile.storageclass.storage.k8s.io/requests.storage 0
9223372036854775807
```

3 Update annotations using the `kubectl annotate` command.

For example, `kubectl annotate --overwrite ns testns my-ann="test-ann-3"`

To delete an annotation, run the command `kubectl annotate --overwrite ns testns my-ann-`

4 Update labels using the `kubectl label` command.

For example, `kubectl label --overwrite ns testns my-label="test-label-3"`

To delete a label, run the command `kubectl label --overwrite ns testns my-label-`

5 Describe the namespace to see the updates.

```
root@localhost [ /tmp ]# kubectl describe ns testns
Name:          testns
Labels:        my-label=test-label-3
               vSphereClusterID=domain-c50
Annotations:   my-ann: test-ann-3
               vmware-system-namespace-owner-count: 2
               vmware-system-resource-pool: resgroup-153
               vmware-system-resource-pool-cpu-limit: 0.4770
               vmware-system-resource-pool-memory-limit: 2000Mi
               vmware-system-self-service-namespace: true
               vmware-system-vm-folder: group-v154
Status:        Active

Resource Quotas
Name:          testns
Resource      Used  Hard
-----
requests.storage 0    5000Mi

Name:          testns-
storagequota
Resource      Used  Hard
-----
namespace-service-storage-profile.storage.k8s.io/requests.storage 0
9223372036854775807

No LimitRange resource.
```

Update a Self-Service Namespace Using `kubectl edit`

As a DevOps engineer you can update self-service namespaces using the `kubectl edit` command.

Prerequisites

Verify that you have owner permissions on the namespace that you want to update.

Procedure**1** Log in to the Supervisor Cluster.

```
kubectl vsphere login --server IP-ADDRESS-SUPERVISOR-CLUSTER --vsphere-username VCENTER-SSO-USERNAME
```

2 Describe the namespace that you want to update.

```
kubectl describe ns testns-1
Name:          testns
Labels:        vSphereClusterID=domain-c50
Annotations:   my-ann: test-ann-2
               vmware-system-namespace-owner-count: 2
               vmware-system-resource-pool: resgroup-153
               vmware-system-resource-pool-cpu-limit: 0.4770
               vmware-system-resource-pool-memory-limit: 2000Mi
               vmware-system-self-service-namespace: true
               vmware-system-vm-folder: group-v154
Status:        Active

Resource Quotas
Name:          testns-1
Resource      Used  Hard
-----
requests.storage 0    5000Mi

Name:          testns-1-
storagequota
Resource      Used  Hard
-----
namespace-service-storage-profile.storageclass.storage.k8s.io/requests.storage 0
9223372036854775807
```

3 Edit the namespace using the `kubectl edit` command.

For example, `kubectl edit ns testns-1`

The `kubectl edit` command opens the namespace manifest in the text editor defined by your `KUBE_EDITOR` or the `EDITOR` environment variable.

4 Update the labels.

For example, `my-label=test-label`

5 Update the annotations.

For example, `my-ann: test-ann`

6 Describe the namespace to see the updates.

```
root@localhost [ /tmp ]# kubectl describe ns testns-1
Name:          testns-1
Labels:        my-label=test-label
```

```

vSphereClusterID=domain-c50
Annotations:  my-ann: test-ann
              vmware-system-namespace-owner-count: 1
              vmware-system-resource-pool: resgroup-173
              vmware-system-resource-pool-cpu-limit: 0.4770
              vmware-system-resource-pool-memory-limit: 2000Mi
              vmware-system-self-service-namespace: true
              vmware-system-vm-folder: group-v174
Status:      Active

```

Resource Quotas

```

Name:         testns-1
Resource      Used  Hard
-----
requests.storage 0    5000Mi

```

```

Name:         testns-1-
storagequota
Resource      Used  Hard
-----
namespace-service-storage-profile.storageclass.storage.k8s.io/requests.storage 0
9223372036854775807

```

No LimitRange resource.

Delete a Self-Service Namespace

As a DevOps Engineer, you can delete a self-service namespace that you create.

Prerequisites

Verify that you have created a self-service namespace using the vSphere Plugin for kubectl.

Procedure

- 1 Using the vSphere Plugin for kubectl, authenticate with the Supervisor Cluster. See [Connect to the Supervisor Cluster as a vCenter Single Sign-On User](#).
- 2 Delete the self-service namespace from the cluster.

```
kubectl delete namespace NAMESPACE NAME
```

For example:

```
kubectl delete namespace test-ns
```


Managing Supervisor Services with vSphere with Tanzu



Supervisor Services are vSphere certified Kubernetes operators that deliver Infrastructure-as-a-Service components and tightly-integrated Independent Software Vendor services to developers. You can install and manage Supervisor Services on the vSphere with Tanzu environment so that to make them available for use with Kubernetes workloads.

When Supervisor Services are installed on Supervisor Clusters, DevOps engineers can consume them in different ways:

- Shared Supervisor Services directly provide functionality to workloads running in TKG clusters, vSphere Pods, or VMs. .
- Supervisor Services that include an operator, such as MinIO, typically provide API or UI interfaces, which DevOps engineers can use to create and manage instances of the service in a vSphere Namespace through CRDs. For example, to create a Minio bucket, you use a CRD to create the bucket in a vSphere Namespace.

Learn more about the supported Supervisor Services and how to download their service YAML files at <http://vmware.com/go/supervisor-service>.

Note In the vSphere 7.x release, only Supervisor Clusters configured with the NSX networking stack support Supervisor Services.

You manage Supervisor Services in the vSphere Services platform from the vSphere Client. By using the platform, you can manage the life cycle of Supervisor Services, install them on Supervisor Clusters, as well as perform version control. A Supervisor Service can have multiple versions that you can install on Supervisor Clusters as only one version at a time can run on a Supervisor Cluster.

Table 8-1. Supervisor Service States

State	Service Version	Entire Service
Active	The service version is ready to be installed on Supervisor Clusters version.	At least one service version is in Active state.
Deactivated	The service version cannot be installed on Supervisor Clusters. It can continue running on any Supervisor Clusters where it is installed, but you cannot install a Deactivated version on new Supervisor Clusters.	When an entire Supervisor Service is Deactivated, all of its versions are also Deactivated and you cannot install any of them on Supervisor Clusters or add new service versions until you reactivate the service.

Supervisor Services Life Cycle Management Operations

Managing the life cycle of a Supervisor Service includes the following operations:

Operation	Description
Add a new Supervisor Service to vCenter Server	When you add a new service to vCenter Server, the service and all information about it is registered with vCenter Server. The service is not yet installed on any Supervisor Cluster. After the service is registered with vCenter Server, its state is Active, which means that you can install that service on Supervisor Clusters.
Add a new Supervisor Service version to vCenter Server	Once, you have added a Supervisor Service to vCenter Server, you can add new versions of that service. After the new service version is registered with vCenter Server it becomes in Active state and you can install the version on Supervisor Clusters.
Install a Supervisor Service on Supervisor Clusters	When you install a service on a Supervisor Cluster, the service YAML file is applied on the cluster, and all the pods and necessary resources are created for the service to operate. Each service you install on a Supervisor Cluster has a dedicated namespace where you can manage the service resources. Supervisor Services might also have a UI plug-in for vCenter Server, where you can manage the service configuration.
Upgrade a Supervisor Service	You can upgrade a service that is installed on a Supervisor Cluster by first adding a new service version to vCenter Server and then installing the new version on the Supervisor Cluster. During the service upgrade, the YAML file of the new version is applied to the Supervisor Cluster. Any resources that are specified in the previous service version that are not required by the new version will be deleted. For example, if version 1 specifies pod A and version 2 specifies pod B, after the upgrade to version 2 a new pod B is created and pod A is deleted. No currently running workloads are impacted during the process.

Operation	Description
Uninstall a Supervisor Service version	Uninstalling a service version from a Supervisor Cluster leads to all of the services resources being removed from the cluster including the service namespace. Application instances of the service in Kubernetes workloads will continue to run.
Delete a Supervisor Service version	To delete a service version, first you must Deactivate that version and uninstall it from the Supervisor Clusters where it runs. Then you can delete the service version from vCenter Server
Delete an entire Supervisor Service	To delete entire service, you must deactivate all of its versions, then uninstall these versions from Supervisor Clusters, and finally delete all the service versions.

Read the following topics next:

- [Add a Supervisor Service to vCenter Server](#)
- [Install a Supervisor Service on Supervisor Clusters](#)
- [Access the Management Interface of a Supervisor Service on the Supervisor Cluster](#)
- [Add a New Version to a Supervisor Service](#)
- [View Supervisor Services Installed on a Supervisor Cluster](#)
- [Deactivate a Supervisor Service or a Version](#)
- [Activate a Supervisor Service Version on vCenter Server](#)
- [Uninstall a Supervisor Service from a Supervisor Cluster](#)
- [Delete a Supervisor Service Version](#)
- [Delete a Supervisor Service](#)

Add a Supervisor Service to vCenter Server

You can add Supervisor Services to the vCenter Server system where your vSphere with Tanzu environment runs. After adding services to vCenter Server, you install Supervisor Services on Supervisor Clusters so that your DevOps engineers can use the services in Kubernetes workloads.

- Learn more about the supported Supervisor Services and how to download their service YAML files at <http://vmware.com/go/supervisor-service>.

Prerequisites

- Verify that you have the **Manage Supervisor Services** privilege on the vCenter Server system where you add the service.

Procedure

- 1 From the vSphere Client home menu, select **Workload Management**.
- 2 Select **Services**.
- 3 Select a vCenter Server system from the drop-down menu at the top.
- 4 Drag and drop the service YAML file in the **Add New Service** card.

New Service

- 1 Register Service
- 2 EULA

Register Service [Close]

YAML was uploaded successfully. Note: YAML content is not verified and could fail during installation into a Supervisor Cluster. [Close]

Running 3rd party services on user workloads has security risks. A 3rd party service has network access to user workloads, Pod VMs, and exposed APIs. [Close]

Upload service definition to support the service on vSphere.

YAML File details [Upload new](#)

minio-supervisorservice-2.0.0.yaml

Service Details

vCenter	sc2-10-185-226-93.eng.vmware.com
Service Name	MinIO
Service ID	minio
Service Description	MinIO is a high-performance, cloud-native object store. Compatible with the Amazon S3 API, it scales seamlessly to hundreds of PBs and is simple to deploy and manage.
Version	2.0.0

[CANCEL](#) [NEXT](#)

- 5 Click **Next** and accept the EULA if any.
- 6 Click **Finish**.

Results

The Supervisor Service and all of its information is registered with the vCenter Server system. The service is in Active state.

Workload Management

Namespaces Supervisor Clusters **Services** Updates

vSphere Services | SC2-10-185-226-93.ENG.VMWARE.COM

vSphere Services is a platform for managing core infrastructure components, such as virtual machines. Application teams are able to deploy instances of vSphere Services within their own Namespaces using industry standard tools and practices.

Sort By: Recently added

Below are the services registered to the selected vCenter. Services with multiple versions can be managed from the same Service card.

MiniO was registered successfully on vCenter sc2-10-185-226-93.eng.vmware.com. This service can now be [View Supervisor Clusters](#) installed on Supervisor Clusters.

Add New Service

or drop a service bundle file

ADD

VM Service

This service allows developers to self-service VMs and allows you to set policies for VM deployment.

MANAGE

MinIO

Status: Active

Active Versions **1** Supervisors **0**

MinIO is a high-performance, cloud-native obje...

ACTIONS

What to do next

Install the Supervisor Service on Supervisor Clusters so that your DevOps engineers can use it in Kubernetes workloads. See [Install a Supervisor Service on Supervisor Clusters](#).

Install a Supervisor Service on Supervisor Clusters

After adding a Supervisor Service to vCenter Server, you can install it on Supervisor Clusters in your vSphere with Tanzu environment. If you install a newer version of a Supervisor Service, it overrides any older service versions on that Supervisor Cluster. Only one version of a Supervisor Service can run on a Supervisor Cluster at a time.

- Learn more about the supported Supervisor Services and how to download their service YAML files at <http://vmware.com/go/supervisor-service>.

Prerequisites

- Add a new Supervisor Service or a newer version of an existing service to vCenter Server. See [Add a Supervisor Service to vCenter Server](#) or [Add a New Version to a Supervisor Service](#).
- Verify that you have the **Manage Supervisor Services** privilege on the vCenter Server system that hosts the Supervisor Cluster where you install the service.
- If the Supervisor Service requires persistent storage, configure the vSAN Data Persistence platform. See [Using vSAN Data Persistence Platform with Modern Stateful Services](#).

Procedure

- 1 From the vSphere Client home menu, select **Workload Management**.
- 2 Select **Services**.
- 3 Select the Supervisor Service version that you want to install.

Note You cannot install Supervisor Service versions that are Deactivated.

- 4 Select the Supervisor Cluster where you want to install the service.
- 5 Fill in the registry endpoint, user name, and password if you have hosted the images in a private registry.

Other key-value pairs might exist in case the service requires them. Refer to the service documentation for more details.

Results

The Supervisor Service is in Configuring state, which means that all the necessary resources are being created in the Supervisor Cluster and the service YAML is being applied to the cluster. Once the YAML is successfully applied to the Supervisor Cluster with all its resources and namespace created or updated, the service state changes to Configured. The service is available to all the namespaces in that cluster and DevOps engineers can use it with their Kubernetes workloads.

vSphere Services

INSTALLED AVAILABLE

Below are the service versions installed on this Supervisor Cluster. [View available services for installation](#)

EDIT UNINSTALL

	Service Version Name	Namespace	Status	Version	Desired version
☐	Hyperstore	svc-hyperstore-domain-c9	✔ Configured	1.0.0	1.0.0

1 item

Note The Supervisor Cluster is not monitoring if the Supervisor Service is actually working. For example, if you have specified incorrect registry user name or password, the service might not be able to fetch any of the images that it needs to run. This way, the service might show as Configured but it's not actually working. To confirm if the Supervisor Service is working or not, check the service namespace and the running pods.

What to do next

Configure the Supervisor Service by using the interface. See where to find it at [Access the Management Interface of a Supervisor Service on the Supervisor Cluster](#)

Access the Management Interface of a Supervisor Service on the Supervisor Cluster

Check out where to find the management UI of Supervisor Services once you install them on a Supervisor Cluster. Supervisor Services can provide their own UI plug-in for vCenter Server that adds the service interface to the Supervisor Cluster view in the vSphere Client. Depending on the Supervisor Service specifics, you can use its interface to configure and manage the service also to deploy service instances of that service.

Procedure

- 1 In the vSphere Client, navigate to the Supervisor Cluster.
- 2 Select **Configure** and scroll down to the service interface, which is usually named after the service, for example **MinIO**.

Add a New Version to a Supervisor Service

Once you have added a Supervisor Service to vCenter Server where you have your vSphere with Tanzu environment, you can add new version to that service. You can install different service versions on Supervisor Clusters.

- Learn more about the supported Supervisor Services and how to download their service YAML files at <http://vmware.com/go/supervisor-service>.

Prerequisites

- Add the service to vCenter Server. See [Add a Supervisor Service to vCenter Server](#).
- Verify that you have the **Manage Supervisor Services** privilege on the vCenter Server system where you add the new service version.

Procedure

- 1 From the vSphere Client home menu, select **Workload Management**.
- 2 Select **Services**.
- 3 In the card of the service to which you want to add a new version, select **Actions > Add New Version**.
- 4 Upload the YAML file of the new service version and click **Next**.
- 5 Accept the EULA if any and click **Finish**.

Results

The new service version is added and it's in Active state.

What to do next

Install the new service version on Supervisor Clusters. See [Install a Supervisor Service on Supervisor Clusters](#).

View Supervisor Services Installed on a Supervisor Cluster

View the vSphere services installed on the Supervisor Clusters in your vSphere with Tanzu environment. Supervisor Services installed on a Supervisor Cluster are available to each namespace on the cluster.

Prerequisites

- Add Supervisor Services to vCenter Server. See [Add a Supervisor Service to vCenter Server](#).
- Install Supervisor Services on Supervisor Clusters. See [Install a Supervisor Service on Supervisor Clusters](#).

Procedure

- 1 From the vSphere Client home menu, select **Workload Management**.
- 2 Select the **Supervisor Clusters** tab.

- 3 On a Supervisor Cluster, in the **Services** column, click **View**.

vSphere Services

INSTALLED **AVAILABLE**

Below are the service versions installed on this Supervisor Cluster. [View available services for installation](#)

EDIT UNINSTALL

	Service Version Name	Namespace	Status	Version	Desired version
☐	Hyperstore	svc-hyperstore-domain-c9	✓ Configured	1.0.0	1.0.0

1 item

- In the **Installed** tab, view the Supervisor Services that are currently installed in the Supervisor Cluster.
- In the **Available** tab, view the Supervisor Services that are available for installation.

What to do next

You can manage the Supervisor Services on that Supervisor Cluster, uninstall services, or install new ones from the services in the **Available** tab.

Deactivate a Supervisor Service or a Version

Deactivate a Supervisor Service version if you no longer want to use it with Kubernetes workloads in your vSphere with Tanzu environment. A deactivated service version continues to run on the Supervisor Clusters where it is installed, but you cannot install a deactivated service version on other Supervisor Clusters. When you deactivate an entire service, all of the service versions become deactivated, you cannot add new service versions or install them on Supervisor Clusters, until you reactivate the service.

Prerequisites

- Verify that you have the **Manage Supervisor Services** privilege on vCenter Server level.

Procedure

- 1 From the vSphere Client home menu, select **Workload Management**.
- 2 Select **Services**.
- 3 In the service card, select **Actions > Manage Versions**.
 - To deactivate a Supervisor Service version, select the version and click **Deactivate**.
 - To deactivate the entire service, click **Confirm** next to **Deactivate entire service**.

Manage Versions: MinIO

Service ID: minio



⚠ Deactivating a version for this service will prevent its installation on supported Supervisor Clusters. Your running instances will not be impacted.



Below are details for all the versions available for MinIO.

- To delete a version, you must deactivate it and remove it on Supervisor Clusters before deleting.
- To delete a service, you must first deactivate the entire service and remove its versions on Supervisor Clusters.

You cannot create instances on Supervisor Clusters with deactivated versions and services.

DEACTIVATE DELETE

	Service Version Name	Version	Status	Supervisor Clusters
	MinIO	3.0.0	Active	0
	MinIO	2.0.0	Active	0
				2 items

Deactivate entire service CONFIRM

You must deactivate a service before deleting it.

- All versions will also be deactivated.
- Versions cannot be added or changed.
- Versions cannot be installed on clusters.

CLOSE

Results

The service version is deactivated and you cannot install on Supervisor Clusters.

Activate a Supervisor Service Version on vCenter Server

Once a Supervisor Service version has been deactivated, you can activate it again in case your DevOps team wants to use that service version in their Kubernetes workloads running on vSphere with Tanzu.

- Verify that you have the **Manage Supervisor Services** privilege on the vCenter Server system where the service is registered.

Procedure

- 1 From the vSphere Client home menu, select **Workload Management**.
- 2 Select **Services**.
- 3 In the Supervisor Service card click **Active Versions**.
- 4 Select **Manage Versions**.
- 5 Select the Supervisor Service version that is in Deactivated status and click **Reactivate**.

Uninstall a Supervisor Service from a Supervisor Cluster

Uninstall a Supervisor Service from a Supervisor Cluster if your DevOps team no longer needs that service for their Kubernetes workloads running in the vSphere with Tanzu environment.

Prerequisites

- Verify that you have the **Manage Supervisor Services** privilege on the vCenter Server system that hosts the Supervisor Cluster where the service is installed.

Procedure

- 1 From the vSphere Client home menu, select **Workload Management**.
- 2 Select **Supervisor Clusters**.
- 3 On a Supervisor Cluster, in the **Services** column, click **View**.
- 4 Select the Supervisor Service that you want to uninstall and click **Uninstall**.

Results

The Supervisor Service is uninstalled from the Supervisor Cluster. All the services resources and the service namespace are removed from the Supervisor Cluster. All the managed instances of services that use the vSAN Data Persistence Platform are removed from the Supervisor Cluster.

Delete a Supervisor Service Version

Delete the version of a Supervisor Service from vCenter Server if this version is obsolete and your DevOps team no longer need it for their Kubernetes workload running in the vSphere with Tanzu environment.

Prerequisites

- Verify that the Supervisor Service version that you want to delete is not installed on Supervisor Clusters. See [Uninstall a Supervisor Service from a Supervisor Cluster](#).
- Verify that you have the **Manage Supervisor Services** privilege on vCenter Server level.

Procedure

- 1 From the vSphere Client home menu, select **Workload Management**.
- 2 Select **Services**.
- 3 In the Supervisor Service card, select **Actions > Manage Versions**.
- 4 Select the version that you want to delete and click **Deactivate**.
- 5 Select the deactivated version and click **Delete**.

Delete a Supervisor Service

Delete a Supervisor Service from the vSphere with Tanzu environment if your DevOps engineers no longer need it for their Kubernetes workloads.

Prerequisites

- Verify that you have the **Manage Supervisor Services** privilege on the vCenter Server system where the service is registered.

Procedure

- 1 From the vSphere Client home menu, select **Workload Management**.
- 2 Select **Services**.
- 3 In the card of the Supervisor Service that you want to remove, select **Actions > Delete**.
- 4 Confirm deactivating all currently available service versions.
- 5 Confirm uninstalling the service from Supervisor Clusters.

Uninstalling a Supervisor Service from the Supervisor Clusters where it runs might take some time. You can close the dialog while the process completes and then reopen it to proceed with the next stage.

Delete Hyperstore | Service ID: hyperstore



You cannot delete the service until all steps are complete.



Uninstalling Hyperstore versions from Supervisor Clusters. You can close this modal and come back.



Hyperstore deactivated.



Impact to services upon uninstallation is dependent on each operator. Running instances might be deleted.



1. Service deactivated.

- All versions will also be deactivated.
- Versions cannot be added or changed.
- Versions cannot be installed on clusters.

[REACTIVATE](#)

2. Uninstall all versions from Supervisor Clusters.

All versions need to be uninstalled from Supervisor Clusters for the Service to be deleted.

CONFIRM

Supervisor Cluster	Service Version Name	Version	Service Status
compute-cluster	Hyperstore	1.0.0	Removing
			1 item

3. Delete all versions of the Service.

All versions needs to be deleted before deleting the service.

DELETE

6 Confirm deleting all available versions of the service.

7 Click **Delete**.

Connecting to vSphere with Tanzu Clusters

9

You connect to the Supervisor Cluster to provision Tanzu Kubernetes clusters. Once provisioned you can connect to Tanzu Kubernetes clusters using various methods and authenticate based on your role and objective.

Read the following topics next:

- [Download and Install the Kubernetes CLI Tools for vSphere](#)
- [Configure Secure Login for vSphere with Tanzu Clusters](#)
- [Connect to the Supervisor Cluster as a vCenter Single Sign-On User](#)
- [Authenticating with Tanzu Kubernetes Clusters](#)
- [Connect to a Tanzu Kubernetes Cluster as a vCenter Single Sign-On User](#)
- [Connect to the Tanzu Kubernetes Cluster Control Plane as the Administrator](#)
- [SSH to Tanzu Kubernetes Cluster Nodes as the System User Using a Private Key](#)
- [SSH to Tanzu Kubernetes Cluster Nodes as the System User Using a Password](#)
- [Grant Developer Access to Tanzu Kubernetes Clusters](#)

Download and Install the Kubernetes CLI Tools for vSphere

You can use Kubernetes CLI Tools for vSphere to view and control vSphere with Tanzu namespaces and clusters.

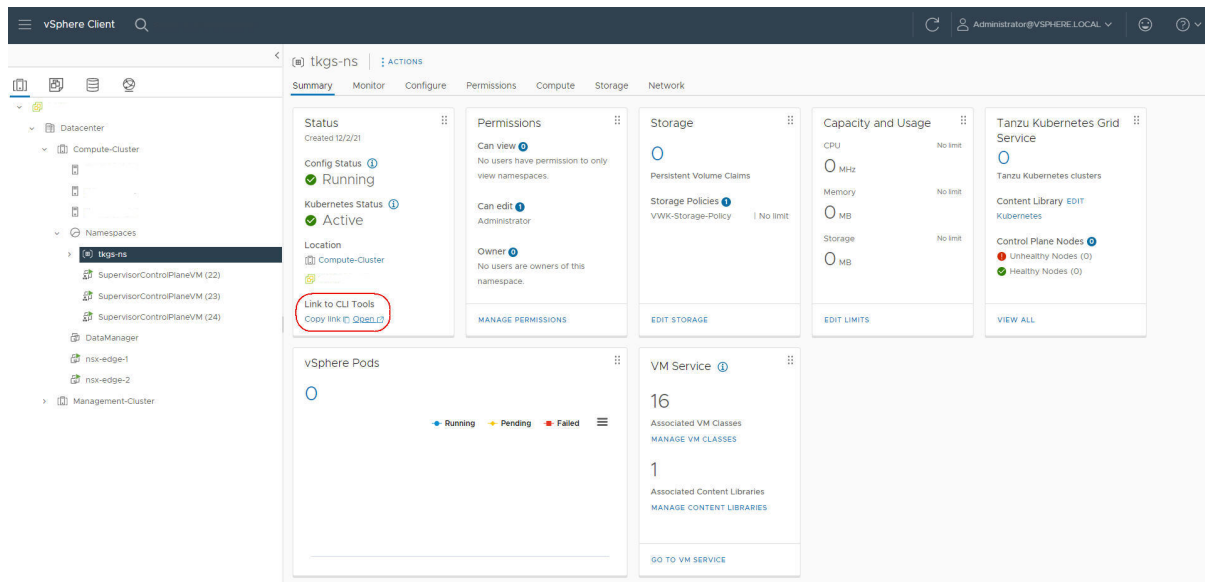
The Kubernetes CLI Tools download package includes two executables: the standard open-source kubectl and the vSphere Plugin for kubectl. The kubectl CLI has a pluggable architecture. The vSphere Plugin for kubectl extends the commands available to kubectl so that you connect to the Supervisor Cluster and to Tanzu Kubernetes clusters using vCenter Single Sign-On credentials.

Note As a best practice, once you have performed a vSphere Namespace update and upgraded the Supervisor Cluster, update the vSphere Plugin for kubectl. See [Update the vSphere Plugin for kubectl](#).

Prerequisites

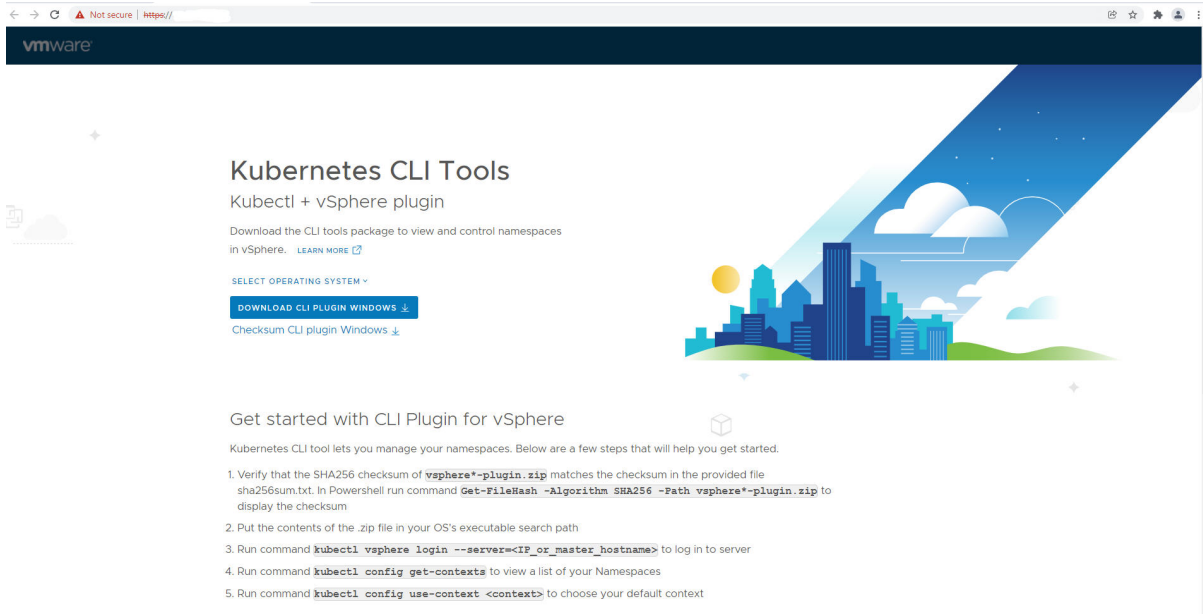
- Get the link for the Kubernetes CLI Tools download page from your vSphere administrator.

- Alternatively, if you have access to the vCenter Server, get the link as follows:
 - Log in to the vCenter Server using the vSphere Client.
 - Navigate to **vSphere Cluster > Namespaces** and select the vSphere Namespace where you are working. In the example shown below, this is the "tkgs-ns" namespace that we created for our Tanzu Kubernetes clusters.
 - Select the **Summary** tab and locate the **Status** area on this page.
 - Select **Open** beneath the **Link to CLI Tools** heading to open the download page. Or, you can **Copy** the link.



Procedure

- 1 Using a browser, navigate to the **Kubernetes CLI Tools** download URL for your environment. Refer to the prerequisites section above for guidance on how to locate the download URL.



- 2 Select the operating system.
- 3 Download the `vsphere-plugin.zip` file.
- 4 Extract the contents of the ZIP file to a working directory.

The `vsphere-plugin.zip` package contains two executable files: `kubectl` and `vSphere Plugin for kubectl`. `kubectl` is the standard Kubernetes CLI. `kubectl-vsphere` is the vSphere Plugin for kubectl to help you authenticate with the Supervisor Cluster and Tanzu Kubernetes clusters using your vCenter Single Sign-On credentials.

- 5 Add the location of both executables to your system's PATH variable.
- 6 To verify the installation of the kubectl CLI, start a shell, terminal, or command prompt session and run the command `kubectl`.

You see the kubectl banner message, and the list of command-line options for the CLI.

- 7 To verify the installation of the vSphere Plugin for kubectl, run the command `kubectl vsphere`.

You see the vSphere Plugin for kubectl banner message, and the list of command-line options for the plugin.

What to do next

[Configure Secure Login for vSphere with Tanzu Clusters.](#)

Configure Secure Login for vSphere with Tanzu Clusters

To log in securely to vSphere with Tanzu clusters, including the Supervisor Cluster and Tanzu Kubernetes clusters, configure the vSphere Plugin for kubectl with the appropriate TLS certificate and ensure that you are running the latest edition of the plugin.

Supervisor Cluster CA Certificate

vSphere with Tanzu supports vCenter Single Sign-On for cluster access using the vSphere Plugin for kubectl command `kubectl vsphere login ....` To install and use this utility, see [Download and Install the Kubernetes CLI Tools for vSphere](#).

The vSphere Plugin for kubectl defaults to secure login and requires a trusted certificate, the default being the certificate signed by the vCenter Server root CA. Although the plugin supports the `--insecure-skip-tls-verify` flag, for security reasons this is not recommended.

To securely log in to the Supervisor Cluster and Tanzu Kubernetes clusters using the vSphere Plugin for kubectl, you have two options:

Option	Instructions
Download and install the vCenter Server root CA certificate on each client machine.	Refer to the VMware knowledge base article How to download and install vCenter Server root certificates .
Replace the VIP certificate used for the Supervisor Cluster with a certificate signed by a CA each client machine trusts.	See Replace the VIP Certificate to Securely Connect to the Supervisor Cluster API Endpoint

Note For additional information on vSphere authentication, including vCenter Single Sign-On, managing and rotating vCenter Server certificates, and troubleshooting authentication, see the [vSphere Authentication](#) documentation.

Tanzu Kubernetes Cluster CA Certificate

To connect securely with the Tanzu Kubernetes cluster API server using the `kubectl` CLI, you need to download the Tanzu Kubernetes cluster CA certificate.

If you are using the latest edition of the vSphere Plugin for kubectl, the first time you log in to the Tanzu Kubernetes cluster, the plugin registers the Tanzu Kubernetes cluster CA certificate in your kubeconfig file. This certificate is stored in the Kubernetes secret named `TANZU-KUBERNETES-CLUSTER-NAME-ca`. The plugin uses this certificate to populate the CA information in the corresponding cluster's certificate authority datastore.

If you are updating vSphere with Tanzu, make sure you update to the latest version of the plugin. See [Update the vSphere Plugin for kubectl](#).

Connect to the Supervisor Cluster as a vCenter Single Sign-On User

To provision vSphere Pods, or Tanzu Kubernetes clusters by using the Tanzu Kubernetes Grid Service, you connect to the Supervisor Cluster by using the vSphere Plugin for kubectl and authenticate with your vCenter Single Sign-On credentials.

After you log in to the Supervisor Cluster, the vSphere Plugin for kubectl generates the context for the cluster. In Kubernetes, a configuration context contains a cluster, a namespace, and a user. You can view the cluster context in the file `.kube/config`. This file is commonly called the `kubeconfig` file.

Note If you have an existing `kubeconfig` file, it is appended with each cluster context. The vSphere Plugin for kubectl respects the `KUBECONFIG` environment variable that kubectl itself uses. Although not required, it can be useful to set this variable before running `kubectl vsphere login ...` so that the information is written to a new file, instead of being added to your current `kubeconfig` file.

Prerequisites

- Get your vCenter Single Sign-On credentials.
- Get the IP address of the Supervisor Cluster control plane.
- Get the name of the vSphere Namespace.
- Get confirmation that you have **Edit** permissions on the vSphere Namespace.
- [Download and Install the Kubernetes CLI Tools for vSphere.](#)
- Verify that the certificate served by the Kubernetes control plane is trusted on your system, either by having the signing CA installed as a Trust Root or by adding the certificate as a Trust Root directly. See [Configure Secure Login for vSphere with Tanzu Clusters](#).

Procedure

- 1 To view the command syntax and options for logging in, run the following command.

```
kubectl vsphere login --help
```

- 2 To connect to the Supervisor Cluster, run the following command.

```
kubectl vsphere login --server=<KUBERNETES-CONTROL-PLANE-IP-ADDRESS> --vsphere-username
<VCENTER-SSO-USER>
```

For example:

```
kubectl vsphere login --server=10.92.42.13 --vsphere-username administrator@example.com
```

This action creates a configuration file with the JSON Web Token (JWT) to authenticate to the Kubernetes API.

- 3 To authenticate, enter the password for the user.

After you connect to the Supervisor Cluster, you are presented with the configuration contexts can access. For example:

```
You have access to the following contexts:
tanzu-ns-1
tkg-cluster-1
tkg-cluster-2
```

- 4 To view details of the configuration contexts which you can access to, run the following `kubectl` command:

```
kubectl config get-contexts
```

The CLI displays the details for each available context.

- 5 To switch between contexts, use the following command:

```
kubectl config use-context <example-context-name>
```

What to do next

[Connect to a Tanzu Kubernetes Cluster as a vCenter Single Sign-On User.](#)

Authenticating with Tanzu Kubernetes Clusters

You can authenticate with the Tanzu Kubernetes cluster environment in various ways depending on your role and purpose.

DevOps engineers provision and operate Tanzu Kubernetes clusters. Developers deploy workloads to Tanzu Kubernetes clusters. Administrators might need to troubleshoot Tanzu Kubernetes clusters. vSphere with Tanzu provides authentication methods supporting each role or objective.

- DevOps engineers connect to the Supervisor Cluster to provision and update Tanzu Kubernetes clusters. Authentication is done using the vSphere Plugin for `kubectl` and vCenter Single Sign-On credentials. See [Connect to the Supervisor Cluster as a vCenter Single Sign-On User](#).
- Cluster administrators connect to a provisioned Tanzu Kubernetes cluster to operate and manage it.
 - A user granted the **Edit** permission on the vSphere Namespace where the cluster is deployed is assigned to the `cluster-admin` role. Cluster administrators authenticate using the vSphere Plugin for `kubectl` and their vCenter Single Sign-On credentials. See [Connect to a Tanzu Kubernetes Cluster as a vCenter Single Sign-On User](#).

- Alternatively, cluster administrators can connect to a Tanzu Kubernetes cluster as the `kubernetes-admin` user. This method might be appropriate if vCenter Single Sign-On authentication is not available. See [Connect to the Tanzu Kubernetes Cluster Control Plane as the Administrator](#).
- Cluster users or developers connect to a Tanzu Kubernetes cluster to deploy workloads, including pods, services, load balancers, and other resources.
 - A cluster administrator grants cluster access to developers by binding the user or group to default or custom pod security policy. For more information, see [Grant Developer Access to Tanzu Kubernetes Clusters](#).
 - Bound developers authenticate with Tanzu Kubernetes clusters using the vSphere Plugin for kubectl and their vCenter Single Sign-On credentials. See [Connect to a Tanzu Kubernetes Cluster as a vCenter Single Sign-On User](#).
- For troubleshooting purposes, system administrators can connect to a Tanzu Kubernetes cluster as the `vmware-system-user` using SSH and a private key. See [SSH to Tanzu Kubernetes Cluster Nodes as the System User Using a Private Key](#).

Connect to a Tanzu Kubernetes Cluster as a vCenter Single Sign-On User

You can connect to a Tanzu Kubernetes cluster using the vSphere Plugin for kubectl and authenticate with your vCenter Single Sign-On credentials.

After you log in to the Tanzu Kubernetes cluster, the vSphere Plugin for kubectl generates the context for the cluster. In Kubernetes, a configuration context contains a cluster, a namespace, and a user. You can view the cluster context in the file `.kube/config`. This file is commonly called the `kubeconfig` file.

Note If you have an existing `kubeconfig` file, it is appended with each cluster context. The vSphere Plugin for kubectl respects the `KUBECONFIG` environment variable that kubectl itself uses. Although not required, it can be useful to set this variable before running `kubectl vsphere login ...` so that the information is written to a new file, instead of being added to your current `kubeconfig` file.

Prerequisites

Obtain the following information from your vSphere administrator:

- Get your vCenter Single Sign-On credentials.
- Get the IP address of the Supervisor Cluster control plane.
- Get the name of the vSphere Namespace.
- [Download and Install the Kubernetes CLI Tools for vSphere](#).

Procedure

- 1 To view the command syntax and options for logging in, run the following command.

```
kubectl vsphere login --help
```

- 2 To connect to the Tanzu Kubernetes cluster, run the following command.

```
kubectl vsphere login --server=SUPERVISOR-CLUSTER-CONTROL-PLANE-IP
--tanzu-kubernetes-cluster-name TANZU-KUBERNETES-CLUSTER-NAME
--tanzu-kubernetes-cluster-namespace SUPERVISOR-NAMESPACE-WHERE-THE-CLUSTER-IS-DEPLOYED
--vsphere-username VCENTER-SSO-USER-NAME
```

For example:

```
kubectl vsphere login --server=10.92.42.137
--tanzu-kubernetes-cluster-name tanzu-kubernetes-cluster-01
--tanzu-kubernetes-cluster-namespace tanzu-ns-1
--vsphere-username administrator@example.com
```

This action creates a configuration file with the JSON Web Token (JWT) to authenticate to the Kubernetes API.

- 3 To authenticate, enter your vCenter Single Sign-On password.

If the operation is successful, you see the message `Logged in successfully`, and you can run `kubectl` commands against the cluster. If the command returns `Error from server (Forbidden)`, typically this error means you do not have the required permissions. For more information, see [Troubleshoot vCenter Single Sign-On Connection Errors](#).

- 4 To get a list of contexts available to you, run the following command:

```
kubectl config get-contexts
```

This command lists the configuration contexts you have access to. You see a configuration context for the target cluster, such as `tkg-cluster-01`.

- 5 To use the context for the target cluster, run the following command:

```
kubectl config use-context CLUSTER-NAME
```

- 6 To list cluster nodes, run the following command:

```
kubectl get nodes
```

You see the control plane and worker nodes in this cluster.

- 7 To list all the cluster pods, run the following command:

```
kubectl get pods -A
```

You see all the pods in this cluster across all Kubernetes namespaces that you have access to. If you have not deployed any workloads, you do not see any pods in the default namespace.

Connect to the Tanzu Kubernetes Cluster Control Plane as the Administrator

You can connect to the Tanzu Kubernetes cluster control plane as the `kubernetes-admin` user to perform administrative tasks and troubleshoot cluster problems.

A valid `kubeconfig` file for a provisioned Tanzu Kubernetes cluster is available on the Supervisor Cluster as a secret object named `TKGS-CLUSTER-NAME-kubeconfig`. You can use this secret to connect to the cluster control plane as the `kubernetes-admin` user. For more information, see [Get Tanzu Kubernetes Cluster Secrets](#).

Procedure

- 1 Connect to the Supervisor Cluster. See [Connect to the Supervisor Cluster as a vCenter Single Sign-On User](#).
- 2 Switch context to the vSphere Namespace where the target Tanzu Kubernetes cluster is provisioned.

```
kubectl config use-context VSPHERE-NAMESPACE
```

- 3 View the secret objects in the namespace.

```
kubectl get secrets
```

The secret is named `TKGS-CLUSTER-NAME-kubeconfig`.

```
kubectl config use-context tkgs-cluster-ns
Switched to context "tkgs-cluster-ns".
ubuntu@ubuntu:~$ kubectl get secrets
```

NAME	TYPE	DATA	AGE
...			
tkgs-cluster-1-kubeconfig	Opaque	1	23h
...			

- 4 Decode the secret by running the following command.

The secret is Base64 encoded. To decode it: on Linux use `base64 --decode` (or `base64 -d`); on MacOS, use `base64 --Decode` (or `base64 -D`); on Windows, use an [online tool](#).

```
kubectl get secret TKGS-CLUSTER-NAME-kubeconfig -o jsonpath='{.data.value}' | base64 -d >
tkgs-cluster-kubeconfig-admin
```

This command decodes the secret and writes it to a local file named `tkgs-cluster-kubeconfig-admin`. Use the `cat` command to verify the file contents.

- 5 Connect to the Tanzu Kubernetes cluster as the Kubernetes administrator using the decoded `tkgs-cluster-kubeconfig-admin` file.

There are two options to do this:

Option	Description
<code>--kubeconfig <path\to\kubeconfig></code>	Use the <code>--kubeconfig</code> flag and the path to the local kubeconfig file. For example, assuming the kubeconfig file is in the same directory where you are running the command: <code>kubectl --kubeconfig tkgs-cluster-kubeconfig-admin get nodes</code>
KUBECONFIG	Set your KUBECONFIG environment variable to point to the decoded kubeconfig file and run <code>kubectl</code> , such as <code>kubectl get nodes</code> .

You should see the nodes in the cluster. For example:

```
kubectl --kubeconfig tkgs-cluster-kubeconfig-admin get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
tkgs-cluster-1-control-plane-4ncm4	Ready	master	23h	v1.18.5+vmware.1
tkgs-cluster-1-control-plane-jj9gq	Ready	master	23h	v1.18.5+vmware.1
tkgs-cluster-1-control-plane-r4hm6	Ready	master	23h	v1.18.5+vmware.1
tkgs-cluster-1-workers-6nj7-84dd7f48c6-nz2n8	Ready	<none>	23h	v1.18.5+vmware.1
tkgs-cluster-1-workers-6nj7-84dd7f48c6-rk9pk	Ready	<none>	23h	v1.18.5+vmware.1
tkgs-cluster-1-workers-6nj7-84dd7f48c6-zzngh	Ready	<none>	23h	v1.18.5+vmware.1

SSH to Tanzu Kubernetes Cluster Nodes as the System User Using a Private Key

You can SSH to a Tanzu Kubernetes cluster node as the `vmware-system-user` using a private key.

You can connect through SSH to any Tanzu Kubernetes cluster node as the `vmware-system-user` user. The secret that contains the SSH private key is named `CLUSTER-NAME-ssh`. For more information, see [Get Tanzu Kubernetes Cluster Secrets](#).

To connect to a Tanzu Kubernetes cluster node over SSH using a private key, you create a jump box vSphere Pod on the Supervisor Cluster.

Prerequisites

In this task you provision a vSphere Pod as a jump host for SSH connectivity. vSphere Pods require NSX-T networking for the Supervisor Cluster. If you are using vDS networking for the Supervisor Cluster, use the following approach instead: [SSH to Tanzu Kubernetes Cluster Nodes as the System User Using a Password](#).

Procedure

- 1 Connect to the Supervisor Cluster.

See [Connect to the Supervisor Cluster as a vCenter Single Sign-On User](#).

- 2 Create an environment variable named **NAMESPACE** whose value is the name of the vSphere Namespace where the target Tanzu Kubernetes cluster is provisioned.

```
export NAMESPACE=VSPHERE-NAMESPACE
```

- 3 Switch context to the vSphere Namespace where the Tanzu Kubernetes cluster is provisioned.

```
kubectl config use-context $NAMESPACE
```

- 4 View the *TKGS-CLUSTER-NAME-ssh* secret object.

```
kubectl get secrets
```

- 5 Create a vSphere Pod using the following *jumpbox.yaml*.

Replace the `namespace` value `YOUR-NAMESPACE` with the vSphere Namespace where the target cluster is provisioned. Replace the `secretName` value `YOUR-CLUSTER-NAME-ssh` with name of the target cluster.

```
apiVersion: v1
kind: Pod
metadata:
  name: jumpbox
  namespace: YOUR-NAMESPACE      #REPLACE
spec:
  containers:
  - image: "photon:3.0"
    name: jumpbox
    command: [ "/bin/bash", "-c", "--" ]
    args: [ "yum install -y openssh-server; mkdir /root/.ssh; cp /root/ssh/ssh-privatekey /
    root/.ssh/id_rsa; chmod 600 /root/.ssh/id_rsa; while true; do sleep 30; done;" ]
    volumeMounts:
    - mountPath: "/root/ssh"
      name: ssh-key
      readOnly: true
  resources:
    requests:
      memory: 2Gi
  volumes:
  - name: ssh-key
    secret:
      secretName: YOUR-CLUSTER-NAME-ssh      #REPLACE
```

- 6 Deploy the pod by applying the *jumpbox.yaml* spec.

```
kubectl apply -f jumpbox.yaml
```

```
pod/jumpbox created
```


7 Verify that the pod is running.

```
kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
jumpbox	1/1	Running	0	3h9m

Note You should also see the jumpbox pod in vCenter in the vSphere Namespace.

8 Create an environment variable with the IP address of the target cluster node by running the following set of commands.

- a Get the name of the target virtual machine.

```
kubectl get virtualmachines
```

- b Create the environment variable `VMNAME` whose value is the name of the target node.

```
export VMNAME=NAME-OF-THE-VIRTUAL-MACHINE
```

- c Create the environment variable `VMIP` whose value is the IP address of the target node VM.

```
export VMIP=$(kubectl -n $NAMESPACE get virtualmachine/$VMNAME -o
jsonpath='{.status.vmIp}')
```

9 SSH to the cluster node using the jump box pod by running the following command.

```
kubectl exec -it jumpbox /usr/bin/ssh vmware-system-user@$VMIP
```

Important It takes approximately 60 seconds to create the container and install the software. If you receive an "error executing command in container: container_linux.go:370: starting container process caused: exec: "/usr/bin/ssh": stat /usr/bin/ssh: no such file or directory," try the command again in a few seconds.

10 Confirm the authenticity of the host by entering **yes**.

```
The authenticity of host '10.249.0.999 (10.249.0.999)' can't be established.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.249.0.999' (ECDSA) to the list of known hosts.
Welcome to Photon 3.0
```

- 11 Confirm that you are logged into the target node as the `vmware-system-user`.

For example, the following output indicates that you are logged into a control plane node as the system user.

```
vmware-system-user@tkgs-cluster-1-control-plane-66tbr [ ~ ]$
```

- 12 Perform the desired operations on the node.

Attention You might need to use `sudo` or `sudo su` to perform certain operations on the node, such as restarting kubelet.

- 13 When done, type `exit` to log out of the SSH session on the vSphere Pod.

- 14 To delete the pod, run the command `kubectl delete pod jumpbox`.

Caution For security, consider deleting the jumpbox pod after you have done your work. If needed you can recreate it at a later time.

SSH to Tanzu Kubernetes Cluster Nodes as the System User Using a Password

You can SSH to a Tanzu Kubernetes cluster node as the `vmware-system-user` using a password.

You can connect to a cluster node as the `vmware-system-user` user with a password. The password is stored as a secret named `CLUSTER-NAME-ssh-password`. The password is base64 encoded in `.data.ssh-passwordkey`. You can provide the password over an SSH session. For more information about this secret, see [Get Tanzu Kubernetes Cluster Secrets](#).

Prerequisites

To route SSH connections to the appropriate workload network, deploy a Linux jump host VM in the vSphere environment where **Workload Management** is enabled. See [Create a Linux Jump Host VM](#).

Note This is a hard requirement if you want to connect to cluster nodes using SSH and you are using vDS networking, which does not support vSphere Pods. You can also use this approach with NSX-T networking if you prefer to use a password instead of a private key to connect over SSH.

Procedure

- 1 Get the IP address of the jump host VM, the username, and the password. See [Create a Linux Jump Host VM](#).
- 2 Connect to the Supervisor Cluster.
[Connect to the Supervisor Cluster as a vCenter Single Sign-On User](#).

- 3 Switch context to the vSphere Namespace where the target Tanzu Kubernetes cluster is provisioned.

```
kubectl config use-context VSPHERE-NAMESPACE
```

- 4 Get the IP address of the target cluster node.

List the nodes.

```
kubectl get virtualmachines
```

Describe the nodes to get the IP address of the target node.

```
kubectl describe virtualmachines
```

- 5 View the `TKGS-CLUSTER-NAME-ssh-password` secret.

```
kubectl get secrets
```

- 6 Get the `ssh-passwordkey` for the target cluster.

```
kubectl get secrets TKGS-CLUSTER-NAME-ssh-password -o yaml
```

The `ssh-passwordkey` is returned, for example.

```
apiVersion: v1
data:
  ssh-passwordkey: RUlpQ1lLTC9TRjVFV0RBcCtmd1zwOTROeURYSWNGeXNReXJhaXRBULLYaz0=
```

- 7 Decode the `ssh-passwordkey`.

The secret is Base64 encoded. To decode it: on Linux use `base64 --decode` (or `base64 -d`); on MacOS, use `base64 --Decode` (or `base64 -D`); on Windows, use an [online tool](#).

```
echo <ssh-passwordkey> | base64 --decode
```

- 8 SSH to the target cluster node as the `vmware-system-user`.

```
ssh vmware-system-user@TKGS-CLUSTER-NODE-IP-ADDRESS
```

- 9 Log in using the password you decoded.

Create a Linux Jump Host VM

To SSH to Tanzu Kubernetes cluster nodes using a password, first create a jump box VM that connects to the workload network and the management or frontend network for SSH tunneling.

Create a Linux Jump Host VM

Follow these steps to create a Linux jump box VM. There are many ways to do this. This is one approach. The instructions use PhotonOS which you can download here: <https://github.com/vmware/photon/wiki/Downloading-Photon-OS>.

- 1 Log into vCenter Server using the vSphere Client.
- 2 Create a new virtual machine.
- 3 Select the Linux guest OS, in this example, VMware Photon OS (64-bit).
- 4 Install the OS. To do this, download the ISO, attach it to the VM and boot it.
- 5 Configure the VM with an IP address on the Workload network.
- 6 Add a second virtual NIC to the VM and assign it to the Frontend network.
- 7 Complete the configuration of the OS and power on the VM after reboot.
- 8 Log into the vSphere console for the VM as the root user.
- 9 Create a network interface for the new NIC and give it an IP on the Frontend network.

```
ifconfig eth1 IP-ADDRESS netmask NETMASK up
```

Note This method is non-persistent on reboots.

- 10 Verify that you can ping the gateway and DNS server through that interface
- 11 In the vSphere console for the VM, set up an SSH user with certificates. Verify that it works by creating a nested shell.
- 12 SSH into the jumpbox from the Frontend network as the SSH user to verify that works.
- 13 Install sshpass onto the VM (so you can log in over SSH using a password). For PhotonOS, the command is as follows:

```
tdnf install -y sshpass
```

- 14 Add the client's public key to the `~/.ssh/authorized_keys` file and restart the `sshd` process so that ssh can work without a password.
 - Get your public key, for example: `cat ~/.ssh/id_rsa.pub`.
 - Access the jumphost VM.
 - Create the SSH directory (if it does not exist): `mkdir -p ~/.ssh`.
 - Append the public key to the `authorized_keys` file: `echo ssh-rsa AAAA.... >> ~/.ssh/authorized_keys`. Replace `ssh-rsa AAAA....` with the entire public key string that was output from the `cat ~/.ssh/id_rsa.pub` command.
 - Ensure that the `~/.ssh` directory and `authorized_keys` file have the appropriate permissions set, for example: `chmod -R go= ~/.ssh`.

Grant Developer Access to Tanzu Kubernetes Clusters

Developers are the target users of Kubernetes. Once a Tanzu Kubernetes cluster is provisioned, you can grant developer access using vCenter Single Sign-On authentication.

Authentication for Developers

A cluster administrator can grant cluster access to other users, such as developers. Developers can deploy pods to clusters directly using their user accounts, or indirectly using service accounts. For more information, see [Using Pod Security Policies with Tanzu Kubernetes Clusters](#).

- For user account authentication, Tanzu Kubernetes clusters support vCenter Single Sign-On users and groups. The user or group can be local to the vCenter Server, or synchronized from a supported directory server.
- For service account authentication, you can use service tokens. For more information, see the Kubernetes documentation.

Adding Developer Users to a Cluster

To grant cluster access to developers:

- 1 Define a Role or ClusterRole for the user or group and apply it to the cluster. For more information, see the Kubernetes documentation.
- 2 Create a RoleBinding or ClusterRoleBinding for the user or group and apply it to the cluster. See the following example.

Example RoleBinding

To grant access to a vCenter Single Sign-On user or group, the subject in the RoleBinding must contain either of the following values for the `name` parameter.

Table 9-1. Supported User and Group Fields

Field	Description
<code>sso:USER-NAME@DOMAIN</code>	For example, a local user name, such as <code>sso:joe@vsphere.local</code> .
<code>sso:GROUP-NAME@DOMAIN</code>	For example, a group name from a directory server integrated with the vCenter Server, such as <code>sso:devs@ldap.example.com</code> .

The following example RoleBinding binds the vCenter Single Sign-On local user named Joe to the default ClusterRole named `edit`. This role permits read/write access to most objects in a namespace, in this case the `default` namespace.

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: rolebinding-cluster-user-joe
  namespace: default
```

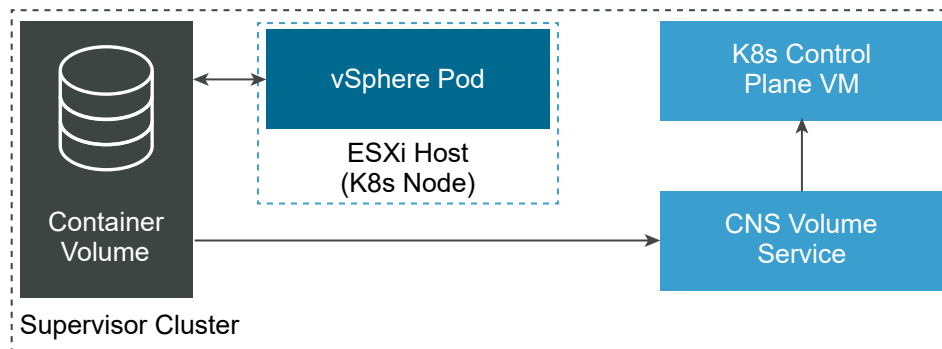
```
roleRef:
  kind: ClusterRole
  name: edit                                #Default ClusterRole
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: User
  name: sso:joe@vsphere.local              #sso:<username>@<domain>
  apiGroup: rbac.authorization.k8s.io
```

Using Persistent Storage in vSphere with Tanzu

10

Certain Kubernetes workloads require persistent storage to store data permanently. To provision persistent storage for Kubernetes workloads, vSphere with Tanzu integrates with Cloud Native Storage (CNS), a vCenter Server component that manages persistent volumes.

Persistent storage is used by vSphere Pods, Tanzu Kubernetes clusters, and VMs. The following example illustrates how persistent storage is used by a vSphere Pod.



To understand how vSphere with Tanzu works with persistent storage, be familiar with the following essential concepts.

Persistent Volume

To provide persistent storage, Kubernetes uses persistent volumes that can retain their state and data. If persistent volumes are mounted by a pod, they continue to exist even when the pod is deleted or reconfigured. In the vSphere with Tanzu environment, the persistent volume objects are backed by the First Class Disks on a datastore.

vSphere with Tanzu supports dynamic and static provisioning of volumes in ReadWriteOnce mode, where the volumes can be mounted by a single pod.

Starting with vSphere 7.0 Update 3 release, vSphere with Tanzu also supports ReadWriteMany mode for persistent volumes in Tanzu Kubernetes clusters. With the ReadWriteMany support, a single volume can be mounted simultaneously by multiple pods or applications running in a cluster. vSphere with Tanzu uses vSAN file shares for persistent volumes of the ReadWriteMany type. For more information, see [Creating ReadWriteMany Persistent Volumes in vSphere with Tanzu](#).

Dynamic and Static Provisioning

With dynamic volume provisioning, storage does not need to be pre-provisioned and persistent volumes can be created on demand. DevOps engineers issue a persistent volume claim that references a storage class available in the namespace. vSphere with Tanzu automatically provisions the corresponding persistent volume and a backing virtual disk.

Both the Supervisor Cluster and Tanzu Kubernetes cluster support dynamic provisioning.

For an example of how to dynamically create a persistent volume, see [Provision a Dynamic Persistent Volume for a Stateful Application](#).

With static provisioning, you can use an existing storage object and make it available to a cluster.

Typically, a DevOps engineer must know the details of the existing storage object, its supported configurations, and mount options to be able to reuse it.

For an example of how to provision a static persistent volume, see [Provision a Static Persistent Volume in a Tanzu Kubernetes Cluster](#).

First Class Disk

vSphere with Tanzu uses the First Class Disk (FCD) type of virtual disks to back persistent volumes. First Class Disk, also known as Improved Virtual Disk, is a named virtual disk not associated with a VM.

First Class Disks are identified by UUID. This UUID is globally unique and is the primary identifier for the FCD. The UUID remains valid even if its FCD is relocated or snapshotted.

Persistent Volume Claim

DevOps engineers create persistent volume claims to request persistent storage resources. The request provisions a persistent volume object and a matching virtual disk. In the vSphere Client, the persistent volume claim manifests as an FCD virtual disk that can be monitored by vSphere administrators.

The claim is bound to the persistent volume. The workloads can use the claim to mount the persistent volumes and access storage.

When the DevOps engineers delete the claim, the corresponding persistent volume object and the provisioned virtual disk are also deleted.

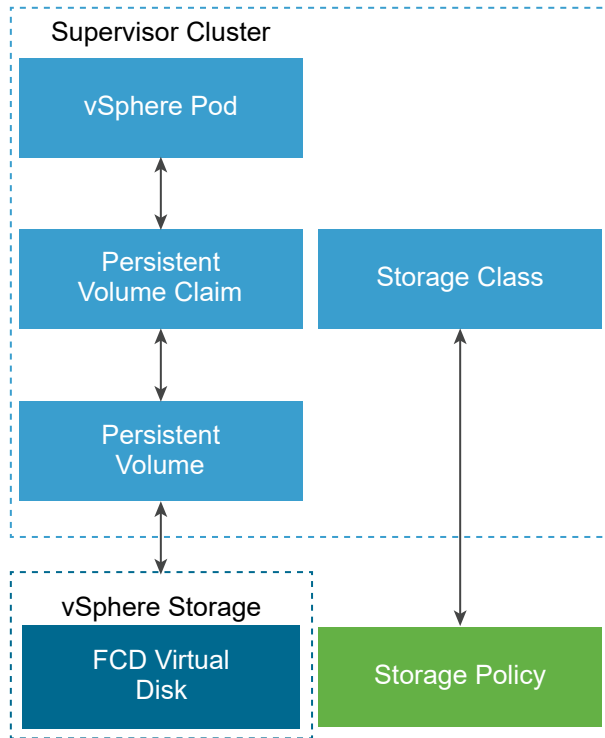
Storage Class

Kubernetes uses storage classes to describe requirements for storage backing the persistent volumes. DevOps engineers can include a specific storage class in their persistent volume claim specification to request the type of storage the class describes.

Persistent Storage Workflow

The workflow for provisioning persistent storage in vSphere with Tanzu generally includes the following sequential actions.

Step	Action	Description
1	vSphere administrators deliver persistent storage resources to the DevOps team.	vSphere administrators create VM storage policies that describe different storage requirements and classes of services. They can then assign the storage policies to a vSphere Namespace.
2	vSphere with Tanzu creates storage classes that match the storage policies assigned to the vSphere Namespace.	The storage classes automatically appear in the Kubernetes environment, and can be used by the DevOps team. If a vSphere administrator assigns multiple storage policies to the vSphere Namespace, a separate storage class is created for each storage policy. If you use the Tanzu Kubernetes Grid Service to provision Tanzu Kubernetes clusters, each Tanzu Kubernetes cluster inherits storage classes from the vSphere Namespace in which the cluster is provisioned.
3	DevOps engineers use the storage classes to request persistent storage resources for a workload.	The request comes in a form of a persistent volume claim that references a specific storage class.
4	vSphere with Tanzu creates a persistent volume object and a matching persistent virtual disk for a workload.	vSphere with Tanzu places the virtual disk into the datastore that meets the requirements specified in the original storage policy and its matching storage class. The virtual disk can be mounted by a workload.
5	vSphere administrators monitor persistent volumes in the vSphere with Tanzu environment.	Using the vSphere Client, vSphere administrators monitor the persistent volumes and their backing virtual disks. They can also monitor storage compliance and health statuses of the persistent volumes.



Watch this video to learn about the persistent storage in vSphere with Tanzu.



([Persistent Storage in vSphere with Tanzu](#))

Storage Management Tasks of a vSphere Administrator

Generally, the persistent storage management tasks in vSphere with Tanzu include the following. As a vSphere administrator, you use the vSphere Client to perform these tasks.

- Perform lifecycle operations for the VM storage policies.

Before enabling a Supervisor Cluster and configuring namespaces, create storage policies for persistent storage. The storage policies are based on the storage requirements communicated to you by the DevOps engineers. See [Create Storage Policies for vSphere with Tanzu](#).

Note Do not delete the storage policy from VMware vCenter or a vSphere Namespace when a persistent volume claim with the corresponding storage class is running in the namespace. This recommendation also applies to Tanzu Kubernetes clusters.

- Provide storage resources to the DevOps engineers by assigning the storage policies to the namespace and by setting storage limits. For information about changing storage policy assignments, see [Change Storage Settings on a Namespace](#). For information on setting limits, see [Configure Object Limitations on a vSphere Namespace](#).

- Monitor Kubernetes objects and their storage policy compliance in the vSphere Client. See [Monitor Persistent Volumes in the vSphere Client](#).

Storage Management Tasks of a DevOps Engineer

Typically, the DevOps engineer uses `kubectl` to perform the following storage tasks.

- Manage storage classes. See [Display Storage Classes in a vSphere Namespace or Tanzu Kubernetes Cluster](#).
- Deploy and manage stateful applications. See [Provision a Dynamic Persistent Volume for a Stateful Application](#).
- Perform lifecycle operations for persistent volumes. [Tanzu Kubernetes Persistent Volume Claim Examples](#).

Read the following topics next:

- [How vSphere with Tanzu Integrates with vSphere Storage](#)
- [Functionality Supported by vSphere CNS-CSI and Paravirtual CSI in vSphere with Tanzu](#)
- [Storage Permissions in vSphere with Tanzu](#)
- [Create Storage Policies for vSphere with Tanzu](#)
- [Change Storage Settings on the Supervisor Cluster](#)
- [Change Storage Settings on a Namespace](#)
- [Display Storage Classes in a vSphere Namespace or Tanzu Kubernetes Cluster](#)
- [Provision a Dynamic Persistent Volume for a Stateful Application](#)
- [Provision a Static Persistent Volume in a Tanzu Kubernetes Cluster](#)
- [Creating ReadWriteMany Persistent Volumes in vSphere with Tanzu](#)
- [Volume Expansion in vSphere with Tanzu](#)
- [Monitor Persistent Volumes in the vSphere Client](#)
- [Monitor Volume Health in a vSphere Namespace or Tanzu Kubernetes Cluster](#)
- [Using vSAN Data Persistence Platform with Modern Stateful Services](#)

How vSphere with Tanzu Integrates with vSphere Storage

vSphere with Tanzu uses several components to integrate with vSphere storage.

Cloud Native Storage (CNS) on vCenter Server

The CNS component resides in vCenter Server. It is an extension of vCenter Server management that implements provisioning and lifecycle operations for persistent volumes.

When provisioning container volumes, the component interacts with the vSphere First Class Disk functionality to create virtual disks that back the volumes. In addition, the CNS server component communicates with the Storage Policy Based Management to guarantee a required level of service to the disks.

The CNS also performs query operations that allow vSphere administrators to manage and monitor persistent volumes and their backing storage objects through vCenter Server.

First Class Disk (FCD)

Also called Improved Virtual Disk. It is a named virtual disk unassociated with a VM. These disks reside on a VMFS, NFS, or vSAN datastore and back ReadWriteOnce persistent volumes.

The FCD technology performs lifecycle operations related to persistent volumes outside of a lifecycle of a VM or pod.

When you use FCDs, keep in mind the following:

- FCDs do not support NFS 4.x protocols. Instead, use NFS 3.
- vCenter Server does not serialize operations on the same FCD. As a result, applications cannot simultaneously perform operations on the same FCD. Performing such operations as clone, relocate, delete, retrieve, and so on simultaneously from different threads causes unpredictable results. To avoid problems, applications must perform operations on the same FCD in a sequential order.
- FCD is not a managed object and does not support a global lock protecting multiple writes to a single FCD. As a result, FCD doesn't support multiple vCenter Server instances managing the same FCD. If you need to use multiple vCenter Server instances with FCDs, you have the following options:
 - Multiple vCenter Server instances can manage different datastores.
 - Multiple vCenter Server instances do not operate on the same FCD.

Storage Policy Based Management

Storage Policy Based Management is a vCenter Server service that supports provisioning of persistent volumes and their backing virtual disks according to storage requirements described in a storage policy. After provisioning, the service monitors compliance of the volume with the storage policy characteristics. For more information about Storage Policy Based Management, see [Storage Policy Based Management](#).

vSphere CNS-CSI

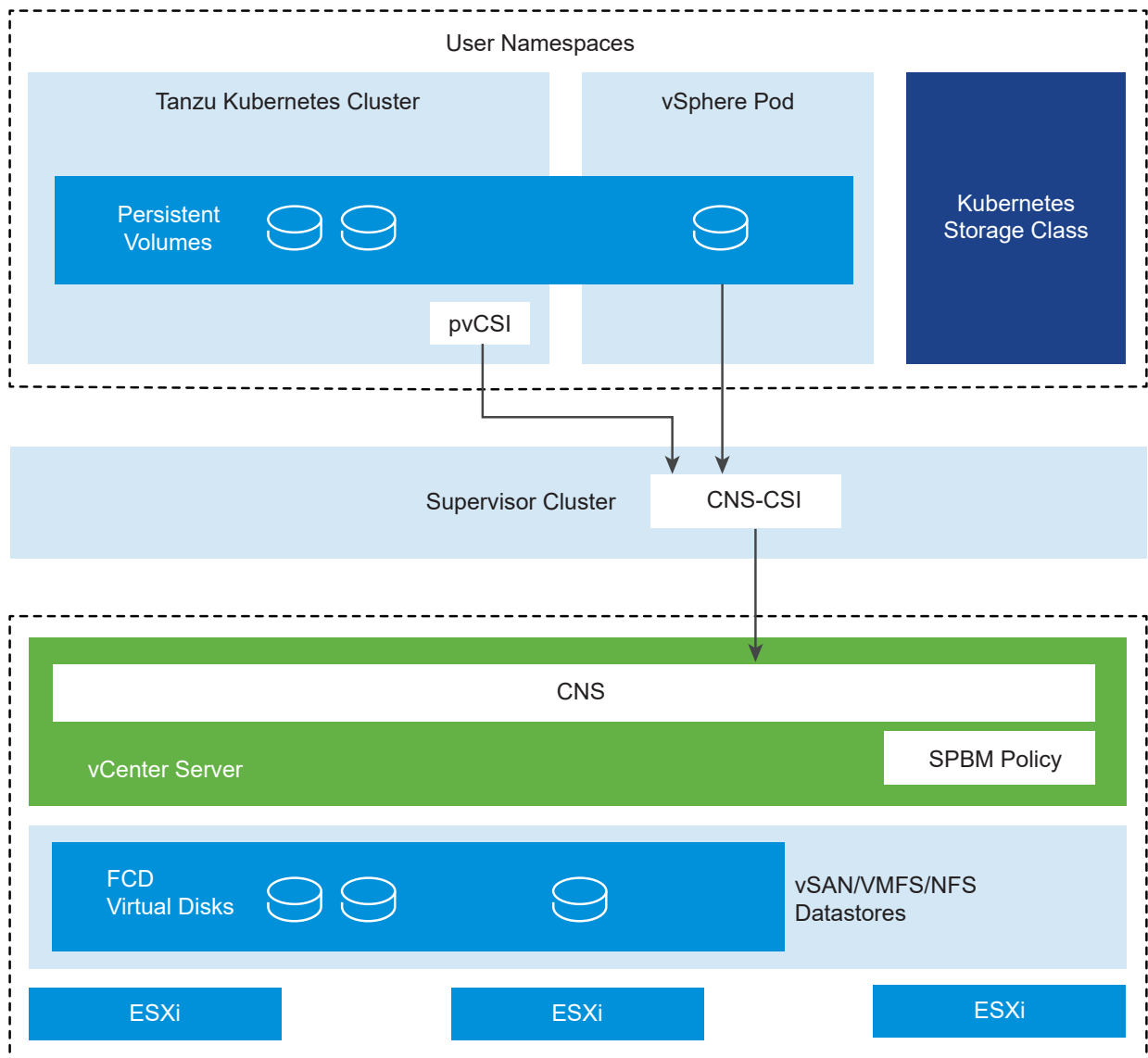
The vSphere CNS-CSI component conforms to Container Storage Interface (CSI) specification, an industry standard designed to provide an interface that container orchestrators like Kubernetes use to provision persistent storage. The CNS-CSI driver runs in the Supervisor Cluster and connects vSphere storage to Kubernetes environment on a vSphere Namespace. The vSphere CNS-CSI communicates directly with the CNS control plane for all storage

provisioning requests that come from vSphere Pods and pods that runs in a Tanzu Kubernetes cluster on the namespace.

Paravirtual CSI (pvCSI)

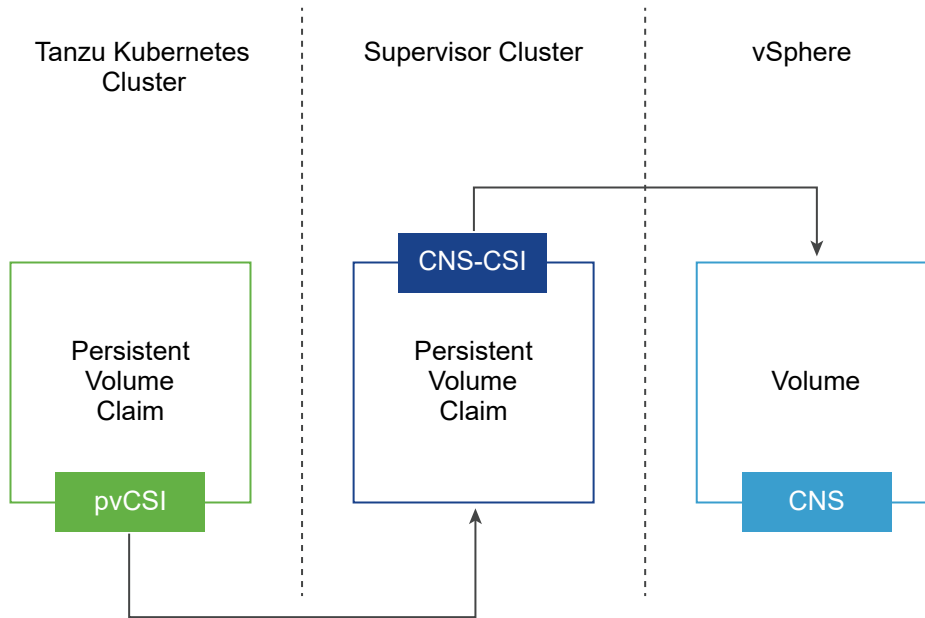
The pvCSI is the version of the vSphere CNS-CSI driver modified for Tanzu Kubernetes clusters. The pvCSI resides in the Tanzu Kubernetes cluster and is responsible for all storage related requests originating from the Tanzu Kubernetes cluster. The requests are delivered to the CNS-CSI, which then propagates them to CNS in vCenter Server. As a result, the pvCSI does not have direct communication with the CNS component, but instead relies on the CNS-CSI for any storage provisioning operations.

Unlike the CNS-CSI, the pvCSI does not require infrastructure credentials. It is configured with a service account in the vSphere Namespace.



The following illustrates how different components interact when a DevOps engineer performs a storage related operation within the Tanzu Kubernetes cluster, for example creates a persistent volume claim (PVC).

The DevOps engineer creates a PVC using the command line on the Tanzu Kubernetes cluster. This action generates a matching PVC on the supervisor cluster and triggers the CNS-CSI. The CNS-CSI invokes the CNS create volume API.



After successful creation of a volume, the operation propagates back through the supervisor cluster to the Tanzu Kubernetes cluster. As a result of this propagation, users can see the persistent volume and the persistent volume claim in the bound state in the supervisor cluster. And they also see the persistent volume and the persistent volume claim in the bound state in the Tanzu Kubernetes cluster.

Functionality Supported by vSphere CNS-CSI and Paravirtual CSI in vSphere with Tanzu

The vSphere CNS-CSI driver that runs in the Supervisor Cluster and pvCSI driver, the CSI version modified for Tanzu Kubernetes clusters, support multiple vSphere and Kubernetes storage features. However, certain limitations apply.

Supported Functionality	vSphere CNS-CSI with Supervisor Cluster	pvCSI with Tanzu Kubernetes Cluster
CNS Support in the vSphere Client	Yes	Yes
Enhanced Object Health in the vSphere Client	Yes (vSAN only)	Yes (vSAN only)
Dynamic Block Persistent Volume (ReadWriteOnce Access Mode)	Yes	Yes

Supported Functionality	vSphere CNS-CSI with Supervisor Cluster	pvCSI with Tanzu Kubernetes Cluster
Dynamic File Persistent Volume (ReadWriteMany Access Mode)	No	Yes (with vSAN File Services)
vSphere Datastore	VMFS/NFS/vSAN/vVols	VMFS/NFS/vSAN/vVols
Static Persistent Volume	Yes	Yes
Encryption	No	No
Offline Volume Expansion	Yes	Yes
Online Volume Expansion	Yes	Yes
Volume Topology and Zones	No	No
Kubernetes Multiple Control Plane Instances	Yes	Yes
WaitForFirstConsumer	No	No
VolumeHealth	Yes	Yes

Storage Permissions in vSphere with Tanzu

vSphere with Tanzu provides a sample role, Workload Storage Manager, that includes a set of privileges for storage operations. You can clone this role to create a similar role.

Privilege Name	Description	Required On
Cns.Searchable	Allows storage administrator to see the Cloud Native Storage UI.	Root vCenter Server
Datastore.Allocate space Datastore.Low level file operations	Allows allocating space on a datastore for a virtual machine, snapshot, clone, or virtual disk. Allows performing read, write, delete, and rename operations in the datastore browser.	Shared datastore where persistent volumes reside
ESX Agent Manager.Modify	Allows modifications to an agent virtual machine such as powering off or deleting the virtual machine.	vSphere Pod
Resource.Assign virtual machine to resource pool	Allows assignment of a virtual machine to a resource pool.	Resource pools

Privilege Name	Description	Required On
Profile-driven storage.Profile-driven storage view	Allows viewing of defined storage policies.	Root vCenter Server
Virtual machine.Change Configuration.Add existing disk Virtual machine.Change Configuration.Add new disk Virtual machine.Change Configuration.Add or remove device Virtual machine.Change Configuration.Change Settings Virtual machine.Change Configuration.Remove disk Virtual machine.Edit Inventory.Create new Virtual machine.Edit Inventory.Remove	Allows creation and deletion of virtual machines. Allows configuration of virtual machine options and devices.	vSphere Pod

Create Storage Policies for vSphere with Tanzu

Before you enable vSphere with Tanzu, create storage policies to be used in the Supervisor Cluster and namespaces. The policies represent datastores available in the vSphere environment. They control the storage placement of such objects as control plane VMs, pod ephemeral disks, container images, and persistent storage volumes. If you use Tanzu Kubernetes clusters, the storage policies also dictate how the Tanzu Kubernetes cluster nodes are deployed.

Depending on your vSphere storage environment and the needs of DevOps, you can create several storage policies to represent different classes of storage. For example, if your vSphere storage environment has three classes of datastores, Bronze, Silver, and Gold, you can create storage policies for all datastores. You can then use the Bronze datastore for ephemeral and container image virtual disks, and use the Silver and Gold datastores for persistent volume virtual disks. For more information about storage policies, see the [Storage Policy Based Management](#) chapter in the *vSphere Storage* documentation.

The following example creates the storage policy for the datastore tagged as Gold.

If you use vSAN Data Persistence platform you can create storage policies for vSAN Direct or vSAN SNA datastores. For information, see [Create vSAN Direct Storage Policy](#) and [Create vSAN SNA Storage Policy](#).

Prerequisites

- Make sure that the datastore you reference in the storage policy is shared between all ESXi hosts in the cluster.
- Required privileges: **VM storage policies. Update** and **VM storage policies. View**.

Procedure

1 Add tags to the datastore.

- a Right-click the datastore you want to tag and select **Tags and Custom Attributes > Assign Tag**.
- b Click **Add Tag** and specify the tag's properties.

Property	Description
Name	Specify the name of the datastore tag, for example, Gold .
Description	Add the description of the tag. For example, Datastore for Kubernetes objects .
Category	Select an existing category or create a new category. For example, Storage for Kubernetes .

2 In the vSphere Client, open the **Create VM Storage Policy** wizard.

- a Click **Menu > Policies and Profiles**.
- b Under **Policies and Profiles**, click **VM Storage Policies**.
- c Click **Create VM Storage Policy**.

3 Enter the policy name and description.

Option	Action
vCenter Server	Select the vCenter Server instance.
Name	Enter the name of the storage policy, for example, goldsp . Note When vSphere with Tanzu converts storage policies that you assign to namespaces into Kubernetes storage classes, it changes all upper case letters into lower case and replaces spaces with dashes (-). To avoid confusion, use lower case and no spaces in the VM storage policy names.
Description	Enter the description of the storage policy.

4 On the **Policy structure** page under **Datastore specific rules**, enable tag-based placement rules.

5 On the **Tag based placement** page, create the tag rules.

Select the options using the following example.

Option	Description
Tag category	From the drop-down menu, select the tag's category, such as Storage for Kubernetes .
Usage option	Select Use storage tagged with .
Tags	Click Browse Tags , and select the datastore tag, for example, Gold .

- 6 On the **Storage compatibility** page, review the list of datastores that match this policy.
In this example, only the datastore that is tagged as Gold is displayed.
- 7 On the **Review and finish** page, review the storage policy settings and click **Finish**.

Results

The new storage policy for the datastore tagged as Gold appears on the list of existing storage policies.

What to do next

After creating storage policies, a vSphere administrator can perform the following tasks:

- Assign the storage policies to the Supervisor Cluster. The storage policies configured on the Supervisor Cluster ensure that the control plane VMs, pod ephemeral disks, and container images are placed on the datastores that the policies represent. See [Enable Workload Management with NSX Networking](#).
- Assign the storage policies to the vSphere Namespace. Storage policies visible to the namespace determine which datastores the namespace can access and use for persistent volumes. The storage policies appear as matching Kubernetes storage classes in the namespace. They are also propagated to the Tanzu Kubernetes cluster on this namespace. DevOps engineers can use the storage classes in their persistent volume claim specifications. See [Create and Configure a vSphere Namespace](#).

Change Storage Settings on the Supervisor Cluster

Storage policies assigned to the Supervisor Cluster manage how such objects as a control plane VM, vSphere Pod, and container image cache are placed within datastores in the vSphere storage environment. As a vSphere administrator, you typically configure storage policies when enabling the Supervisor Cluster. If you need to make any changes in the storage policy assignments after initial Supervisor Cluster configuration, perform this task. You can also use this task to activate or deactivate file volume support for ReadWriteMany persistent volumes.

The changes you make to the storage settings apply only to new objects.

Prerequisites

If you plan to activate file volume support for persistent volumes in ReadWriteMany mode, follow prerequisites in [Creating ReadWriteMany Persistent Volumes in vSphere with Tanzu](#).

Procedure

- 1 In the vSphere Client, navigate to the host cluster that has vSphere with Tanzu enabled.
- 2 Click the **Configure** tab and click **Storage** under **Namespaces**.

- 3 Change storage policy assignments for the following items.

Option	Description
Control Plane Node	Select the storage policy for placement of the control plane VMs.
Pod Ephemeral Disks	Select the storage policy for placement of the vSphere Pods.
Container Image Cache	Select the storage policy for placement of the cache of container images.

- 4 Enable file volume support to deploy ReadWriteMany persistent volumes.

Change Storage Settings on a Namespace

Storage policies that a vSphere administrator assigns to a namespace in a Supervisor Cluster control how persistent volumes and Tanzu Kubernetes cluster nodes are placed within vSphere datastores. The persistent volume claims that correspond to persistent volumes can originate from a vSphere Pod or from a Tanzu Kubernetes cluster. You can change the original storage policy assignments.

Prerequisites

Before deleting a storage policy from VMware vCenter or a vSphere Namespace, or changing the storage policy assignment, make sure that no persistent volume claim with the corresponding storage class runs in the namespace. Also, ensure that no Tanzu Kubernetes cluster is using the storage class.

Procedure

- 1 In the vSphere Client, navigate to the namespace.
 - a From the vSphere Client home menu, select **Workload Management**.
 - b Click the **Namespaces** tab and click the namespace.
- 2 Click the **Storage** tab and click **Storage Policies**.
- 3 Click the **Edit** icon to change storage policy assignments.

Display Storage Classes in a vSphere Namespace or Tanzu Kubernetes Cluster

After a vSphere administrator creates a storage policy and assigns it to the vSphere Namespace, the storage policy appears as a matching Kubernetes storage class in the namespace and any available Tanzu Kubernetes clusters. As a DevOps engineer, you can verify that the storage class is available.

Your ability to run the commands depends on your permissions.

Prerequisites

Make sure that your vSphere administrator has created an appropriate storage policy and assigned the policy to the vSphere Namespace.

Procedure

- 1 Use one of the following commands to verify that the storage classes are available.

- **kubectl get storageclass**

Note This command is available only to a user with administrator privileges.

You get the output similar to the following. The name of the storage class matches the name of the storage policy on the vSphere side.

NAME	PROVISIONER	AGE
silver	csi.vsphere.vmware.com	2d
gold	csi.vsphere.vmware.com	1d

- **kubectl describe namespace *namespace_name***

In the output, the name of the storage class appears as a part of the ***storageclass_name.storage.k8s.io/requests.storage*** parameter. For example:

```

-----
Name:                                     namespace_name
Resource                                Used  Hard
-----
silver.storageclass.storage.k8s.io/requests.storage  1Gi
9223372036854775807
gold.storageclass.storage.k8s.io/requests.storage    0
9223372036854775807

```

- 2 To check the amount of storage space available on the namespace, run the following command.

- **kubectl describe resourcequotas -namespace *namespace***

You get the output similar to the following.

```

Name:          ns-my-namespace
Namespace:     ns-my-namespace
Resource       Used  Hard
-----
requests.storage  0    200Gi

```

Provision a Dynamic Persistent Volume for a Stateful Application

Stateful applications, for example databases, save data between sessions and require persistent storage to store the data. The retained data is called the application's state. You can later retrieve the data and use it in the next session. Kubernetes offers persistent volumes as objects capable of retaining their state and data.

In the vSphere environment, the persistent volume objects are backed by virtual disks that reside on datastores. Datastores are represented by storage policies. After the vSphere administrator creates a storage policy, for example **gold**, and assigns it to a namespace in a Supervisor Cluster, the storage policy appears as a matching Kubernetes storage class in the vSphere Namespace and any available Tanzu Kubernetes clusters.

As a DevOps engineer, you can use the storage class in your persistent volume claim specifications. You can then deploy an application that uses storage from the persistent volume claim. In this example, the persistent volume for the application is created dynamically.

Prerequisites

Make sure that your vSphere administrator has created an appropriate storage policy and assigned the policy to the namespace.

Procedure

- 1 Access your namespace in the vSphere Kubernetes environment.
- 2 Verify that the storage classes are available.

See [Display Storage Classes in a vSphere Namespace or Tanzu Kubernetes Cluster](#).

3 Create a persistent volume claim.

- a Create a YAML file that contains the persistent volume claim configuration.

In this example, the file references the **gold** storage class.

To provision a ReadWriteMany persistent volume, set `accessModes` to `ReadWriteMany`. See [Creating ReadWriteMany Persistent Volumes in vSphere with Tanzu](#).

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: gold
  resources:
    requests:
      storage: 3Gi
```

- b Apply the persistent volume claim to the Kubernetes cluster.

```
kubectl apply -f pvc_name.yaml
```

This command dynamically creates a Kubernetes persistent volume and a vSphere volume with a backing virtual disk that satisfies the claim's storage requirements.

- c Check the status of the persistent volume claim.

```
kubectl get pvc my-pvc
```

The output shows that the volume is bound to the persistent volume claim.

NAME	STATUS	VOLUME	CAPACITY	ACCESSMODES	STORAGECLASS	AGE
my-pvc	Bound	my-pvc	2Gi	RWO	gold	30s

- 4 Create a pod that mounts your persistent volume.
 - a Create a YAML file that includes the persistent volume.

The file contains these parameters.

```
...
volumes:
  - name: my-pvc
    persistentVolumeClaim:
      claimName: my-pvc
```

- b Deploy the pod from the YAML file.

```
kubectl create -f pv_pod_name.yaml
```

- c Verify that the pod has been created.

```
kubectl get pod
```

NAME	READY	STATUS	RESTARTS	AGE
pod_name	1/1	Ready	0	40s

Results

The pod that you configured uses persistent storage described in the persistent volume claim.

What to do next

To monitor health status of the persistent volume, see [Monitor Volume Health in a vSphere Namespace or Tanzu Kubernetes Cluster](#). To review and monitor the persistent volume in the vSphere Client, see [Monitor Persistent Volumes in the vSphere Client](#).

Provision a Static Persistent Volume in a Tanzu Kubernetes Cluster

You can statically create a block volume in a Tanzu Kubernetes cluster using an unused persistent volume claim (PVC) from the Supervisor Cluster.

The PVC must satisfy the following conditions:

- The PVC is present in the same namespace where the Tanzu Kubernetes cluster resides.
- The PVC is not attached to a vSphere Pod in the Supervisor Cluster or a pod in any Tanzu Kubernetes cluster.

Using static provisioning, you can also reuse in a new Tanzu Kubernetes cluster a PVC that is no longer needed by another Tanzu Kubernetes cluster. To do this, change the `Reclaim` policy of the persistent volume (PV) in the original Tanzu Kubernetes cluster to `Retain`, and then delete the corresponding PVC.

Follow these steps to statically create a PVC in a new Tanzu Kubernetes cluster using the information from the leftover underlying volume.

Procedure

- 1 Note the name of the original PVC in the Supervisor Cluster.

If you are reusing the PVC from an old Tanzu Kubernetes cluster, you can retrieve the PVC name from the `volumeHandle` of the old PV object in the Tanzu Kubernetes cluster.

- 2 Create a PV.

In the YAML file, specify the values of the following items:

- For `storageClassName`, you can enter the storage class name that is used by your PVC in the Supervisor Cluster.
- For `volumeHandle`, enter the PVC name that you obtained in [Step 1](#).

If you are reusing a volume from another Tanzu Kubernetes cluster, delete the PVC and PV objects from the old Tanzu Kubernetes cluster before creating a PV in the new Tanzu Kubernetes cluster.

Use the following YAML manifest as an example.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: static-tkg-block-pv
  annotations:
    pv.kubernetes.io/provisioned-by: csi.vsphere.vmware.com
spec:
  storageClassName: gc-storage-profile
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Delete
  claimRef:
    namespace: default
    name: static-tkg-block-pvc
  csi:
    driver: "csi.vsphere.vmware.com"
    volumeAttributes:
      type: "vSphere CNS Block Volume"
      volumeHandle: "supervisor-block-pvc-name" # Enter the PVC name from the Supervisor
cluster.
```

- 3 Create a PVC to match the PV object you created in [Step 2](#).

Set the `storageClassName` to the same value as in the PV.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: static-tkg-block-pvc
```



```
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
  storageClassName: gc-storage-profile
  volumeName: static-tkg-block-pv
```

4 Verify that the PVC is bound to the PV that you created.

```
$ kubectl get pv,pvc
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY
STATUS CLAIM	STORAGECLASS	REASON	AGE
persistentvolume/static-tkg-block-pv	2Gi	RWO	Delete
Bound default/static-tkg-block-pvc	gc-storage-profile		10s

NAME	STATUS	VOLUME	CAPACITY
ACCESS MODES STORAGECLASS AGE			
persistentvolumeclaim/static-tkg-block-pvc	Bound	static-tkg-block-pv	2Gi
RWO gc-storage-profile 10s			

Creating ReadWriteMany Persistent Volumes in vSphere with Tanzu

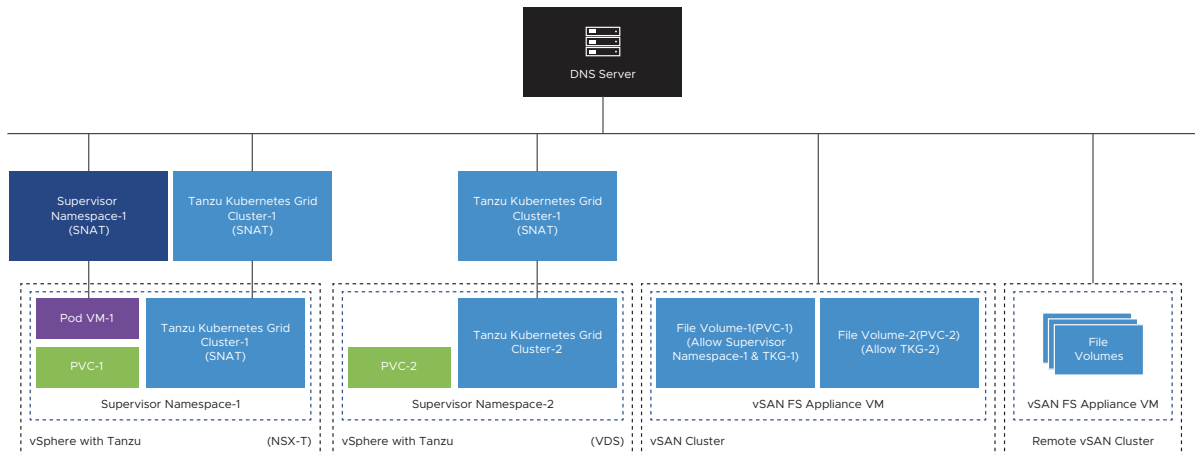
Starting with the vSphere 7.0 Update 3 release, vSphere with Tanzu supports persistent volumes in ReadWriteMany mode. With the ReadWriteMany support, a single volume can be mounted simultaneously by multiple pods or applications running in a cluster. vSphere with Tanzu uses vSAN File Services to provide file shares for the ReadWriteMany persistent volumes.

Considerations for ReadWriteMany Persistent Volumes

When you enable ReadWriteMany support for persistent volumes in vSphere with Tanzu, keep in mind the following considerations.

- With Tanzu Kubernetes clusters, use TKr version 1.22 or later. For information about the TKr versions, see [VMware Tanzu Kubernetes releases Release Notes](#).
- When you enable file volume support for vSphere with Tanzu, be aware of the potential security weaknesses:
 - The volumes are mounted without encryption. The unencrypted data might be accessed while the data transits the network.
 - Access Control Lists (ACLs) is used for the file shares to isolate file share access within a supervisor namespace. It might have risk of IP spoofing.
- Follow these guidelines for networking:
 - Make sure that the vSphere Namespace is in NAT mode. See [Create and Configure a vSphere Namespace](#).

- Make sure the vSAN File Services is routable from the Workload network and there is no NAT between the Workload network and vSAN File Services IP addresses.
- Use common DNS server for vSAN File Services and the vSphere cluster.



- If after enabling file volume support, you later deactivate it, existing ReadWriteMany persistent volumes that you provisioned in the cluster remain unaffected and usable. You will not be able to create new ReadWriteMany persistent volumes.

Workflow for Enabling ReadWriteMany Support for Persistent Volumes

Follow this process to enable ReadWriteMany support for persistent volumes.

- 1 A vSphere administrator sets up a vSAN cluster with configured vSAN File Services. See [Configure File Services](#).
- 2 A vSphere administrator activates file volume support on the Supervisor Cluster.

Action	Description
Activate file volume support when enabling the Workload Management platform.	■ Enable Workload Management with vSphere Networking ■ Enable Workload Management with NSX Networking
Activate file volume support on the existing cluster, for example, after an upgrade of vSphere with Tanzu.	Change Storage Settings on the Supervisor Cluster

- 3 A DevOps engineer provisions a persistent volume setting the PVC `accessMode` as `ReadWriteMany`.

Several pods can be provisioned with the same PVC.

See [Provision a Dynamic Persistent Volume for a Stateful Application](#).

Volume Expansion in vSphere with Tanzu

As a DevOps engineer, you can use the Kubernetes volume expansion feature to expand a persistent block volume after its creation. Both types of clusters, Supervisor Clusters and Tanzu Kubernetes clusters, support offline and online volume expansion.

Storage classes that appear in the vSphere with Tanzu environment have `allowVolumeExpansion` set to `true` by default. This parameter makes it possible to modify the size of an offline or online volume.

A volume is considered to be offline when it is not attached to a node or pod. An online volume is a volume that is available on a node or pod.

The level of support of the volume expansion functionality depends on the vSphere version. You can expand volumes created in the earlier versions of vSphere when you upgrade your vSphere environment to appropriate versions that support expansions.

If you use a Tanzu Kubernetes cluster, make sure to upgrade both the Tanzu Kubernetes cluster and the Supervisor Cluster to the appropriate version for the functionality to be supported. The functionality in the Tanzu Kubernetes cluster depends on the enablement of that feature in the Supervisor Cluster.

For example, if you upgrade the Tanzu Kubernetes cluster to vSphere 7.0 Update 2 and leave the Supervisor Cluster at 7.0 Update 1, the online volume expansion will not work in the Tanzu Kubernetes cluster.

	Supervisor Cluster 7.0	Supervisor Cluster 7.0 Update 1	Supervisor Cluster 7.0 Update 2
Tanzu Kubernetes Cluster 7.0	Offline and online expansions in a Tanzu Kubernetes cluster or Supervisor Cluster: not supported	Offline and online expansions in a Tanzu Kubernetes cluster or Supervisor Cluster: not supported	■ Offline and online expansions in a Tanzu Kubernetes cluster: not supported ■ Offline and online expansions in a Supervisor Cluster: supported

Tanzu Kubernetes Cluster 7.0 Update 1	Offline and online expansions in a Tanzu Kubernetes cluster or Supervisor Cluster: not supported	■ Offline expansion in a Tanzu Kubernetes cluster: supported ■ Offline expansion in a Supervisor Cluster: not supported ■ Online expansion in a Tanzu Kubernetes cluster or Supervisor Cluster: not supported	■ Offline expansion in a Tanzu Kubernetes cluster: supported ■ Online expansion in a Tanzu Kubernetes cluster: not supported ■ Offline and online expansions in a Supervisor Cluster: supported
Tanzu KubernetesCluster 7.0 Update 2	Offline and online expansions in a Tanzu Kubernetes cluster or Supervisor Cluster: not supported	■ Offline expansion in a Tanzu Kubernetes cluster: supported ■ Offline expansion in a Supervisor Cluster: not supported ■ Online expansion in a Tanzu Kubernetes cluster or Supervisor Cluster: not supported	Offline and online expansions in a Tanzu Kubernetes cluster or Supervisor Cluster: supported

When you expand the volumes, keep in mind the following:

- You can expand the volumes up to the limits specified by storage quotas. vSphere with Tanzu supports consecutive resize requests for a persistent volume claim object.
- All types of datastores, including VMFS, vSAN, vSAN Direct, vVols, and NFS, support volume expansion.
- You can perform volume expansion for deployments or standalone pods.
- You can resize statically provisioned volumes in a Supervisor Cluster and Tanzu Kubernetes cluster if the volumes have storage classes associated with them.
- You cannot expand volumes that are created as a part of a StatefulSet when you use the StatefulSet definition. Currently, Kubernetes does not support this feature. As a result, your attempts to expand the volumes by increasing the storage size in the StatefulSet definition fail.
- If a virtual disk that backs a volume has snapshots, it cannot be resized.
- vSphere with Tanzu does not support volume expansion for in-tree or migrated volumes.

Expand a Persistent Volume in Offline Mode

A volume is considered to be offline when it is not attached to a node or pod. Both types of clusters, Supervisor Clusters and Tanzu Kubernetes clusters, support offline volume expansion.

Prerequisites

Make sure to upgrade your vSphere environment to an appropriate version that supports offline volume expansion. See [Volume Expansion in vSphere with Tanzu](#).

Procedure

- 1 Create a persistent volume claim (PVC) with a storage class.

- a Define a PVC using the following YAML manifest as an example.

In the example, the size of the requested storage is 1 Gi.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: example-block-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: example-block-sc
```

- b Apply the PVC to the Kubernetes cluster.

```
kubectl apply -f example-block-pvc.yaml
```

- 2 Patch the PVC to increase its size.

If the PVC is not attached to a node or being used by a pod, use the following command to patch the PVC. In this example, the requested storage increase is 2 Gi.

```
kubectl patch pvc example-block-pvc -p '{"spec": {"resources": {"requests": {"storage": "2Gi"}}}}'
```

This action triggers an expansion in the volume associated with the PVC.

3 Verify that the size of the volume has increased.

```
kubectl get pv
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS
CLAIM	STORAGECLASS	REASON	AGE	
pvc-9e9a325d-ee1c-11e9-a223-005056ad1fc1	2Gi	RWO	Delete	Bound
default/example-block-pvc	example-block-sc	6m44s		

Note The size of the PVC remains unchanged until the PVC is used by a pod.

The following example shows that the PVC size hasn't changed. If you describe the PVC, you can see the `FilesystemResizePending` condition applied on the PVC.

```
kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS
MODES	STORAGECLASS	AGE		
example-block-pvc	Bound	pvc-9e9a325d-ee1c-11e9-a223-005056ad1fc1	1Gi	
RWO	example-block-sc	6m57s		

4 Create a pod to use the PVC.

When the PVC is used by the pod, the filesystem is expanded.

5 Verify that the size of the PVC has been modified.

```
kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS	MODES
STORAGECLASS	AGE				
example-block-pvc	Bound	pvc-24114458-9753-428e-9c90-9f568cb25788	2Gi		RWO
example-block-sc	2m12s				

The `FilesystemResizePending` condition has been removed from the PVC. Volume expansion is complete.

What to do next

A vSphere administrator can see the new volume size in the vSphere Client. See [Monitor Persistent Volumes in the vSphere Client](#).

Expand a Persistent Volume in Online Mode

An online volume is a volume that is available on a node or pod. As a DevOps engineer, you can expand an online persistent block volume. Both types of clusters, Supervisor Clusters and Tanzu Kubernetes clusters, support online volume expansion.

Prerequisites

Make sure to upgrade your vSphere environment to an appropriate version that supports online volume expansion. See [Volume Expansion in vSphere with Tanzu](#).

Procedure

1 Find the persistent volume claim to resize.

```
$ kubectl get pv,pvc,pod
```

NAME	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	CAPACITY	REASON	ACCESS MODES	AGE
persistentvolume/pvc-5cd51b05-245a-4610-8af4-f07e77fdc984	Delete	Bound	default/block-pvc	block-sc	1Gi		RWO	4m56s

NAME	CAPACITY	ACCESS MODES	STORAGECLASS	STATUS	VOLUME	AGE
persistentvolumeclaim/block-pvc	1Gi	RWO	block-sc	Bound	pvc-5cd51b05-245a-4610-8af4-f07e77fdc984	5m3s

NAME	READY	STATUS	RESTARTS	AGE
pod/block-pod	1/1	Running	0	26s

Note that the size of storage the volume uses is 1 Gi.

2 Patch the PVC to increase its size.

For example, increase the size to 2 Gi.

```
$ kubectl patch pvc block-pvc -p '{"spec": {"resources": {"requests": {"storage": "2Gi"}}}}'
persistentvolumeclaim/block-pvc edited
```

This action triggers an expansion in the volume associated with the PVC.

3 Verify that the size of both PVC and PV has increased.

```
$ kubectl get pvc,pv,pod
```

NAME	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	CAPACITY	REASON	ACCESS MODES	AGE
persistentvolume/pvc-5cd51b05-245a-4610-8af4-f07e77fdc984	Delete	Bound	default/block-pvc	block-sc	2Gi		RWO	6m11s

NAME	CAPACITY	ACCESS MODES	STORAGECLASS	STATUS	VOLUME	AGE
persistentvolumeclaim/block-pvc	2Gi	RWO	block-sc	Bound	pvc-5cd51b05-245a-4610-8af4-f07e77fdc984	6m18s

NAME	READY	STATUS	RESTARTS	AGE
pod/block-pod	1/1	Running	0	101s

What to do next

A vSphere administrator can see the new volume size in the vSphere Client. See [Monitor Persistent Volumes in the vSphere Client](#).

Monitor Persistent Volumes in the vSphere Client

When DevOps engineers deploy a stateful application with a persistent volume claim, vSphere with Tanzu creates a persistent volume object and a matching persistent virtual disk. As a vSphere administrator, you can review details of the persistent volume in the vSphere Client. You can also monitor its storage compliance and health status.

Procedure

- 1 In the vSphere Client, navigate to the namespace that has persistent volumes.

- a From the vSphere Client home menu, select **Workload Management**.
- b Click the namespace.

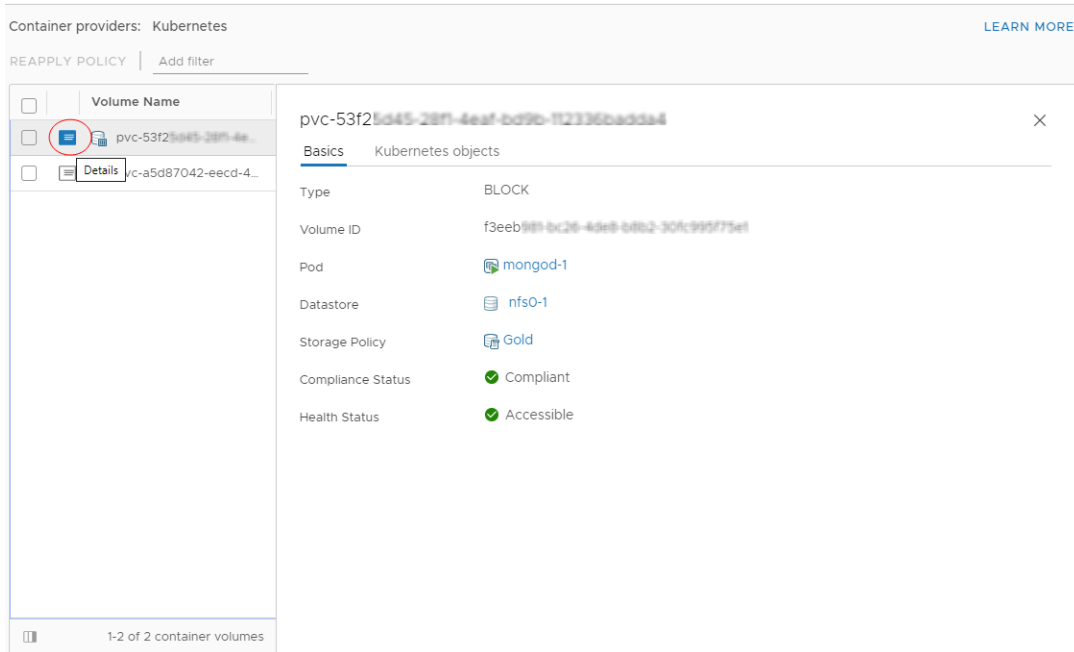
- 2 Click the **Storage** tab and click **Persistent Volume Claims**.

The vSphere Client lists all persistent volume claim objects and corresponding volumes available in the namespace.

- 3 To view details of a selected persistent volume claim, click the name of the volume in the **Persistent Volume Name** column.

- 4 On the **Container Volumes** page, check the volume's health status and storage policy compliance.
- a Click the **Details** icon, and switch between the **Basics** and **Kubernetes objects** tabs to view additional information for the Kubernetes persistent volume.

To monitor volume health status using the `kubectl` command, see [Monitor Volume Health in a vSphere Namespace or Tanzu Kubernetes Cluster](#).



- b Verify the health status of the volume.

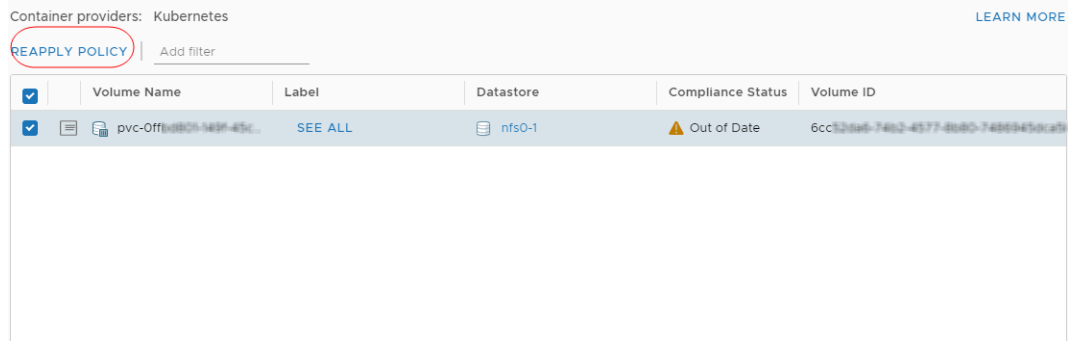
Health Status	Description
Accessible	The persistent volume is accessible and available for use.
Inaccessible	The persistent volume is inaccessible and cannot be used. The persistent volume becomes inaccessible if the datastore that stores the volume cannot be reached by the hosts that connect to the datastore.

- c Verify the storage compliance status.

You can see one of the following in the **Compliance Status** column.

Compliance Status	Description
Compliant	The datastore where the virtual disk backing the volume resides has the storage capabilities that the policy requires.
Out of Date	This status indicates that the policy has been edited, but the new storage requirements have not been communicated to the datastore. To communicate the changes, reapply the policy to the volume that is out of date.
Noncompliant	The datastore supports specified storage requirements, but cannot currently satisfy the storage policy. For example, the status might become Noncompliant when physical resources of the datastore are unavailable. You can bring the datastore into compliance by making changes in the physical configuration of your host cluster, for example by adding hosts or disks to the cluster. If additional resources satisfy the storage policy, the status changes to Compliant.
Not Applicable	The storage policy references datastore capabilities that are not supported by the datastore.

- d If the compliance status is Out of Date, select the volume and click **Reapply Policy**.



The status changes to Compliant.

Monitor Volume Health in a vSphere Namespace or Tanzu Kubernetes Cluster

As a DevOps engineer, you can check health status of a persistent volume in a bound state.

For each persistent volume in a bound state, the health status appears in the `Annotations: volumehealth.storage.kubernetes.io/messages:` field of the persistent volume claim bound to the persistent volume. Two possible values for health status exist.

Health Status	Description
Accessible	The persistent volume is accessible and available for use.
Inaccessible	The persistent volume is inaccessible and cannot be used. The persistent volume becomes inaccessible if the datastore that stores the volume cannot be reached by the hosts that connect to the datastore.

To monitor volume health status in the vSphere Client, see [Monitor Persistent Volumes in the vSphere Client](#).

Procedure

- 1 Access your namespace in the vSphere Kubernetes environment.
- 2 Create a persistent volume claim.
 - a Create a YAML file that contains the persistent volume claim configuration.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: gold
  resources:
    requests:
      storage: 2Gi
```

- b Apply the persistent volume claim to the Kubernetes cluster.

```
kubectl apply -f pvc_name.yaml
```

This command creates a Kubernetes persistent volume and a vSphere volume with a backing virtual disk that satisfies the claim's storage requirements.

- c Check whether the persistent volume claim is bound to a volume.

```
kubectl get pvc my-pvc
```

The output shows that the persistent volume claim and the volume are in the bound state.

NAME	STATUS	VOLUME	CAPACITY	ACCESSMODES	STORAGECLASS	AGE
my-pvc	Bound	my-pvc	2Gi	RWO	gold	30s

- 3 Check the health status of the volume.

Run the following command to check the volume health annotation of the persistent volume claim bound to the persistent volume.

```
kubectl describe pvc my-pvc
```

In the following sample output, the `volumehealth.storage.kubernetes.io/messages` field shows the health status as accessible.

```
Name:          my-pvc
Namespace:     test-ns
StorageClass:  gold
Status:        Bound
```

```

Volume:      my-pvc
Labels:      <none>
Annotations:  pv.kubernetes.io/bind-completed: yes
              pv.kubernetes.io/bound-by-controller: yes
              volume.beta.kubernetes.io/storage-provisioner: csi.vsphere.vmware.com
              volumehealth.storage.kubernetes.io/messages: accessible
Finalizers:   [kubernetes.io/pvc-protection]
Capacity:    2Gi
Access Modes: RWO
VolumeMode:   Filesystem

```

Using vSAN Data Persistence Platform with Modern Stateful Services

You can use the vSAN Data Persistence platform for modern stateful services that require persistent storage. The platform provides a framework that enables third parties to integrate their service applications with underlying vSphere infrastructure, so that third-party software can run on vSphere with Tanzu optimally.

The benefits of using vSAN Data Persistence include the following:

Automatic Service Deployment and Scaling

Using the vSphere Client, administrators can install and deploy a modern stateful service on a Supervisor Cluster and grant access to the service namespace to DevOps engineers. The DevOps engineers can provision and scale instances of the stateful service dynamically in a self-service manner through Kubernetes APIs.

Service Monitoring Integrated with vCenter Server

Partners can build dashboard plugins that integrate with vCenter Server. Using the UI plugins, the vSphere administrators can manage and monitor the stateful services. In addition, vSAN offers health and capacity monitoring capabilities for these integrated third-party services.

Optimized Storage Configuration with vSAN Direct

vSAN Direct enables modern stateful services to interface directly with the underlying direct-attached storage for optimized I/O and storage efficiency.

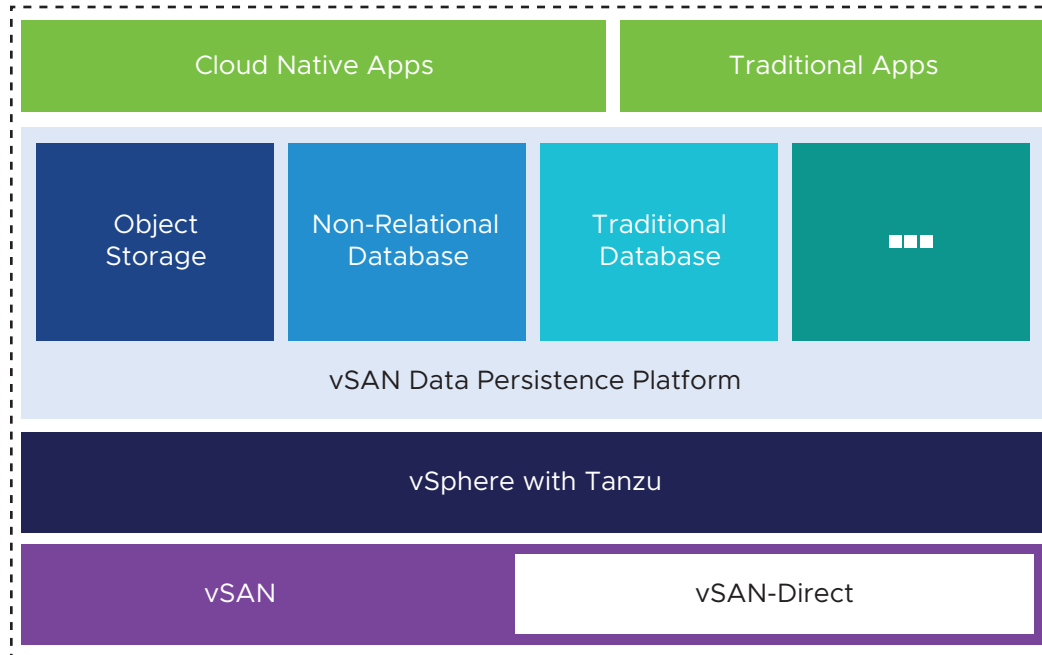
The platform supports the following types of services:

- Object storage, such as MinIO.
- NoSQL databases, also called non-relational databases.
- Traditional databases.

vSphere Shared Nothing Storage

Most modern stateful services have a Shared Nothing Architecture (SNA). They consume non-replicated local storage and offer their own storage replication, compression, and other data operations. As a result, the services do not benefit when the same operations are performed by the underlying storage.

To avoid duplicating the operations, the vSAN Data Persistence platform offers two vSAN solutions with optimized data paths. The persistent service can run either on vSAN with the SNA storage policy or on a mostly raw local storage called vSAN Direct.



vSAN with SNA Storage Policy

With this technology, you can use a distributed replicated vSAN datastore with the vSAN host-local SNA policy. As a result, the SNA service application can control placement and take over the duty of maintaining data availability. The technology makes it easy for the persistent service to co-locate its compute instance and a storage object on the same physical ESXi host. With the host-local placement, it is possible to perform such operations as replication at the service layer and not at the storage layer.

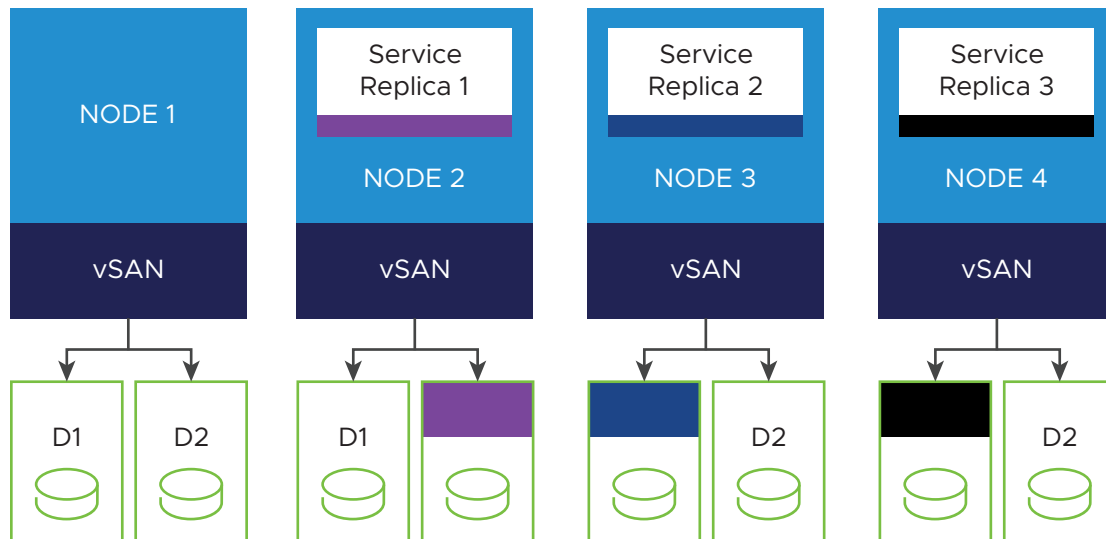
The compute instance, such as a pod, comes up first on one of the nodes in the vSAN cluster. And then the vSAN object created with the vSAN SNA policy automatically has all its data placed on the same node where the pod is running.

The following example illustrates storage deployment of an application that uses the SNA storage class for its persistent volume. vSAN can select any disk group on the node for the persistent volume placement.

Total Copies of Data = 3

Expected Fault Tolerance = 2

Actual Failures Guaranteed to Be Tolerated = 2

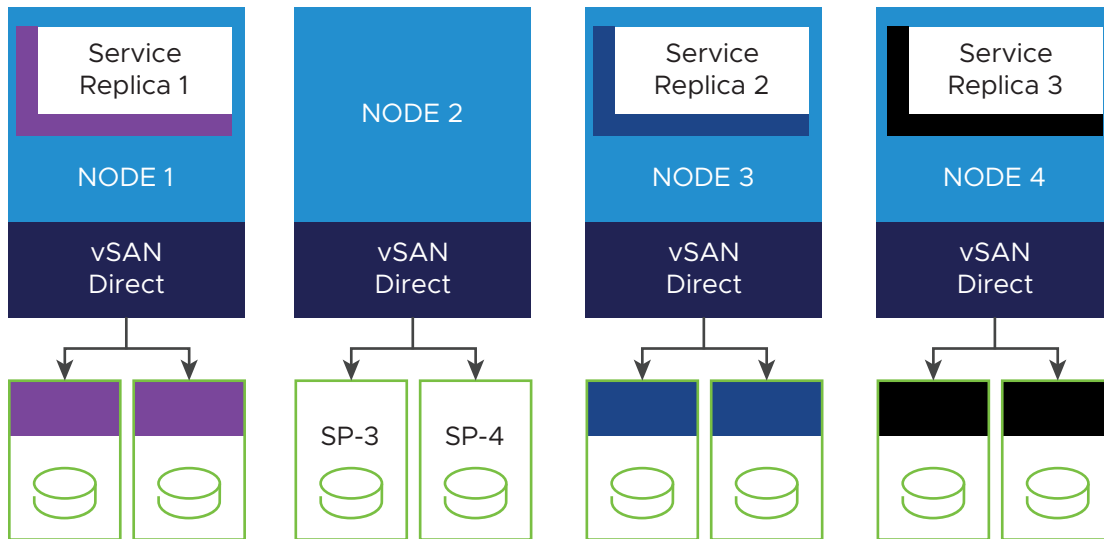


vSAN Direct

Even though vSAN with the SNA storage policy can place data locally to the compute instance, an overhead exists of a distributed vSAN data path between the application and the physical storage device. With vSAN Direct, the stateful services applications can access mostly raw non-vSAN local storage through a more direct data path, which offers the most performance optimized solution.

With vSAN Direct, the vSphere administrator can claim host-local devices, and then manage and monitor the devices. vSAN Direct provides insights into the device health, performance, and capacity. On every local device it claims, vSAN Direct creates an independent VMFS datastore and makes it available as a placement choice to the application. The VMFS datastores that vSAN Direct manages are exposed as storage pools in Kubernetes. In the vSphere Client, they appear as vSAN Direct datastores.

The following illustrates persistent volumes placed locally on vSAN Direct disks.



When to Use vSAN with SNA or vSAN Direct

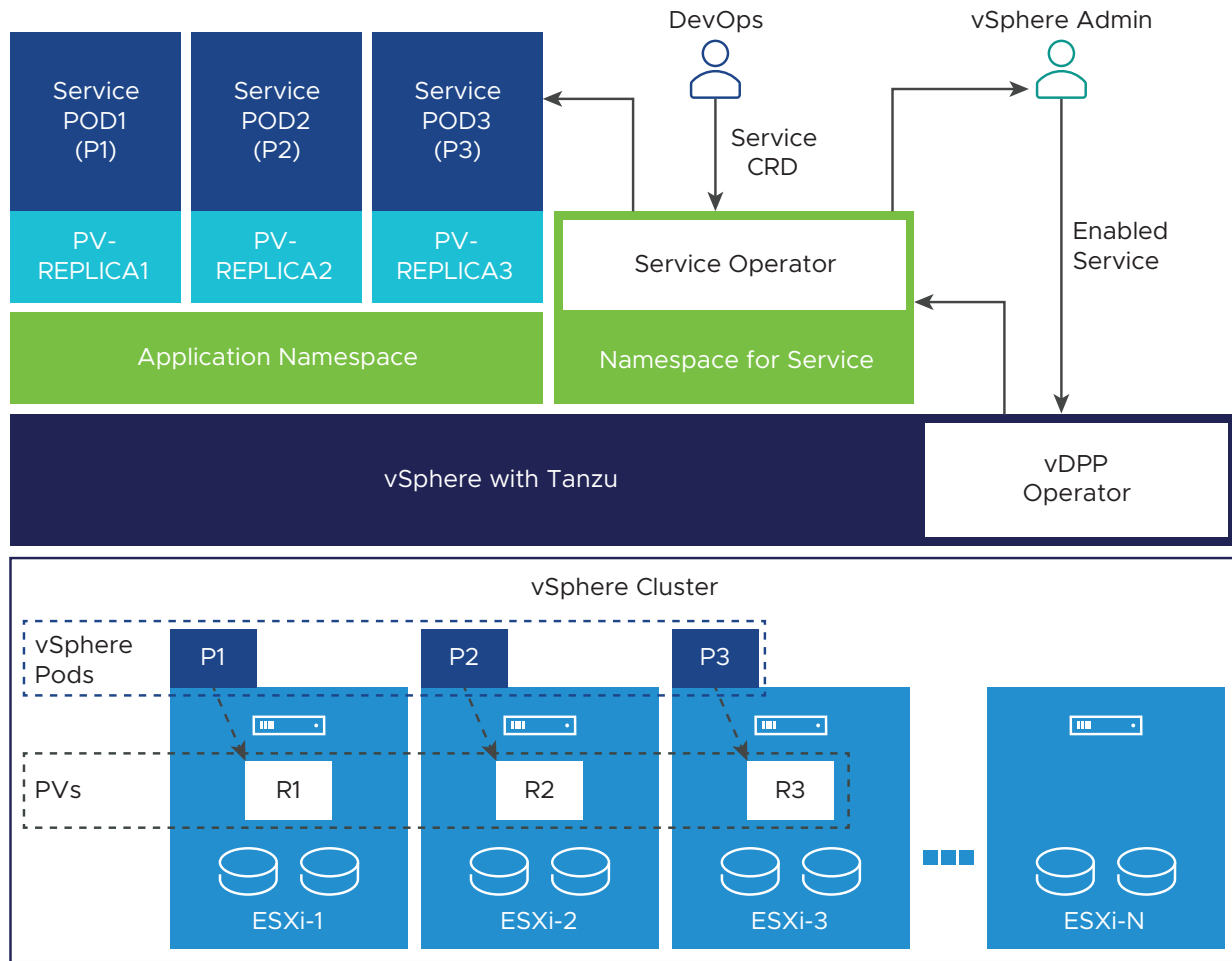
Follow these general recommendations when deciding which type of vSAN to use.

- Use vSAN with SNA when you want the cloud-native stateful application to share the physical infrastructure with other regular VMs or Kubernetes workloads. Each workload can define its own storage policy and can get the best of both worlds from a single cluster.
- Use vSAN Direct if you are creating a dedicated hardware cluster for the shared nothing cloud-native services.

vSAN Data Persistence Platform Operator

The vSAN Data Persistence platform (vDPP) operator is a component responsible for running and managing partner stateful services integrated with vSphere. The vDPP operator exposes available stateful services to the vSphere administrator. When the vSphere administrator enables a persistent service, for example, MinIO, the vDPP operator deploys an application-specific operator for the service on the supervisor cluster.

The application-specific operators are provided by the third party and must be vDPP compliant. The operator typically offers a CRD which provides a self-service interface for Kubernetes users to instantiate instances. vSphere with Tanzu uses this operator and CRD to provision new service instances and manage and monitor them through the stateful services layer. Most of these operators use stateful sets for deploying their instances.



After the vSphere administrator enables a service, the following takes place.

- The vDPP operator activates a service-specific operator.
- The service-specific operator registers the UI plug-in.
- Storage-optimized storage policies are created.

Configuration Limits for vSAN Data Persistence Platform

VMware provides configuration limits in the [VMware Configuration Maximums](#) tool.

vSAN Data Persistence Maximums	Limits
Maximum number of persistent volumes per vSAN Data Persistence platform	1000
Maximum number of persistent volumes per service instance on the vSAN Data Persistence platform	60 to 80

Tag Storage Devices for vSAN Direct

In VMware Cloud Foundation deployments, vSAN automatically claims all local storage devices on your ESXi host. You can make the devices ineligible for regular vSAN and available for vSAN Direct.

This topic describes how you can use the `esxcli` command to mark the devices as vSAN Direct. Alternately, you can use a script. See [Using Script to Tag Storage Devices for vSAN Direct](#).

Procedure

- 1 Tag local storage device for vSAN Direct.

```
esxcli vsan storage tag add -d diskName -t vsanDirect
```

For example,

```
esxcli vsan storage tag add -d mpv.vmhba0:C0:T1:L0 -t vsanDirect
```

The device becomes ineligible for regular vSAN.

- 2 Remove the vSAN Direct tag from the device.

```
esxcli vsan storage tag remove -d diskName -t vsanDirect
```

For example,

```
esxcli vsan storage tag remove -d mpv.vmhba0:C0:T1:L0 -t vsanDirect
```

Using Script to Tag Storage Devices for vSAN Direct

In VMware Cloud Foundation deployments, you can use a script to tag HDD devices attached to your ESXi host. After you run the script, the devices become ineligible for regular vSAN and are available for vSAN Direct.

```
#!/usr/bin/env python3

# Copyright 2020 VMware, Inc. All rights reserved.

# Abstract
#
# This script helps manage tagging of Direct Attached HDD disks
# on ESXi systems for vSAN Direct in preparation for a VCF deployment.
#
# It is expected to be used with ESX systems of version 7.0.1 or later.
#

import argparse
from enum import Enum
import logging
import sys
import os
import paramiko
import subprocess
import traceback
import ast
```

```

import getpass
from six.moves import input
from distutils.util import strtobool
from argparse import ArgumentParser

class ParseState(Enum):
    OPEN = 0
    DEVICE = 1

class RemoteOperationError(Exception):
    pass

class EsxVersion:

    def __init__(self, major, minor, release):
        self.major = major
        self.minor = minor
        self.release = release

    def __str__(self):
        return '{}.{}.{}'.format(self.major, self.minor, self.release)

    @staticmethod
    def build(str):
        tokens = str.split(b'.',3)
        return EsxVersion(int(tokens[0]), int(tokens[1]), int(tokens[2]))

class StorageDevice:

    def __init__(self, deviceId, isSSD, isVsanDirectEnabled):
        self.deviceId = str(deviceId.decode())
        self.isSSD = isSSD
        self.isVsanDirectCapable = True
        self.isVsanDirectEnabled = isVsanDirectEnabled

    def __str__(self):
        return '{}:\n\tIs SSD: {}\n\tvsanDirect enabled:{}'.format(
            self.deviceId,
            self.isSSD,
            self.isVsanDirectEnabled)

    @staticmethod
    def strToBool(v):
        return bool(strtobool(str(v.decode())))

    @staticmethod
    def build(deviceId, props):
        vsanDirectEnabled = False
        isLocal = StorageDevice.strToBool(props[b'Is Local'])
        status = props[b'Status']
        isOffline = StorageDevice.strToBool(props[b'Is Offline'])
        isSSD = StorageDevice.strToBool(props[b'Is SSD'])
        isBootDevice = StorageDevice.strToBool(props[b'Is Boot Device'])
        deviceType = props[b'Device Type']
        if deviceType == b'Direct-Access' and isLocal and (not isOffline) and (not

```

```

isBootDevice) and status == b'on':
    return StorageDevice(deviceId, isSSD, vsanDirectEnabled)
else:
    print("Skipping device {}".format(deviceId))
    return None

def parse_arguments():
    """
    Parses the command line arguments to the function
    """
    parser = argparse.ArgumentParser()
    parser.add_argument('--hostname', dest='hostname',
                        help='specify hostname for the ESX Server', required=True)
    parser.add_argument('--username', dest='username',
                        help='specify username to connect to the ESX Server', required=True)
    parser.add_argument('--password', dest='password',
                        help='specify password to connect to the ESX Server', required=False)
    return parser.parse_args()

def get_esx_version(sshClient):
    global logger
    stdin_, stdout_, stderr_ = sshClient.exec_command('vmware -v')
    exit_status = stdout_.channel.recv_exit_status()
    if exit_status != 0:
        logger.error('Command exited with non-zero status: %s' % exit_status)
        logger.error('Error message: %s' % stderr_.read())
        raise RemoteOperationError('Failed to determine ESX version')
    output = stdout_.read()
    tokens = output.split()
    if len(tokens) < 3:
        raise RemoteOperationError('Invalid ESX Version - %s', output)
    return EsxVersion.build(tokens[2])

def check_esx_version(esxVersion):
    return esxVersion.major >= 7 and esxVersion.minor >= 0 and esxVersion.release >= 1

def query_devices(sshClient):
    global logger
    stdin_, stdout_, stderr_ = sshClient.exec_command('esxcli storage core device list')
    exit_status = stdout_.channel.recv_exit_status()
    if exit_status != 0:
        logger.error('Command exited with non-zero status: %s' % exit_status)
        logger.error('Error message: %s' % stderr_.read())
        raise RemoteOperationError('Failed to query core storage device list')
    output = stdout_.read()
    # Build the device list from the output
    return create_device_list(output)

def create_device_list(str):
    devices = []

    deviceId=""
    deviceProps={}

    parseState = ParseState.OPEN

```

```

for line in str.splitlines():
    if parseState == ParseState.OPEN:
        if line.strip():
            deviceId=line.strip()
            parseState = ParseState.DEVICE
    elif parseState == ParseState.DEVICE:
        if line.strip():
            props = line.strip().split(b':',1)
            deviceProps[props[0]] = props[1].strip()
        else:
            if deviceId:
                device = StorageDevice.build(deviceId, deviceProps)
                if device:
                    devices.append(device)
            else:
                logger.debug("Skipping device {}".format(deviceId))
            deviceId=""
            deviceProps={}
            parseState = ParseState.OPEN
    if deviceId:
        device = StorageDevice.build(deviceId, deviceProps)
        if device:
            devices.append(device)
return devices

def tag_device_for_vsan_direct(sshClient, deviceId):
    global logger
    logger.info("Tagging device [{}] for vSAN Direct".format(deviceId))
    command = "esxcli vsan storage tag add -d " + deviceId + " -t vsanDirect"
    stdin_, stdout_, stderr_ = sshClient.exec_command(command)
    exit_status = stdout_.channel.recv_exit_status()
    if exit_status != 0:
        logger.error('Command exited with non-zero status: %s' % exit_status)
        logger.error('Error message: %s' % stderr_.read())
        raise RemoteOperationError('Failed to tag device [{}] for vSAN
Direct'.format(deviceId))
    logger.info('Successfully tagged device [{}] for vSAN Direct'.format(deviceId))

def untag_device_for_vsan_direct(sshClient, deviceId):
    global logger
    logger.info("Untagging device [{}] for vSAN Direct".format(deviceId))
    command = "esxcli vsan storage tag remove -d " + deviceId + " -t vsanDirect"
    stdin_, stdout_, stderr_ = sshClient.exec_command(command)
    exit_status = stdout_.channel.recv_exit_status()
    if exit_status != 0:
        logger.error('Command exited with non-zero status: %s' % exit_status)
        logger.error('Error message: %s' % stderr_.read())
        raise RemoteOperationError('Failed to untag device [{}] for vSAN
Direct'.format(deviceId))
    logger.info('Successfully untagged device [{}] for vSAN Direct'.format(deviceId))

def get_vsan_info_for_device(sshClient, deviceId):
    global logger
    command = "vdq -q -d {}".format(deviceId)
    stdin_, stdout_, stderr_ = sshClient.exec_command(command)

```

```

    exit_status = stdout_.channel.recv_exit_status()
    if exit_status != 0:
        logger.error('Command exited with non-zero status: %s' % exit_status)
        logger.error('Error message: %s' % stderr_.read())
        raise RemoteOperationError('Failed to query vsan direct status on device [%s]' %
deviceId)
    output = stdout_.read()
    return ast.literal_eval(str(output.decode()))

def update_vsan_direct_status(sshClient, devices):
    for device in devices:
        vsanInfo = get_vsan_info_for_device(sshClient, device.deviceId)
        device.isVsanDirectEnabled = vsanInfo[0]['IsVsanDirectDisk'].strip() == "1"
        device.isVsanDirectCapable = vsanInfo[0]['State'].strip() == 'Eligible for use by
VSAN'

def getVsanDirectCapableDevices(devices):
    selectDevices = []
    # Cull devices incapable of vSAN Direct
    for device in devices:
        if device.isVsanDirectCapable:
            selectDevices.append(device)
    return selectDevices

def print_devices(devices):
    print("Direct-Attach Devices:")
    print("=====")
    iDevice = 0
    for device in devices:
        iDevice = iDevice + 1
        print ("{}. {}".format(iDevice, device))
    print("=====")

def tag_devices(sshClient, devices):
    for device in devices:
        tag_device_for_vsan_direct(sshClient, device.deviceId)

def untag_devices(sshClient, devices):
    for device in devices:
        untag_device_for_vsan_direct(sshClient, device.deviceId)

def tag_all_hdd_devices(sshClient, devices):
    hddDevices = []
    for device in devices:
        if not device.isSSD:
            hddDevices.append(device)
    if len(hddDevices) > 0:
        tag_devices(sshClient, hddDevices)

def show_usage():
    print ("=====")
    print ("commands: {tag-all-hdd, tag, untag}")
    print ("\tttag <comma separated serial numbers of devices>")
    print ("\tuntag <comma separated serial numbers of devices>")
    print ("\tttag-all-hdd")

```

```

print ("=====")

def main():
    global logger
    logger.info('Tag disks for vSAN Direct')

    try:
        # Parse arguments
        args = parse_arguments()

        # 1. Setup SSH connection to ESX system
        sshClient = paramiko.SSHClient()
        sshClient.load_system_host_keys()
        sshClient.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        passwd = args.password
        if passwd == None:
            passwd = getpass.getpass(prompt='Password: ')
        logger.info('Connecting to ESX System (IP: %s)' % args.hostname)
        sshClient.connect(args.hostname, username=args.username, password=passwd)
        # version check
        esxVersion = get_esx_version(sshClient)
        print('ESX Version on {} is {}'.format(args.hostname, esxVersion))
        logger.info('Checking ESX Version...')
        if not check_esx_version(esxVersion):
            raise Exception('ESX Version must be 7.0.1 or greater')

        print ('This script helps tag direct-attached disks for vSAN Direct on ESX')
        print ('Note: Only disks of type HDD are supported at this time.')
        print ()
        print ("For help, type help")
        show_usage()

    while True:
        # get device list
        print("Querying devices...")
        devices = query_devices(sshClient)
        # update devices with vSAN Direct status
        update_vsan_direct_status(sshClient, devices)
        # cull device list
        selectDevices = getVsanDirectCapableDevices(devices)
        # List the devices for the user to see
        print_devices(selectDevices)
        # find out what the user wants to do to these devices
        args = input('Command> ').split()
        if len(args) == 0:
            break
        cmd = args[0]
        if cmd == 'q' or cmd == 'quit' or cmd == 'exit':
            break
        elif cmd == 'help':
            show_usage()
        elif cmd == 'tag-all-hdd':
            print("Tagging all HDD devices...")
            tag_all_hdd_devices(sshClient, selectDevices)
        elif cmd == 'tag' or cmd == 'untag':

```

```

        chosenDevices = []
        if len(args) > 1:
            serials = args[1].split(',')
            for serialStr in serials:
                serial = int(serialStr)
                if serial < 1 or serial > len(selectDevices):
                    raise Exception("Error: Serial {} is out of range".format(serial))
                chosenDevices.append(selectDevices[serial-1])
        if len(chosenDevices) == 0:
            print("No devices specified")
            continue
        if cmd == 'tag':
            print("Tagging devices...")
            tag_devices(sshClient, chosenDevices)
        else:
            print("Untagging devices...")
            untag_devices(sshClient, chosenDevices)
    else:
        print ("Error: Unrecognized command - %s" % cmd)
except paramiko.ssh_exception.AuthenticationException as e:
    logger.error(e)
    sys.exit(5)
except Exception as e:
    logger.error('Disk tagging failed with error: %s' % e)
    logger.error(traceback.format_exc())
    sys.exit(1)
finally:
    # Close SSH client
    try:
        sshClient.close()
    except:
        pass

# Set up logging
logging.basicConfig()
logger = logging.getLogger('tag-disks-for-vsan-direct')

if __name__ == "__main__":
    main()

```

Set Up vSAN Direct for vSphere with Tanzu

As a vSphere administrator, set up vSAN Direct to be used with vSAN Data Persistence platform. Use unclaimed storage devices local to your ESXi host.

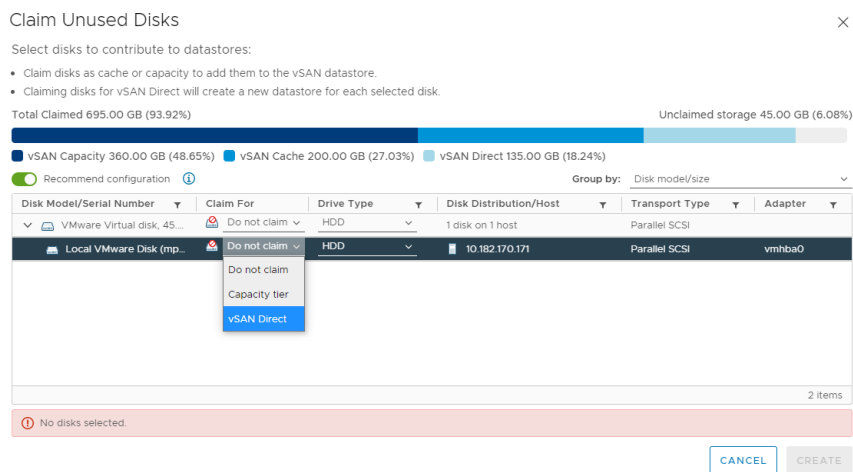
Prerequisites

If vSAN automatically claims all local storage devices in your deployment, use the `esxcli` command to tag the devices you want to be available for vSAN Direct. See [Tag Storage Devices for vSAN Direct](#). Alternately, you can use a script. For information, see [Using Script to Tag Storage Devices for vSAN Direct](#).

Procedure

- 1 In the vSphere Client, navigate to the vSAN cluster.
- 2 Click the **Configure** tab.
- 3 Under vSAN, click **Disk Management**.
- 4 Click **Claim Unused Disks**.
- 5 On the **Claim Unused Disks** dialog box, click the **vSAN Direct** tab.
- 6 Select a device to claim and select a check box in the **Claim for vSAN Direct** column.

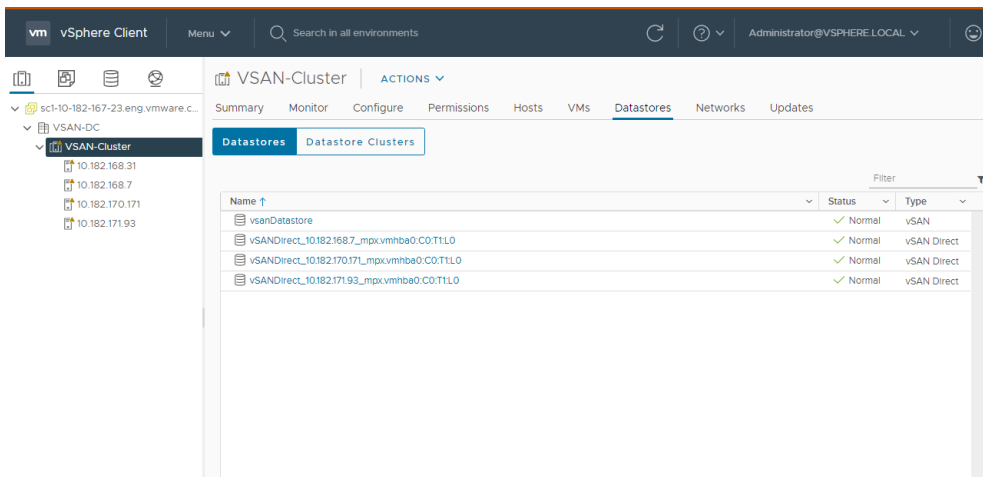
Note If you claimed the devices for a regular vSAN datastore, these devices do not appear in the **vSAN Direct** tab.



- 7 Click **Create**.

For each device you claim, vSAN Direct creates a new datastore.

- 8 Click the **Datastores** tab to display all vSAN Direct datastores in your cluster.



What to do next

You can use vSAN Direct with external storage. For more information, see [Use External Storage with vSAN Direct](#).

Enable Stateful Services in vSphere with Tanzu

vSphere with Tanzu integrates with several third-party services that use the vSAN Data Persistence platform for their persistent storage needs. As a vSphere administrator, enable the services on vCenter Server.

Starting with the vSphere with Tanzu 7.0 Update 3 release, you can download available third-party services from a VMware supported repository.

When you enable the stateful service, you first register the service with vCenter Server using the downloaded YAML file that describes the service. You then install the service on Supervisor Clusters so that your DevOps engineers can use the service in Kubernetes workloads.

Prerequisites

- Required privilege: **Supervisor Services.Manage Supervisor Services**
- Make sure that your Supervisor Cluster uses the NSX networking stack. vSAN Data Persistence platform does not support vSphere Distributed Switch (vDS) networking.

For information on setting up NSX-T, see [Configuring NSX for vSphere with Tanzu](#).

- Download a partner service YAML file from the repository maintained by VMware.

When you download the service YAML files, make sure to use correct service version compatible with your version of vSphere.

If you installed earlier versions of partner services, MinIO and Cloudian Hyperstore, upgrade them to compatible versions after upgrading vSphere to version 7.0 Update 3. The newer versions of partner operators fix certain issues and use new platform features. For more information, see the partner documentation.

Table 10-1. Compatibility Matrix for vSphere and Partner Services

vSphere Version	Partner Service	Service Version	Kubernetes Version
vSphere 7.0 Update 3	MinIO	2.0.0	1.19, 1.20, 1.21
	Cloudian	1.2.0	1.19, 1.20, 1.21

Use one of the following methods to download the YAML file:

- In the <https://vmwareesaas.jfrog.io/> repository, navigate to an appropriate partner folder in **Artifacts > vDPP-Partner-YAML** and select a YAML file to download.

The latest version of the partner YAML is located in the top level partner directory.

- Use the `wget` or `curl` commands to download the YAML files.

For example:

```
wget https://vmwaresaas.jfrog.io/artifactory/vDPP-Partner-YAML/Cloudian/Hyperstore/SupervisorService/hyperstore-supervisor-service.yaml
```

Procedure

1 Configure vSAN or vSAN Direct storage.

For information on setting up vSAN storage, see the *Administering VMware vSAN*. To set up vSAN Direct, see [Set Up vSAN Direct for vSphere with Tanzu](#).

vSAN Direct datastores appear in Kubernetes as StoragePools.

2 Add a stateful service to the vCenter Server system.

Use the partner service YAML file that you downloaded from the repository maintained by VMware.

See [Add a Supervisor Service to vCenter Server](#).

3 Install the service on Supervisor Clusters.

See [Install a Supervisor Service on Supervisor Clusters](#).

After you enable the service, the vSAN Data Persistence platform performs the following actions to create necessary resources for the service:

- Creates a namespace for this service in the supervisor cluster.
- Creates default storage policies and corresponding storage classes to be used with vSAN Shared-Nothing-Architecture (SNA) and vSAN Direct datastores.

Note The vSAN Data Persistence platform automatically creates vsan-direct and vsan-sna storage classes in the namespace after the vSphere administrator enables the stateful service. Only applications that run on the Supervisor Cluster can use the vsan-direct and vsan-sna storage classes. These storage classes cannot be used inside a Tanzu Kubernetes cluster.

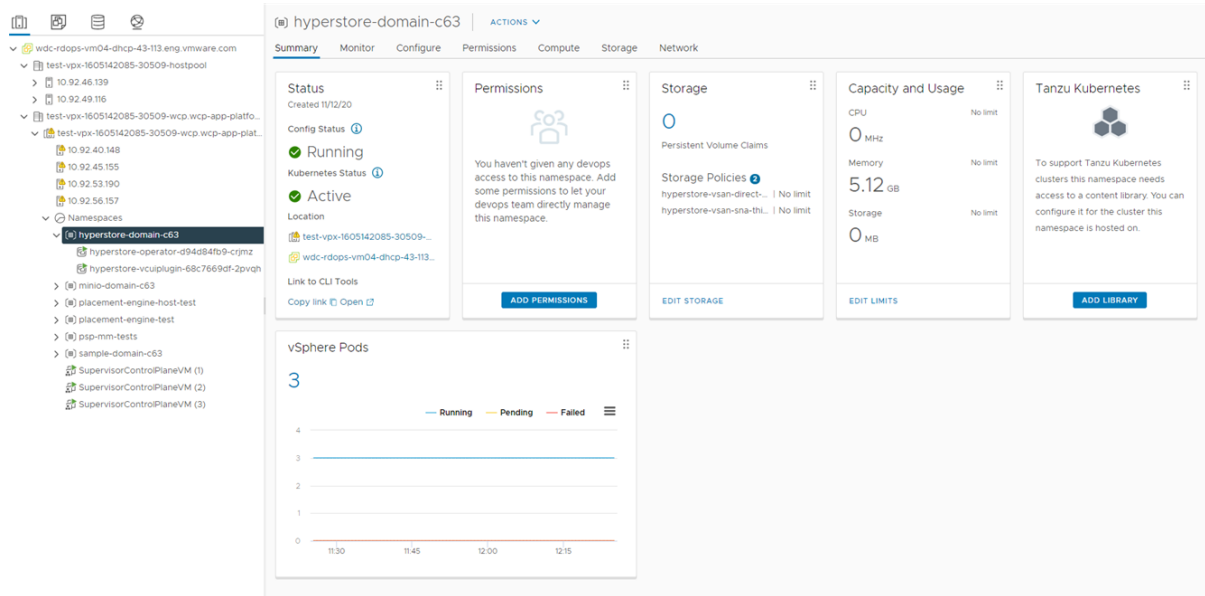
In vSphere 7.0 Update 2 and later, the vSAN Direct storage policy is capability-based. If you created tag-based policies in vSphere 7.0 Update 1, they are automatically converted to capability-based after an upgrade to vSphere 7.0 Update 2 and later.

If you want to create new storage policies and assign them to the service namespace instead of the default, see [Create vSAN Direct Storage Policy](#) and [Create vSAN SNA Storage Policy](#).

- Creates DevOps roles, including the roles with edit and view permissions.

When the service operator is deployed, its custom CRDs are installed in the Supervisor Cluster. Users with the edit permission can CRUD resources of these CRDs in the namespace. Users with the view permission can only view resources of this CRD.

- If the third party has provided a custom UI plugin, it appears in the vSphere Client. The vSphere administrator can use the plugin to manage the service.
- 4 Select the namespace created for the service and click the **Summary** tab to verify that all appropriate resources for the service have been created.



What to do next

- The DevOps engineer uses the `kubectl` command to access the service namespace and uses the third-party CRDs to deploy instances of the third-party application service. For more information, see the third-party documentation.

To verify that the namespace you use for stateful services has appropriate storage classes, see [Check Storage Policies Available for Stateful Services](#).

- If the third party has provided a custom UI plugin, the vSphere administrator can use the plugin to manage and monitor the service.

For more information, see the third-party UI plugin documentation. In addition, the vSphere administrator can use the Skyline Health checks to monitor the services. See [Monitor Stateful Services in vSphere with Tanzu](#)

Monitor Stateful Services in vSphere with Tanzu

After you enable integrated third-party stateful services, use vSAN health and capacity monitoring capabilities to view status and analyze space usage of the service objects.

Procedure

- 1 In the vSphere Client, navigate to the Supervisor Cluster.
- 2 Click the **Monitor** tab.

3 Monitor virtual objects that run in the namespace corresponding to the enabled service.

a Under **vSAN**, click **Virtual Objects**.

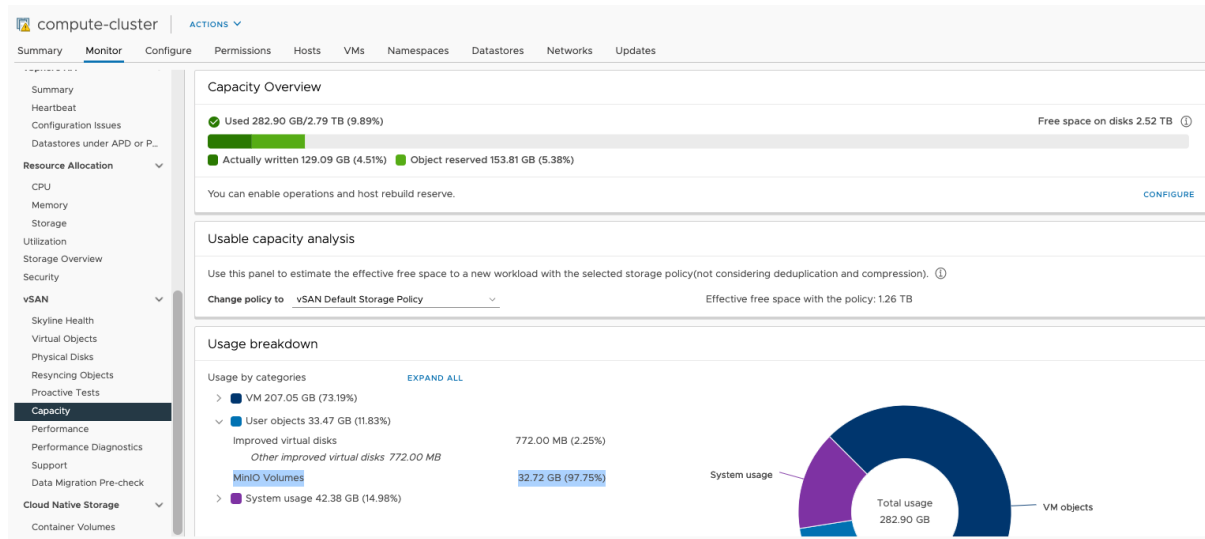
You can browse the virtual objects, for example, MinIO operator objects, and check their state.

b To view the object's placement across the physical infrastructure, select a specific object and click **VIEW PLACEMENT DETAILS**.

4 Monitor capacity that the service objects use.

a Under **vSAN**, click **Capacity**.

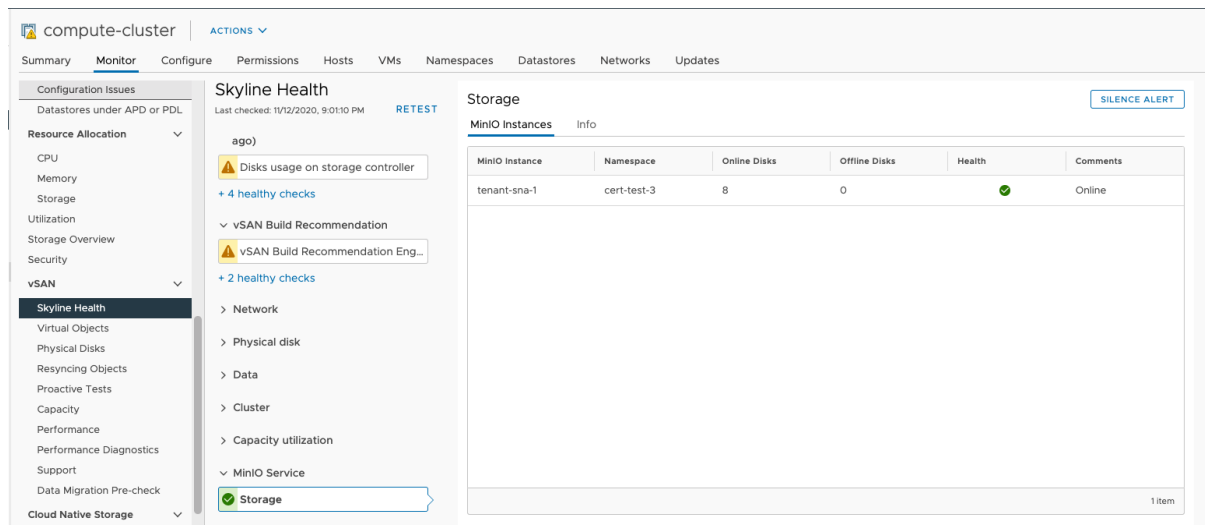
b In the **Usage breakdown** pane, display your service objects under **User objects**.



5 Monitor health of your service instances.

a Under **vSAN**, select **Skyline Health**.

b Select an individual service health check to view the detailed information.



Check Storage Policies Available for Stateful Services

As a DevOps engineer, verify that the namespace you use for stateful services in the vSphere with Tanzu environment has appropriate storage classes. The storage classes can be vSAN Shared-Nothing-Architecture (SNA) and vSAN Direct.

The vSAN Data Persistence platform automatically creates these storage classes in the namespace after a vSphere administrator enables the stateful service. See [Enable Stateful Services in vSphere with Tanzu](#).

Note Only applications that run on the Supervisor Cluster can use the vsan-direct and vsan-sna storage classes. These storage classes cannot be used inside a Tanzu Kubernetes cluster.

In addition to default storage classes, vSphere administrator can also create custom storage policies and assign them to the namespace. See [Create vSAN Direct Storage Policy](#) and [Create vSAN SNA Storage Policy](#).

Procedure

- ◆ Verify that the storage policies to be used with vSAN SNA and vSAN Direct are available in your namespace.

```
# kubectl get sc
NAME                                PROVISIONER                                RECLAIMPOLICY    VOLUMEBINDINGMODE
ALLOWVOLUMEEXPANSION    AGE
sample-vsan-direct-thick  csi.vsphere.vmware.com    Delete            WaitForFirstConsumer
true                    3m36s
sample-vsan-sna-thick    csi.vsphere.vmware.com    Delete            WaitForFirstConsumer
true                    13m
```

Create vSAN SNA Storage Policy

If you use vSAN with vSAN Data Persistence platform, you can create a vSAN Shared Nothing Architecture (SNA) storage policy to be used with the namespace where stateful services run.

Procedure

- 1 In the vSphere Client, open the **Create VM Storage Policy** wizard.
 - a In the **Home** menu, click **Policies and Profiles**.
 - b Under **Policies and Profiles**, click **VM Storage Policies**.
 - c Click **Create**.

2 Enter the policy name and description.

Option	Action
vCenter Server	Select the vCenter Server instance.
Name	Enter the name of the storage policy, for example, Sample SNA Thick .
Description	Enter the description of the storage policy.

3 On the **Policy structure** page under **Datastore specific rules**, enable rules for vSAN storage placement.

4 On the **vSAN** page, click the **Availability** tab and select the following values. The values are only applicable for SNA workloads on the vSAN Data Persistence platform. They cannot be used for provisioning VM workloads.

Option	Description
Option	Value
Site disaster tolerance	None - standard cluster
	Note vSAN Data Persistence platform supports only standard clusters.
Failures to tolerate	No data redundancy with host affinity

Thick provisioning enforced for SNA workloads and is selected as a value for object space reservation on the **Advanced Policy Rules** tab. You cannot change this value.

5 On the **Storage compatibility** page, review the list of vSAN datastores that match this policy.

6 On the **Review and finish** page, review the storage policy settings and click **Finish**.

To change any settings, click **Back** to go to the relevant page.

What to do next

After you create the policy, you can assign it to the namespace where your stateful service runs. See [Change Storage Settings on a Namespace](#).

Create vSAN Direct Storage Policy

If you use vSAN Direct with vSAN Data Persistence platform, you can create a capability based storage policy to be used with the namespace where stateful services run.

Procedure

- 1 In the vSphere Client, open the **Create VM Storage Policy** wizard.
 - a In the **Home** menu, click **Policies and Profiles**.
 - b Under **Policies and Profiles**, click **VM Storage Policies**.
 - c Click **Create**.

2 Enter the policy name and description.

Option	Action
vCenter Server	Select the vCenter Server instance.
Name	Enter the name of the storage policy, for example, Sample vSAN Direct Thick .
Description	Enter the description of the storage policy.

- 3 On the **Policy structure** page under **Datastore specific rules**, enable rules for vSAN Direct storage placement.
- 4 On the **vSAN Direct rules** page, specify vSAN Direct as a storage placement type.
- 5 On the **Storage compatibility** page, review the list of vSAN Direct datastores that match this policy.
- 6 On the **Review and finish** page, review the storage policy settings and click **Finish**.
To change any settings, click **Back** to go to the relevant page.

What to do next

After you create the policy, you can assign it to the namespace where your stateful service runs. See [Change Storage Settings on a Namespace](#).

Deploying Workloads to vSphere Pods

11

As a DevOps engineer, you can deploy and manage the life cycle of vSphere Pods within the resources boundaries of a namespace that is running on a Supervisor Cluster. You must have write permissions on a namespace to deploy vSphere Pods on it.

Read the following topics next:

- [Get and Use the Supervisor Cluster Context](#)
- [Deploy an Application to a vSphere Pod on a vSphere Namespace](#)
- [Deploy an Application to a vSphere Pod Using the Embedded Harbor Registry](#)
- [Scale a vSphere Pod Application](#)
- [Deploy a Confidential vSphere Pod](#)

Get and Use the Supervisor Cluster Context

After the vSphere administrator provides you with the IP address of the Kubernetes control plane on the Supervisor Cluster, you can log in to the Supervisor Cluster and obtain the contexts to which you have access. Contexts correspond to the namespaces on the Supervisor Cluster.

Prerequisites

- Get the IP address of the Kubernetes control plane on the Supervisor Cluster from your vSphere administrator.
- Get your user account in vCenter Single Sign-On.
- Verify with your vSphere administrator that you have permissions to access the contexts that you need.
- Verify that the certificate served by the Kubernetes control plane is trusted on your system, either by having the signing CA installed as a Trust Root or by adding the certificate as a Trust Root directly.

Procedure

- 1 In a browser window, open the URL of the Kubernetes control plane.
- 2 Verify that the SHA256 checksum of the `vsphere-plugin.zip` matches the checksum from the `sha256sum.txt` file.

- 3 Download the `vsphere-plugin.zip` file on your machine and set it on your OS's executable search path.
- 4 In a command prompt window, run the following command to log in to vCenter Server:

```
kubectl vsphere login --server=https://<server_adress> --vsphere-username <your user account name>
```

- 5 To view details of the configuration contexts which you can access to, run the following `kubectl` command:

```
kubectl config get-contexts
```

The CLI displays the details for each available context.

- 6 To switch between contexts, use the following command:

```
kubectl config use-context <example-context-name>
```

Results

The login API on the Kubernetes control plane is invoked. An authenticating proxy redirects a request for authentication to vCenter Single Sign-On. vCenter Server returns a JSON Web token and adds it to the `kubeconfig` file. That token is sent to the Kubernetes control plane with every new `kubectl` command to authenticate the user.

Deploy an Application to a vSphere Pod on a vSphere Namespace

You can deploy an application on a namespace on a Supervisor Cluster. Once the application is deployed, the respective number of vSphere Pods are created on the Supervisor Cluster within the namespace.

You can also deploy applications from images that are stored in the Harbor image registry. See [Deploy an Application to a vSphere Pod Using the Embedded Harbor Registry](#).

Prerequisites

- Get the IP address of the Kubernetes control plane on the Supervisor Cluster from your vSphere administrator.
- Get your user account in vCenter Single Sign-On.
- Verify with your vSphere administrator that you have permissions to access the contexts that you need.

Procedure

- 1 Authenticate with the Supervisor Cluster.

See [Connect to the Supervisor Cluster as a vCenter Single Sign-On User](#).

- 2 Switch to the context where you want to deploy the application.

```
kubectl config use-context <namespace>
```

- 3 Deploy the application.

```
kubectl apply -f <application name>.yaml
```

Deploy an Application to a vSphere Pod Using the Embedded Harbor Registry

You can use images that are stored in Harbor Registry to deploy vSphere Pods in namespaces on the Supervisor Cluster.

Prerequisites

- Push images to a project in Harbor Registry that has the same name as the namespace where you want to deploy your application. See [Push Images to the Embedded Harbor Registry](#).
- Add the contents of the `vsphere-plugin.zip` to the execution file path of your environment.

Procedure

- 1 Create a YAML file that contains the following parameters:

```
...
namespace: <namespace-name>
...
spec:
...
image: <image registry URL>/<namespace name>/<image name>
```

- 2 Login to the Supervisor Cluster:

```
kubectl vsphere login --server=https://<server_adress> --vsphere-username <your user
account name>
```

- 3 Switch to the namespace where you want to deploy the application.

```
kubectl config use-context <namespace>
```

- 4 Deploy a vSphere Pod from that YAML file:

```
kubectl apply -f <yaml file name>.yaml
```

- 5 Run the following command to verify that the image is pulled from the Harbor Registry and the vSphere Pod is in running state:

```
kubectl describe pod/<yaml name>
```

Results

The YAML file that you created is deployed to the specified namespace by using the image from the project on Harbor Registry that is named after the namespace.

Example:

Create and deploy the following YAML file on the namespace demoapp1 by using the busybox image from the demoapp1 project in Harbor Registry:

```
apiVersion: v1
kind: Pod
metadata:
  name: busybox
  namespace: demoapp1
spec:
  containers:
  - name: busybox
    image: <harbor_IP>/demoapp1/busybox:latest
    command:
      - sleep
      - "3600"
    imagePullPolicy: IfNotPresent
  restartPolicy: Always
```

Scale a vSphere Pod Application

You can scale up and down the number of replicas for each application that is running on a Supervisor Cluster.

Prerequisites

- Get the IP address of the Kubernetes control plane on the Supervisor Cluster from your vSphere administrator.
- Get your user account in vCenter Single Sign-On.
- Verify with your vSphere administrator that you have permissions to access the contexts that you need.

Procedure

- 1 Authenticate with the Supervisor Cluster.

```
kubectl vsphere login --server <control plane load balancer IP address> --vsphere-username
<vSphere user account name>
```

- 2 Scale up or scale down an application.

```
kubectl get deployments
kubectl scale deployment <deployment-name> --replicas=<number-of-replicas>
```

Deploy a Confidential vSphere Pod

With vSphere with Tanzu, you can run confidential vSphere Pods on a Supervisor Cluster. A confidential vSphere Pod uses a hardware technology that keeps the guest OS memory encrypted, protecting it against access from the hypervisor.

Starting from vSphere 7.0 Update 2, you can create confidential vSphere Pods by adding Secure Encrypted Virtualization-Encrypted State (SEV-ES) as an extra security enhancement. SEV-ES prevents CPU registers from leaking information in registers to components like the hypervisor. SEV-ES can also detect malicious modifications to a CPU register state. For more information about using SEV-ES technology in the vSphere environment, see [Securing Virtual Machines with AMD Secure Encrypted Virtualization-Encrypted State](#).

Prerequisites

To enable SEV-ES on an ESXi host, a vSphere administrator must follow these guidelines:

- Use the hosts that support the SEV-ES functionality. Currently, SEV-ES supports only AMD EPYC 7xx2 CPUs (code named *Rome*) and later CPUs.
- Use the ESXi version of 7.0 Update 2 or later.
- Enable SEV-ES in an ESXi system's BIOS configuration. See your system's documentation for more information about accessing the BIOS configuration.
- When enabling SEV-ES in the BIOS, enter a value for the **Minimum SEV non-ES ASID** setting equal the number of SEV-ES VMs and confidential vSphere Pods on the host plus one. For example, if you plan to run 100 SEV-ES VMs and 128 vSphere Pods, enter at least 229. You can enter a setting as high as 500.

Procedure**1** Create a YAML file that contains the following parameters.

- a In annotations, enable the confidential vSphere Pods feature.

```
...
annotations:
  vmware/confidential-pod: enabled
...
```

- b Specify memory resources for containers.

Make sure to set memory requests and memory limits to the same value, as in this example.

```
resources:
  requests:
    memory: "512Mi"
  limits:
    memory: "512Mi"
```

Use the following YAML file as an example:

```
apiVersion: v1
kind: Pod
metadata:
  name: photon-pod
  namespace: my-podvm-ns
  annotations:
    vmware/confidential-pod: enabled
spec: # specification of the pod's contents
  restartPolicy: Never
  containers:
  - name: photon
    image: wcp-docker-ci.artifactory.eng.vmware.com/vmware/photon:1.0
    command: ["/bin/sh"]
    args: ["-c", "while true; do echo hello, world!; sleep 1; done"]
    resources:
      requests:
        memory: "512Mi"
      limits:
        memory: "512Mi"
```

2 Log in to the Supervisor Cluster.

```
kubectl vsphere login --server=https://<server_adress> --vsphere-username <your user
account name>
```

3 Switch to the namespace where you want to deploy the application.

```
kubectl config use-context <namespace>
```

4 Deploy a confidential vSphere Pod from the YAML file.

```
kubectl apply -f <yaml file name>.yaml
```

Note When the vSphere Pod is deployed, DRS places it to the ESXi node that supports SEV-ES. If no such node is available, the vSphere Pod is marked as failed.

The confidential vSphere Pod that is launched provides hardware memory encryption support for all workloads that are running on that pod.

5 Run the following command to verify that the confidential vSphere Pod has been created.

```
kubectl describe pod/<yaml name>
```

What to do next

A vSphere administrator can view the confidential vSphere Pod. In the vSphere Client, it appears with the **Encryption Mode: Confidential Compute** tag.

vm vSphere Client

Menu

Search

vm

pod-vm-1

ACTION

Summary

Monitor

Configure

Compute

Storage

vm

pod-vm-1

ACTION

Summary

Monitor

Configure

Compute

Storage

Status

Running

Tue, 12 Feb 2019 14:57:30

Namespace

work-auth

Node

m-04.eng.vmware.com

Restart Policy

Inactive

Containers

8 Total

Container 1

ImageName 1

Container 2

ImageName 2

Container 3

ImageName 3

VIEW ALL

Metadata

UID

fd726e00-180f-11e8-8fa1-0050568e3cc9

Labels

Application Windows

Application Windows

Application Windows

and 9 more

QoS Class

BestEffort

Encryption mode

Confidential Compute

VIEW YAML

Deploying and Managing Virtual Machines in vSphere with Tanzu

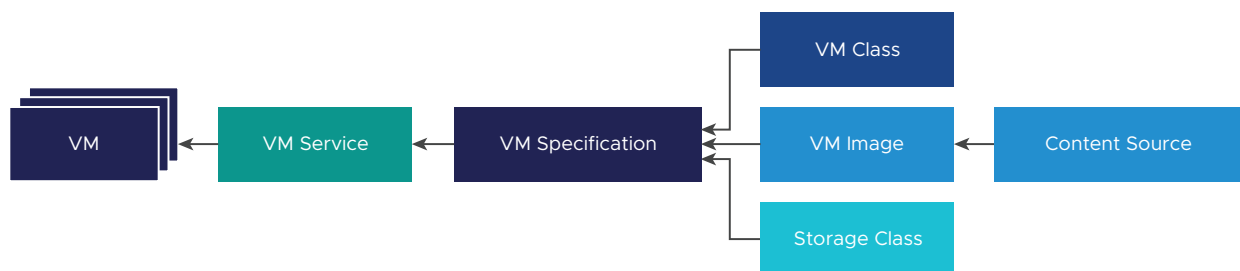
12

vSphere with Tanzu offers a VM Service functionality that enables DevOps engineers to deploy and run VMs, in addition to containers, in a common, shared Kubernetes environment. You can use the VM Service to manage the lifecycle of virtual machines in a vSphere Namespace. VM Service manages stand-alone VMs and VMs that make up Tanzu Kubernetes clusters.

Generally, your business needs and goals dictate your decision to run workloads in a VM rather than in a container. For information about when to run a VM, see [Using Virtual Machines in vSphere with Tanzu](#).

Concepts of VM Service

To describe the state of a VM to be deployed in a vSphere Namespace, you use such parameters as a VM class, a VM image, and a storage class. The VM Service then brings together these specifications to create stand-alone VMs or VMs that support Tanzu Kubernetes clusters.



VM Service

The VM Service is a component of vSphere with Tanzu that provides a declarative, Kubernetes-style API for management of VMs and associated vSphere resources. The VM Service enables the vSphere administrators to deliver resources and provide templates, such as VM classes and VM images, to Kubernetes. DevOps engineers can use these resources to describe the desired state of a VM. After the DevOps engineers specify the VM state, the VM Service converts the desired state in to a realized state against backing infrastructure resources.

A VM created through the VM Service can be managed only from the Kubernetes namespace with the `kubectl` commands. vSphere administrators cannot manage the VM from the vSphere Client, but can display its details and monitor resources it uses. For information, see [Monitor Virtual Machines Available in vSphere with Tanzu](#).

VM Class

The VM class is a VM specification that can be used to request a set of resources for a VM. The VM class is controlled and managed by a vSphere administrator, and defines such parameters as the number of virtual CPUs, memory capacity, and reservation settings. The defined parameters are backed and guaranteed by the underlying infrastructure resources of a Supervisor Cluster.

A vSphere administrator can create custom VM classes.

In addition, Workload Management offers several default VM classes. Generally, each default class type comes in two editions: guaranteed and best effort. A guaranteed edition fully reserves resources that a VM specification requests. A best effort class edition does not and allows resources to be overcommitted. Typically, a guaranteed type is used in a production environment.

Examples of default VM classes include the following.

Class	CPU	Memory (GB)	Reserved CPU and Memory
guaranteed-large	4	16	Yes
best-effort-large	4	16	No
guaranteed-small	2	4	Yes
best-effort-small	2	4	No

The vSphere administrator can assign any number of existing VM classes to make them available to DevOps engineers within a specific namespace.

The VM class provides a simplified experience for the DevOps engineers. The DevOps don't need to understand the full configuration of each VM that they plan to create. Instead, they can select a VM class from available options, and the VM Service manages the VM configuration.

On the Kubernetes side, the VM classes appear as `VirtualMachineClass` and `VirtualMachineClassBinding` resources.

VM Image

A VM image is a template that contains a software configuration, including an operating system, applications, and data.

When DevOps engineers create VMs, they can select images from the content library associated with the namespace. To the DevOps, the images are exposed as `VirtualMachineImage` objects.

VM Service supports a limited number of VM images and guest OSes. Compatible VM images appear on VMware Marketplace as OVF's. Make sure to use only those VM images that are supported by VM Service. To find compatible images, search for **VM Service image** on the [VMware Cloud Marketplace](#) web site. See an example of the VM Service image for CentOS at [VM Service Image for CentOS](#).

Content Source

A DevOps engineer uses a content library as a source of images to create a VM. Similar to VM classes, a vSphere administrator can assign existing content libraries to a namespace to make them available to DevOps engineers.

Storage Class

VM Service uses storage classes for placing virtual disks and dynamically attaching persistent volumes. For more information about storage classes, see [Chapter 10 Using Persistent Storage in vSphere with Tanzu](#).

VM Specification

DevOps engineers describe the desired state of a VM in a YAML file that brings together the VM image, the VM class, and the storage class.

Networking

VM Service does not have any specific requirements and relies on the network configuration available in vSphere with Tanzu. VM Service supports both types of networking, the vSphere networking or NSX-T. When VMs are deployed, an available network provider allocates static IP addresses to the VMs. For information, see [Chapter 4 Networking for vSphere with Tanzu](#).

vSphere Administrator Workflow for Provisioning a VM

As a vSphere administrator, you set guardrails for the policy and governance of the VMs, and deliver VM resources, such as VM classes and VM templates to DevOps engineers. After a VM is deployed, you can monitor it using the vSphere Client.

Step	Description	Instructions
1	Create and manage VM classes.	■ Create a VM Class in vSphere with Tanzu ■ To use NVIDIA vGPU, configure a PCI device in the VM class. See Add PCI Devices to a VM Class in vSphere with Tanzu. ■ Edit or Delete a VM Class in vSphere with Tanzu
2	Associate a set of VM classes with a namespace.	Associate a VM Class with a Namespace in vSphere with Tanzu

Step	Description	Instructions
3	Create and manage content libraries.	■ For stand-alone VMs, see Creating and Managing Content Libraries for Stand-Alone VMs in vSphere with Tanzu. ■ For Tanzu Kubernetes clusters, see Creating and Managing Content Libraries for Tanzu Kubernetes releases.
4	Associate a content library with a namespace.	■ For stand-alone VMs, see Associate a VM Content Library with a Namespace in vSphere with Tanzu. ■ For Tanzu Kubernetes clusters, see Configure a vSphere Namespace for Tanzu Kubernetes releases.
5	Associate storage classes with a namespace.	Create and Configure a vSphere Namespace
6	Monitor deployed VMs.	Monitor Virtual Machines Available in vSphere with Tanzu

DevOps Engineer Workflow for Provisioning a VM

DevOps engineers with permissions, can review available VM resources and deploy VMs into the namespace. They use the `kubectl` command to perform the following tasks.

Step	Description	Instructions
1	List VM classes, images, and other resources associated with the namespace.	View VM Resources Available on a Namespace in vSphere with Tanzu
2	Create a VM.	■ For stand-alone VMs, see Deploy a Virtual Machine in vSphere with Tanzu. ■ For Tanzu Kubernetes cluster VMs, see Workflow for Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API.

Read the following topics next:

- [Create a VM Class in vSphere with Tanzu](#)
- [Add PCI Devices to a VM Class in vSphere with Tanzu](#)
- [Edit or Delete a VM Class in vSphere with Tanzu](#)
- [Associate a VM Class with a Namespace in vSphere with Tanzu](#)
- [Manage VM Classes on a Namespace in vSphere with Tanzu](#)
- [View VM Resources Available on a Namespace in vSphere with Tanzu](#)

- [Deploy a Virtual Machine in vSphere with Tanzu](#)
- [Monitor Virtual Machines Available in vSphere with Tanzu](#)

Create a VM Class in vSphere with Tanzu

As a vSphere administrator, create custom VM classes to be used for a VM deployment in a namespace in vSphere with Tanzu. Custom VM classes can be used by stand-alone VMs that run in namespaces and by VMs hosting a Tanzu Kubernetes cluster.

A VM class is a template that defines CPU, memory, and reservations for VMs. The VM class helps to set guardrails for the policy and governance of VMs by anticipating development needs and accounting for resource availability and constraints. vSphere with Tanzu offers several default VM classes. You can use them as is, edit, or delete them.

You can also create custom VM classes. When you create new classes, keep in mind the following considerations.

- VM classes that you create in a vCenter Server instance are available to all vCenter Server clusters and all namespaces in these clusters.
- VM classes are available to all namespaces in vCenter Server. However, DevOps engineers can use only those VM classes that you associate with a particular namespace.

Prerequisites

Required privileges:

- **Namespaces.Modify cluster-wide configuration**
- **Namespaces.Modify namespace configuration**
- **Virtual Machine Classes.Manage Virtual Machine Classes**

Procedure

- 1 Go to the **VM Service** page.
 - a From the vSphere Client home menu, select **Workload Management**.
 - b Click the **Services** tab and click **Manage** on the **VM Service** pane.
- 2 On the **VM Service** page, click **VM Classes** and click **Create VM Class**.

3 On the **Configuration** page, specify the general VM class attributes.

VM Class Attribute	Description
Name	Identifies the VM class. Enter a unique DNS compliant name that follows these requirements: ■ Use a unique name that does not duplicate the names of default or custom VM classes in your environment. ■ Use alphanumeric string with maximum length of 63 characters. ■ Do not use uppercase letters or spaces. ■ Use a dash anywhere except as a first or last character. For example, vm-class1. After you create the VM class, you cannot change its name.
vCPU Count	Defines the number of virtual CPUs (vCPUs) for a VM. This is a VM hardware configuration. When a DevOps user assigns the VM class to a VM, this count becomes the configured number of vCPUs for the VM.
CPU Resource Reservation	Optional parameter. Specifies the guaranteed minimum CPU resource allocation for a virtual machine. This value is expressed in percentage (%). Value of 0 % defines no CPU reservation. The percentage you enter is multiplied by the minimum CPU available among all the cluster nodes. The resulting value, in MHz, specifies the amount of CPU resources that vSphere guarantees for a VM.
Memory	Defines the memory configured for a VM in MB, GB, or TB. This is a VM hardware configuration. When a DevOps user assigns the VM class policy to a VM, the VM receives the amount of memory defined by this attribute. The value must be between 4 MB and 24 TB and a multiple of 4 MB.
Memory Resource Reservation	Optional parameter. Defines the reserved amount of memory that is configured for a VM. The value of the attribute ranges between 0 and 100%. If you add PCI devices to the VM class configuration, set the parameter to 100%.

4 (Optional) To add PCI devices, on the **Configuration** page, select **Yes** from the **PCI Devices** drop-down menu and click **Next**.

If you select this option, the memory resource reservation value automatically changes to 100%.

For requirements and additional details, see [Add PCI Devices to a VM Class in vSphere with Tanzu](#).

5 On the **Review and Confirm** page, review the details and click **Finish**.

What to do next

After you create a VM class, you can edit its parameters or delete it from your environment. See [Edit or Delete a VM Class in vSphere with Tanzu](#).

To make the VM class available to DevOps engineers, associate it with a namespace. The association of the VM class occurs on a per namespace level. See [Associate a VM Class with a Namespace in vSphere with Tanzu](#).

Attributes of VM Classes in vSphere with Tanzu

As a vSphere administrator, you can create or edit virtual machine (VM) classes that VMs in a vSphere Namespace use. For each VM class, you specify a subset of available attributes.

The following table lists all attributes that you can define within a VM class.

VM Class Attribute	Description
Name	Identifies the VM class. Enter a unique DNS compliant name that follows these requirements: ■ Use a unique name that does not duplicate the names of default or custom VM classes in your environment. ■ Use alphanumeric string with maximum length of 63 characters. ■ Do not use uppercase letters or spaces. ■ Use a dash anywhere except as a first or last character. For example, vm-class1. After you create the VM class, you cannot change its name.
vCPU Count	Defines the number of virtual CPUs (vCPUs) for a VM. This is a VM hardware configuration. When a DevOps user assigns the VM class to a VM, this count becomes the configured number of vCPUs for the VM.
CPU Resource Reservation	Optional parameter. Specifies the guaranteed minimum CPU resource allocation for a virtual machine. This value is expressed in percentage (%). Value of 0 % defines no CPU reservation. The percentage you enter is multiplied by the minimum CPU available among all the cluster nodes. The resulting value, in MHz, specifies the amount of CPU resources that vSphere guarantees for a VM.

VM Class Attribute	Description
Memory	Defines the memory configured for a VM in MB, GB, or TB. This is a VM hardware configuration. When a DevOps user assigns the VM class policy to a VM, the VM receives the amount of memory defined by this attribute. The value must be between 4 MB and 24 TB and a multiple of 4 MB.
Memory Resource Reservation	Optional parameter. Defines the reserved amount of memory that is configured for a VM. The value of the attribute ranges between 0 and 100%. If you add PCI devices to the VM class configuration, set the parameter to 100%.

Add PCI Devices to a VM Class in vSphere with Tanzu

If ESXi hosts in your vSphere with Tanzu environment have one or more NVIDIA GRID GPU graphics devices, you can configure VMs to use the NVIDIA GRID virtual GPU (vGPU) technology. You can also configure other PCI devices on an ESXi host to make them available to a VM in a passthrough mode.

NVIDIA GRID GPU

NVIDIA GRID GPU graphics devices are designed to optimize complex graphics operations and enable them to run at high performance without overloading the CPU. NVIDIA GRID vGPU provides unparalleled graphics performance, cost-effectiveness, and scalability by sharing a single physical GPU among multiple VMs as separate vGPU-enabled passthrough devices.

When you configure NVIDIA vGPU for a VM, you add a PCI device for vGPU to a VM class.

The following considerations apply when you use NVIDIA vGPU:

- VMs with vGPU devices that are managed by VM Service are automatically powered off when an ESXi host enters maintenance mode. This might temporarily affect workloads running in the VMs. The VMs are automatically powered on after the host exists the maintenance mode.

Dynamic DirectPath I/O

Using Dynamic DirectPath I/O, the VM can directly access the physical PCI and PCIe devices connected to a host.

You can use Dynamic DirectPath I/O to assign multiple PCI passthrough devices to a VM. Each passthrough device can be specified by its PCI vendor and device identifier.

Prerequisites

- Verify that the host machine is supported in the [VMware Compatibility Guide](#), and check with the vendor to verify the host meets power and configuration requirements. Install a PCI device on the ESXi host.

- To configure NVIDIA vGPU, follow these prerequisites:
 - Use vSphere version 7.0 Update 3 or later.
 - Configure ESXi host graphics settings with at least one device in **Shared Direct** mode. See [Configuring Host Graphics](#).
 - Install NVIDIA vGPU software. NVIDIA provides a vGPU software package that includes the following components.

For more information, see appropriate NVIDIA Virtual GPU Software documentation.

 - vGPU Manager that a vSphere administrator installs on the ESXi host. See [VMware Knowledge Base article 2033434](#).
 - Guest VM driver that a DevOps engineer installs in the VM after deploying and booting the VM. See [Install the NVIDIA Guest Driver in a VM in vSphere with Tanzu](#).
- To configure Dynamic DirectPath I/O for PCI passthrough devices, follow these prerequisites:
 - Use vSphere version 7.0 Update 3 MP01.
 - Connect the PCI devices to the host and mark them as available for passthrough. See [Mark a PCI Device as Passthrough](#).
- Required privileges:
 - **Namespaces.Modify cluster-wide configuration**
 - **Namespaces.Modify namespace configuration**
 - **Virtual Machine Classes.Manage Virtual Machine Classes**

Procedure

- 1 Add a PCI device to a VM class when you create or edit an existing VM class.

Option	Action
Create a new VM class	- a From the vSphere Client home menu, select Workload Management. - b Click the Services tab and click Manage on the VM Service pane. - c On the VM Service page, click VM Classes and click Create VM Class. - d On the Configuration page, specify the general VM class attributes. See Attributes of VM Classes in vSphere with Tanzu. Make sure that the memory resource reservation value is set to 100%. - e To add PCI devices, on the Configuration page, select Yes from the PCI Devices drop-down menu and click Next.
Edit a VM class	- a From the vSphere Client home menu, select Workload Management. - b Click the Services tab and click Manage on the VM Service pane. - c On the VM Service page, click VM Classes. - d In the existing VM class pane, click Manage and click Edit. Make sure that the memory resource reservation value is set to 100%. - e To add PCI devices, on the Configuration page, select Yes from the PCI Devices drop-down menu and click Next.

- On the **PCI Devices** page, expand the **Add PCI Device** menu, select the access type and other appropriate options, and click **Next**.

Option	Action
NVIDIA GRID vGPU	Specify the following options: ■ Model. Name of physical device. Select the device from the list of devices available on the host. ■ GPU Sharing. Indicates how physical GPU is shared across VMs. For example, Time Sharing. ■ GPU Mode. The GPU mode within a VM. For example, Compute is a configuration that is optimized for high-performance computing applications. While Workstation is used for graphics intensive workloads. ■ GPU Memory. Minimum GPU memory in GB per VM. ■ Number of vGPUs. Number of vGPU devices per VM.
Dynamic DirectPath IO	From the PCI Device list, select the PCI passthrough devices by their vendor, model name, or hardware label.

- On the **Review and Confirm** page, review the details and click **Finish**.

Results

A **GPUs** tag on the VM class pane indicates that the VM class is GPU-enabled.

Edit or Delete a VM Class in vSphere with Tanzu

As a vSphere administrator, you can create custom VM classes for VMs in a vSphere Namespace. After you create a VM class, you can edit its parameters. You can also edit default VM classes that vSphere with Tanzu offers. If you no longer need an existing VM class, you can delete it from your environment.

Editing a VM class does not result in automatic reconfiguring of the VMs that were previously deployed from this class. For example, if a DevOps user created a Tanzu Kubernetes cluster with the VM class and you later change the VM class definition, existing Tanzu Kubernetes VMs remain unaffected. New Tanzu Kubernetes VMs will use the modified class definition.

Caution If you scale out a Tanzu Kubernetes cluster after editing a VM class used by that cluster, new cluster nodes use the updated class definition, but existing cluster nodes continue to use the initial class definition, resulting in a mismatch. Both control plane and worker nodes can be scaled. For information about scaling, [Scale a Tanzu Kubernetes Cluster Using the Tanzu Kubernetes Grid Service v1alpha1 API](#).

When you delete a VM class, it is removed from all associated namespaces. DevOps users can no longer self-service VMs using this VM class. VMs that have already been created with this VM class are not affected.

Prerequisites

- Verify that you have at least one VM class. See [Create a VM Class in vSphere with Tanzu](#).

- Required privileges:
 - **Namespaces.Modify cluster-wide configuration**
 - **Namespaces.Modify namespace configuration**
 - **Virtual Machine Classes.Manage Virtual Machine Classes**

Procedure

- 1 In the vSphere Client, display available VM classes.
 - a From the vSphere Client home menu, select **Workload Management**.
 - b Click the **Services** tab and click the **VM Service** pane.
 - c On the **VM Service** page, click **VM Classes**.

All default or user-created VM classes appear under **Available VM Classes**.
- 2 Edit or delete an existing VM class.

Option	Description
Edit a VM class	a In the selected VM class pane, click Manage and click Edit .
	b Modify the VM class parameters. See Attributes of VM Classes in vSphere with Tanzu .
Note You cannot change the name of the VM class.	
Delete a VM class	a In the selected VM class pane, click Manage and click Delete .
	b Confirm that you want to delete the VM class.

What to do next

To make a VM class available to DevOps engineers, associate the VM class with a namespace. The association of the VM class occurs on a per namespace level. See [Associate a VM Class with a Namespace in vSphere with Tanzu](#).

Associate a VM Class with a Namespace in vSphere with Tanzu

As a vSphere administrator, you can add a VM class to one or more namespaces on a Supervisor Cluster. When you add a VM class to a namespace, you make the class available to DevOps users, so that they can start self-servicing VMs in the Kubernetes namespace environment. The VM classes you assign to the namespace are also used by the VMs that make up Tanzu Kubernetes clusters.

You can add multiple VM classes to a single namespace. Different VM classes serve as indicators of different levels of service. If you publish multiple VM classes, DevOps users can select between all custom and default classes when creating and managing virtual machines in the namespace.

Note To be able to deploy a Tanzu Kubernetes cluster in a newly created namespace, DevOps engineers need to have access to VM classes. As a vSphere administrator, you must explicitly associate default or custom VM classes to any new namespace where the Tanzu Kubernetes cluster is deployed. Existing namespaces, where Tanzu Kubernetes clusters have been already provisioned, continue to have automatic access to default VM classes. But you can also associate custom VM classes with any existing namespace.

Prerequisites

Use default VM classes that VMware provides or create new classes. See [Create a VM Class in vSphere with Tanzu](#).

Required privileges:

- **Namespaces.Modify cluster-wide configuration**
- **Namespaces.Modify namespace configuration**
- **Virtual Machine Classes.Manage Virtual Machine Classes**

Procedure

- 1 In the vSphere Client, go to the namespace.
 - a From the vSphere Client home menu, select **Workload Management**.
 - b Click the **Namespaces** tab and click the namespace.
- 2 Add a VM class.
 - a On the **VM Service** pane, click **Add VM Class**.
 - b Select one or several VM classes and click **OK**.

Results

The VM classes you added become available in the namespace for the DevOps to self-service VMs. These classes can also be used by the VMs that make up Tanzu Kubernetes clusters.

What to do next

After you associate a VM class with a namespace, you can add more VM classes or remove the class to unpublish it from the namespace. See [Manage VM Classes on a Namespace in vSphere with Tanzu](#).

Manage VM Classes on a Namespace in vSphere with Tanzu

As a vSphere administrator, you can associate a VM class with one or more namespaces on a Supervisor Cluster. After you associate a VM class with a namespace, you can add more VM classes or remove the class to unpublish it from the Kubernetes namespace.

Prerequisites

- Verify that at least one VM class is associated with the namespace. See [Associate a VM Class with a Namespace in vSphere with Tanzu](#).
- If you want to remove a VM class from a namespace, verify that it is not used by Tanzu Kubernetes Grid Service. Removing it can affect operations Tanzu Kubernetes Grid Service.
- Required privileges:
 - **Namespaces.Modify cluster-wide configuration**
 - **Namespaces.Modify namespace configuration**
 - **Virtual Machine Classes.Manage Virtual Machine Classes**

Procedure

- 1 In the vSphere Client, go to the namespace.
 - a From the vSphere Client home menu, select **Workload Management**.
 - b Click the **Namespaces** tab and click the namespace.
- 2 Add or remove a VM class.
 - a On the **VM Service** pane, click **Manage VM Class**.
 - b Perform one of the following operations.

Option	Description
Remove a VM class	Deselect the VM class and click OK .
Add a VM class	Select one or several VM classes and click OK .

View VM Resources Available on a Namespace in vSphere with Tanzu

To be able to deploy a stand-alone VM in vSphere with Tanzu, a DevOps engineer must have access to specific VM resources. As a DevOps engineer, verify that you can access these resources and view VM classes and VM templates available in your environment. You can also list storage classes and other items you might need to self-service a VM.

This task covers commands you use to access resources available for a deployment of a stand-alone VM. For information about resources necessary to deploy Tanzu Kubernetes clusters and VMs that make up the clusters, see [Virtual Machine Classes for Tanzu Kubernetes Clusters](#).

Prerequisites

A vSphere administrator has performed these steps:

- Created a namespace and associated it with storage policies. See [Create and Configure a vSphere Namespace](#).
- Associated default or custom VM classes with a namespace. See [Associate a VM Class with a Namespace in vSphere with Tanzu](#).
- Created a content library and associated it with a namespace. See [Associate a VM Content Library with a Namespace in vSphere with Tanzu](#).

Note If a content library is protected by a security policy, all library items must be complaint. If a protected library includes a mix of compliant and non-compliant items, the `kubectl get virtualmachineimages` command fails to present VM images to the DevOps engineers.

Procedure

- 1 Access your namespace in the Kubernetes environment.

See [Get and Use the Supervisor Cluster Context](#).

- 2 To view VM classes available in your namespace, run the following command.

```
kubectl get virtualmachineclassbindings
```

You can see the following output.

Note Because the best effort VM class type allows resources to be overcommitted, you can run out of resources if you have set limits on the namespace where you are provisioning the VMs. For this reason, use the guaranteed VM class type in the production environment.

NAME	VIRTUALMACHINECLASS	AGE
best-effort-large	best-effort-large	44m
best-effort-medium	best-effort-medium	44m
best-effort-small	best-effort-small	44m
best-effort-xsmall	best-effort-xsmall	44m
custom	custom	44m

- 3 To view details of a specific VM class, run the following commands.

- `kubectl describe virtualmachineclasses name_vm_class`

If a VM class includes a vGPU device, you can see its profile under `spec: hardware: devices: vgpuDevices`.

```
.....
spec:
  hardware:
    cpus: 4
```

```

    devices:
      vgpuDevices:
        - profileName: grid_v100-q4
        .....

```

- `kubectl get virtualmachineclasses -o wide`

If the VM class includes a vGPU or a passthrough device, the output shows it in the `VGPUDevicesProfileNames` or `PassthroughDeviceIDs` column.

4 View the VM images.

```
kubectl get virtualmachineimages
```

The output you see is similar to the following.

NAME	VERSION	OSTYPE	FORMAT
IMAGESUPPORTED AGE			
centos-stream-8-vmervice-v1alpha1-xxxxxxxxxxxxxx		centos8_64Guest	ovf
true 4d3h			

5 To describe a specific image, use the following command.

```
kubectl describe virtualmachineimage/centos-stream-8-vmervice-v1alpha1-xxxxxxxxxxxxxx
```

VMs with vGPU devices require images that have boot mode set to EFI, such as CentOS. Make sure to have access to these images. For information about supported images, search for **VM Service image** on the [VMware Cloud Marketplace](#) web site.

6 Verify that you can access storage classes.

```
kubectl get resourcequotas
```

For more information, see [Display Storage Classes in a vSphere Namespace or Tanzu Kubernetes Cluster](#).

NAME	AGE
REQUEST	
LIMIT	
my-ns-ubuntu-storagequota	24h wcpglobal-storage-profile.storageclass.storage.k8s.io/requests.storage: 0/9223372036854775807

7 If you are using vSphere Distributed Switch for your workload networking, obtain the name of the network.

Note You use this information to specify the `networkName` parameter in the VM YAML file when the `networkType` is **vsphere-distributed**. You do not need to obtain and specify the network name if you use VMware NSX-T.

```
kubectl get network
```

```
NAME      AGE
primary   7d2h
```

What to do next

You can now deploy VMs. See [Deploy a Virtual Machine in vSphere with Tanzu](#).

Deploy a Virtual Machine in vSphere with Tanzu

As a DevOps engineer, you can provision a VM and its guest OS in a declarative manner by writing VM deployment specifications in a Kubernetes YAML file.

Prerequisites

- Verify that you have available resources to deploy a VM in your namespace. See [View VM Resources Available on a Namespace in vSphere with Tanzu](#).
- If you use NVIDIA vGPU or other PCI devices for your VMs, the following considerations apply:
 - Make sure to use appropriate VM class with PCI configuration. See [Add PCI Devices to a VM Class in vSphere with Tanzu](#).
 - VMs with vGPU devices require images that have boot mode set to EFI, such as CentOS. Make sure to have access to these images. For information about supported images, search for **VM Service image** on the [VMware Cloud Marketplace](#) web site.
 - VMs with vGPU devices that are managed by VM Service are automatically powered off when an ESXi host enters maintenance mode. This might temporarily affect workloads running in the VMs. The VMs are automatically powered on after the host exits the maintenance mode.

Procedure

- 1 Prepare the VM YAML file.

In the file, specify the following parameters:

Option	Description
apiVersion	Specifies the version of the VM Service API. Such as vmoperator.vmware.com/v1alpha1 .
kind	Specifies the type of Kubernetes resource to create. The only available value is VirtualMachine .
spec.imageName	Specifies the content library image the VM should use. For example, centos-stream-8-vmervice-v1alpha1-xxxxxxxxxxxxxx .
spec.storageClass	Identifies the storage class to be used for storage of the persistent volumes. For example, wcpglobal-storage-profile .

Option	Description
spec.className	Specifies the name of the VM class that describes the virtual hardware settings to be used. For example, custom .
spec.networkInterfaces	Specifies network-related settings for the VM. ■ networkType. Values for this key can be nsx-t or vsphere-distributed. ■ networkName. Specify the name only if networkType is vsphere-distributed. You can obtain this information using the <code>kubectl get network</code> command. If networkType is nsx-t, you do not need to indicate networkName.
spec.vmMetadata	Includes additional metadata to pass to the VM. You can use this key to customize the guest OS image and set such items as the hostname of the VM and user-data , including passwords, ssh keys, and so on. The example YAML below uses ConfigMap to store the metadata.

Use the following as an example of a YAML file `vmSvc-centos-vm.yaml`.

```

apiVersion: vmoperator.vmware.com/v1alpha1
kind: VirtualMachine
metadata:
  name: vmSvc-centos-vm
  namespace: my-ns-centos
spec:
  imageName: centos-stream-8-vmService-v1alpha1-xxxxxxxxxxxxx
  className: custom
  powerState: poweredOn
  storageClass: wcpGlobal-storage-profile
  networkInterfaces:
    - networkName: primary
      networkType: vsphere-distributed
  vmMetadata:
    configMapName: vmSvc-centos-nginx-cm
    transport: OvfEnv
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: vmSvc-centos-nginx-cm
  namespace: my-ns-centos
data:
  user-data: >-

I2Nsb3VklWNvbmZpZwoKcGFzc3dvcmQ6IFZNV0FSRQpzc2hfcHdhbXR0OiB0cnVlCgp1c2VyczoKICAtIG5hbWU6IHZtd2FyZQogICAgc3VkbzogQUxMPShBTEwpIE5PUEFTUldEOkFMTAogICAgbG9ja19wYXNzd2Q6IGZhbHNlCiAgICAgIFBhc3N3b3JkIHNdCB0byBBZG1pbiEyMwogICAgcGFzc3dkOiAnJDEkc2FsdCRTT0MzM2ZWYkEvWnhlSXdENXl3MXUxJwogICAgc2h1bGw6IC9iaW4vYmFzaAoKd3JpdGVfZmlsZXNM6CiAgLSBjb250ZW50OiB8CiAgICAgIFZNU1ZDIFNheXMgSGVsbG8gV29ybGQKICAgIHBBhdGg6IC9oZWxs3dvcmxkCg==

```


ConfigMap contains the cloud-config blob that specifies the username and password for the guest OS. In this example, `user-data` in `vmsvc-centos-nginx-cm` ConfigMap represents the following snippet in base64 format:

```
#cloud-config
password: VMWARE
ssh_pwauth: true
users:
  - name: vmware
    sudo: ALL=(ALL) NOPASSWD:ALL
    lock_passwd: false
    passwd: '$1$salt$SOC33fVbA/ZxeIwD5ywlul'
    shell: /bin/bash
write_files:
  - content: |
      VMSVC Says Hello World
    path: /helloworld
```

For more information about cloud-config specifications, see <https://cloudinit.readthedocs.io/en/latest/topics/examples.html>.

2 Deploy the VM.

```
kubectl apply -f vmsvc-centos-vm.yaml
```

3 Verify that the VM has been created.

```
kubectl get vm -n my-ns-centos
NAME                                AGE
vmsvc-centos-vm                    28s
```

4 Check the status of the VM and associated events.

```
kubectl describe virtualmachine vmsvc-centos-vm
```

The output is similar to the following. From the output, you can also obtain the IP address of the VM, which appears in the `Vm Ip` field.

```
Name:                vmsvc-centos-vm
Namespace:           my-ns-centos
Annotations:         vmoperator.vmware.com/image-supported-check: disabled
API Version:         vmoperator.vmware.com/v1alpha1
Kind:                VirtualMachine
Metadata:
  Creation Timestamp: 2021-03-23T19:07:36Z
  Finalizers:
    virtualmachine.vmoperator.vmware.com
  Generation:        1
  Managed Fields:
  ...
  ...
Spec:
  Class Name:        custom
```

```

Image Name:  vmervice-centos-20-10-server-cloudimg-amd64
Network Interfaces:
  Network Name:  primary
  Network Type:  vsphere-distributed
Power State:  poweredOn
Storage Class:  wcpglobal-storage-profile
Vm Metadata:
  Config Map Name:  vmsvc-centos-nginx-cm
  Transport:  OvfEnv
Status:
  Bios UUID:  4218ec42-aeb3-9491-fe22-19b6f954ce38
  Change Block Tracking:  false
  Conditions:
    Last Transition Time:  2021-03-23T19:08:59Z
    Status:  True
    Type:  VirtualMachinePrereqReady
  Host:  10.185.240.10
  Instance UUID:  50180b3a-86ee-870a-c3da-90ddbaffc950
  Phase:  Created
  Power State:  poweredOn
  Unique ID:  vm-73
  Vm Ip:  10.161.75.162
Events:  <none>
...

```

5 Verify that the VM IP is reachable.

```

ping 10.161.75.162
PING 10.161.75.162 (10.161.75.162): 56 data bytes
64 bytes from 10.161.75.162: icmp_seq=0 ttl=59 time=43.528 ms
64 bytes from 10.161.75.162: icmp_seq=1 ttl=59 time=53.885 ms
64 bytes from 10.161.75.162: icmp_seq=2 ttl=59 time=31.581 ms

```

Results

A VM created through the VM Service can be managed only by DevOps from the Kubernetes namespace. Its life cycle cannot be managed from the vSphere Client, but vSphere administrators can monitor the VM and its resources. For more information, see [Monitor Virtual Machines Available in vSphere with Tanzu](#).

What to do next

For additional details, see the [Introducing Virtual Machine Provisioning](#) blog.

If the VM includes a PCI device configured for vGPU, install the NVIDIA display driver. See [Install the NVIDIA Guest Driver in a VM in vSphere with Tanzu](#).

Install the NVIDIA Guest Driver in a VM in vSphere with Tanzu

After you create and boot a VM in your vSphere with Tanzu environment, install the NVIDIA vGPU graphics driver in the VM to fully enable GPU operation.

Prerequisites

- Create a VM with NVIDIA vGPU device. The VM must reference the VM class that includes a vGPU definition. See [Add PCI Devices to a VM Class in vSphere with Tanzu](#).
- Verify that you downloaded the vGPU software package from the NVIDIA download site, uncompressed the package, and have the guest drive component ready. For information, see appropriate NVIDIA Virtual GPU Software documentation.

Note The version of the driver component must correspond to the version of the vGPU Manager that a vSphere administrator installed on the ESXi host. See [Add PCI Devices to a VM Class in vSphere with Tanzu](#).

Procedure

- 1 Copy the NVIDIA vGPU software Linux driver package, for example `NVIDIA-Linux-x86_64-version-grid.run`, to the guest VM.
- 2 Before attempting to run the driver installer, terminate all applications.
- 3 Start the NVIDIA vGPU driver installer.

```
sudo ./NVIDIA-Linux-x86_64-version-grid.run
```

- 4 Accept the NVIDIA software license agreement and select **Yes** to update the X configuration settings automatically.
- 5 Verify that the driver has been installed.

For example,

```
~$ nvidia-smi
Wed May 19 22:15:04 2021
+-----+
| NVIDIA-SMI 460.63      Driver Version: 460.63      CUDA Version: 11.2      |
+-----+-----+-----+-----+-----+-----+
| GPU   Name                Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan   Temp   Perf   Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|               |                    |          MIG M. |
+=====+=====+=====+=====+=====+=====+
|    0   GRID V100-4Q           On      | 00000000:02:00.0 Off |                  N/A |
| N/AN/AP0        N/A/  N/A|    304MiB /  4096MiB |      0%      Default |
|               |                    |          |                  N/A |
+-----+-----+-----+-----+-----+-----+

+-----+
| Processes:                                     |
|  GPU   GI    CI          PID    Type    Process name                        GPU Memory |
|          ID    ID                                   Usage                        |
+=====+=====+=====+=====+=====+=====+
| No running processes found                                     |
+-----+
```

Monitor Virtual Machines Available in vSphere with Tanzu

As a vSphere administrator, use the vSphere Client to monitor a VM that was deployed by DevOps in the Kubernetes environment.

You cannot manage the VM lifecycle from the vSphere Client.

Prerequisites

A DevOps engineer has deployed a VM. See [Deploy a Virtual Machine in vSphere with Tanzu](#).

Procedure

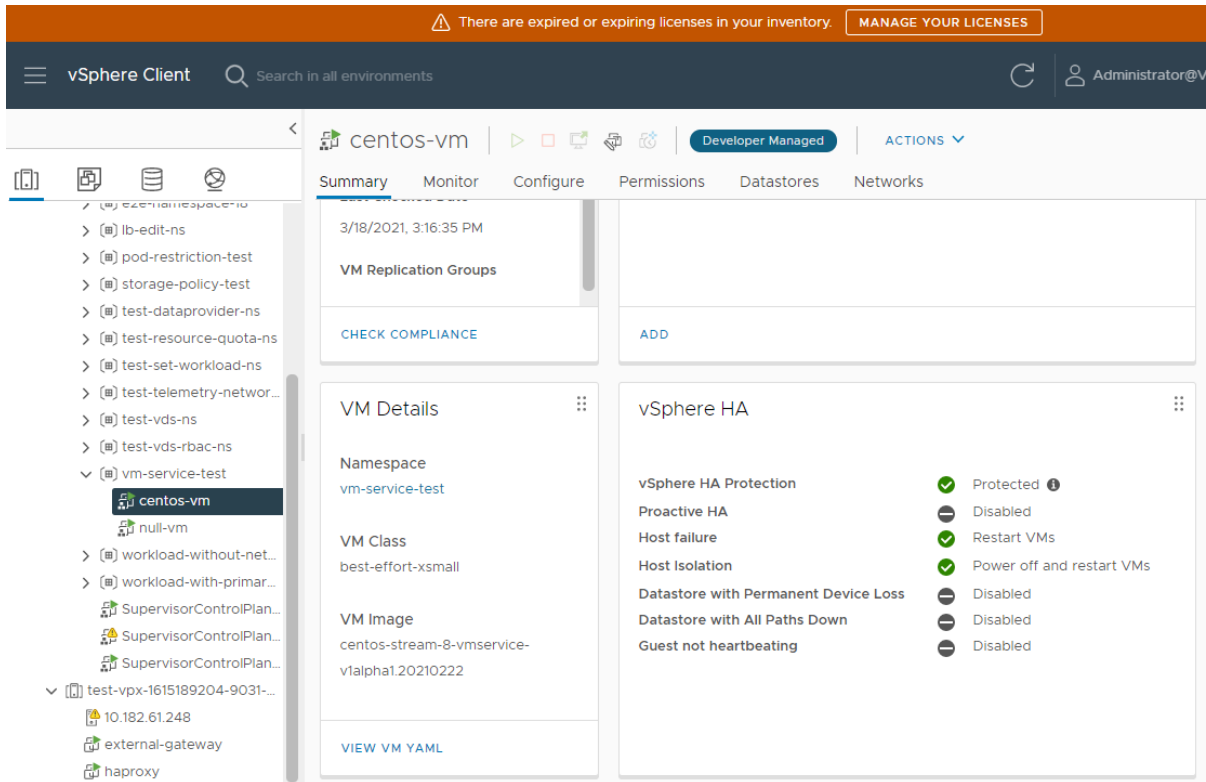
- 1 In the vSphere Client, navigate to the host cluster that has vSphere with Tanzu enabled.
- 2 Under **Namespaces**, expand the namespace where a VM was deployed.
- 3 Select the VM to view and click the **Summary** tab.

Make sure that you see the **Developer Managed** tag at the top of the **Summary** page.

The page displays information about the VM, including its guest OS and IP addresses.

The screenshot shows the vSphere Client interface. The left sidebar displays a tree view of namespaces, with 'vm-service-test' expanded and 'centos-vm' selected. The main panel shows the 'Summary' tab for 'centos-vm'. At the top, a 'Developer Managed' tag is visible. The VM status is 'Powered On'. Key information includes: Guest OS: CentOS 8 (64-bit), Compatibility: ESXi 7.0 and later (VM version 17), VMware Tools: Running, version:11328 (Guest Managed), DNS Name: centos-vm, IP Addresses: 10.182.59.181, Host: 10.182.51.8, and Managed By: WCP Service. On the right, resource usage is shown: CPU USAGE 167 MHz, MEMORY USAGE 20 MB, and STORAGE USAGE 3.83 GB. Below this, there are sections for 'VM Hardware', 'Related Objects' (listing Cluster, Host, and Namespace), 'Notes', and 'Custom Attributes'.

- 4 Click **Switch to New View** in the upper-right corner of the page to display additional details such as the VM class and VM image, as well as the namespace, where the VM runs.



Provisioning and Operating TKGS Clusters

13

vSphere with Tanzu provides convenient tooling and straightforward workflows for provisioning and operating Tanzu Kubernetes clusters. Refer to the procedures and examples to create and customize clusters with various configurations to meet your needs.

Read the following topics next:

- [Workflow for Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API](#)
- [Virtual Machine Classes for Tanzu Kubernetes Clusters](#)
- [Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API](#)
- [Provisioning Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API](#)
- [Delete a Tanzu Kubernetes Cluster](#)
- [Specify a Default Text Editor for Kubectl](#)
- [Operating Tanzu Kubernetes Clusters](#)

Workflow for Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API

You provision Tanzu Kubernetes clusters by invoking the Tanzu Kubernetes Grid Service declarative API using `kubectl` and a cluster specification defined in YAML. After you provision a cluster, you operate it and deploy workloads to it using `kubectl`.

This workflow supports the Tanzu Kubernetes Grid Service [TKGS v1alpha2 API for Provisioning Tanzu Kubernetes Clusters](#). If you are using the [Configuration Parameters for Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API](#), refer to that [Workflow for Provisioning Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API](#).

Prerequisites

Verify the completion of the following prerequisites before starting the workflow procedure:

- Install or update your environment to support the Tanzu Kubernetes Grid Service [TKGS v1alpha2 API for Provisioning Tanzu Kubernetes Clusters](#). Refer to the [Requirements for Using the TKGS v1alpha2 API](#) for details. The minimum Tanzu Kubernetes release that supports the [TKGS v1alpha2 API for Provisioning Tanzu Kubernetes Clusters](#) is `v1.21.2`. Refer to the [VMware Tanzu Kubernetes releases Release Notes](#) for details.

- Configure a vSphere Namespace for hosting Tanzu Kubernetes clusters. The namespace requires edit permissions for DevOps engineers and shared storage. See [Create and Configure a vSphere Namespace](#).
- Create a content library for Tanzu Kubernetes releases and synchronize the releases you want to use. See [Creating and Managing Content Libraries for Tanzu Kubernetes releases](#).
- Decide which default VM classes you want to use and if you require custom VM classes. See [Virtual Machine Classes for Tanzu Kubernetes Clusters](#).
- Associate the content library and the virtual machine classes with the vSphere Namespace. See [Configure a vSphere Namespace for Tanzu Kubernetes releases](#).

Procedure

- 1 Download and install the Kubernetes CLI Tools for vSphere.

For guidance see [Download and Install the Kubernetes CLI Tools for vSphere](#).

- 2 Authenticate with the Supervisor Cluster using the vSphere Plugin for kubectl.

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME
```

For guidance see [Connect to the Supervisor Cluster as a vCenter Single Sign-On User](#).

- 3 Verify successful login to the Supervisor Cluster.

You should see a message similar to the following:

```
Logged in successfully.

You have access to the following contexts:
  192.197.2.65
  tkgs-ns
```

Where `192.197.2.65` is the Supervisor Cluster context and `tkgs-ns` is the context for the vSphere Namespace where you plan to provision the Tanzu Kubernetes cluster.

- 4 Verify that the target vSphere Namespace is the current context.

```
kubectl config get-contexts
```

CURRENT	NAME	CLUSTER	AUTHINFO	NAMESPACE
	192.197.2.65	192.197.2.65	wcp:192.197.2.65:user@vsphere.local	
*	tkgs-ns	192.197.2.65	wcp:192.197.2.65:user@vsphere.local	tkgs-ns

If the target vSphere Namespace is not the current context, switch to it.

```
kubectl config use-context tkgs-ns
```

- 5 List the virtual machine class bindings that are available in the target vSphere Namespace.

```
kubectl get virtualmachineclassbindings
```

You can only use those VM classes that are bound to the target namespace. If you do not see any VM classes, check that the vSphere Namespace has the default VM classes added to it.

- 6 Get the available persistent volume storage classes.

```
kubectl describe storageclasses
```

- 7 List the available Tanzu Kubernetes releases.

You can use either of the following commands to perform this operation:

```
kubectl get tkr
```

```
kubectl get tanzukubernetesreleases
```

You can only use those releases that are returned by this command. If you do not see any releases, or the releases that you want, verify that you have synchronized the desired OVA files with the content library.

- 8 Craft the YAML file for provisioning your Tanzu Kubernetes cluster.

- a Review the [TKGS v1alpha2 API for Provisioning Tanzu Kubernetes Clusters](#) specification.

Note The TKG v1alpha1 API is deprecated and should not be used to create new TKG clusters. Use the TKG v1alpha2 API.

- b Start with one of the [Example YAML for Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API](#) for provisioning a cluster, either the [Example YAML for Provisioning a Default Tanzu Kubernetes Cluster](#) or [Example YAML for Provisioning a Custom Tanzu Kubernetes Cluster](#) depending on your requirements.
 - c Save your YAML file as `tkgs-cluster-1.yaml`, or similar.

- d Populate your YAML file based on your requirements and using the information you gleaned from the output of the preceding commands, including:
 - The name of the cluster, such as `tkgs-cluster-1`
 - The target vSphere Namespace, such as `tkgs-ns`
 - Bound VM classes, such as `guaranteed-medium` and `guaranteed-small`
 - Storage classes for cluster nodes and workloads, such as `vwt-storage-policy`
 - The number of control plane and worker nodes (replicas)
 - The Tanzu Kubernetes release specified by the TKR NAME string, such as `v1.23.8---vmware.3-tkg.1`
- e Customize your YAML file as needed. For example:
 - Specify a default persistent storage class for cluster nodes
 - Customize the cluster networking, including the CNI, pod and service CIDRs

The result of this step is a valid YAML for provisioning your TKGS cluster. For example:

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-1
  namespace: tkgs-ns
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.23.8---vmware.3-tkg.1
    nodePools:
      - name: worker-nodepool-a1
        replicas: 3
        vmClass: guaranteed-medium
        storageClass: vwt-storage-policy
        tkr:
          reference:
            name: v1.23.8---vmware.3-tkg.1
      - name: worker-nodepool-a2
        replicas: 2
        vmClass: guaranteed-small
        storageClass: vwt-storage-policy
        tkr:
          reference:
```

```

    name: v1.23.8---vmware.3-tkg.1
  settings:
    storage:
      defaultClass: vwt-storage-policy

```

Note The above example uses the default cluster networking, that is, the Antrea CNI and default CIDR ranges for cluster pods and services.

- 9 Provision the cluster by running the following `kubectl` command.

```
kubectl apply -f tkgs-cluster-1.yaml
```

Expected result:

```
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 created
```

- 10 Monitor the deployment of cluster nodes using `kubectl`.

```
kubectl get tanzukubernetesclusters
```

Initially the cluster is not ready because it is being provisioned.

NAME	CONTROL PLANE	WORKER	TKR NAME	AGE	READY	TKR
COMPATIBLE	UPDATES AVAILABLE					
tkgs-cluster-1	3	5	v1.23.8---vmware.3-tkg.1	2m4s	False	True

After a few minutes the READY status should be True.

NAME	CONTROL PLANE	WORKER	TKR NAME	AGE	READY	TKR
COMPATIBLE	UPDATES AVAILABLE					
tkgs-cluster-1	3	5	v1.23.8---vmware.3-tkg.1	13m	True	True

For additional guidance, see [Monitor Tanzu Kubernetes Cluster Status Using kubectl](#).

- 11 Monitor the deployment of cluster nodes using the vSphere Client.

In the vSphere **Hosts and Clusters** inventory, you should see the virtual machine nodes being deployed in the target vSphere Namespace.

For additional guidance, see [Monitor Tanzu Kubernetes Cluster Status Using the vSphere Client](#).

- 12 Run additional `kubectl` commands to verify cluster provisioning.

```
kubectl get tanzukubernetescluster,cluster-api,virtualmachinesetresourcepolicy,virtualmachineservice,virtualmachine
```

For additional guidance, see [Use Tanzu Kubernetes Cluster Operational Commands](#).

For troubleshooting, see [Troubleshooting Tanzu Kubernetes Clusters](#).

13 Using the vSphere Plugin for kubectl, log in to the cluster.

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME \
--tanzu-kubernetes-cluster-name CLUSTER-NAME \
--tanzu-kubernetes-cluster-namespace NAMESPACE-NAME
```

For example:

```
kubectl vsphere login --server=192.197.2.65 --vsphere-username user@vsphere.local \
--tanzu-kubernetes-cluster-name tkgs-cluster-1 --tanzu-kubernetes-cluster-namespace tkgs-ns
```

For additional guidance, see [Connect to a Tanzu Kubernetes Cluster as a vCenter Single Sign-On User](#).

14 Verify successful login to the Tanzu Kubernetes cluster.

You should see a message similar to the following.

```
Logged in successfully.

You have access to the following contexts:
  192.197.2.65
  tkgs-cluster-1
  tkgs-ns
```

Where 192.197.2.65 is the Supervisor Cluster context, tkgs-ns is the vSphere Namespace context, and tkgs-cluster-1 is the Tanzu Kubernetes cluster context.

15 List available cluster contexts using kubectl.

```
kubectl config get-contexts
```

For example:

CURRENT	NAME	CLUSTER	AUTHINFO
	192.197.2.65	192.197.2.65	wcp:192.197.2.65:administrator@vsphere.local
*	tkgs-cluster-1	192.197.2.67	wcp:192.197.2.67:administrator@vsphere.local
	tkgs-ns	192.197.2.65	wcp:192.197.2.65:administrator@vsphere.local

If necessary use `kubect config use-context tkgs-cluster-1` to switch to the Tanzu Kubernetes cluster so that it is the current context.

16 Verify cluster provisioning using the following kubectl command.

```
kubectl get nodes
```

For example:

NAME	STATUS	ROLES

AGE	VERSION		
tkgs-cluster-1-control-plane-6ln2h		Ready	control-plane,master
30m	v1.21.6+vmware.1		
tkgs-cluster-1-control-plane-6q67n		Ready	control-plane,master
33m	v1.21.6+vmware.1		
tkgs-cluster-1-control-plane-jw964		Ready	control-plane,master
37m	v1.21.6+vmware.1		
tkgs-cluster-1-worker-nodepool-a1-4vvkb-65494d66d8-h5fp8		Ready	<none>
32m	v1.21.6+vmware.1		
tkgs-cluster-1-worker-nodepool-a1-4vvkb-65494d66d8-q4g24		Ready	<none>
33m	v1.21.6+vmware.1		
tkgs-cluster-1-worker-nodepool-a1-4vvkb-65494d66d8-vdcn4		Ready	<none>
33m	v1.21.6+vmware.1		
tkgs-cluster-1-worker-nodepool-a2-2n22f-bd59d7b96-nh4dg		Ready	<none>
34m	v1.21.6+vmware.1		
tkgs-cluster-1-worker-nodepool-a2-2n22f-bd59d7b96-vvfmf		Ready	<none>
33m	v1.21.6+vmware.1		

17 Verify cluster provisioning using additional kubectl commands.

```
kubectl get namespaces

kubectl get pods -A

kubectl cluster-info

kubectl api-resources
```

18 Define appropriate pod security policy.

Tanzu Kubernetes clusters have the PodSecurityPolicy Admission Controller enabled by default. For guidance, see [Using Pod Security Policies with Tanzu Kubernetes Clusters](#).

Depending on the workload and user, you will need to create bindings for system-provided PodSecurityPolicy, or create custom PodSecurityPolicy. See [Example Role Bindings for Pod Security Policy](#).

19 Deploy an example workload and verify cluster creation.

For guidance, see [Deploy Workloads on Tanzu Kubernetes Clusters](#).

20 Operationalize the cluster by deploying TKG Extensions.

For guidance, see [Install Packages on Tanzu Kubernetes Clusters](#).

Virtual Machine Classes for Tanzu Kubernetes Clusters

To size Tanzu Kubernetes cluster nodes, you specify the virtual machine class. vSphere with Tanzu provides default classes, and you can create your own. To use a class, associate it with the target vSphere Namespace and reference the class in the manifest.

About Virtual Machine Classes

A virtual machine class is a request for resource reservations for processing power on the virtual machine (VM), including CPU and memory (RAM). For example, the VM class type named "guaranteed-large" reserves 4 CPU and 16 GB RAM. See [Default Virtual Machine Classes](#) for a list of default VM classes and their corresponding CPU and RAM reservations.

Note The VM disk size is set by the OVA template, not the VM class definition. For Tanzu Kubernetes releases, the disk size is 16GB. See [About Tanzu Kubernetes release Distributions](#).

There are two reservation types for VM classes: guaranteed and best effort. The guaranteed class fully reserves its configured resources. This means that for a given cluster the `spec.policies.resources.requests` matches the `spec.hardware` settings. The best effort class allows resources to be overcommitted. For production workloads it is recommended that you use the guaranteed VM class type.

Warning Because the best effort VM class type allows resources to be overcommitted, you can run out of resources if you have set limits on the vSphere Namespace where you are provisioning the Tanzu Kubernetes cluster. If contention occurs and the control plane is impacted, the cluster may stop running. For this reason you should always use the guaranteed VM class type for production clusters. If you cannot use the guaranteed VM class type for all production nodes, at a minimum you should use guaranteed for the control plane nodes.

Using Virtual Machine Classes

To use a virtual machine class with a Tanzu Kubernetes cluster, the VM class must be bound to the vSphere Namespace where the cluster is provisioned. To do this, you associate the class with the target namespace. See [Configure a vSphere Namespace for Tanzu Kubernetes releases](#).

To list the VM classes available in the target vSphere Namespace, use the command `kubectl get virtualmachineclassbinding`. To view all virtual machine classes present on the Supervisor Cluster, run the command `kubectl describe virtualmachineclasses`. Note, however, that because only bound classes can be used to provision a cluster, the latter command is informational-only. See [Workflow for Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API](#).

Note The requirement to associate VM classes with the vSphere Namespace only applies to new clusters. Existing Tanzu Kubernetes clusters using default VM classes continue to function without requiring namespace association.

Default Virtual Machine Classes

The table [Table 13-1. Default Virtual Machine Classes](#) lists default virtual machine class types that are used as deployment sizes for Tanzu Kubernetes cluster nodes.

To avoid overcommitting resources, production workloads should use the guaranteed class type. To avoid running out of memory, do not use the small or extra small class size for any worker node where you are deploying workloads in any environment (development, test, or production).

Table 13-1. Default Virtual Machine Classes

Class	CPU	Memory (GB)	Reserved CPU and Memory
guaranteed-8xlarge	32	128	Yes
best-effort-8xlarge	32	128	No
guaranteed-4xlarge	16	128	Yes
best-effort-4xlarge	16	128	No
guaranteed-2xlarge	8	64	Yes
best-effort-2xlarge	8	64	No
guaranteed-xlarge	4	32	Yes
best-effort-xlarge	4	32	No
guaranteed-large	4	16	Yes
best-effort-large	4	16	No
guaranteed-medium	2	8	Yes
best-effort-medium	2	8	No
guaranteed-small	2	4	Yes
best-effort-small	2	4	No
guaranteed-xsmall	2	2	Yes
best-effort-xsmall	2	2	No

Custom Virtual Machine Classes

vSphere with Tanzu supports custom virtual machine classes for use with Tanzu Kubernetes clusters. Once you have defined a custom VM class, you must associate it with the target vSphere Namespace before you can use it with a cluster. See [Create a VM Class in vSphere with Tanzu](#).

Editing Virtual Machine Classes

VM class definitions are not immutable. Any VM class can be [Edit or Delete a VM Class in vSphere with Tanzu](#), including the [Virtual Machine Classes for Tanzu Kubernetes Clusters](#). If a VM class is edited, existing Tanzu Kubernetes cluster nodes remain unaffected. New Tanzu Kubernetes clusters will use the modified class definition.

Caution If you edit a VM class that is in use by a Tanzu Kubernetes cluster, and then scale out that cluster, the new node(s) will use the edited class definition, but the existing nodes will use the initial class definition, resulting in a mismatch of classes.

Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API

This section describes how to provision Tanzu Kubernetes clusters using the TKGS v1alpha2 API.

Requirements for Using the TKGS v1alpha2 API

To use the Tanzu Kubernetes Grid Service v1alpha2 API for provisioning Tanzu Kubernetes clusters, adhere to the full list of requirements.

Requirements for Using the TKGS v1alpha2 API

The Tanzu Kubernetes Grid Service v1alpha2 API provides a robust set of enhancements for provisioning Tanzu Kubernetes clusters. For more information, see [TKGS v1alpha2 API for Provisioning Tanzu Kubernetes Clusters](#).

To take advantage of the new functionality provided by the Tanzu Kubernetes Grid Service v1alpha2 API, your environment must satisfy each of the following requirements.

Requirement	Reference
Workload Management is enabled with supported networking, either NSX-T Data Center or native vSphere vDS. Note A particular feature may require a specific type of networking. If so this is called out in the topic for that feature.	See Prerequisites for Configuring vSphere with Tanzu on a vSphere Cluster. See Enable Workload Management with NSX Networking. See Enable Workload Management with vSphere Networking.
The vCenter Server hosting Workload Management is updated to version 7 Update 3 or later.	Refer to the vCenter Server Update and Patch Releases release notes. For update instructions, see Upgrading the vCenter Server Appliance.
All ESXi hosts supporting the vCenter Server cluster where Workload Management is enabled are updated to version 7 Update 3 or later.	Refer to the ESXi Update and Patch Release Notes. For update instructions, see Upgrading ESXi Hosts.

Requirement	Reference
vSphere Namespaces is updated to v0.0.11 or later.	For release details, see the vSphere with Tanzu Release Notes. For update instructions, see Chapter 17 Updating the vSphere with Tanzu Environment.
Supervisor Cluster is updated to v1.21.0+vmware.wcp.2 or later.	For release details, see the vSphere with Tanzu Release Notes. For update instructions, see Update the Supervisor Cluster by Performing a vSphere Namespaces Update.
You must use Tanzu Kubernetes release v1.21.2---vmware.1-tkg.1.ee25d55 or later.	For release details, see Verify Tanzu Kubernetes Cluster Compatibility for Update. For instructions on provisioning new clusters, see Example YAML for Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API. For instructions on updating an existing cluster, see Updating a Tanzu Kubernetes Release After the Cluster Spec Is Converted to the TKGS v1alpha2 API.
CNI Considerations for Node Limits	The cluster specification setting <code>spec.settings.network.pods.cidrBlocks</code> defaults to 192.168.0.0/16. See TKGS v1alpha2 API for Provisioning Tanzu Kubernetes Clusters. If you customize, the minimum pods CIDR block size is /24. However, be cautious about restricting the <code>pods.cidrBlocks</code> subnet mask beyond /16. TKGS allocates to each cluster node a /24 subnet carved from the <code>pods.cidrBlocks</code>. This allocation is determined by the Kubernetes Controller Manager > NodeIPAMController parameter named NodeCIDRMaskSize which sets the subnet mask size for the Node CIDR in the cluster. The default node subnet mask is /24 for IPv4. Because each node in a cluster gets a /24 subnet from the <code>pods.cidrBlocks</code>, you can run out of node IP addresses if you use a subnet mask size that is too restrictive for the cluster you are provisioning. The following node limits apply to a Tanzu Kubernetes cluster provisioned with either the Antrea or Calico CNI. /16 == 150 nodes max (per ConfigMax) /17 == 128 nodes max /18 == 64 nodes max /19 == 32 nodes max /20 == 16 nodes max /21 == 8 nodes max /22 == 4 nodes max /23 == 2 nodes max /24 == 1 node max

TKGS v1alpha2 API for Provisioning Tanzu Kubernetes Clusters

The Tanzu Kubernetes Grid Service v1alpha2 API lets you declaratively provision Tanzu Kubernetes clusters. Refer to the list and description of all parameters and usage guidelines to create and customize your clusters.

Tanzu Kubernetes Grid Service v1alpha2 API Specification for Provisioning Tanzu Kubernetes Clusters

The YAML specification lists all available parameters for provisioning a Tanzu Kubernetes cluster using the Tanzu Kubernetes Grid Service v1alpha2 API.

Important A valid key name must consist only of alphanumeric characters, a dash (such as key-name), an underscore (such as KEY_NAME) or a dot (such as key.name). You cannot use the space character in a key name.

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: string
  namespace: string
spec:
  topology:
    controlPlane:
      replicas: int32
      vmClass: string
      storageClass: string
      volumes:
        - name: string
          mountPath: string
          capacity:
            storage: size in GiB
    tkr:
      reference:
        name: string
      nodeDrainTimeout: string
  nodePools:
  - name: string
    labels: map[string]string
    taints:
      - key: string
        value: string
        effect: string
        timeAdded: time
    replicas: int32
    vmClass: string
    storageClass: string
    volumes:
      - name: string
        mountPath: string
        capacity:
          storage: size in GiB
    tkr:
```

```

    reference:
      name: string
    nodeDrainTimeout: string
  settings:
    storage:
      classes: [string]
      defaultClass: string
    network:
      cni:
        name: string
      pods:
        cidrBlocks: [string]
      services:
        cidrBlocks: [string]
      serviceDomain: string
    proxy:
      httpProxy: string
      httpsProxy: string
      noProxy: [string]
    trust:
      additionalTrustedCAs:
        - name: string
          data: string

```

Annotated Tanzu Kubernetes Grid Service v1alpha2 API Specification for Provisioning Tanzu Kubernetes Clusters

The annotated YAML specification lists all available parameters for provisioning a Tanzu Kubernetes cluster using the Tanzu Kubernetes Grid Service v1alpha2 API with documentation for each field.

Note Currently all `tkr.reference.name` fields must match. In the future different Tanzu Kubernetes releases for node pools may be supported.

```

apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
#valid config key must consist of alphanumeric characters, '-', '_' or '.'
#metadata defines cluster information
metadata:
  #name for this Tanzu Kubernetes cluster
  name: string
  #namespace vSphere Namespace where to provision this cluster
  namespace: string
#spec defines cluster configuration
spec:
  #topology describes the number, purpose, organization
  #of nodes and the resources allocated for each
  #nodes are grouped into pools based on their purpose
  #`controlPlane` is special kind of a node pool
  #`nodePools` is for groups of worker nodes
  #each node pool is homogeneous: its nodes have the same
  #resource allocation and use the same storage
  topology:

```

```

#controlPlane defines the topology of the cluster
#controller, including the number of nodes and
#the resources allocated for each
#control plane must have an odd number of nodes
controlPlane:
  #replicas is the number of nodes in the pool
  #the control plane can have 1 or 3 nodes
  #defaults to 1 if `nil`
  replicas: int32
  #vmClass is the name of the VirtualMachineClass
  #which describes the virtual hardware settings
  #to be used for each node in the node pool
  #vmClass controls the CPU and memory available
  #to the node and the requests and limits on
  #those resources; to list available vm classes run
  #`kubectl describe virtualmachineclasses`
  vmClass: string
  #storageClass to be used for storage of the disks
  #which store the root filesystems of the nodes
  #to list available storage classes run
  #`kubectl describe storageclasses`
  storageClass: string
  #volumes is the optional set of PVCs to create
  #and attach to each node; use for high-churn
  #control plane components such as etcd
  volumes:
    #name of the PVC to be used as the suffix (node.name)
    - name: string
      #mountPath is the directory where the volume
      #device is mounted; takes the form /dir/path
      mountPath: string
      #capacity is the PVC capacity
      capacity:
        #storage to be used for the disk
        #volume; if not specified defaults to
        #`spec.controlPlane.storageClass`
        storage: size in GiB
      #tkr.reference.name is the TKR NAME
      #to be used by control plane nodes; supported
      #format is `v1.21.2---vmware.1-tkg.1.ee25d55`
      #currently all `tkr.reference.name` fields must match
      tkr:
        reference:
          name: string
      #nodeDrainTimeout is the total amount of time
      #the controller will spend draining a node
      #the default value is 0 which means the node is
      #drained without any time limit
      nodeDrainTimeout: string
  #nodePools is an array that describes a group of
  #worker nodes in the cluster with the same configuration
  nodePools:
    #name of the worker node pool
    #must be unique in the cluster
    - name: string

```

```

#labels are an optional map of string keys and values
#to organize and categorize objects
#propagated to the created nodes
labels: map[string]string
#taints specifies optional taints to register the
#Node API object with; user-defined taints are
#propagated to the created nodes
taints:
  #key is the taint key to be applied to a node
  - key: string
  #value is the taint value corresponding to the key
    value: string
  #effect is the effect of the taint on pods
  #that do not tolerate the taint; valid effects are
  #`NoSchedule`, `PreferNoSchedule`, `NoExecute`
    effect: string
  #timeAdded is the time when the taint was added
  #only written by the system for `NoExecute` taints
    timeAdded: time
#replicas is the number of nodes in the pool
#worker nodePool can have from 0 to 150 nodes
#value of `nil` means the field is not reconciled,
#allowing external services like autoscalers
#to choose the number of nodes for the nodePool
#by default CAPI's `MachineDeployment` will pick 1
#NOTE: a cluster provisioned with 0 worker nodes/nodepools
#is not assigned any load balancer services
replicas: int32
#vmClass is the name of the VirtualMachineClass
#which describes the virtual hardware settings
#to be used for each node in the pool
#vmClass controls the CPU and memory available
#to the node and the requests and limits on
#those resources; to list available vm classes run
#`kubectl describe virtualmachineclasses`
vmClass: string
#storageClass to be used for storage of the disks
#which store the root filesystems of the nodes
#to list available storage classes run
#`kubectl describe ns`
storageClass: string
#volumes is the optional set of PVCs to create
#and attach to each node for high-churn worker node
#components such as the container runtime
volumes:
  #name of this PVC to be used as the suffix (node.name)
  - name: string
    #mountPath is the directory where the volume
    #device is mounted; takes the form /dir/path
    mountPath: string
    #capacity is the PVC capacity
    capacity:
      #storage to be used for the disk
      #volume; if not specified defaults to
      #`topology.nodePools[*].storageClass`

```

```

    storage: size in GiB
    #tkr.reference.name points to the TKR NAME
    #to be used by `spec.topology.nodePools[*]` nodes; supported
    #format is `v1.21.2---vmware.1-tkg.1.ee25d55`
    #currently all `tkr.reference.name` fields must match
    tkr:
      reference:
        name: string
    #nodeDrainTimeout is the total amount of time
    #the controller will spend draining a node
    #the default value is 0 which means the node is
    #drained without any time limit
    nodeDrainTimeout: string
#settings are optional runtime configurations
#for the cluster, including persistent storage
#for pods and node network customizations
settings:
  #storage defines persistent volume (PV) storage entries
  #for container workloads; note that the storage used for
  #node disks is defined by `topology.controlPlane.storageClass`
  #and by `spec.topology.nodePools[*].storageClass`
  storage:
    #classes is a list of persistent volume (PV) storage
    #classes to expose for container workloads on the cluster
    #any class specified must be associated with the
    #vSphere Namespace where the cluster is provisioned
    #if omitted, all storage classes associated with the
    #namespace will be exposed in the cluster
    classes: [string]
    #defaultClass treats the named storage class as the default
    #for the cluster; because all namespaced storage classes
    #are exposed if specific `classes` are not named,
    #classes is not required to specify a defaultClass
    #many workloads, including TKG Extensions and Helm,
    #require a default storage class
    #if omitted, no default storage class is set
    defaultClass: string
#network defines custom networking for cluster workloads
network:
  #cni identifies the CNI plugin for the cluster
  #use to override the default CNI set in the
  #tkgservicesonfiguration spec, or when customizing
  #network settings for the default CNI
  cni:
    #name is the name of the CNI plugin to use; supported
    #values are `antrea`, `calico`, `antrea-nsx-routed`
    name: string
  #pods configures custom networks for pods
  #defaults to 192.168.0.0/16 if CNI is `antrea` or `calico`
  #defaults to empty if CNI is `antrea-nsx-routed`
  #custom subnet size must equal or exceed /24
  #use caution before setting CIDR range other than /16
  #cannot overlap with Supervisor Cluster workload network
  pods:
    #cidrBlocks is an array of network ranges; supplying

```

```

#multiple ranges may not be supported by all CNI plugins
cidrBlocks: [string]
#services configures custom network for services
#defaults to 10.96.0.0/12
#cannot overlap with Supervisor Cluster workload network
services:
  #cidrBlocks is an array of network ranges; supplying
  #multiple ranges many not be supported by all CNI plugins
  cidrBlocks: [string]
#serviceDomain specifies the service domain for the cluster
#defaults to `cluster.local`
serviceDomain: string
#proxy configures proxy server to be used inside the cluster
#if omitted no proxy is configured
proxy:
  #httpProxy is the proxy URI for HTTP connections
  #to endpoints outside the cluster
  #takes form `http://<user>:<pwd>@<ip>:<port>`
  httpProxy: string
  #httpsProxy is the proxy URL for HTTPS connections
  #to endpoints outside the cluster
  #takes the from `http://<user>:<pwd>@<ip>:<port>`
  httpsProxy: string
  #noProxy is the list of destination domain names, domains,
  #IP addresses, and other network CIDRs to exclude from proxying
  #must include Supervisor Cluster Pod, Egress, Ingress CIDRs
  noProxy: [string]
#trust configures additional certificates for the cluster
#if omitted no additional certificate is configured
trust:
  #additionalTrustedCAs are additional trusted certificates
  #can be additional CAs or end certificates
  additionalTrustedCAs:
    #name is the name of the additional trusted certificate
    #must match the name used in the filename
    - name: string
      #data holds the contents of the additional trusted cert
      #PEM Public Certificate data encoded as base64 string
      #such as `LS0tLS1C...LS0tCg==` where "..." is the
      #middle section of the long base64 string
      data: string

```

Example YAML for Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API

The TKGS API provides intelligent defaults and an array of options for customizing Tanzu Kubernetes clusters. Refer to the examples to provision clusters of various types with different configurations and customizations to meet your needs.

Example YAML for Provisioning a Default Tanzu Kubernetes Cluster

The following example YAML provisions a default Tanzu Kubernetes cluster using the TKGS [TKGS v1alpha2 API for Provisioning Tanzu Kubernetes Clusters](#). This example YAML uses all available defaults and thus represents the minimal configuration required to provision a cluster.

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-v2-cluster-default
  namespace: tkgs-cluster-ns
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.23.8---vmware.3-tkg.1
    nodePools:
    - name: worker-nodepool-a1
      replicas: 3
      vmClass: guaranteed-large
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.23.8---vmware.3-tkg.1
```

Example YAML for Provisioning a Custom Tanzu Kubernetes Cluster

The following example YAML provisions a custom Tanzu Kubernetes cluster using the Tanzu Kubernetes Grid Service [TKGS v1alpha2 API for Provisioning Tanzu Kubernetes Clusters](#). This example YAML specifies separate volumes for high-churn node components such as `etcd` and `containerd` and customizes certain storage and network settings.

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-v2-cluster-custom
  namespace: tkgs-cluster-ns
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.23.8---vmware.3-tkg.1
    nodePools:
    - name: worker-nodepool-a1
      replicas: 3
```

```

vmClass: guaranteed-large
storageClass: vwt-storage-policy
volumes:
  - name: containerd
    mountPath: /var/lib/containerd
    capacity:
      storage: 16Gi
tkr:
  reference:
    name: v1.23.8---vmware.3-tkg.1
- name: worker-nodepool-a2
  replicas: 2
  vmClass: guaranteed-medium
  storageClass: vwt-storage-policy
  tkr:
    reference:
      name: v1.23.8---vmware.3-tkg.1
- name: worker-nodepool-a3
  replicas: 1
  vmClass: guaranteed-small
  storageClass: vwt-storage-policy
  tkr:
    reference:
      name: v1.23.8---vmware.3-tkg.1
settings:
  storage:
    defaultClass: vwt-storage-policy
  network:
    cni:
      name: antrea
    services:
      cidrBlocks: ["198.53.100.0/16"]
    pods:
      cidrBlocks: ["192.0.5.0/16"]
  serviceDomain: cluster.local
  proxy:
    httpProxy: http://<user>:<pwd>@<ip>:<port>
    httpsProxy: http://<user>:<pwd>@<ip>:<port>
    noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]
  trust:
    additionalTrustedCAs:
      - name: CompanyInternalCA-1
        data: LS0tLS1C...LS0tCg==
      - name: CompanyInternalCA-2
        data: MTLtMT1C...MT0tPg==

```

Updating a Tanzu Kubernetes Release After the Cluster Spec Is Converted to the TKGS v1alpha2 API

After a Tanzu Kubernetes cluster specification has been auto-converted to the v1alpha2 API format, to perform a rolling update of the Tanzu Kubernetes cluster, which is typically done by changing the Tanzu Kubernetes release version, you may need to do some pre-processing of the cluster spec to avoid errors.

Auto-Conversion of Cluster Specs

To update your vSphere with Tanzu environment to the Tanzu Kubernetes Grid Service v1alpha2 API, you update the Supervisor Cluster where the service runs.

Once the Tanzu Kubernetes Grid Service is running the v1alpha2 API, the system automatically converts all existing Tanzu Kubernetes cluster specifications from the v1alpha1 format to the v1alpha2 format. During the auto-conversion process, the system creates and populates the expected fields for each cluster manifest. [API Deprecations and Additions](#) lists the cluster specification fields that are new and deprecated in the v1alpha2 API.

To update the Tanzu Kubernetes release version for a cluster whose manifest has been auto-converted to the v1alpha2 format, you need to perform some manual pre-processing to avoid errors. [Cluster Update Examples](#) lists various options.

API Deprecations and Additions

Note The TKG v1alpha1 API is deprecated and should not be used to create new TKG clusters. Use the TKG v1alpha2 API instead. See [Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API](#).

The table lists the cluster specification settings that are deprecated in the v1alpha2 API and replaced by new settings.

Deprecated Settings	New Settings	Comments
spec.distribution.version spec.distribution.fullVersion	spec.topology.controlPlane.tkr.reference.name spec.topology.nodePools[*].tkr.reference.name	Must use the TKR NAME format. See the examples.
spec.topology.workers	spec.topology.nodePools[*]	In a converted cluster, the block spec.topology.workers becomes spec.topology.nodePools[0]. The first entry in the nodePools list is name: workers.
spec.topology.controlPlane.count spec.topology.workers.count	spec.topology.controlPlane.replicas spec.topology.nodePools[*].replicas	count is replaced by replicas
spec.topology.controlPlane.class spec.topology.workers.class	spec.topology.controlPlane.vmClasses spec.topology.nodePools[*].vmClasses	class is replaced by vmClass
N/A	spec.topology.nodePools[*].labels	Optional key-pair values to organize and categorize objects; labels are propagated to the created nodes
N/A	spec.topology.nodePools[*].taints	Optional taints to register the nodes with; user-defined taints are propagated to the created nodes

TKR NAME Format Is Required

In addition to the `spec.distribution.version` fields being deprecated, the DISTRIBUTION format for specifying the Tanzu Kubernetes release version is not supported. This means that you cannot use the following string formats to reference the target release: `1.21.2+vmware.1-tkg.1.ee25d55`, `1.21.2`, and `1.21`.

When referencing the Tanzu Kubernetes release version in a v1alpha2 API cluster spec, you must use the TKR NAME format, not the deprecated DISTRIBUTION format. Although the deprecated format is displayed in the UPDATES AVAILABLE column, the only supported format is the one listed in the TKR NAME column.

```
kubectl get tanzukubernetescluster
```

NAMESPACE	NAME	CONTROL PLANE	WORKER	TKR	UPDATES AVAILABLE
NAME	AGE	READY	TKR COMPATIBLE		
tkgs-cluster-1	test-cluster	3	3	v1.21.2---vmware.1-tkg.1.ee25d55	
	38h	True	True	[1.21.2+vmware.1-tkg.1.ee25d55]	

Use kubectl edit to Update a Cluster Spec

If you need to make edits to a cluster specification so that it conforms with the TKGS v1alpha2 API, use the `kubectl edit` method. Do not attempt to use the `kubectl patch` method for this type of update. See [Methods for Editing the Cluster Manifest](#). To configure `kubectl` with an editor, see [Specify a Default Text Editor for Kubectl](#).

Cluster Update Examples

Because changing the `spec.distribution.version` is the most common way to trigger a rolling update of the cluster (see [Update Tanzu Kubernetes Clusters](#)), and this field is deprecated in the v1alpha2 API, there are some considerations to be aware of and some pre-processing recommendations to follow to avoid potential cluster update problems.

The following examples demonstrate how to update the version of a Tanzu Kubernetes cluster that was provisioned using the v1alpha1 API to a system that is running the v1alpha2 API.

Cluster Upgrade Example 1: Use a Single TKR NAME Reference in the Control Plane

The recommended approach is to remove all `nodePools[*].tkr.reference.name` blocks from the converted spec and update the `controlPlane.tkr.reference.name` with the TKR NAME of the target release. In this case the same Tanzu Kubernetes release is propagated to all `nodePools[*]` nodes.

In the future, the Tanzu Kubernetes release versions can be different across `controlPlane` and `nodePools[*]`. Currently, however, all releases in a cluster must match so putting a single TKR NAME reference in `controlPlane` is sufficient.

For example:

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-update-example1
  namespace: tkgs-cluster-ns
spec:
  settings:
    network:
      cni:
        name: antrea
    pods:
      cidrBlocks:
        - 192.0.2.0/16
      serviceDomain: cluster.local
    services:
      cidrBlocks:
        - 198.51.100.0/12
  topology:
    controlPlane:
      replicas: 3
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
          vmClass: best-effort-medium
    nodePools:
      - name: workers
        replicas: 3
        storageClass: vwt-storage-policy
        vmClass: best-effort-medium
```

Cluster Upgrade Example 2: Use a TKR NAME Reference for Each Node Pool

The second example is to put the TKR NAME in the `tkr.reference.name` block for both the `controlPlane` and `nodePools[*]` topologies.

This approach has the advantage of being ready for future releases when the Tanzu Kubernetes release can be different across node pools. Currently, they must match.

For example:

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-update-example2
  namespace: tkgs-cluster-ns
spec:
  settings:
    network:
      cni:
        name: antrea
    pods:
```

```

    cidrBlocks:
      - 192.0.2.0/16
    serviceDomain: cluster.local
    services:
      cidrBlocks:
        - 198.51.100.0/12
  topology:
    controlPlane:
      replicas: 3
      storageClass: vwt-storage-policy
      vmClass: best-effort-medium
      tkr:
        reference:
          name: v1.21.2---vmware.1-tkg.1.ee25d55
    nodePools:
      - name: workers
        replicas: 3
        storageClass: vwt-storage-policy
        vmClass: best-effort-medium
        tkr:
          reference:
            name: v1.21.2---vmware.1-tkg.1.ee25d55

```

Cluster Upgrade Example 3: Use Deprecated Distribution Fields

The final option is to use the deprecated fields `spec.distribution.fullVersion` and `spec.distribution.version`, and manually remove all `tkr.reference.name` blocks. You must include both fields with one using the TKR NAME format and the other nulled out. Version shortcuts such as `v1.21.2` and `v1.21` are not supported.

Note For the Tanzu Kubernetes release on Ubuntu, the use of `spec.distribution.version` is not supported.

The following example uses the `fullVersion` with the TKR NAME and a null (empty) value in the `version` field. All `tkr.reference.name` entries are removed.

```

apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-update-example3a
  namespace: tkgs-cluster-ns
spec:
  distribution:
    fullVersion: v1.21.2---vmware.1-tkg.1.ee25d55
    version: ""
  settings:
    network:
      cni:
        name: antrea
    pods:
      cidrBlocks:
        - 192.0.2.0/16
      serviceDomain: cluster.local

```

```

    services:
      cidrBlocks:
        - 198.51.100.0/12
  topology:
    controlPlane:
      replicas: 3
      storageClass: vwt-storage-policy
      vmClass: best-effort-medium
    nodePools:
      - name: workers
        replicas: 3
        storageClass: vwt-storage-policy
        vmClass: best-effort-medium

```

Alternatively you can use the `version` field with the TKR NAME and a null (empty) value in the `fullVersion` field. Even though you are using the `version` field, version shortcuts are not supported. All `tkr.reference.name` entries are removed.

```

apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-update-example3b
  namespace: tkgs-cluster-ns
spec:
  distribution:
    fullVersion: ""
    version: v1.21.2---vmware.1-tkg.1.ee25d55
  settings:
    network:
      cni:
        name: antrea
      pods:
        cidrBlocks:
          - 192.0.2.0/16
        serviceDomain: cluster.local
      services:
        cidrBlocks:
          - 198.51.100.0/12
  topology:
    controlPlane:
      replicas: 3
      storageClass: vwt-storage-policy
      vmClass: best-effort-medium
    nodePools:
      - name: workers
        replicas: 3
        storageClass: vwt-storage-policy
        vmClass: best-effort-medium

```

Configuring a Tanzu Kubernetes Cluster with a Routable Pod Network Using the v1alpha2 API

You can configure a Tanzu Kubernetes cluster to use routable pod networking by specifying the `antrea-nsx-routed` as the CNI for your cluster.

Introducing Routable Pod Networking

The [Kubernetes network model](#) requires that a pod in the node network of a cluster be able to communicate with all pods on all nodes in the same cluster without network address translation (NAT). To satisfy this requirement, each Kubernetes pod is given an IP address that is allocated from a dedicated pods network.

When you provision a Tanzu Kubernetes cluster using the `antrea` or `calico` CNI plugins, the system creates the default pods network `192.168.0.0/16`. This subnet is a private address space that is only unique within the cluster and not routable on the internet. Although you can customize the `network.pods.cidrBlocks`, the pods network cannot be routable using these CNI plugins. For more information, see [TKGS v1alpha2 API for Provisioning Tanzu Kubernetes Clusters](#).

The Tanzu Kubernetes Grid Service v1alpha2 API supports routable pod networking using the `antrea-nsx-routed` CNI plugin. This network interface is a customized Antrea plugin configured to support routable pod networks for Tanzu Kubernetes clusters. In the cluster specification, the pods CIDR blocks field must be explicitly null so that IP address management (IPAM) is handled by the Supervisor Cluster.

Enabling routable pod networking lets pods be directly addressed from a client external to the cluster. In addition, pod IP addressees are preserved so external network services and servers can identify the source pods and apply policies based on IP addresses. Supported traffic patterns including the following:

- Traffic is allowed between a Tanzu Kubernetes cluster pod and a vSphere Pod in the same vSphere Namespace.
- Traffic is dropped between a Tanzu Kubernetes cluster pod and a vSphere Pod in different vSphere Namespaces.
- Supervisor Cluster control plane nodes can reach Tanzu Kubernetes cluster pods.
- Tanzu Kubernetes cluster pods can reach the external network.
- External network cannot reach Tanzu Kubernetes cluster pods. Traffic is dropped by distributed firewall (DFW) isolation rules on the Tanzu Kubernetes cluster nodes.

System Requirements for Routable Pods

Routable pods networking requires the Supervisor Cluster to be configured with NSX-T Data Center. You cannot use routable pods with native vSphere vDS networking.

Routable pods requires the Tanzu Kubernetes Grid Service v1alpha2 API. See [Requirements for Using the TKGS v1alpha2 API](#).

NSX-T Configuration Requirements for Routable Pods

Other than the base requirements, there is no special NSX-T configuration required to use routable pods networks with Tanzu Kubernetes clusters. A vSphere with Tanzu environment running vSphere U3 with NSX-T includes the NCP version to support routable pods networks. There is no extra NSX-T configuration needed.

NCP creates an IP Pool for the routable pods network from one of two sources:

- If the Workload Network is configured with a Namespace Network, NCP will create one or more IP pools from the IP blocks specified for this Namespace Network.
- If there is no Namespace Network specified for the Workload Network, NCP will create one or more IP pools from the Supervisor Cluster Pod CIDR.

For more information, see [Add Workload Networks to a Supervisor Cluster Configured with VDS Networking](#) and [Change Workload Network Settings on a Supervisor Cluster Configured with NSX-T Data Center](#).

Supervisor Cluster Configuration Requirements for Routable Pods

Other than the base requirements, there is no special Supervisor Cluster configuration required to use routable pods networks with Tanzu Kubernetes clusters.

If routable pods networking is enabled as described below, the Tanzu Kubernetes cluster pods CIDR is allocated from the IP pool created from the Namespace Network or, if none, from the Supervisor Cluster Pod CIDR.

You must make sure that the Supervisor Cluster Services CIDR which allocates the IP addresses for cluster nodes does not overlap with the Namespace Network CIDR or the with Supervisor Cluster Pod CIDR.

Example Cluster Configuration for Routable Pods

The following example YAML shows how to configure a cluster with a routable pods network. is a custom configuration for invoking the Tanzu Kubernetes Grid Service and provision a Tanzu Kubernetes cluster using the v1alpha2 API.

The cluster spec declares `antrea-nsx-routed` as the CNI to enable routable pods networking. When the CNI is `antrea-nsx-routed` is specified, the `pod CIDRBlock` field must be empty. If `antrea-nsx-routed` is specified, cluster provisioning will fail if NSX-T networking is not being used.

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-v2-cluster-routable-pods
  namespace: tkgs-cluster-ns
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
```

```

storageClass: vwt-storage-policy
tkr:
  reference:
    name: v1.21.2---vmware.1-tkg.1.ee25d55
nodePools:
- name: worker-nodepool-a1
  replicas: 3
  vmClass: guaranteed-large
  storageClass: vwt-storage-policy
  tkr:
    reference:
      name: v1.21.2---vmware.1-tkg.1.ee25d55
settings:
  storage:
    defaultClass: vwt-storage-policy
  network:
    #`antrea-nsx-routed` is the required CNI
    #for routable pods
    cni:
      name: antrea-nsx-routed
    services:
      cidrBlocks: ["10.97.0.0/24"]
      serviceDomain: tanzukubernetescluster.local
      #`pods.cidrBlocks` value must be empty
      #when `antrea-nsx-routed` is the CNI
    pods:
      cidrBlocks:

```

Configuration Parameters for the TKGS v1alpha2 API

You can customize the Tanzu Kubernetes Grid Service with global settings for key features, including the container network interface (CNI), proxy server, and TLS certificates. Be aware of trade-offs and considerations when implementing global versus per-cluster functionality.

TkgServiceConfiguration v1alpha2 Specification

The `TkgServiceConfiguration` specification provides fields for configuring the Tanzu Kubernetes Grid Service instance.

Important A valid key name must consist only of alphanumeric characters, a dash (such as key-name), an underscore (such as KEY_NAME) or a dot (such as key.name). You cannot use the space character in a key name.

```

apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration-spec
spec:
  defaultCNI: string
  proxy:
    httpProxy: string
    httpsProxy: string

```



```

    noProxy: [string]
  trust:
    additionalTrustedCAs:
      - name: string
        data: string
  defaultNodeDrainTimeout: time

```

Caution Configuring the Tanzu Kubernetes Grid Service is a global operation. Any change you make to the `TkgServiceConfiguration` specification applies to all Tanzu Kubernetes clusters provisioned by that service. If a rolling update is initiated, either manually or by upgrade, clusters are updated by the changed service specification.

Annotated TkgServiceConfiguration v1alpha2 Specification

The following YAML lists and describes the configurable fields for each of the `TkgServiceConfiguration` specification parameters. For examples, see [Examples for Configuring the Tanzu Kubernetes Grid Service v1alpha1 API](#).

```

apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TkgServiceConfiguration
#valid config key must consist of alphanumeric characters, '-', '_' or '.'
metadata:
  name: tkg-service-configuration-spec
spec:
  #defaultCNI is the default CNI for all Tanzu Kubernetes
  #clusters to use unless overridden on a per-cluster basis
  #supported values are antrea, calico, antrea-nsx-routed
  #defaults to antrea
  defaultCNI: string
  #proxy configures a proxy server to be used inside all
  #clusters provisioned by this TKGS instance
  #if implemented all fields are required
  #if omitted no proxy is configured
  proxy:
    #httpProxy is the proxy URI for HTTP connections
    #to endpoints outside the clusters
    #takes the form http://<user>:<pwd>@<ip>:<port>
    httpProxy: string
    #httpsProxy is the proxy URI for HTTPS connections
    #to endpoints outside the clusters
    #takes the form http://<user>:<pwd>@<ip>:<port>
    httpsProxy: string
    #noProxy is the list of destination domain names, domains,
    #IP addresses, and other network CIDRs to exclude from proxying
    #must include Supervisor Cluster Pod, Egress, Ingress CIDRs
    noProxy: [string]
  #trust configures additional trusted certificates
  #for the clusters provisioned by this TKGS instance
  #if omitted no additional certificate is configured
  trust:
    #additionalTrustedCAs are additional trusted certificates
    #can be additional CAs or end certificates
    additionalTrustedCAs:

```

```

#name is the name of the additional trusted certificate
#must match the name used in the filename
- name: string
  #data holds the contents of the additional trusted cert
  #PEM Public Certificate data encoded as a base64 string
  data: string
#defaultNodeDrainTimeout is the total amount of time the
#controller spends draining a node; default is undefined
#which is the value of 0, meaning the node is drained
#without any time limitations; note that `nodeDrainTimeout`
#is different from `kubectl drain --timeout`
defaultNodeDrainTimeout: time

```

Proxy Server Configuration Requirements

If configured the proxy server will be used by the cluster for outbound HTTP and HTTPS traffic. Keep in mind the following requirements for configuring a proxy server for Tanzu Kubernetes clusters.

- The required `proxy` parameters are `httpProxy`, `httpsProxy`, and `noProxy`. If you add the `proxy` stanza, all three fields are mandatory.
- You can connect to the proxy server using HTTP. HTTPS connections are not supported.
- You obtain the required `spec.proxy.noProxy` values from the **Workload Network**. You must not proxy the **Namespace Network** (formerly called Pods CIDRs), **Ingress** (formerly called Ingress CIDRs), and **Egress** (formerly called Egress CIDRs) by including them in the `noProxy` field. Refer to the example images below.
- You do not need to include the Services CIDR in the `noProxy` field. Tanzu Kubernetes clusters do not interact with this subnet.
- You do not need to include the `network.services.cidrBlocks` and the `network.pods.cidrBlocks` values from the Tanzu Kubernetes cluster specification in the `noProxy` field. These subnets are automatically not proxied for you.
- You do not need to include `localhost` and `127.0.0.1` in the `noProxy` field. The endpoints are automatically not proxied for you.

The screenshot displays the 'Compute-Cluster' configuration page in the vSphere with Tanzu interface. The left sidebar contains a navigation menu with categories like Licensing, vSphere Cluster Services, Supervisor Cluster, vSphere Services, TKG Service, and vSAN. The 'Network' option under the Supervisor Cluster is selected. The main panel shows the 'Network' configuration for the Supervisor Cluster, with a green arrow pointing to the 'Workload Network' section. The configuration includes settings for the vSphere Distributed Switch, Edge Cluster, DNS Server(s), Services CIDR, Tier-0 Gateway, NAT Mode, and Namespace Network. The Namespace Network is highlighted in yellow, showing a /18 CIDR and an 'EDIT' button. Below it, the Namespace subnet prefix is /28. At the bottom, the Ingress and Egress settings are also highlighted in yellow, showing /26 CIDRs and 'EDIT' buttons.

Compute-Cluster | ACTIONS

Summary Monitor **Configure** Permissions Hosts VMs Namespaces Datastores Networks Updates

VM Overrides
I/O Filters
Host Options
Host Profile

Licensing ▾
vSAN Cluster
Supervisor Cluster
Trust Authority
Alarm Definitions
Scheduled Tasks

vSphere Cluster Ser... ▾
Datastores

Supervisor Cluster ▾
General
Network
Storage
Certificates
Image Registry

vSphere Services ▾
Overview

TKG Service ▾
Default CNI
Tanzu Mission Control

vSAN ▾
Services
Disk Management
Fault Domains
Remote Datastores

Network

Below are the network settings for supporting namespaces on this Supervisor Cluster.

> Management Network

▾ **Workload Network**

vSphere Namespaces uses the Workload Network to allow you to reach the user workloads. You can assign a vSphere Distributed Switch to the Workload Network.

vSphere Distributed Switch ⓘ DSwitch

Edge Cluster ⓘ ☐

DNS Server(s) ⓘ EDIT

Services CIDR ⓘ /19

Tier-0 Gateway ⓘ Tier-0_VWK

NAT Mode ⓘ ☒ Enabled

Namespace Network ⓘ [redacted] /18 EDIT

Namespace subnet prefix ⓘ /28

Ingress ⓘ [redacted] /26 EDIT

Egress ⓘ [redacted] /26 EDIT

compute-cluster ACTIONS ▾

Summary Monitor **Configure** Permissions Hosts VMs Namespaces Datastores Network

Services ▾

- vSphere DRS
- vSphere Availabili...

Configuration ▾

- Quickstart
- General
- Key Provider
- VMware EVC
- VM/Host Groups
- VM/Host Rules
- VM Overrides
- I/O Filters
- Host Options
- Host Profile

Licensing ▾

- vSAN Cluster
- Supervisor Cluster
- Trust Authority
- Alarm Definitions
- Scheduled Tasks

Namespaces ▾

- General
- Network**
- Storage
- Certificates
- Image Registry

vSAN ▾

- Services
- Disk Management
- Fault Domains
- Datastore Sharing

Supervisor Ser... ▾

Network

Below are the network settings for supporting namespaces on this cluster.

> Management Network

▾ Workload Network

vSphere Distributed Switch	wcp_vds_1	
Edge Cluster	EDGECLUSTER1	
DNS Servers	10.20.145.1	EDIT
Pod CIDRs	10.246.0.0/16	EDIT
Services CIDR	10.94.0.0/12	
Ingress CIDRs	192.168.144.0/20	EDIT
Egress CIDRs	192.168.128.0/20	EDIT

When To Use Global or Per-Cluster Configuration Options

The `TkgServiceConfiguration` is a global specification that impacts all Tanzu Kubernetes clusters provisioned by the Tanzu Kubernetes Grid Service instance.

Before editing the `TkgServiceConfiguration` specification, be aware of the per-cluster alternatives that might satisfy your use case instead of a global configuration.

Table 13-2. Global vs. Per-Cluster Configuration Options

Setting	Global Option	Per-Cluster Option
Default CNI	Edit the <code>TkgServiceConfiguration</code> spec. See Examples for Configuring the Tanzu Kubernetes Grid Service v1alpha1 API .	Specify the CNI in the cluster specification. For example, Antrea is the default CNI. To use Calico, specify it in the cluster YAML. See Examples for Provisioning Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API
Proxy Server	Edit the <code>TkgServiceConfiguration</code> spec. See Examples for Configuring the Tanzu Kubernetes Grid Service v1alpha1 API .	Include the proxy server configuration parameters in the cluster spec. See Examples for Provisioning Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API .
Trust Certificates	Edit the <code>TkgServiceConfiguration</code> spec. There are two use cases: configuring an external container registry and certificate-based proxy configuration. See Examples for Configuring the Tanzu Kubernetes Grid Service v1alpha1 API	Yes, you can include custom certificates on a per-cluster basis or override the globally-set <code>trust</code> settings in the cluster specification. See Examples for Provisioning Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API .

Note If a global proxy is configured on the `TkgServiceConfiguration`, that proxy information is propagated to the cluster manifest after the initial deployment of the cluster. The global proxy configuration is added to the cluster manifest only if there is no proxy configuration fields present when creating the cluster. In other words, per-cluster configuration takes precedence and will overwrite a global proxy configuration. For more information, see [Configuration Parameters for the Tanzu Kubernetes Grid Service v1alpha1 API](#).

Before editing the `TkgServiceConfiguration` specification, be aware of the ramifications of applying the setting at the global level.

Field	Applied	Impact on Existing Clusters If Added/ Changed	Per-Cluster Overriding on Cluster Creation	Per-Cluster Overriding on Cluster Update
defaultCNI	Globally	None	Yes, you can override the global setting on cluster creation	No, you cannot change the CNI for an existing cluster; if you used the globally set default CNI on cluster creation, it cannot be changed
proxy	Globally	None	Yes, you can override the global setting on cluster creation	Yes, with U2+, you can override the global setting on cluster update
trust	Globally	None	Yes, you can override the global setting on cluster creation	Yes, with U2+, you can override the global setting on cluster update

Propagating Global Configuration Changes to Existing Clusters

Settings made at the global level in the `TkgServiceConfiguration` may not be automatically propagated to existing clusters. For example, if you make changes to either the `proxy` or the `trust` settings in the `TkgServiceConfiguration`, such changes may not affect clusters that are already provisioned.

To manually propagate a global change to an existing cluster, you must patch the Tanzu Kubernetes cluster to make the cluster inherit the changes made to the `TkgServiceConfiguration`.

For example:

```
kubectl patch tkc <CLUSTER_NAME> -n <NAMESPACE> --type merge -p '{"spec":{"settings":{"network":{"proxy": null}}}}'
```

```
kubectl patch tkc <CLUSTER_NAME> -n <NAMESPACE> --type merge -p '{"spec":{"settings":{"network":{"trust": null}}}}'
```

Examples for Configuring the TKGS Instance Using the v1alpha2 API

Refer to the examples to customize the Tanzu Kubernetes Grid Service v1alpha2 API with global configuration settings for the container network interface, proxy server, and TLS certificates.

Configuring the Tanzu Kubernetes Grid Service Using the v1alpha2 API

You can customize the Tanzu Kubernetes Grid Service by changing the default CNI, adding a global proxy server, and adding trusted certificates. See [Configuration Parameters for the Tanzu Kubernetes Grid Service v1alpha1 API](#).

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-v2-configuration-example
spec:
  defaultCNI: antrea
  proxy:
    #supported format is `http://<user>:<pwd>@<ip>:<port>`
    httpProxy: http://admin:PaSsWoRd@10.66.100.22:80
    httpsProxy: http://admin:PaSsWoRd@10.66.100.22:80
    noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]
  trust:
    additionalTrustedCAs:
      - name: CompanyInternalCA-1
        data: LS0tLS1C...LS0tCg==
        #where "... " is the middle section of the long base64 string
      - name: CompanyInternalCA-2
        data: MTLtMT1C...MT0tPg==
    defaultNodeDrainTimeout: 0
```

Caution Editing the Tanzu Kubernetes Grid Service specification results in global changes to all clusters provisioned by that service, including new clusters and existing clusters that are upgraded manually or automatically.

Prerequisite: Configure Kubectl Editing

To scale a Tanzu Kubernetes cluster, you update the cluster manifest using the command `kubectl edit tanzukubernetescluster/CLUSTER-NAME`. The `kubectl edit` command opens the cluster manifest in the text editor defined by your `KUBE_EDITOR` or `EDITOR` environment variable. For instructions on setting up the environment variable, see [Specify a Default Text Editor for Kubectl](#).

When you save the specification changes, `kubectl` reports that the edits were successfully recorded. To cancel, simply close the editor without saving.

Configure the Default CNI

The Tanzu Kubernetes Grid Service provides a default container network interface (CNI) for Tanzu Kubernetes clusters. The default configuration lets you create clusters without the need to specify the CNI. You can change the default CNI by editing the service specification.

The Tanzu Kubernetes Grid Service supports two CNIs: Antrea and Calico, with Antrea being the default. For more information, see [Tanzu Kubernetes Grid Service Cluster Networking](#).

You can override the default CNI by explicitly specifying the CNI to use. Alternatively, you can change the default CNI by editing the TKG Service controller for CNIs.

- 1 Authenticate with the Supervisor Cluster.

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

- 2 Switch context to the target vSphere Namespace.

```
kubectl config use-context tkgs-cluster-ns
```

- 3 List the default CNI.

```
kubectl get tkgserviceconfigurations
```

Example result:

NAME	DEFAULT CNI
tkg-service-configuration	antrea

- 4 Load for editing the Tanzu Kubernetes Grid Service specification.

```
kubectl edit tkgserviceconfigurations tkg-service-configuration
```

The system opens the `tkg-service-configuration` specification in the default text editor defined by your `KUBE_EDITOR` or `EDITOR` environment variable.

- 5 Edit the `spec.defaultCNI` value.

For example, change from:

```
spec:
  defaultCNI: antrea
```

Change to:

```
spec:
  defaultCNI: calico
```

- 6 To apply the changes, save the file in the text editor. To cancel, close the editor without saving.

When you save the change in the text editor, `kubectl` updates the `tkg-service-configuration` service specification.

- 7 Verify that the default CNI is updated.

```
kubectl get tkgserviceconfigurations
```


The default CNI is updated. Any cluster provisioned with default network settings uses the default CNI.

NAME	DEFAULT CNI
tkg-service-configuration	calico

Configure a Global Proxy Server

To enable a global proxy server, add the proxy server parameters to the `TkgServiceConfiguration`. For a description of the required fields, see [Configuration Parameters for the Tanzu Kubernetes Grid Service v1alpha1 API](#).

- 1 Authenticate with the Supervisor Cluster.

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

- 2 Switch context to the target vSphere Namespace.

```
kubectl config use-context tkgs-cluster-ns
```

- 3 Get the current configuration.

```
kubectl get tkgserviceconfigurations
```

Example result:

NAME	DEFAULT CNI
tkg-service-configuration	antrea

- 4 Load for editing the Tanzu Kubernetes Grid Service specification.

```
kubectl edit tkgserviceconfigurations tkg-service-configuration
```

The system opens the `tkg-service-configuration` specification in the default text editor defined by your `KUBE_EDITOR` or `EDITOR` environment variable.

- 5 Add the `spec.proxy` subsection with each required field, including `httpProxy`, `httpsProxy`, and `noProxy`.

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  ...
  name: tkg-service-configuration-example
  resourceVersion: "44170525"
  selfLink: /apis/run.tanzu.vmware.com/v1alpha1/tkgserviceconfigurations/tkg-service-configuration
  uid: 10347195-5f0f-490e-8ae1-a758a724c0bc
spec:
  defaultCNI: antrea
```

```

proxy:
  httpProxy: http://<user>:<pwd>@<ip>:<port>
  httpsProxy: http://<user>:<pwd>@<ip>:<port>
  noProxy: [SVC-POD-CIDRs, SVC-EGRESS-CIDRs, SVC-INGRESS-CIDRs]

```

- 6 Populate each proxy field with the appropriate values. For a description of each field, see [Configuration Parameters for the Tanzu Kubernetes Grid Service v1alpha1 API](#).

The required values for the `noProxy` field come from the **Workload Network** on the Supervisor Cluster. Refer to the picture at the above topic on where to get these values.

For example:

```

apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  ...
  name: tkg-service-configuration-example
  resourceVersion: "44170525"
  selfLink: /apis/run.tanzu.vmware.com/v1alpha1/tkgserviceconfigurations/tkg-service-configuration
  uid: 10347195-5f0f-490e-8ae1-a758a724c0bc
spec:
  defaultCNI: antrea
  proxy:
    httpProxy: http://user:password@10.186.102.224:3128
    httpsProxy: http://user:password@10.186.102.224:3128
    noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]

```

- 7 To apply the changes, save the file in the text editor. To cancel, close the editor without saving.

When you save the changes in the text editor, `kubectl` updates Tanzu Kubernetes Grid Service with the configuration defined in the `tkg-service-configuration` service specification.

- 8 Verify that the Tanzu Kubernetes Grid Service is updated with the proxy settings.

```
kubectl get tkgserviceconfigurations -o yaml
```

- 9 To verify, provision a Tanzu Kubernetes cluster. See [Workflow for Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API](#).

Use the following command to confirm that the cluster is using the proxy.

```
kubectl get tkc CLUSTER-NAME -n NAMESPACE -o yaml
```

Certificate-Based Proxy Configuration

Using a proxy server to route internet traffic is a hard requirement for some environments. For example, a company in a highly regulated industry such as a financial institution requires all internet traffic go through a corporate proxy.

You can configure the Tanzu Kubernetes Grid Service to provision Tanzu Kubernetes clusters to use a proxy server for outbound HTTP/S traffic. For more information, see [Configuration Parameters for the Tanzu Kubernetes Grid Service v1alpha1 API](#).

As shown in the example, you can add trusted certificates for the proxy server to the `TkgServiceConfiguration` specification.

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration-example
spec:
  defaultCNI: antrea
  proxy:
    httpProxy: http://user:password@10.186.102.224:3128
    httpsProxy: http://user:password@10.186.102.224:3128
    noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]
  trust:
    additionalTrustedCAs:
      - name: first-cert-name
        data: base64-encoded string of a PEM encoded public cert 1
      - name: second-cert-name
        data: base64-encoded string of a PEM encoded public cert 2
  defaultNodeDrainTimeout: 0
```

External Private Registry Configuration

You can configure the Tanzu Kubernetes Grid Service with custom certificates for connecting Tanzu Kubernetes clusters with an external private registry. For more information, see [Use an External Container Registry with Tanzu Kubernetes Clusters](#).

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration-example
spec:
  defaultCNI: antrea
  trust:
    additionalTrustedCAs:
      - name: harbor-vm-cert
        data: <<<base64-encoded string of a PEM encoded public cert>>>>
```

Scale a Tanzu Kubernetes Cluster Using the TKGS v1alpha2 API

You can scale a Tanzu Kubernetes cluster horizontally by changing the number of nodes or vertically by changing the virtual machine class hosting the nodes.

Supported Scaling Operations

The table lists the supported scaling operations for Tanzu Kubernetes clusters.

Table 13-3. Supported Scaling Operations for Tanzu Kubernetes Clusters

Node	Horizontal Scale Out	Horizontal Scale In	Vertical Scale	Volume Scale
Control Plane	Yes	No	Yes	No
Worker	Yes	Yes	Yes	Yes

Keep in mind the following considerations:

- While vertically scaling a cluster node, workloads may no longer be able to run on the node for lack of available resource. For this reason horizontal scaling may be the preferred approach.
- VM classes are not immutable. If you scale out a Tanzu Kubernetes cluster after editing a VM class used by that cluster, new cluster nodes use the updated class definition, but existing cluster nodes continue to use the initial class definition, resulting in a mismatch. See [Virtual Machine Classes for Tanzu Kubernetes Clusters](#).
- Worker node volumes can be changed after provisioning; control plane node volumes cannot.

Scaling Prerequisite: Configure Kubectl Editing

To scale a Tanzu Kubernetes cluster, you update the cluster manifest using the command `kubectl edit tanzukubernetescluster/CLUSTER-NAME`. The `kubectl edit` command opens the cluster manifest in the text editor defined by your `KUBE_EDITOR` or `EDITOR` environment variable. For instructions on setting up the environment variable, see [Specify a Default Text Editor for Kubectl](#).

When you save the manifest changes, `kubectl` reports that the edits were successfully recorded, and the cluster is updated with the changes.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 edited
```

To cancel, simply close the editor without saving.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
Edit cancelled, no changes made.
```

Scale Out the Control Plane

You can scale out a Tanzu Kubernetes cluster by increasing number of control plane nodes from 1 to 3. The number of control plane nodes must be odd. You cannot scale in the control plane.

- 1 Authenticate with the Supervisor Cluster.

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

- 2 Switch context to the vSphere Namespace where the Tanzu Kubernetes cluster is running.

```
kubectl config use-context tkgs-cluster-ns
```

- 3 List the Kubernetes clusters that are running in the namespace.

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

- 4 Get the number of nodes running in the target cluster.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

For example, the following cluster has 1 control plane nodes and 3 worker nodes.

NAMESPACE	NAME	CONTROL PLANE	WORKER	TKR
NAME	AGE	READY		
tkgs-cluster-ns	test-cluster	1	3	v1.21.2---vmware.1-
tkg.1.13da849	5d12h	True		

- 5 Load the cluster manifest for editing using the `kubectl edit` command.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
```

The cluster manifest opens in the text editor defined by your `KUBE_EDITOR` or `EDITOR` environment variables.

- 6 Locate the `spec.topology.controlPlane.count` parameter and increase the number of nodes from 1 to 3.

```
...
controlPlane:
  replicas: 1
...
```

```
...
ControlPlane:
  replicas: 3
...
```

- 7 To apply the changes, save the file in the text editor. To cancel, close the editor without saving.

When you save the file, `kubectl` applies the changes to the cluster. In the background, the Virtual Machine Service on the Supervisor Cluster provisions the new worker node.

- 8 Verify that the new nodes are added.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

The scaled out control plane now has 3 nodes.

NAMESPACE	NAME	CONTROL PLANE	WORKER	TKR
NAME	AGE	READY		
tkgs-cluster-ns	test-cluster	3	3	v1.21.2---vmware.1-
tkg.1.13da849	5d12h	True		

Scale Out Worker Nodes

You can scale out a Tanzu Kubernetes cluster by increasing the number of worker nodes using `kubectl`.

- 1 Authenticate with the Supervisor Cluster.

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

- 2 Switch context to the vSphere Namespace where the Tanzu Kubernetes cluster is running.

```
kubectl config use-context tkgs-cluster-ns
```

- 3 List the Kubernetes clusters that are running in the namespace.

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

- 4 Get the number of nodes running in the target cluster.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

For example, the following cluster has 3 control plane nodes and 3 worker nodes.

NAMESPACE	NAME	CONTROL PLANE	WORKER	TKR
NAME	AGE	READY		
tkgs-cluster-ns	test-cluster	3	3	v1.21.2---vmware.1-
tkg.1.13da849	5d12h	True		

- 5 Load the cluster manifest for editing using the `kubectl edit` command.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
```

The cluster manifest opens in the text editor defined by your `KUBE_EDITOR` or `EDITOR` environment variables.

- 6 Locate the `spec.topology.workers.count` parameter and increase the number of nodes.

```
...
workers:
  replicas: 3
...
```

```
...
workers:
  replicas: 4
...
```

- 7 To apply the changes, save the file in the text editor. To cancel, close the editor without saving.

When you save the file, `kubectl` applies the changes to the cluster. In the background, the Virtual Machine Service on the Supervisor Cluster provisions the new worker node.

- 8 Verify that the new worker node is added.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

After scaling out, the cluster has 4 worker nodes.

NAMESPACE	NAME	AGE	CONTROL PLANE	WORKER	TKR
tkgs-cluster-ns	test-cluster	3	READY	4	v1.21.2---vmware.1-
tkg.1.13da849	5d12h	True			

Scale In Worker Nodes

You can scale in a Tanzu Kubernetes cluster by decreasing the number of worker nodes. Scaling in the control plane is not supported.

- 1 Authenticate with the Supervisor Cluster.

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

- 2 Switch context to the vSphere Namespace where the Tanzu Kubernetes cluster is running.

```
kubectl config use-context tkgs-cluster-ns
```

- 3 List the Kubernetes clusters that are running in the namespace.

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

- 4 Get the number of nodes running in the target cluster.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

For example, the following cluster has 3 control plane nodes and 4 worker nodes.

NAMESPACE	NAME	CONTROL PLANE	WORKER	TKR
NAME	AGE	READY		
tkgs-cluster-ns	test-cluster	3	4	v1.21.2---vmware.1-
tkg.1.13da849	5d12h	True		

- 5 Load the cluster manifest for editing using the `kubectl edit` command.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
```

The cluster manifest opens in the text editor defined by your `KUBE_EDITOR` or `EDITOR` environment variables.

- 6 Locate the `spec.topology.workers.count` parameter and decrease the number of nodes.

```
...
workers:
  replicas: 4
...
```

```
...
workers:
  replicas: 2
...
```

- 7 To apply the changes, save the file in the text editor. To cancel, close the editor without saving.

When you save the file, `kubectl` applies the changes to the cluster. In the background, the Virtual Machine Service on the Supervisor Cluster provisions the new worker node.

- 8 Verify that the worker nodes were removed.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

After scaling in, the cluster has 2 worker nodes.

NAMESPACE	NAME	CONTROL PLANE	WORKER	TKR
NAME	AGE	READY		
tkgs-cluster-ns	test-cluster	3	2	v1.21.2---vmware.1-
tkg.1.13da849	5d12h	True		

Scale a Cluster Vertically

You can vertically scale a Tanzu Kubernetes cluster by changing the virtual machine class used to host the cluster nodes. Vertical scaling is supported for both control plane and worker nodes.

The Tanzu Kubernetes Grid Service supports scaling cluster nodes vertically through the rolling update mechanism built into the service. If you change the `VirtualMachineClass` definition, the service rolls out new nodes with that new class and spins down the old nodes. See [Update Tanzu Kubernetes Clusters](#).

- 1 Authenticate with the Supervisor Cluster.

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

- 2 Switch context to the vSphere Namespace where the Tanzu Kubernetes cluster is running.

```
kubectl config use-context tkgs-cluster-ns
```

- 3 List the Kubernetes clusters that are running in the namespace.

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

- 4 Describe the target Tanzu Kubernetes cluster and check the VM class.

```
kubectl describe tanzukubernetescluster tkgs-cluster-2
```

For example, the following cluster is using the best-effort-medium VM class.

```
Spec:
  ...
  Topology:
    Control Plane:
      Class:          best-effort-medium
      ...
    nodePool-a1:
      Class:          best-effort-medium
      ...
```

- 5 List and describe the available VM classes.

```
kubectl get virtualmachineclassbinding
```

```
kubectl describe virtualmachineclassbinding
```

Note The VM class you want to use must be bound the vSphere Namespace. See [Virtual Machine Classes for Tanzu Kubernetes Clusters](#).

- 6 Open for editing the target cluster manifest.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-2
```

The cluster manifest opens in the text editor defined by your `KUBE_EDITOR` or `EDITOR` environment variables.

- 7 Edit the manifest by changing the VM class.

For example, edit the cluster manifest to use the `guaranteed-large` VM class for control plane and worker nodes.

```
spec:
  topology:
    controlPlane:
      class: guaranteed-large
      ...
    nodePool-a1:
      class: guaranteed-large
      ...
```

- 8 To apply the changes, save the file in the text editor. To cancel, close the editor without saving.

When you save the file, `kubectl` applies the changes to the cluster. In the background, the Tanzu Kubernetes Grid Service provisions the new nodes and deletes the old ones. For a description of the rolling update process, see [About Tanzu Kubernetes Grid Service Cluster Updates](#).

- 9 Verify that the cluster is updated.

```
kubectl get tanzukubernetescluster
```

NAMESPACE	NAME	AGE	CONTROL PLANE	WORKER	TKR
tkgs-cluster-ns	test-cluster	3		3	v1.21.2---vmware.1-
tkg.1.13da849	5d12h	True			

Scale Node Volumes

In the Tanzu Kubernetes cluster specification for nodes, you have the option of declaring one or more persistent volumes. Declaring a node volume is useful for high-churn components, such as the etcd database on the control plane and the container runtime on worker nodes. An excerpted cluster specification with both of these node volumes declared is provided below for reference. (A full example cluster specification is available [Example YAML for Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API](#).)

If you want to add or change one or more node volumes after cluster creation, keep in mind the following considerations:

Volume Node	Description
Worker node volume changes are allowed.	After a Tanzu Kubernetes cluster is provisioned, you can add or update a worker node volume. When you initiate a rolling update, the cluster is updated with the new or changed volume. Warning If you scale the worker node with a new or changed volume, data in the current volume is deleted during the rolling update.
Control plane node volume changes are not allowed.	After a Tanzu Kubernetes cluster is provisioned, you cannot add or update a control plane node volume. The Kubernetes Cluster API (CAPI) forbids post-creation changes to <code>spec.topology.controlPlane.volumes</code>. If you attempt to add or change a control plane volume after cluster creation, the request is denied and you receive the error message "updates to immutable fields are not allowed."

```
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-medium
      storageClass: vwt-storage-policy
      volumes:
        - name: etcd
          mountPath: /var/lib/etcd
          capacity:
            storage: 4Gi
    tkr:
      reference:
        name: v1.21.2---vmware.1-tkg.1.ee25d55
  nodePools:
    - name: worker-nodepool-a1
      replicas: 3
      vmClass: guaranteed-large
      storageClass: vwt-storage-policy
      volumes:
        - name: containerd
          mountPath: /var/lib/containerd
          capacity:
            storage: 16Gi
```

Provisioning Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API

This section describes how to provision Tanzu Kubernetes clusters using the Tanzu Kubernetes Grid Service v1alpha1 API.

Note The TKG v1alpha1 API is deprecated and should not be used to create new TKG clusters. Use the TKG v1alpha2 API instead. See [Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API](#).

Workflow for Provisioning Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API

You provision Tanzu Kubernetes clusters by invoking the Tanzu Kubernetes Grid Service declarative API using `kubectl` and a cluster specification defined using YAML. After you provision a cluster, you operate it and deploy workloads to it using `kubectl`.

The workflow provides an end-to-end procedure for the cluster provisioning process. Each of the steps has links for more information about the specific task.

Prerequisites

Note The TKG v1alpha1 API is deprecated and should not be used to create new TKG clusters. Use the TKG v1alpha2 API instead. See [Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API](#).

Complete the following prerequisites:

- Configure a Supervisor Cluster instance. See [Chapter 5 Configuring and Managing a Supervisor Cluster](#).
- Create a vSphere Namespace where you plan to provision Tanzu Kubernetes clusters. See [Create and Configure a vSphere Namespace](#).

The initial namespace configuration requires the following:

- Edit permissions for one or more DevOps engineers to access the namespace using vCenter Single Sign-On credentials.
- Tag-based shared storage policy for the namespace.
- Capacity and usage quotas for the namespace, verified and adjusted, as necessary.
- Create a content library for Tanzu Kubernetes releases on a shared datastore and synchronize the releases you want to use. See [Creating and Managing Content Libraries for Tanzu Kubernetes releases](#).
- Associate the content library and the virtual machine classes with the vSphere Namespace. See [Configure a vSphere Namespace for Tanzu Kubernetes releases](#).

Procedure

- 1 Download and install the Kubernetes CLI Tools for vSphere. See [Download and Install the Kubernetes CLI Tools for vSphere](#).
- 2 Using the vSphere Plugin for `kubectl`, authenticate with the Supervisor Cluster. See [Connect to the Supervisor Cluster as a vCenter Single Sign-On User](#).

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME
```

- 3 Using kubectl, switch context to the vSphere Namespace where you plan to provision the Tanzu Kubernetes cluster.

```
kubectl config get-contexts
```

```
kubectl config use-context SUPERVISOR-NAMESPACE
```

For example:

```
kubectl config use-context tkgs-cluster-ns
```

- 4 List the available virtual machine class bindings. See [Virtual Machine Classes for Tanzu Kubernetes Clusters](#).

Use the following command to list all VM class bindings that are available in the vSphere Namespace where you deploy the cluster.

```
kubectl get virtualmachineclassbindings
```

Note The command `kubectl get virtualmachineclasses` lists all the VM classes present on the Supervisor Cluster. Because you must associate VM classes with the vSphere Namespace, you can only use those VM classes that are bound to the target namespace.

- 5 Get the available default storage class by describing the namespace.

```
kubectl describe namespace SUPERVISOR-NAMESPACE
```

- 6 List the available Tanzu Kubernetes releases.

Note Refer to the list of Tanzu Kubernetes releases for compatibility. See [Verify Tanzu Kubernetes Cluster Compatibility for Update](#).

```
kubectl get tanzukubernetesreleases
```

Note The command `kubectl get virtualmachineimages` returns generic information about the virtual machines.

7 Construct the YAML file for provisioning a Tanzu Kubernetes cluster.

- a Start with one of the example YAML files. See [Examples for Provisioning Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API](#).

For example, the following YAML file provisions a minimal cluster using all available defaults:

```
apiVersion: run.tanzu.vmware.com/v1alpha1 #TKGS API endpoint
kind: TanzuKubernetesCluster              #required parameter
metadata:
  name: tkgs-cluster-1                    #cluster name, user defined
  namespace: tkgs-cluster-ns             #vsphere namespace
spec:
  distribution:
    version: v1.19                        #Resolves to latest TKR 1.19 version
  topology:
    controlPlane:
      count: 1                            #number of control plane nodes
      class: best-effort-medium            #vmclass for control plane nodes
      storageClass: vwt-storage-policy     #storageclass for control plane
    workers:
      count: 3                            #number of worker nodes
      class: best-effort-medium            #vmclass for worker nodes
      storageClass: vwt-storage-policy     #storageclass for worker nodes
```

- b Use the information you gleaned from the output of the preceding commands to populate the cluster YAML, including the namespace, storage class, and virtual machine class.
- c Customize the cluster as needed by referring to the full list of cluster configuration parameters. See [Configuration Parameters for Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API](#).
- d Save the file as `tkgs-cluster-1.yaml`, or similar.

8 Provision the cluster by running the following kubectl command.

```
kubectl apply -f CLUSTER-NAME.yaml
```

For example:

```
kubectl apply -f tkgs-cluster-1.yaml
```

Expected result:

```
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 created
```

9 Monitor the deployment of cluster nodes using kubectl. See [Monitor Tanzu Kubernetes Cluster Status Using kubectl](#).

```
kubectl get tanzukubernetesclusters
```

Sample result:

NAME	CONTROL PLANE	WORKER	DISTRIBUTION	AGE	PHASE
tkgs-cluster-2	1	3	v1.19.7+vmware.1-tkg.1.c40d30d	7m59s	running

- 10 Monitor the deployment of cluster nodes using the vSphere Client. See [Monitor Tanzu Kubernetes Cluster Status Using the vSphere Client](#).

For example, in the vSphere inventory you should see the virtual machine nodes being deployed in the namespace.

- 11 Run additional commands to verify cluster provisioning. See [Use Tanzu Kubernetes Cluster Operational Commands](#).

For example:

```
kubectl get tanzukubernetescluster,cluster-  
api,virtualmachinesetresourcepolicy,virtualmachineservice,virtualmachine
```

Note For additional troubleshooting, see [Troubleshooting Tanzu Kubernetes Clusters](#).

- 12 Using the vSphere Plugin for kubectl, log in to the cluster. See [Connect to a Tanzu Kubernetes Cluster as a vCenter Single Sign-On User](#).

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME \  
--tanzu-kubernetes-cluster-name CLUSTER-NAME --tanzu-kubernetes-cluster-namespace  
NAMESPACE-NAME
```

- 13 Verify cluster provisioning using the following kubectl commands.

```
kubectl cluster-info
```

```
kubectl get nodes
```

```
kubectl get namespaces
```

```
kubectl api-resources
```

- 14 Deploy an example workload and verify cluster creation. See [Deploy Workloads on Tanzu Kubernetes Clusters](#).

Note Tanzu Kubernetes clusters have pod security policy enabled. Depending on the workload and user, you might need to create an appropriate RoleBinding or custom PodSecurityPolicy. See [Using Pod Security Policies with Tanzu Kubernetes Clusters](#).

- 15 Operationalize the cluster by deploying TKG Extensions. See [Install Packages on Tanzu Kubernetes Clusters](#).

Configuration Parameters for Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API

The Tanzu Kubernetes Grid Service declarative API exposes several parameters for configuring Tanzu Kubernetes clusters. Refer to the list and description of all parameters and usage guidelines to provision and customize your clusters.

Annotated YAML for Provisioning a Tanzu Kubernetes Cluster

Note The TKG v1alpha1 API is deprecated and should not be used to create new TKG clusters. Use the TKG v1alpha2 API instead. See [Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API](#).

The annotated YAML lists all available parameters for provisioning a Tanzu Kubernetes cluster with summarized comments for each field.

Note The annotated YAML is not validated for provisioning a cluster. Refer to the examples for such guidance: [Examples for Provisioning Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API](#).

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
#valid config key must consist of alphanumeric characters, '-', '_' or '.'
metadata:
  name: <tanzu kubernetes cluster name>
  namespace: <vsphere namespace where the cluster will be provisioned>
spec:
  distribution:
    version: <tanzu kubernetes release version string: full, point, short>
  topology:
    controlPlane:
      count: <integer either 1 or 3>
      class: <vm class bound to the target vsphere namespace>
      storageClass: <vsphere storage policy bound to the target vsphere namespace>
      volumes: #optional setting for high-churn control plane component
        - name: <user-defined string>
          mountPath: </dir/path>
          capacity:
            storage: <size in GiB>
    workers:
      count: <integer from 0 to 150>
      class: <vm class bound to the target vsphere namespace>
      storageClass: <vsphere storage policy bound to the target vsphere namespace>
      volumes: #optional setting for high-churn worker node component (such as containerd)
        - name: <user-defined string>
          mountPath: </dir/path>
          capacity:
            storage: <size in GiB>
  settings: #all spec.settings are optional
    storage: #optional storage settings
    classes: [<array of kubernetes storage classes for dynamic pvc provisioning>]
```



```

defaultClass: <default kubernetes storage class>
network: #optional network settings
  cni: #override default cni set in the tkgservicesonfiguration spec
    name: <antrea or calico>
  pods: #custom pod network
    cidrBlocks: [<array of pod cidr blocks>]
  services: #custom service network
    cidrBlocks: [<array of service cidr blocks>]
  serviceDomain: <custom service domain>
proxy: #proxy server for outbound connections
  httpProxy: http://<IP:PORT>
  httpsProxy: http://<IP:PORT>
  noProxy: [<array of CIDRs to not proxy>]
trust: #trust fields for custom public certs for tls
  additionalTrustedCAs:
    - name: <first-cert-name>
      data: <base64-encoded string of PEM encoded public cert 1>
    - name: <second-cert-name>
      data: <base64-encoded string of PEM encoded public cert 2>

```

Parameters for Provisioning Tanzu Kubernetes Clusters

The table lists and describes all parameters and acceptable values for provisioning a Tanzu Kubernetes cluster. For examples, see [Examples for Configuring the Tanzu Kubernetes Grid Service v1alpha1 API](#).

Important A valid key name must consist only of alphanumeric characters, a dash (such as key-name), an underscore (such as KEY_NAME) or a dot (such as key.name). You cannot use the space character in a key name.

Table 13-4. Parameters for Provisioning Tanzu Kubernetes Clusters

Name	Value	Description
apiVersion	run.tanzu.vmware.com/v1alpha1	Specifies the version of the Tanzu Kubernetes Grid Service API.
kind	TanzuKubernetesCluster	Specifies the type of Kubernetes resource to create. The only allowed value is <code>TanzuKubernetesCluster</code> (case-sensitive).
metadata	Section for cluster metadata	Includes cluster metadata, such as name and namespace. This is standard Kubernetes metadata, so you can use <code>generateName</code> instead of <code>name</code> , add labels and annotations, and so on.

Table 13-4. Parameters for Provisioning Tanzu Kubernetes Clusters (continued)

Name	Value	Description
name	A user-defined string that accepts alphanumeric characters and dashes, for example: <code>my-tkg-cluster-1</code>	Specifies the name of the cluster to create. Current cluster naming constraints: ■ Name length must be 41 characters or less. ■ Name must begin with a letter. ■ Name may contain letters, numbers, and hyphens. ■ Name must end with a letter or a number.
namespace	A user-defined string that accepts alphanumeric characters and dashes, for example: <code>my-sns-1</code>	Identifies the name of the Supervisor Namespace where the cluster will be deployed. This is a reference to a Supervisor Namespace that exists on the Supervisor Cluster.
spec	Section for technical specifications for the cluster	Includes the specification, expressed in declarative fashion, for the end-state of the cluster, including the node <code>topology</code> and Kubernetes <code>software distribution</code> .
distribution	Section for specifying the Tanzu Kubernetes Release version	Indicates the distribution for the cluster: the Tanzu Kubernetes cluster software installed on the control plane and worker nodes, including Kubernetes itself.
version	Alphanumeric string with dashes representing the Kubernetes version, for example: <code>v1.20.2+vmware.1-tkg.1</code> or <code>v1.20.2</code> or <code>v1.20</code>	Specifies the software version of the Kubernetes distribution to install on cluster nodes using semantic version notation. Can specify the fully qualified version or use version shortcuts, such as "version: v1.20.2", which is resolved to the most recent image matching that patch version, or "version: v1.20", which is resolved to the most recent matching patch version. The resolved version displays as the "fullVersion" on the cluster description after you have created it.

Table 13-4. Parameters for Provisioning Tanzu Kubernetes Clusters (continued)

Name	Value	Description
<code>topology</code>	Section for cluster node topologies	Includes fields that describe the number, purpose, and organization of cluster nodes and the resources allocated to each. Cluster nodes are grouped into pools based on their intended purpose: either <code>control-plane</code> or <code>worker</code> . Each pool is homogeneous, having the same resource allocation and using the same storage.
<code>controlPlane</code>	Section for control plane settings	Specifies the topology of the cluster control plane, including the number of nodes (<code>count</code>), type of VM (<code>class</code>), and the storage resources allocated for each node (<code>storageClass</code>).
<code>count</code>	An integer that is either 1 or 3	Specifies the number of control plane nodes. The control plane must have an odd number of nodes.
<code>class</code>	A system-defined element in the form of a string from an enumerated set, for example: <code>guaranteed-small</code> or <code>best-effort-large</code>	Specifies the name of the <code>VirtualMachineClass</code> that describes the virtual hardware settings to be used each node in the pool. This controls the hardware available to the node (CPU and memory) as well as the requests and limits on those resources. See Virtual Machine Classes for Tanzu Kubernetes Clusters .
<code>storageClass</code>	<code>node-storage</code> (for example)	Identifies the storage class to be used for storage of the disks which store the root file systems of the control plane nodes. Run <code>kubectl describe ns</code> on the namespace to view the available storage classes. The available storage classes for the namespace depend on the storage set by the vSphere administrator. Storage classes associated with the Supervisor Namespace are replicated in the cluster. In other words, the storage class must be available on the Supervisor Namespace to be a valid value for this field. See Chapter 7 Configuring and Managing vSphere Namespaces .

Table 13-4. Parameters for Provisioning Tanzu Kubernetes Clusters (continued)

Name	Value	Description
volumes	Optional storage setting ■ volumes: ■ name: <i>string</i> ■ mountPath: <i>/dir/path</i> ■ capacity ■ storage: GiB size	Can specify separate disk and storage parameters for etcd on control plane nodes. See the example Cluster with Separate Disks and Storage Parameters .
workers	Section for worker node settings	Specifies the topology of the cluster worker nodes, including the number of nodes (<i>count</i>), type of VM (<i>class</i>), and the storage resources allocated for each node (<i>storageClass</i>).
count	An integer between 0 and 150, for example: 1 or 2 or 7	Specifies the number of worker nodes in the cluster. A cluster with zero worker nodes can be created, allowing for a cluster with only control plane nodes. There is no hard maximum for the number of worker nodes, but a reasonable limit is 150. Note A cluster provisioned with 0 worker nodes is not assigned any load balancer services.
class	A system-defined element in the form of a string from an enumerated set, for example: <i>guaranteed-small</i> or <i>best-effort-large</i>	Specifies the name of the VirtualMachineClass that describes the virtual hardware settings to be used each node in the pool. This controls the hardware available to the node (CPU and memory) as well as the requests and limits on those resources. See Virtual Machine Classes for Tanzu Kubernetes Clusters .
storageClass	<i>node-storage</i> (for example)	Identifies the storage class to be used for storage of the disks that store the root file systems of the worker nodes. Run <code>kubectl describe ns</code> on the namespace to list available storage classes. The available storage classes for the namespace depend on the storage set by the vSphere administrator. Storage classes associated with the Supervisor Namespace are replicated in the cluster. In other words, the storage class must be available on the Supervisor Namespace to be valid. See Chapter 7 Configuring and Managing vSphere Namespaces .

Table 13-4. Parameters for Provisioning Tanzu Kubernetes Clusters (continued)

Name	Value	Description
volumes	Optional storage setting ■ volumes: ■ name: <i>string</i> ■ mountPath: <i>/dir/path</i> ■ capacity ■ storage: GiB size	Can specify separate disk and storage parameters for container images on worker nodes. See the example Cluster with Separate Disks and Storage Parameters .
settings	Section for cluster-specific settings; all <code>spec.settings</code> are optional	Identifies optional runtime configuration information for the cluster, including node <code>network</code> details and persistent <code>storage</code> for pods.
storage	Section for specifying storage	Identifies persistent volume (PV) storage entries for container workloads.
classes	Array of one or more user-defined strings, for example: ["gold", "silver"]	Specifies named persistent volume (PV) storage classes for container workloads. Storage classes associated with the Supervisor Namespace are replicated in the cluster. In other words, the storage class must be available on the Supervisor Namespace to be a valid value. See the example Cluster with Storage Classes and a Default Class for Persistent Volumes .
defaultClass	<code>silver</code> (for example)	Specifies a named storage class to be annotated as the default in the cluster. If you do not specify it, there is no default. You do not have to specify one or more <code>classes</code> to specify a <code>defaultClass</code> . Some workloads may require a default class, such as Helm. See the example Cluster with Storage Classes and a Default Class for Persistent Volumes .
network	Section marker for networking settings	Specifies network-related settings for the cluster.
cni	Section marker for specifying the CNI	Identifies the Container Networking Interface (CNI) plug-in for the cluster. The default is Antrea, which does not need to be specified for new clusters.
name	String <code>antrea</code> or <code>calico</code>	Specifies the CNI to use. Antrea and Calico are supported. System configuration sets Antrea as the default CNI. The default CNI can be changed. If using the default, this field does not need to be specified.

Table 13-4. Parameters for Provisioning Tanzu Kubernetes Clusters (continued)

Name	Value	Description
services	Section marker for specifying Kubernetes services subnets	Identifies network settings for Kubernetes services. Default is 10.96.0.0/12.
cidrBlocks	Array ["198.51.100.0/12"] (for example)	Specifies a range of IP addresses to use for Kubernetes services. Default is 10.96.0.0/12. Must not overlap with the settings chosen for the Supervisor Cluster. Although this field is an array, allowing for multiple ranges, currently only a single IP range is allowed. See the networking examples at Examples for Provisioning Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API .
pods	Section marker for specifying Kubernetes pods subnets	Specifies network settings for pods. Default is 192.168.0.0/16. Minimum block size is /24.
cidrBlocks	Array ["192.0.2.0/16"] (for example)	Specifies a range of IP addresses to use for Kubernetes pods. Default is 192.168.0.0/16. Must not overlap with the settings chosen for the Supervisor Cluster. Pods subnet size must be equal to or larger than /24. Although this field is an array, allowing for multiple ranges, currently only a single IP range is allowed. See the networking examples at Examples for Provisioning Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API .
serviceDomain	"cluster.local"	Specifies the service domain for the cluster. Default is cluster.local.
proxy	Section that specifies HTTP(s) proxy configuration for the cluster. If implemented all fields are required.	Provides fields for specified proxy settings; will be auto-populated if a global proxy is configured and individual cluster proxy is not configured. See the example Cluster with a Proxy Server .
httpProxy	http://<user>:<pwd>@<ip>:<port>	Specifies a proxy URL to use for creating HTTP connections outside the cluster.
httpsProxy	http://<user>:<pwd>@<ip>:<port>	Specifies a proxy URL to use for creating HTTPS connections outside the cluster.

Table 13-4. Parameters for Provisioning Tanzu Kubernetes Clusters (continued)

Name	Value	Description
<code>noProxy</code>	Array of CIDR blocks to not proxy, for example: <code>[10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]</code>. Get the required values come from the Workload Network on the Supervisor Cluster: Pod CIDRs, Ingress CIDRs, and Egress CIDRs. Refer to the image below for what values to include in the <code>noProxy</code> array field.	You must not proxy the subnets used by the Workload Network on the Supervisor Cluster for Pods, Ingress, and Egress. You do not need to include the Services CIDR from the Supervisor Cluster in the <code>noProxy</code> field. Tanzu Kubernetes clusters do not interact with such services. The endpoints <code>localhost</code> and <code>127.0.0.1</code> are automatically not proxied. You do not need to add them to the <code>noProxy</code> field. The Pod and Service CIDRs for Tanzu Kubernetes clusters are automatically not proxied. You do not need to add them to the <code>noProxy</code> field. See the example Cluster with a Proxy Server.
<code>trust</code>	Section marker for <code>trust</code> parameters.	Accepts no data.
<code>additionalTrustedCAs</code>	Accepts an array of certificates with <code>name</code> and <code>data</code> for each.	Accepts no data.
<code>name</code>	String	The name of the TLS certificate.
<code>data</code>	String	The base64-encoded string of a PEM encoded public certificate.

Get the required `noProxy` values from the **Workload Network** on the Supervisor Cluster as shown in the image.

The screenshot displays the 'compute-cluster' configuration page in the vSphere with Tanzu interface. The left sidebar contains a navigation menu with categories: Services, Configuration, Licensing, Namespaces, vSAN, and Supervisor Ser... The 'Namespaces' category is expanded, and 'Network' is selected. The main content area is titled 'Network' and includes a sub-header 'Below are the network settings for supporting namespaces on this cluster.' The settings are organized into two sections: 'Management Network' and 'Workload Network'. The 'Workload Network' section is expanded, showing a list of network parameters. The 'Pod CIDRs' and 'Ingress CIDRs' rows are highlighted with red boxes. Each row includes an 'EDIT' link.

Management Network	
Workload Network	
vSphere Distributed Switch	wcp_vds_1
Edge Cluster	EDGECLUSTER1
DNS Servers	10.20.145.1 EDIT
Pod CIDRs	10.246.0.0/16 EDIT
Services CIDR	10.94.0.0/12
Ingress CIDRs	192.168.144.0/20 EDIT
Egress CIDRs	192.168.128.0/20 EDIT

Examples for Provisioning Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API

The Tanzu Kubernetes Grid Service API provides intelligent defaults and an array of options for customizing Tanzu Kubernetes clusters. Refer to the examples to provision clusters of various types with different configurations and customizations to meet your needs.

The v1alpha1 API Is Deprecated

Note The TKG v1alpha1 API is deprecated and should not be used to create new TKG clusters. Use the TKG v1alpha2 API instead. See [Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API](#).

Minimal YAML for Provisioning a Tanzu Kubernetes Cluster

The following example YAML is the minimal configuration required to invoke the Tanzu Kubernetes Grid Service and provision a Tanzu Kubernetes cluster that uses all of the default settings.

Characteristics of the minimal example YAML include:

- The Tanzu Kubernetes release version, listed as v1.19, is resolved to the most recent distribution matching that minor version, for example v1.19.7+vmware.1-tkg.1.xxxxxx. See [Verify Tanzu Kubernetes Cluster Compatibility for Update](#).
- The VM class `best-effort-<size>` has no reservations. For more information, see [Virtual Machine Classes for Tanzu Kubernetes Clusters](#).
- The cluster does not include persistent storage for containers. If needed this is set in `spec.settings.storage`. See the storage example below.
- Some workloads, such as Helm, may require a `spec.settings.storage.defaultClass`. See the storage example below.
- The `spec.settings.network` section is not specified. This means that the cluster uses the following default network settings:
 - Default CNI: `antrea`
 - Default Pod CIDR: `192.168.0.0/16`
 - Default Services CIDR: `10.96.0.0/12`
 - Default Service Domain: `cluster.local`

Note The default IP range for Pods is 192.168.0.0/16. If this subnet is already in use, you must specify a different CIDR range. See the custom network examples below.

```
apiVersion: run.tanzu.vmware.com/v1alpha1      #TKGS API endpoint
kind: TanzuKubernetesCluster                   #required parameter
metadata:
```

```

name: tkgs-cluster-1                #cluster name, user defined
namespace: tgks-cluster-ns          #vsphere namespace
spec:
  distribution:
    version: v1.20                  #Resolves to latest TKR 1.20
  topology:
    controlPlane:
      count: 1                      #number of control plane nodes
      class: best-effort-medium      #vmclass for control plane nodes
      storageClass: vwt-storage-policy #storageclass for control plane
    workers:
      count: 3                      #number of worker nodes
      class: best-effort-medium      #vmclass for worker nodes
      storageClass: vwt-storage-policy #storageclass for worker nodes

```

Cluster with Separate Disks and Storage Parameters

The following example YAML shows how to provision a cluster with separate disks and storage parameters for cluster control plane and worker nodes.

Using separating disks and storage parameters for high-churn data help to minimize read-write overhead related to the use of linked clones, among other benefits. There are two primary use cases:

- Customize storage performance on control plane nodes for the etcd database
- Customize the size of the disk for container images on the worker nodes

The example has the following characteristics:

- The `spec.topology.workers.volumes` settings specify the separate volume for the container images.
- The `mountPath: /var/lib/containerd` for container images is supported for Tanzu Kubernetes releases 1.17 and later.

```

apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-5
  namespace: tgks-cluster-ns
spec:
  distribution:
    version: v1.20
  topology:
    controlPlane:
      count: 3
      class: best-effort-medium
      storageClass: vwt-storage-policy
    workers:
      count: 3
      class: best-effort-medium
      storageClass: vwt-storage-policy
      volumes:

```

```

- name: containerd
  mountPath: /var/lib/containerd
  capacity:
    storage: 16Gi

```

Cluster with a Custom Antrea Network

The following YAML demonstrates how to provision a Tanzu Kubernetes cluster with custom network ranges for the Antrea CNI.

- Because custom network settings are applied, the `cni.name` parameter is required even though the default Antrea CNI is used.
 - CNI name: `antrea`
 - Custom Pods CIDR: `193.0.2.0/16`
 - Custom Services CIDR: `195.51.100.0/12`
 - Custom Service Domain: `managedcluster.local`
- The custom CIDR blocks cannot overlap with the Supervisor Cluster. For more information, see [Configuration Parameters for Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API](#).

```

apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: tkg-cluster-3-antrea
  namespace: tkgs-cluster-ns
spec:
  distribution:
    version: v1.20
  topology:
    controlPlane:
      class: guaranteed-medium
      count: 3
      storageClass: vwt-storage-policy
    workers:
      class: guaranteed-medium
      count: 5
      storageClass: vwt-storage-policy
  settings:
    network:
      cni:
        name: antrea #Use Antrea CNI
      pods:
        cidrBlocks:
          - 193.0.2.0/16 #Must not overlap with SVC
      services:
        cidrBlocks:
          - 195.51.100.0/12 #Must not overlap with SVC
        serviceDomain: managedcluster.local

```

Cluster with a Custom Calico Network

The following YAML demonstrates how to provision a Tanzu Kubernetes cluster with a custom Calico network.

- Calico is not the default CNI, so it is explicitly named in the manifest. To change the default CNI at the service-level, see [Examples for Configuring the Tanzu Kubernetes Grid Service v1alpha1 API](#).
 - CNI name: `calico`
 - Custom Pods CIDR: `198.51.100.0/12`
 - Custom Services CIDR: `192.0.2.0/16`
 - Custom Service Domain: `managedcluster.local`
- The network uses custom CIDR ranges, not the defaults. These ranges must not overlap with the Supervisor Cluster. See [Configuration Parameters for Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API](#).

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-2
  namespace: tkgs-cluster-ns
spec:
  distribution:
    version: v1.20
  topology:
    controlPlane:
      count: 3
      class: guaranteed-large
      storageClass: vwt-storage-policy
    workers:
      count: 5
      class: guaranteed-xlarge
      storageClass: vwt-storage-policy
  settings:
    network:
      cni:
        name: calico                                #Use Calico CNI for this cluster
      services:
        cidrBlocks: ["198.51.100.0/12"]             #Must not overlap with SVC
      pods:
        cidrBlocks: ["192.0.2.0/16"]                #Must not overlap with SVC
      serviceDomain: managedcluster.local
```

Cluster with Storage Classes and a Default Class for Persistent Volumes

The following example YAML demonstrates how to provision a cluster with a storage classes for dynamic PVC provisioning and a default storage class.

- The `spec.settings.storage.classes` setting specifies two storage classes for persistent storage for containers in the cluster.
- The `spec.settings.storage.defaultClass` is specified. Some applications require a default class. For example, if you want to use Helm or Kubeapps as the `defaultClass`, which is referenced by many charts.

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: default-storage-spec
  namespace: tkgs-cluster-ns
spec:
  topology:
    controlPlane:
      count: 3
      class: best-effort-medium
      storageClass: vwt-storage-policy
    workers:
      count: 3
      class: best-effort-medium
      storageClass: vwt-storage-policy
  distribution:
    version: v1.20
  settings:
    network:
      cni:
        name: antrea
      services:
        cidrBlocks: ["198.51.100.0/12"]
      pods:
        cidrBlocks: ["192.0.2.0/16"]
        serviceDomain: "tanzukubernetescluster.local"
    storage:
      classes: ["gold", "silver"]           #Array of named PVC storage classes
      defaultClass: silver                 #Default PVC storage class
```

Cluster with a Proxy Server

You can use a proxy server with an individual Tanzu Kubernetes cluster by applying the proxy server configuration to the cluster manifest.

Note the following characteristics:

- The section `spec.settings.network.proxy` specifies HTTP(s) proxy configuration for this Tanzu Kubernetes cluster.
- The syntax for both `proxy server` values is `http://<user>:<pwd>@<ip>:<port>`.

- Specific endpoints are automatically not proxied, including `localhost` and `127.0.0.1`, and the Pod and Service CIDRs for Tanzu Kubernetes clusters. You do not need to include these in the `noProxy` field.
- The `noProxy` field accepts an array of CIDRs to not proxy. Get the required values from the Workload Network on the Supervisor Cluster. Refer to the image at [Configuration Parameters for Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API](#).
- If a global proxy is configured on the `TkgServiceConfiguration`, that proxy information is propagated to the cluster manifest after the initial deployment of the cluster. The global proxy configuration is added to the cluster manifest only if there is no proxy configuration fields present when creating the cluster. In other words, per-cluster configuration takes precedence and will overwrite a global proxy configuration.

```

apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-with-proxy
  namespace: tkgs-cluster-ns
spec:
  distribution:
    version: v1.20
  topology:
    controlPlane:
      count: 3
      class: guaranteed-medium
      storageClass: vwt-storage-policy
    workers:
      count: 5
      class: guaranteed-xlarge
      storageClass: vwt-storage-policy
  settings:
    storage:
      classes: ["gold", "silver"]
      defaultClass: silver
    network:
      cni:
        name: antrea
      pods:
        cidrBlocks:
          - 193.0.2.0/16
      services:
        cidrBlocks:
          - 195.51.100.0/12
      serviceDomain: managedcluster.local
    proxy:
      httpProxy: http://10.186.102.224:3128 #Proxy URL for HTTP connections
      httpsProxy: http://10.186.102.224:3128 #Proxy URL for HTTPS connections
      noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20] #SVC Pod, Egress, Ingress
  CIDRs

```

Cluster with Custom Certificates for TLS

Similar to how you can specify a `trust.additionalTrustedCAs` in the `TkgServiceConfiguration` (see [Configuration Parameters for the Tanzu Kubernetes Grid Service v1alpha1 API](#)), you can include `trust.additionalTrustedCAs` under the `spec.settings.network` in the `TanzuKubernetesCluster` `spec`. For example:

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-cluster-with-custom-certs-tls
  namespace: tkgs-cluster-ns
spec:
  topology:
    controlPlane:
      count: 3
      class: guaranteed-medium
      storageClass: vwt-storage-profile
    workers:
      count: 3
      class: guaranteed-large
      storageClass: vwt-storage-profile
  distribution:
    version: 1.20.2
  settings:
    network:
      cni:
        name: antrea
      services:
        cidrBlocks: ["198.51.100.0/12"]
      pods:
        cidrBlocks: ["192.0.2.0/16"]
        serviceDomain: "managedcluster.local"
      trust:
        additionalTrustedCAs:
          - name: custom-selfsigned-cert-for-tkc
            data: |
              LS0aaaaaaaaaaaaaaaaabase64...
```

Cluster that Does or Does Not Inherit Global Settings from the TkgServiceConfiguration Spec

Note The following example cluster configurations require vCenter Server version 7.0U2a or later and at least Supervisor Cluster version 1.18.10.

To provision a Tanzu Kubernetes cluster that inherits a global setting from the `TkgServiceConfiguration`, configure the cluster with the global setting not specified or nulled out.

For example, if you wanted to configure a cluster that would inherit the `proxy` setting you could use either of the following approaches:

Option 1: Do not include the `proxy` setting in the cluster specification:

```
...
settings:
  network:
    cni:
      name: antrea
    services:
      cidrBlocks: ["198.51.100.0/12"]
    pods:
      cidrBlocks: ["192.0.2.0/16"]
      serviceDomain: "tanzukubernetescluster.local"
```

Option 2: Include the `proxy` setting in the specification but explicitly set its value to `null`:

```
settings:
  network:
    proxy: null
```

To provision a Tanzu Kubernetes cluster that does not inherit the default value from `TkgServiceConfiguration`, configure the cluster specification with all elements included but empty values.

For example, if the `TkgServiceConfiguration` has a global `proxy` configured and you want to provision a cluster that does not inherit the global `proxy` settings, include the following syntax in your cluster specification:

```
...
settings:
  network:
    proxy:
      httpProxy: ""
      httpsProxy: ""
      noProxy: null
```

Cluster Using a Local Content Library

To provision a Tanzu Kubernetes cluster in an air-gapped environment, create a cluster using the virtual machine image synchronized from a Local Content Library.

To provision a cluster using a local content library image, you must specify that image in the cluster spec. For the `distribution.version` value, you can enter either the full image name or, if you kept the name format from the image directory, you can shorten it to the Kubernetes version. If you want to use a fully qualified version number, replace `-----` with `+`. For example, if you have an OVA image named `photon-3-k8s-v1.20.2---vmware.1-tkg.1.1d4f79a`, the following formats are acceptable.

```
spec:
  distribution:
    version: v1.20
```

```
spec:
  distribution:
    version: v1.20.2
```

```
spec:
  distribution:
    version: v1.20.2+vmware.1-tkg.1
```

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TanzuKubernetesCluster
metadata:
  name: tgks-cluster-9
  namespace: tkgs-cluster-ns
spec:
  topology:
    controlPlane:
      count: 3
      class: best-effort-medium
      storageClass: vwt-storage-policy
    workers:
      count: 3
      class: best-effort-medium
      storageClass: vwt-storage-policy
  distribution:
    version: v1.20.2
  settings:
    network:
      cni:
        name: antrea
    services:
      cidrBlocks: ["198.51.100.0/12"]
    pods:
      cidrBlocks: ["192.0.2.0/16"]
```

Configuration Parameters for the Tanzu Kubernetes Grid Service v1alpha1 API

You can customize the Tanzu Kubernetes Grid Service with global settings for key features, including the container network interface (CNI), proxy server, and TLS certificates. Be aware of trade-offs and considerations when implementing global versus per-cluster functionality.

Note The TKG v1alpha1 API is deprecated and should not be used to create new TKG clusters. Use the TKG v1alpha2 API instead. See [Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API](#).

Caution Configuring the Tanzu Kubernetes Grid Service is a global operation. This means that any change you make to the `TkgServiceConfiguration` specification applies to all Tanzu Kubernetes clusters provisioned by that service. If a rolling update is initiated, either manually or by upgrade, clusters are updated by the changed service spec.

TkgServiceConfiguration Specification

The `TkgServiceConfiguration` specification provides fields for configuring the Tanzu Kubernetes Grid Service instance.

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration-example
spec:
  defaultCNI: <antrea or calico>
  proxy:
    httpProxy: http://<user>:<pwd>@<ip>:<port>
    httpsProxy: http://<user>:<pwd>@<ip>:<port>
    noProxy: [<array of CIDRs to not proxy>]
  trust:
    additionalTrustedCAs:
      - name: <first-cert-name>
        data: <base64-encoded string of a PEM encoded public cert 1>
      - name: <second-cert-name>
        data: <base64-encoded string of a PEM encoded public cert 2>
```

TkgServiceConfiguration Specification Parameters

The table lists and describes each of the `TkgServiceConfiguration` specification parameters. For examples, see [Examples for Configuring the Tanzu Kubernetes Grid Service v1alpha1 API](#).

Field	Value	Description
defaultCNI	antrea or calico	Default CNI for clusters to use. The default is antrea. The other supported CNI is calico.
proxy	Section marker for proxy parameters.	The proxy parameters are httpProxy, httpsProxy, and noProxy. All parameters are required. If any proxy parameter is missing, you cannot create Tanzu Kubernetes clusters.
httpProxy	URI in the form http://<user>:<pwd>@<ip>:<port>	Does not allow the https protocol. If https is used, you cannot create Tanzu Kubernetes clusters.
httpsProxy	URI in the form http://<user>:<pwd>@<ip>:<port>	Does not allow the https protocol. If https is used, you cannot create Tanzu Kubernetes clusters.
noProxy	Array of CIDR blocks to not proxy, for example: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]. Get the required values from the Workload Network on the Supervisor Cluster: Pod CIDRs, Ingress CIDRs, and Egress CIDRs. Refer to the image below for what values to include in the noProxy array field.	You must not proxy the subnets used by the Workload Network on the Supervisor Cluster for Pods, Ingress, and Egress. You do not need to include the Services CIDR from the Supervisor Cluster in the noProxy field. Tanzu Kubernetes clusters do not interact with such services. The endpoints localhost and 127.0.0.1 are automatically not proxied. You do not need to add them to the noProxy field. The Pod and Service CIDRs for Tanzu Kubernetes clusters are automatically not proxied. You do not need to add them to the noProxy field.
trust	Section marker for trust parameters.	Accepts no data.
additionalTrustedCAs	Accepts an array of certificates with name and data for each.	Accepts no data.
name	String	The name of the TLS certificate.
data	String	The base64-encoded string of a PEM encoded public certificate.

Get the required `noProxy` values from the **Workload Network** on the Supervisor Cluster as shown in the image.

compute-cluster ACTIONS ▾

Summary Monitor **Configure** Permissions Hosts VMs Namespaces Datastores Network

Services ▾

- vSphere DRS
- vSphere Availabili...

Configuration ▾

- Quickstart
- General
- Key Provider
- VMware EVC
- VM/Host Groups
- VM/Host Rules
- VM Overrides
- I/O Filters
- Host Options
- Host Profile

Licensing ▾

- vSAN Cluster
- Supervisor Cluster
- Trust Authority
- Alarm Definitions
- Scheduled Tasks

Namespaces ▾

- General
- Network**
- Storage
- Certificates
- Image Registry

vSAN ▾

- Services
- Disk Management
- Fault Domains
- Datastore Sharing

Supervisor Ser... ▾

Network

Below are the network settings for supporting namespaces on this cluster.

> Management Network

▾ Workload Network

vSphere Distributed Switch	wcp_vds_1	
Edge Cluster	EDGECLUSTER1	
DNS Servers	10.20.145.1	EDIT
Pod CIDRs	10.246.0.0/16	EDIT
Services CIDR	10.94.0.0/12	
Ingress CIDRs	192.168.144.0/20	EDIT
Egress CIDRs	192.168.128.0/20	EDIT

When To Use Global or Per-Cluster Configuration Options

The `TkgServiceConfiguration` is a global specification that impacts all Tanzu Kubernetes clusters provisioned by the Tanzu Kubernetes Grid Service instance.

Before editing the `TkgServiceConfiguration` specification, be aware of the per-cluster alternatives that might satisfy your use case instead of a global configuration.

Table 13-5. Global vs. Per-Cluster Configuration Options

Setting	Global Option	Per-Cluster Option
Default CNI	Edit the <code>TkgServiceConfiguration</code> spec. See Examples for Configuring the Tanzu Kubernetes Grid Service v1alpha1 API .	Specify the CNI in the cluster specification. For example, Antrea is the default CNI. To use Calico, specify it in the cluster YAML. See Examples for Provisioning Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API
Proxy Server	Edit the <code>TkgServiceConfiguration</code> spec. See Examples for Configuring the Tanzu Kubernetes Grid Service v1alpha1 API .	Include the proxy server configuration parameters in the cluster spec. See Examples for Provisioning Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API .
Trust Certificates	Edit the <code>TkgServiceConfiguration</code> spec. There are two use cases: configuring an external container registry and certificate-based proxy configuration. See Examples for Configuring the Tanzu Kubernetes Grid Service v1alpha1 API	Yes, you can include custom certificates on a per-cluster basis or override the globally-set <code>trust</code> settings in the cluster specification. See Examples for Provisioning Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API .

Note If a global proxy is configured on the `TkgServiceConfiguration`, that proxy information is propagated to the cluster manifest after the initial deployment of the cluster. The global proxy configuration is added to the cluster manifest only if there is no proxy configuration fields present when creating the cluster. In other words, per-cluster configuration takes precedence and will overwrite a global proxy configuration. For more information, see [Configuration Parameters for the Tanzu Kubernetes Grid Service v1alpha1 API](#).

Before editing the `TkgServiceConfiguration` specification, be aware of the ramifications of applying the setting at the global level.

Field	Applied	Impact on Existing Clusters If Added/ Changed	Per-Cluster Overriding on Cluster Creation	Per-Cluster Overriding on Cluster Update
defaultCNI	Globally	None	Yes, you can override the global setting on cluster creation	No, you cannot change the CNI for an existing cluster; if you used the globally set default CNI on cluster creation, it cannot be changed
proxy	Globally	None	Yes, you can override the global setting on cluster creation	Yes, with U2+, you can override the global setting on cluster update
trust	Globally	None	Yes, you can override the global setting on cluster creation	Yes, with U2+, you can override the global setting on cluster update

Propagating Global Configuration Changes to Existing Clusters

Settings made at the global level in the `TkgServiceConfiguration` may not be automatically propagated to existing clusters. For example, if you make changes to either the `proxy` or the `trust` settings in the `TkgServiceConfiguration`, such changes may not affect clusters that are already provisioned.

To manually propagate a global change to an existing cluster, you must patch the Tanzu Kubernetes cluster to make the cluster inherit the changes made to the `TkgServiceConfiguration`.

For example:

```
kubectl patch tkc <CLUSTER_NAME> -n <NAMESPACE> --type merge -p '{"spec":{"settings":{"network":{"proxy": null}}}}'
```

```
kubectl patch tkc <CLUSTER_NAME> -n <NAMESPACE> --type merge -p '{"spec":{"settings":{"network":{"trust": null}}}}'
```

Examples for Configuring the Tanzu Kubernetes Grid Service v1alpha1 API

Refer to the examples to customize the Tanzu Kubernetes Grid Service with global configuration settings for the container network interface, proxy server, and TLS certificates.

About Configuring the Tanzu Kubernetes Grid Service

You can customize the Tanzu Kubernetes Grid Service by changing the default CNI, adding a global proxy server, and adding trusted certificates. See [Configuration Parameters for the Tanzu Kubernetes Grid Service v1alpha1 API](#).

Caution Editing the Tanzu Kubernetes Grid Service specification results in global changes to all clusters provisioned by that service, including new clusters and existing clusters that are upgraded manually or automatically.

Note The TKG v1alpha1 API is deprecated and should not be used to create new TKG clusters. Use the TKG v1alpha2 API instead. See [Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API](#).

Prerequisite: Configure Kubectl Editing

To scale a Tanzu Kubernetes cluster, you update the cluster manifest using the command `kubectl edit tanzukubernetescluster/CLUSTER-NAME`. The `kubectl edit` command opens the cluster manifest in the text editor defined by your `KUBE_EDITOR` or `EDITOR` environment variable. For instructions on setting up the environment variable, see [Specify a Default Text Editor for Kubectl](#).

When you save the specification changes, `kubectl` reports that the edits were successfully recorded. To cancel, simply close the editor without saving.

Configure the Default CNI

The Tanzu Kubernetes Grid Service provides a default container network interface (CNI) for Tanzu Kubernetes clusters. The default configuration lets you create clusters without the need to specify the CNI. You can change the default CNI by editing the service specification.

The Tanzu Kubernetes Grid Service supports two CNIs: Antrea and Calico, with Antrea being the default. For more information, see [Tanzu Kubernetes Grid Service Cluster Networking](#).

You can override the default CNI by explicitly specifying the CNI to use. Alternatively, you can change the default CNI by editing the TKG Service controller for CNIs.

- 1 Authenticate with the Supervisor Cluster.

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

- 2 Switch context to the target vSphere Namespace.

```
kubectl config use-context tkgs-cluster-ns
```

- 3 List the default CNI.

```
kubectl get tkgserviceconfigurations
```

Example result:

NAME	DEFAULT CNI
tkg-service-configuration	antrea

- 4 Load for editing the Tanzu Kubernetes Grid Service specification.

```
kubectl edit tkgserviceconfigurations tkg-service-configuration
```

The system opens the `tkg-service-configuration` specification in the default text editor defined by your `KUBE_EDITOR` or `EDITOR` environment variable.

- 5 Edit the `spec.defaultCNI` value.

For example, change from:

```
spec:
  defaultCNI: antrea
```

Change to:

```
spec:
  defaultCNI: calico
```

- 6 To apply the changes, save the file in the text editor. To cancel, close the editor without saving.

When you save the change in the text editor, `kubectl` updates the `tkg-service-configuration` service specification.

- 7 Verify that the default CNI is updated.

```
kubectl get tkgserviceconfigurations
```

The default CNI is updated. Any cluster provisioned with default network settings uses the default CNI.

NAME	DEFAULT CNI
tkg-service-configuration	calico

Configure a Global Proxy Server

To enable a global proxy server, add the proxy server parameters to the `TkgServiceConfiguration`. For a description of the required fields, see [Configuration Parameters for the Tanzu Kubernetes Grid Service v1alpha1 API](#).

- 1 Authenticate with the Supervisor Cluster.

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

- 2 Switch context to the target vSphere Namespace.

```
kubectl config use-context tkgs-cluster-ns
```

- 3 Get the current configuration.

```
kubectl get tkgserviceconfigurations
```

Example result:

NAME	DEFAULT CNI
tkg-service-configuration	antrea

- 4 Load for editing the Tanzu Kubernetes Grid Service specification.

```
kubectl edit tkgserviceconfigurations tkg-service-configuration
```

The system opens the `tkg-service-configuration` specification in the default text editor defined by your `KUBE_EDITOR` or `EDITOR` environment variable.

- 5 Add the `spec.proxy` subsection with each required field, including `httpProxy`, `httpsProxy`, and `noProxy`.

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  ...
  name: tkg-service-configuration-example
  resourceVersion: "44170525"
  selfLink: /apis/run.tanzu.vmware.com/v1alpha1/tkgserviceconfigurations/tkg-service-configuration
  uid: 10347195-5f0f-490e-8ae1-a758a724c0bc
spec:
  defaultCNI: antrea
  proxy:
    httpProxy: http://<user>:<pwd>@<ip>:<port>
    httpsProxy: http://<user>:<pwd>@<ip>:<port>
    noProxy: [SVC-POD-CIDRs, SVC-EGRESS-CIDRs, SVC-INGRESS-CIDRs]
```

- 6 Populate each proxy field with the appropriate values. For a description of each field, see [Configuration Parameters for the Tanzu Kubernetes Grid Service v1alpha1 API](#).

The required values for the `noProxy` field come from the **Workload Network** on the Supervisor Cluster. Refer to the picture at the above topic on where to get these values.

For example:

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  ...
  name: tkg-service-configuration-example
  resourceVersion: "44170525"
  selfLink: /apis/run.tanzu.vmware.com/v1alpha1/tkgserviceconfigurations/tkg-service-configuration
  uid: 10347195-5f0f-490e-8ae1-a758a724c0bc
spec:
  defaultCNI: antrea
  proxy:
    httpProxy: http://user:password@10.186.102.224:3128
    httpsProxy: http://user:password@10.186.102.224:3128
    noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]
```

- 7 To apply the changes, save the file in the text editor. To cancel, close the editor without saving.

When you save the changes in the text editor, `kubectl` updates Tanzu Kubernetes Grid Service with the configuration defined in the `tkg-service-configuration` service specification.

- 8 Verify that the Tanzu Kubernetes Grid Service is updated with the proxy settings.

```
kubectl get tkgserviceconfigurations -o yaml
```

- 9 To verify, provision a Tanzu Kubernetes cluster. See [Workflow for Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API](#).

Use the following command to confirm that the cluster is using the proxy.

```
kubectl get tkc CLUSTER-NAME -n NAMESPACE -o yaml
```

Certificate-Based Proxy Configuration

Using a proxy server to route internet traffic is a hard requirement for some environments. For example, a company in a highly regulated industry such as a financial institution requires all internet traffic go through a corporate proxy.

You can configure the Tanzu Kubernetes Grid Service to provision Tanzu Kubernetes clusters to use a proxy server for outbound HTTP/S traffic. For more information, see [Configuration Parameters for the Tanzu Kubernetes Grid Service v1alpha1 API](#).

As shown in the example, you can add trusted certificates for the proxy server to the `TkgServiceConfiguration` specification.

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration-example
spec:
  defaultCNI: antrea
  proxy:
    httpProxy: http://user:password@10.186.102.224:3128
    httpsProxy: http://user:password@10.186.102.224:3128
    noProxy: [10.246.0.0/16,192.168.144.0/20,192.168.128.0/20]
  trust:
    additionalTrustedCAs:
      - name: first-cert-name
        data: base64-encoded string of a PEM encoded public cert 1
      - name: second-cert-name
        data: base64-encoded string of a PEM encoded public cert 2
```

External Private Registry Configuration

You can configure the Tanzu Kubernetes Grid Service with custom certificates for connecting Tanzu Kubernetes clusters with an external private registry. For more information, see [Use an External Container Registry with Tanzu Kubernetes Clusters](#).

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration-example
spec:
  defaultCNI: antrea
  trust:
    additionalTrustedCAs:
      - name: harbor-vm-cert
        data: <<<base64-encoded string of a PEM encoded public cert>>>>
```

Scale a Tanzu Kubernetes Cluster Using the Tanzu Kubernetes Grid Service v1alpha1 API

You can scale a Tanzu Kubernetes cluster horizontally by changing the number of nodes or vertically by changing the virtual machine class hosting the nodes.

Supported Scaling Operations

The table lists the supported scaling operations for Tanzu Kubernetes clusters.

Table 13-6. Supported Scaling Operations for Tanzu Kubernetes Clusters

Node	Horizontal Scale Out	Horizontal Scale In	Vertical Scale
Control Plane	Yes	No	Yes
Worker	Yes	Yes	Yes

Keep in mind the following considerations:

- While vertically scaling a cluster node, workloads may no longer be able to run on the node for lack of available resource. For this reason horizontal scaling may be the preferred approach.
- VM classes are not immutable. If you scale out a Tanzu Kubernetes cluster after editing a VM class used by that cluster, new cluster nodes use the updated class definition, but existing cluster nodes continue to use the initial class definition, resulting in a mismatch. See [Virtual Machine Classes for Tanzu Kubernetes Clusters](#).

Scaling Prerequisite: Configure Kubectl Editing

To scale a Tanzu Kubernetes cluster, you update the cluster manifest using the command `kubectl edit tanzukubernetescluster/CLUSTER-NAME`. The `kubectl edit` command opens the cluster manifest in the text editor defined by your `KUBE_EDITOR` or `EDITOR` environment variable. For instructions on setting up the environment variable, see [Specify a Default Text Editor for Kubectl](#).

When you save the manifest changes, `kubectl` reports that the edits were successfully recorded, and the cluster is updated with the changes.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 edited
```

To cancel, simply close the editor without saving.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
Edit cancelled, no changes made.
```

Scale Out the Control Plane

You can scale out a Tanzu Kubernetes cluster by increasing number of control plane nodes from 1 to 3. The number of control plane nodes must be odd. You cannot scale in the control plane.

- 1 Authenticate with the Supervisor Cluster.

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

- 2 Switch context to the vSphere Namespace where the Tanzu Kubernetes cluster is running.

```
kubectl config use-context tkgs-cluster-ns
```

- 3 List the Kubernetes clusters that are running in the namespace.

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

- 4 Get the number of nodes running in the target cluster.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

For example, the following cluster has 1 control plane nodes and 3 worker nodes.

NAME	CONTROL PLANE	WORKER	DISTRIBUTION	AGE	PHASE
tkgs-cluster-1	1	3	v1.18.5+vmware.1-tkg.1.886c781	1d	running

- 5 Load the cluster manifest for editing using the `kubectl edit` command.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
```

The cluster manifest opens in the text editor defined by your `KUBE_EDITOR` or `EDITOR` environment variables.

- 6 Locate the `spec.topology.controlPlane.count` parameter and increase the number of nodes from 1 to 3.

```
...
controlPlane:
  count: 1
...
```

```
...
ControlPlane:
  count: 3
...
```

- 7 To apply the changes, save the file in the text editor. To cancel, close the editor without saving.

When you save the file, `kubectl` applies the changes to the cluster. In the background, the Virtual Machine Service on the Supervisor Cluster provisions the new worker node.

- 8 Verify that the new nodes are added.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

The scaled out control plane now has 3 nodes.

NAME	CONTROL PLANE	WORKER	DISTRIBUTION	AGE	PHASE
tkgs-cluster-1	3	3	v1.18.5+vmware.1-tkg.1.886c781	1d	running

Scale Out Worker Nodes

You can scale out a Tanzu Kubernetes cluster by increasing the number of worker nodes using `kubectl`.

- 1 Authenticate with the Supervisor Cluster.

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

- 2 Switch context to the vSphere Namespace where the Tanzu Kubernetes cluster is running.

```
kubectl config use-context tkgs-cluster-ns
```

- 3 List the Kubernetes clusters that are running in the namespace.

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

- 4 Get the number of nodes running in the target cluster.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

For example, the following cluster has 3 control plane nodes and 3 worker nodes.

NAME	CONTROL PLANE	WORKER	DISTRIBUTION	AGE	PHASE
tkgs-cluster-1	3	3	v1.18.5+vmware.1-tkg.1.886c781	1d	running

- 5 Load the cluster manifest for editing using the `kubectl edit` command.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
```

The cluster manifest opens in the text editor defined by your `KUBE_EDITOR` or `EDITOR` environment variables.

- 6 Locate the `spec.topology.workers.count` parameter and increase the number of nodes.

```
...
workers:
  count: 3
...
```

```
...
workers:
  count: 4
...
```

- 7 To apply the changes, save the file in the text editor. To cancel, close the editor without saving.

When you save the file, `kubectl` applies the changes to the cluster. In the background, the Virtual Machine Service on the Supervisor Cluster provisions the new worker node.

- Verify that the new worker node is added.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

After scaling out, the cluster has 4 worker nodes.

NAME	CONTROL PLANE	WORKER	DISTRIBUTION	AGE	PHASE
tkgs-cluster-1	3	4	v1.18.5+vmware.1-tkg.1.886c781	1d	running

Scale In Worker Nodes

You can scale in a Tanzu Kubernetes cluster by decreasing the number of worker nodes. Scaling in the control plane is not supported.

- Authenticate with the Supervisor Cluster.

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

- Switch context to the vSphere Namespace where the Tanzu Kubernetes cluster is running.

```
kubectl config use-context tkgs-cluster-ns
```

- List the Kubernetes clusters that are running in the namespace.

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```

- Get the number of nodes running in the target cluster.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

For example, the following cluster has 3 control plane nodes and 3 worker nodes.

NAME	CONTROL PLANE	WORKER	DISTRIBUTION	AGE	PHASE
tkgs-cluster-1	3	4	v1.18.5+vmware.1-tkg.1.886c781	1d	running

- Load the cluster manifest for editing using the `kubectl edit` command.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
```

The cluster manifest opens in the text editor defined by your `KUBE_EDITOR` or `EDITOR` environment variables.

- 6 Locate the `spec.topology.workers.count` parameter and increase the number of nodes.

```
...
workers:
  count: 4
...
```

```
...
workers:
  count: 2
...
```

- 7 To apply the changes, save the file in the text editor. To cancel, close the editor without saving.

When you save the file, `kubectl` applies the changes to the cluster. In the background, the Virtual Machine Service on the Supervisor Cluster provisions the new worker node.

- 8 Verify that the worker nodes were removed.

```
kubectl get tanzukubernetescluster tkgs-cluster-1
```

After scaling in, the cluster has 2 worker nodes.

NAME	CONTROL PLANE	WORKER	DISTRIBUTION	AGE	PHASE
tkgs-cluster-1	3	2	v1.18.5+vmware.1-tkg.1.886c781	1d	running

Scale a Cluster Vertically

You can vertically scale a Tanzu Kubernetes cluster by changing the virtual machine class used to host the cluster nodes. Vertical scaling is supported for both control plane and worker nodes.

The Tanzu Kubernetes Grid Service supports scaling cluster nodes vertically through the rolling update mechanism built into the service. If you change the `VirtualMachineClass` definition, the service rolls out new nodes with that new class and spins down the old nodes. See [Update Tanzu Kubernetes Clusters](#).

- 1 Authenticate with the Supervisor Cluster.

```
kubectl vsphere login --server=SVC-IP-ADDRESS --vsphere-username USERNAME
```

- 2 Switch context to the vSphere Namespace where the Tanzu Kubernetes cluster is running.

```
kubectl config use-context tkgs-cluster-ns
```

- 3 List the Kubernetes clusters that are running in the namespace.

```
kubectl get tanzukubernetescluster -n tkgs-cluster-ns
```


- Describe the target Tanzu Kubernetes cluster and check the VM class.

```
kubectl describe tanzukubernetescluster tkgs-cluster-2
```

For example, the following cluster is using the best-effort-small VM class.

```
Spec:
  ...
  Topology:
    Control Plane:
      Class:      best-effort-small
      ...
    Workers:
      Class:      best-effort-small
      ...
```

- List and describe the available VM classes.

```
kubectl get virtualmachineclassbinding
```

```
kubectl describe virtualmachineclassbinding
```

Note The VM class you want to use must be bound the vSphere Namespace. See [Virtual Machine Classes for Tanzu Kubernetes Clusters](#).

- Open for editing the target cluster manifest.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-2
```

The cluster manifest opens in the text editor defined by your KUBE_EDITOR or EDITOR environment variables.

- Edit the manifest by changing the VM class.

For example, edit the cluster manifest to use the `guaranteed-xlarge` VM class for control plane and worker nodes.

```
spec:
  topology:
    controlPlane:
      class: guaranteed-xlarge
      ...
    workers:
      class: guaranteed-xlarge
      ...
```

- To apply the changes, save the file in the text editor. To cancel, close the editor without saving.

When you save the file, `kubectl` applies the changes to the cluster. In the background, the Tanzu Kubernetes Grid Service provisions the new nodes and deletes the old ones. For a description of the rolling update process, see [About Tanzu Kubernetes Grid Service Cluster Updates](#).

- 9 Verify that the cluster is being updated.

```
kubectl get tanzukubernetescluster
```

NAME	CONTROL PLANE	WORKER	DISTRIBUTION	AGE	PHASE
tkgs-cluster-1	3	3	v1.18.5+vmware.1-tkg.1.c40d30d	21h	updating

Delete a Tanzu Kubernetes Cluster

Use `kubectl` to delete a Tanzu Kubernetes cluster provisioned by the Tanzu Kubernetes Grid Service.

When you delete a Tanzu Kubernetes cluster using `kubectl`, Kubernetes garbage collection ensures that all dependent resources are deleted.

Note Do not attempt to delete a Tanzu Kubernetes cluster using the vSphere Client or the vCenter Server CLI.

Procedure

- 1 Authenticate with the Supervisor Cluster.

See [Connect to the Supervisor Cluster as a vCenter Single Sign-On User](#).

- 2 Change context to the vSphere Namespace where the Tanzu Kubernetes you want to delete is provisioned.

```
kubectl config use-context CLUSTER-NAMESPACE
```

For example:

```
kubectl config use-context tkgs-ns-1
```

- 3 List the Tanzu Kubernetes clusters in the namespace.

```
kubectl get clusters
```

For example:

```
kubectl get clusters
```

NAME	PHASE
tkgs-cluster-1	provisioned

- 4 Delete the Tanzu Kubernetes cluster using the following syntax.

```
kubectl delete tanzukubernetescluster --namespace CLUSTER-NAMESPACE CLUSTER-NAME
```

For example:

```
kubectl delete tanzukubernetescluster --namespace tkgs-ns-1 tkgs-cluster-1
```

Expected result:

```
tanzukubernetescluster.run.tanzu.vmware.com "tkgs-cluster-1" deleted
```

5 Verify that the cluster is deleted.

```
kubectl get clusters
```

For example:

```
kubectl get clusters
No resources found in tkgs-ns-1 namespace.
```

6 Delete the cluster context from the kubeconfig file.

```
kubectl config delete-context CONTEXT
```

For example:

```
kubectl config get-contexts
CURRENT  NAME              CLUSTER          AUTHINFO
NAMESPACE
          192.0.2.1        192.0.2.1        wcp:192.0.2.1:administrator@vsphere.local
          tkgs-cluster-1 192.0.2.6        wcp:192.0.2.6:administrator@vsphere.local
*         tkgs-ns-1      192.0.2.7        wcp:192.0.2.7:administrator@vsphere.local
tkgs-ns-1
```

```
kubectl config delete-context tkgs-cluster-1
deleted context tkgs-cluster-1 from $HOME/.kube/config
```

```
kubectl config get-contexts
CURRENT  NAME              CLUSTER          AUTHINFO
NAMESPACE
          192.0.2.1        192.0.2.1        wcp:192.0.2.1:administrator@vsphere.local
*         tkgs-ns-1      192.0.2.7        wcp:192.0.2.7:administrator@vsphere.local
tkgs-ns-1
```

Specify a Default Text Editor for Kubectl

To help you provision, operate, and maintain Tanzu Kubernetes clusters, specify a default text editor for kubectl .

Purpose

After you provision a Tanzu Kubernetes cluster you need to maintain it. Typical maintenance tasks include upgrading the Kubernetes version and scaling cluster nodes. To perform such tasks you update the cluster manifest.

The most convenient way to update the manifest for a provisioned cluster is to use the `kubectl edit` command. This command opens the Kubernetes manifest in a text editor of your choice. When you save the changes, Kubernetes automatically applies the changes and updates the cluster.

To use the `kubectl edit` command, create a `KUBE_EDITOR` environment variable and specify your preferred text editor as the variable value. In addition, append the watch flag (`-w`) to the value so that `kubectl` knows when you have committed (saved) your changes.

Windows

On Windows, create a system environment variable named `KUBE_EDITOR` with the value set to the path of your preferred text editor. Append the watch flag (`-w`) to the value.

For example, the following environment variable sets Visual Studio Code as the default text editor for `kubectl` and includes the watch flag so that Kubernetes knows when you save your changes:

```
KUBE_EDITOR=code -w
```

Mac OS

On Mac OS, create an environment variable named `KUBE_EDITOR` with the value set to the path of your preferred text editor. Append the watch flag (`-w`) to the value so that `kubectl` knows when you have committed (saved) your changes.

For example, the following addition to the `.bash_profile` sets Sublime as the default text editor for `kubectl` and includes the watch flag so `kubectl` knows when you have saved any changes.

```
export KUBE_EDITOR="/Applications/Sublime.app/Contents/SharedSupport/bin/subl -w"
```

Linux

On Linux (Ubuntu, for example), typically the default command-line `EDITOR` is Vim. If so, no further action is needed to use the `kubectl edit` command. If you want to use a different editor, create an environment variable named `KUBE_EDITOR` with the value set to the path of your preferred text editor.

Operating Tanzu Kubernetes Clusters

The Tanzu Kubernetes Grid Service includes custom resources that you use to operate Tanzu Kubernetes clusters. In addition, because of the tight integration with the vSphere infrastructure, you can use familiar vSphere tools to help you manage and maintain Tanzu Kubernetes clusters.

Monitor Tanzu Kubernetes Cluster Status Using kubectl

You can monitor the status of provisioned Tanzu Kubernetes clusters using kubectl.

Procedure

- 1 Authenticate with the Supervisor Cluster. See [Connect to the Supervisor Cluster as a vCenter Single Sign-On User](#).
- 2 Switch to the vSphere Namespace where the cluster is running.

```
kubectl config use-context SUPERVISOR-NAMESPACE
```

- 3 View a list of the Tanzu Kubernetes clusters running in the namespace.

```
kubectl get tanzukubernetesclusters
```

This command returns the status of the cluster. For a description of the status fields, see [Tanzu Kubernetes Cluster Lifecycle Status in kubectl](#).

- 4 View the details of the cluster.

```
kubectl describe tanzukubernetescluster <cluster-name>
```

The command returns the details of the cluster. In the Status section of the command output, you see detailed information about the cluster.

```
...
Status:
  Addons:
    Cni:
      Name:      calico
      Status:    applied
    Csi:
      Name:      pvcsi
      Status:    applied
    Psp:
      Name:      defaultpsp
      Status:    applied
  Cloudprovider:
    Name: vmware-guest-cluster
  Cluster API Status:
    API Endpoints:
      Host: 10.161.90.22
      Port: 6443
    Phase: provisioned
```

```

Node Status:
  test-tanzu-cluster-control-plane-0:      ready
  test-tanzu-cluster-workers-0-749458f97c-971jv:  ready
Phase:                                     running
Vm Status:
  test-tanzu-cluster-control-plane-0:      ready
  test-tanzu-cluster-workers-0-749458f97c-971jv:  ready
Events:                                     <none>

```

- 5 Run additional `kubectl` commands to view more details about the cluster. See [Use Tanzu Kubernetes Cluster Operational Commands](#).

Check Tanzu Kubernetes Cluster Readiness

When the Tanzu Kubernetes Grid Service provisions a Tanzu Kubernetes cluster, several status conditions are reported that you can use to get direct insight into key aspects of machine health.

Check TanzuKubernetesCluster Readiness

You can use the `TanzuKubernetesCluster` readiness conditions to determine which, if any, phase or component is not ready. See [ControlPlaneReady Condition and Reasons](#).

Once you check for cluster readiness, to diagnose further you can use `capwcluster` and machine conditions to look further in more detail failure. See [Check Tanzu Kubernetes Machine Health](#) and [Check Tanzu Kubernetes Cluster Health](#).

To check the readiness of a Tanzu Kubernetes cluster:

- 1 Log in to the Supervisor Cluster.
- 2 Switch context to the namespace where the target cluster is provisioned. For example:

```
kubectl config use-context tkgs-cluster-ns
```

- 3 Run the command `kubectl get tkc -o yaml`. The system displays the cluster readiness conditions. For example:

```

status:
  addons:
    authsvc:
      conditions:
        - lastTransitionTime: "2021-01-30T19:53:54Z"
          status: "True"
          type: AuthServiceProvisioned
      name: authsvc
      status: applied
      version: 0.1-66-g8b8f07f
    cloudprovider:
      conditions:
        - lastTransitionTime: "2021-01-30T19:53:53Z"
          status: "True"
          type: CPIProvisioned
      name: vmware-guest-cluster

```

```

    status: applied
    version: 0.1-77-g5875817
  cni:
    conditions:
      - lastTransitionTime: "2021-01-30T19:53:53Z"
        status: "True"
        type: CNIProvisioned
    name: calico
    status: applied
    version: 1.16.14+vmware.1-tkg.1.ada4837
  csi:
    conditions:
      - lastTransitionTime: "2021-01-30T19:53:54Z"
        status: "True"
        type: CSIProvisioned
    name: pvcsi
    status: applied
    version: v0.0.1.alpha+vmware.79-7ecdcb1
  dns:
    conditions:
      - lastTransitionTime: "2021-01-30T19:53:48Z"
        status: "True"
        type: CoreDNSProvisioned
    name: CoreDNS
    status: applied
    version: v1.6.2_vmware.10
  proxy:
    conditions:
      - lastTransitionTime: "2021-01-30T19:53:48Z"
        status: "True"
        type: KubeProxyProvisioned
    name: kube-proxy
    status: applied
    version: 1.16.14+vmware.1
  psp:
    conditions:
      - lastTransitionTime: "2021-01-30T19:53:47Z"
        status: "True"
        type: PSPProvisioned
    name: defaultpsp
    status: applied
    version: v1.16.14+vmware.1-tkg.1.ada4837
  clusterApiStatus:
    apiEndpoints:
      - host: 192.168.1.2
        port: 6443
    phase: Provisioned
  conditions:
    - lastTransitionTime: "2021-01-30T19:53:54Z"
      status: "True"
      type: AddonsReady
    - lastTransitionTime: "2021-01-30T19:51:11Z"
      status: "True"
      type: ControlPlaneReady
    - lastTransitionTime: "2021-01-30T19:51:04Z"

```

```

message: 3/3 Control Plane Node(s) healthy. 1/1 Worker Node(s) healthy
status: "True"
type: NodesHealthy
- lastTransitionTime: "2021-01-31T21:22:45Z"
  status: "True"
  type: ProviderServiceAccountsReady
- lastTransitionTime: "2021-01-30T19:53:50Z"
  status: "True"
  type: RoleBindingSynced
- lastTransitionTime: "2021-01-30T19:53:58Z"
  status: "True"
  type: ServiceDiscoveryReady
- lastTransitionTime: "2021-01-30T19:53:59Z"
  status: "True"
  type: StorageClassSynced
- lastTransitionTime: "2021-01-27T11:34:53Z"
  status: "True"
  type: TanzuKubernetesReleaseCompatible
- lastTransitionTime: "2021-01-27T11:34:54Z"
  message: '[1.17.13+vmware.1-tkg.2.2c133ed]'
  severity: Info
  status: "True"
  type: UpdatesAvailable

```

ControlPlaneReady Condition and Reasons

The table lists and describes the `ControlPlaneReady` condition.

Table 13-7. ControlPlaneReady Condition

Condition Type	Description
<code>ControlPlaneReady</code>	Reports if the control plane nodes are ready and functional for the cluster.

The table lists and describes the reasons why the `ControlPlaneReady` condition may be false.

Table 13-8. ControlPlaneReady False Reasons

Reason	Severity	Description
<code>WaitingForClusterInfrastructure</code>		Indicates that the cluster is waiting for prerequisites that are necessary for running machines, such as a load balancer. This reason is only used if the <code>InfrastructureCluster</code> is not reporting its own ready condition.
<code>WaitingForControlPlaneInitialized</code>		Indicates that the first control plane node is initializing.

Table 13-8. ControlPlaneReady False Reasons (continued)

Reason	Severity	Description
WaitingForControlPlane		Reflects the condition of KubeadmControlPlane. This reason is used if the KubeadmControlPlane is not reporting its own ready condition.
Waiting for cluster infrastructure to be ready	Message	Indicates that the cluster is waiting for prerequisites that are necessary for running machines, such as networking and load balancers.

NodesHealthy Condition and Reasons

The table lists and describes the `NodesHealthy` condition.

Table 13-9. NodesHealthy Condition

Condition Type	Description
<code>NodesHealthy</code>	Reports the status of <code>TanzuKubernetesCluster</code> nodes.

The table lists and describes the reason for the `NodesHealthy` condition not being true.

Table 13-10. NodesHealthy False Reason

Reason	Severity	Description
WaitingForNodesHealthy		Documents that not all the Nodes are healthy.

Addons Conditions and Reasons

The table lists and describes the conditions related to cluster addon components.

Table 13-11. Addons Conditions

Condition Type	Description
<code>AddonsReady</code>	Summary of conditions for <code>TanzuKubernetesCluster</code> addons (CoreDNS, KubeProxy, CSP, CPI, CNI, AuthSvc) .
<code>CNIProvisioned</code>	Documents the status of <code>TanzuKubernetesCluster</code> container network interface (CNI) addon.
<code>CSIProvisioned</code>	Documents the status of <code>TanzuKubernetesCluster</code> container storage interface (CSI) addon.
<code>CPIProvisioned</code>	Documents the status of <code>TanzuKubernetesCluster</code> cloud provider interface (CPI) addon.
<code>KubeProxyProvisioned</code>	Documents the status of <code>TanzuKubernetesCluster</code> KubeProxy addon.
<code>CoreDNSProvisioned</code>	Documents the status of <code>TanzuKubernetesCluster</code> CoreDNS addon.

Table 13-11. Addons Conditions (continued)

Condition Type	Description
AuthServiceProvisioned	Documents the status of TanzuKubernetesCluster AuthService addon.
PSPProvisioned	Documents the status of PodSecurityPolicy.

The table lists and describes the reasons addon conditions not being true.

Table 13-12. Addons False Reasons

Reason	Severity	Description
AddonsReconciliationFailed		Summarized reason for all addons reconciliation failures.
CNIProvisioningFailed	Warning	Documents CNI addon failed to create or update.
CSIProvisioningFailed	Warning	Documents CSI addon failed to create or update.
CPIProvisioningFailed	Warning	Documents CPI addon failed to create or update.
KubeProxyProvisioningFailed	Warning	Documents KubeProxy addon failed to create or update.
CoreDNSProvisioningFailed	Warning	Documents CoreDNS addon failed to create or update.
AuthServiceProvisioningFailed	Warning	Documents AuthService addon failed to create or update.
AuthServiceUnManaged		Documents AuthService is not managed by controller.
PSPProvisioningFailed	Warning	Documents PodSecurityPolicy addons failed to create or update.

Other Conditions and Reasons

The table lists and describes conditions for StorageClass and RoleBinding synchronization, ProviderServiceAccount resource reconciliation, ServiceDiscovery, and TanzuKubernetesCluster compatability.

Table 13-13. Other Conditions

Condition	Description
StorageClassSynced	Documents the status of StorageClass synchronization from Supervisor Cluster to workload cluster.
RoleBindingSynced	Documents the status of RoleBinding synchronization from the Supervisor Cluster to the workload cluster.
ProviderServiceAccountsReady	Documents the status of provider service accounts and related Roles, RoleBindings and Secrets are created.

Table 13-13. Other Conditions (continued)

Condition	Description
<code>ServiceDiscoveryReady</code>	Documents the status of service discoveries.
<code>TanzuKubernetesReleaseCompatible</code>	Indicates if the <code>TanzuKubernetesCluster</code> is compatible with the <code>TanzuKubernetesRelease</code> .

The table lists and describes the reasons for other conditions not being true.

Table 13-14. Other Reasons

Reason	Severity	Description
<code>StorageClassSyncFailed</code>		Reports the <code>StorageClass</code> synchronization failed.
<code>RoleBindingSyncFailed</code>		Reports the <code>RoleBinding</code> synchronization failed.
<code>ProviderServiceAccountsReconciliationFailed</code>		Reports that provider service accounts related resources reconciliation failed.
<code>SupervisorHeadlessServiceSetupFailed</code>		Documents the headless service setup for Supervisor Cluster API server failed.

View the Full Resource Hierarchy for a Tanzu Kubernetes Cluster

You can view the full resource hierarchy for a Tanzu Kubernetes cluster using `kubectl`. Viewing the complete list of cluster resources can help you pinpoint resources that might be causing problems.

Prerequisites

Connect to the Supervisor Cluster. See [Connect to the Supervisor Cluster as a vCenter Single Sign-On User](#).

Procedure

- 1 Switch context to use the target cluster context.

```
kubectl config use-context CLUSTER-NAME
```

- 2 Run the following command to view the Cluster API cluster resource.

```
kubectl describe clusters.cluster.x-k8s.io CLUSTER-NAME
```

This command returns the resource hierarchy for the named cluster, such as the following: Namespace, API version, Resource version.

View Tanzu Kubernetes Cluster Lifecycle Status

You can view the lifecycle status of Tanzu Kubernetes clusters in the vSphere inventory and using kubectl.

Tanzu Kubernetes Cluster Lifecycle Status in vSphere

The table lists and describes the Tanzu Kubernetes cluster status information that appears in the vSphere inventory. To view this information, see [Monitor Tanzu Kubernetes Cluster Status Using the vSphere Client](#).

Table 13-15. Tanzu Kubernetes Cluster Status in the vSphere Inventory

Field	Description	Example
Name	The user-defined name of the cluster.	tkg-cluster-01
Creation Time	The date and time of cluster creation.	Mar 17, 2020, 11:42:46 PM
Phase	The lifecycle status of the cluster. See Table 13-17. Cluster Lifecycle Phase Status .	creating
Worker Count	The number of worker nodes in the cluster.	1 or 2 or 5
Distribution Version	The version of Kubernetes software that the cluster is running.	v1.16.6+vmware.1-tkg.1.7144628
Control Plane Address	The IP address of the cluster control plane load balancer.	192.168.123.2

Tanzu Kubernetes Cluster Lifecycle Status in kubectl

The table lists and describes the Tanzu Kubernetes cluster status information that appears in kubectl. To view this information, see [Monitor Tanzu Kubernetes Cluster Status Using kubectl](#).

Table 13-16. Tanzu Kubernetes Cluster Status in kubectl

Field	Description	Example
NAME	Name of the cluster.	tkg-cluster-01
CONTROL PLANE	Number of control plane nodes in the cluster.	3
WORKER	Number of worker nodes in the cluster.	5
DISTRIBUTION	Kubernetes version that the cluster is running.	v1.16.6+vmware.1-tkg.1.5b5608b
AGE	Number of days the cluster has been running.	13d
PHASE	The lifecycle status of the cluster. See Table 13-17. Cluster Lifecycle Phase Status .	running

Cluster Lifecycle Phase Status

The table lists and describes the status for each phase of a cluster lifecycle. See [Table 13-17. Cluster Lifecycle Phase Status](#).

Table 13-17. Cluster Lifecycle Phase Status

Phase	Description
creating	Cluster provisioning can start, the control plane is being created, or the control plane is created but not initialized.
deleting	The cluster is being deleted.
failed	The creation of the cluster control plane failed and user intervention is likely required.
running	The infrastructure is created and configured, and the control plane is fully initialized.
updating	The cluster is being updated.

Use Tanzu Kubernetes Cluster Operational Commands

You can manage Tanzu Kubernetes clusters using custom kubectl commands. These commands are made available by custom resources created by the Tanzu Kubernetes Grid Service.

Custom Commands to Manage Tanzu Kubernetes Clusters

The table lists and describes kubectl commands for managing Tanzu Kubernetes clusters.

Table 13-18. Custom Commands to Manage Tanzu Kubernetes Clusters

Command	Description
<code>kubectl get tanzukubernetescluster</code>	Lists the clusters in the current namespace.
<code>kubectl get tkc</code>	Short form version of the preceding command.
<code>kubectl describe tanzukubernetescluster CLUSTER-NAME</code>	Describe the specified cluster, showing the expressed state, status, and events. When provisioning is complete, this command shows the virtual IP created for the load balancer that fronts Kubernetes API endpoints.
<code>kubectl get cluster-api</code>	Lists the Cluster API resources supporting the clusters in the current namespace, including resources from the Cluster API project and from the Cluster API Provider used by the Tanzu Kubernetes Grid Service.
<code>kubectl get tanzukubernetesreleases</code>	List available Tanzu Kubernetes releases.
<code>kubectl get tkr</code>	Short form version of the preceding command.

Table 13-18. Custom Commands to Manage Tanzu Kubernetes Clusters (continued)

Command	Description
<code>kubectl get tkr v1.17.8---vmware.1-tkg.1.5417466 -o yaml</code>	Provides details on the named Tanzu Kubernetes release.
<code>kubectl get virtualmachine</code>	Lists the virtual machine resources supporting the cluster nodes in the current namespace.
<code>kubectl get vm</code>	Short form version of the preceding command.
<code>kubectl describe virtualmachine VIRTUAL-MACHINE-NAME</code>	Describe the specified virtual machine, showing the state, current status, and events.
<code>kubectl describe virtualmachinesetresourcepolicy</code>	List the Virtual Machine Set Resource Policy resources supporting the cluster in the current namespace. This resource represents the vSphere objects resource pool and folder used for the cluster.
<code>kubectl get virtualmachineservice</code>	Lists the Virtual Machine Service resources supporting the cluster nodes in the current namespace. These resources are analogous to a service, but for virtual machines instead of pods. Virtual machine services are used both for providing a load balancer for the control plane nodes of a cluster and by the paravirtual cloud provider to support a Kubernetes Service of type LoadBalancer within a cluster. See also the command <code>kubectl loadbalancer</code> .
<code>kubectl get vmservice</code>	Short form version of the preceding command.
<code>kubectl describe virtualmachineservice VIRTUAL-MACHINE-SERVICE-NAME</code>	Describe the specified Virtual Machine Service, showing the expressed cluster state, current status, and events.
<code>kubectl get virtualmachineimage</code>	List available Tanzu Kubernetes releases.
<code>kubectl get vmimage</code>	Short cut version of the preceding command.
<code>kubectl describe vmimage VM_IMAGE_NAME</code>	View details on the named VM image.
<code>kubectl get loadbalancer</code>	Lists the load balancer resources in the current namespace, including those used for clusters. A load balancer is created for the Virtual Machine Service.
<code>kubectl get virtualnetwork</code>	Lists the virtual network resources in the current namespace, including resources used for clusters. A virtual network is created for each namespace where a cluster is provisioned, and for each cluster itself.

Table 13-18. Custom Commands to Manage Tanzu Kubernetes Clusters (continued)

Command	Description
<code>kubectl get persistentvolumeclaim</code>	Lists the persistent volume claim resources in the current namespace, including resources used for clusters. See Chapter 10 Using Persistent Storage in vSphere with Tanzu .
<code>kubectl get cnsnodevmattachment</code>	Lists the CNS node virtual machine attachment resources in the current namespace. These resources represent the attachment of a persistent volume managed by CNS to a virtual machine serving as the node of a cluster. See Chapter 10 Using Persistent Storage in vSphere with Tanzu .
<code>kubectl get configmap</code>	Lists the configuration maps in the current namespace, including maps used for the creation of cluster nodes. Configuration maps are not intended to be user-modifiable, and any changes are overwritten.
<code>kubectl get secret</code>	Lists the secrets in the current namespace, including secrets used for the creation and management of cluster nodes. See Get Tanzu Kubernetes Cluster Secrets .

Use Tanzu Kubernetes Cluster Networking Commands

The Tanzu Kubernetes Grid Service provisions Tanzu Kubernetes clusters with default networking for nodes, pods, and services. You can verify cluster networking using custom `kubectl` commands.

Custom Commands to Verify Tanzu Kubernetes Cluster Networking

Refer to the following commands to verify cluster networking.

Table 13-19. Custom kubectl Commands to Verify Cluster Networking

Command	Description
Switch context to the vSphere Namespace. For example: <pre>kubectl config use-context tkgs-ns</pre> Run the command. <pre>kubectl get tkgserviceconfigurations</pre> Sample result. <pre> NAME DEFAULT CNI tkg-service-configuration antrea </pre>	Returns the default CNI, which is <code>antrea</code> unless changed. The default CNI is used for cluster creation unless explicitly overridden in the cluster YAML. To change the default CNI, see Examples for Configuring the Tanzu Kubernetes Grid Service v1alpha1 API.
Switch context to the vSphere Namespace. For example: <pre>kubectl config use-context tkgs-ns</pre> Run the command. <pre>kubectl get virtualnetwork</pre> Sample result. <pre> NAME AGE SNAT tkgs-cluster-12-vnet 10.191.152.133 True 4h3m </pre>	Returns the virtual network for cluster nodes. Use to verify that the source network address translation (SNAT) IP address is assigned.

Table 13-19. Custom kubectl Commands to Verify Cluster Networking (continued)

Command	Description
Switch context to the vSphere Namespace. For example: <pre>kubectl config use-context tkgs-ns</pre> Run the command. <pre>kubectl get virtualmachines -o wide</pre> Sample result. <pre>NAME POWERSTATE CLASS IMAGE PRIMARY-IP AGE tkgs-cluster-12-control-plane-... poweredOn guaranteed-medium ob-...-v1.21.6---vmware.1-tkg.1.b3d708a 10.244.0.66 4h6m tkgs-cluster-12-worker-... poweredOn guaranteed-medium ob-...-v1.21.6---vmware.1-tkg.1.b3d708a 10.244.0.68 4h3m tkgs-cluster-12-worker-... poweredOn guaranteed-medium ob-...-v1.21.6---vmware.1-tkg.1.b3d708a 10.244.0.67 4h3m</pre>	Returns the virtual network interface for cluster nodes. Use to verify that the virtual machine for each cluster node has an IP address assigned.
Switch context to the vSphere Namespace. For example: <pre>kubectl config use-context tkgs-ns</pre> Run the command. <pre>kubectl get virtualmachineservices</pre> Sample result. <pre>NAME TYPE AGE tkgs-cluster-12-control-plane-service LoadBalancer 3h53m</pre>	Returns the virtual machine service for each cluster node. Use to verify that the status is updated and includes the load balancer virtual IP (VIP) address.

Table 13-19. Custom kubectl Commands to Verify Cluster Networking (continued)

Command	Description
Switch context to the TKGS cluster namespace. For example. <pre>kubectl config use-context tkgs-cluster-10</pre> Run the command. <pre>kubectl get services -n NAMESPACE</pre> Verify. <pre>curl -k https://EXTERNAL-IP:PORT/healthz</pre>	Returns the Kubernetes service load balancer created for Cluster API access. Use to verify that an external IP is assigned. Use <code>curl</code> to verify access to the API using the external IP address and port of the load balancer service.
Switch context to the vSphere Namespace. For example: <pre>kubectl config use-context tkgs-ns</pre> Run the command. <pre>kubectl get endpoints</pre> Sample result. <pre>NAME ENDPOINTS AGE tkgs-cluster-12-control-plane-service 10.244.0.66:6443 3h44m</pre>	Returns the control plane nodes (endpoints) for the cluster. Use to verify that each endpoint is created and included in the endpoint pool.

Get Tanzu Kubernetes Cluster Secrets

Tanzu Kubernetes clusters use secrets to store tokens, keys, and passwords for operating Tanzu Kubernetes clusters.

List of Tanzu Kubernetes Cluster Secrets

A Kubernetes secret is an object that stores a small amount of sensitive data such as a password, a token, or an SSH key. Tanzu Kubernetes cluster administrators might use several secrets while operating clusters. The table lists and describes key secrets cluster administrators might use.

Note The list is not exhaustive. It includes only those secrets that might need to be manually rotated or used to access cluster nodes for troubleshooting purposes.

Secret	Description
<code>TANZU-KUBERNETES-CLUSTER-NAME-ccm-token-RANDOM</code>	A service account token used by the paravirtual cloud provider's cloud controller manager to connect to the vSphere Namespace. To trigger rotation of this credential, delete the secret.
<code>TANZU-KUBERNETES-CLUSTER-NAME-pvcsi-token-RANDOM</code>	A service account token used by the paravirtual CSI plugin to connect to the vSphere Namespace. To trigger rotation of this credential, delete the secret. See How vSphere with Tanzu Integrates with vSphere Storage .
<code>TANZU-KUBERNETES-CLUSTER-NAME-kubeconfig</code>	A kubeconfig file that can be used to connect to the cluster control plane as the <code>kubernetes-admin</code> user. This secret can be used access a cluster and troubleshoot it when vCenter Single Sign-On authentication is not available. See Connect to the Tanzu Kubernetes Cluster Control Plane as the Administrator .
<code>TANZU-KUBERNETES-CLUSTER-NAME-ssh</code>	An SSH private key that can be used to connect to any cluster node as the <code>vmware-system-user</code> . This secret can be used to SSH to any cluster node and troubleshoot it. See SSH to Tanzu Kubernetes Cluster Nodes as the System User Using a Private Key .
<code>TANZU-KUBERNETES-CLUSTER-NAME-ssh-password</code>	A password that can be used to connect to any cluster node as the <code>vmware-system-user</code> . See SSH to Tanzu Kubernetes Cluster Nodes as the System User Using a Private Key .
<code>TANZU-KUBERNETES-CLUSTER-NAME-ca</code>	The root CA certificate for the Tanzu Kubernetes cluster control plane that is used by <code>kubectl</code> to securely connect to the Kubernetes API server.

Check Tanzu Kubernetes Machine Health

When the Tanzu Kubernetes Grid Service provisions a Tanzu Kubernetes cluster, several status conditions are reported that you can use to get direct insight into key aspects of machine health.

About Machine Health Conditions

A Tanzu Kubernetes cluster provisioned by the Tanzu Kubernetes Grid Service comprises several moving parts, all operated by independent but related controllers, working together to build and maintain a set of Kubernetes nodes. The `TanzuKubernetesCluster` object provides status conditions that give you with fine-grained information about machine health.

Check Machine Health

To check the health of a Tanzu Kubernetes machine:

- 1 Run the command `kubectl describe machine`.

If the status is `ready`, the machine is healthy. But, if a machine condition is `false`, such as `InfrastructureReady`, the machine is not ready.

- If the machine is not ready, run the following command and determine what is wrong with the infrastructure:

```
kubectl describe wcpmachine
```

List of Machine Health Conditions

The table lists and defines the available machine health conditions for a Tanzu Kubernetes cluster.

Condition	Description
ResourcePolicyReady	Reports the successful creation of a Resource Policy.
ResourcePolicyCreationFailed	Reported when any errors occur during ResourcePolicy creation.
ClusterNetworkReady	Reports the successful provision of a Cluster Network.
ClusterNetworkProvisionStarted	Reported when waiting for Cluster Network to be Ready.
ClusterNetworkProvisionFailed	Reported when any errors occur during network provisioning.
LoadBalancerReady	Reports the successful reconciliation of a static control plane endpoint.
LoadBalancerCreationFailed	Reported when load balancer related resources creation fails.
WaitingForLoadBalancerIP	Reported when waiting for load balancer IP to exist.
VMProvisioned	Reports that a Virtual Machine is created, powered on and assigned an IP.
WaitingForBootstrapData	Reported when a vSphereMachine is waiting for the bootstrap script to be ready before starting the provisioning process.
VMCreationFailed	Reports that creating VM CRD or corresponding bootstrap ConfigMap failed.
VMProvisionStarted	Reported when a virtual machine currently is in creation process.
PoweringOn	Reported when a virtual machine is currently executing the power on sequence.
WaitingForNetworkAddress	Reported when waiting for the machine network settings to become active.
WaitingForBIOSUUID	Reported when waiting for the machine to have a BIOS UUID

Condition Fields

Each condition may contain several fields.

Type	Describes the type of condition. For example, <code>ResourcePolicyReady</code> . For the <code>Ready</code> condition, it is a summary of all the other conditions.
Status	Describes the status of the type. States can be <code>True</code> , <code>False</code> , or <code>Unknown</code> .
Severity	Classification of the Reason. <code>Info</code> means the reconciliation is happening. <code>Warning</code> means something might wrong and retry. <code>Error</code> means an error occurred and manual action is required to resolve.
Reason	Provides a reason why the status is <code>False</code> . It can be a waiting for ready or a failure reason. Usually is thrown when the status is <code>False</code> .
Message	Human readable information that explains the Reason.

Check Tanzu Kubernetes Cluster Health

When the Tanzu Kubernetes Grid Service provisions a Tanzu Kubernetes cluster, several status conditions are reported that you can use to get direct insight into key aspects of cluster health.

About Cluster Health Conditions

A Tanzu Kubernetes cluster provisioned by the Tanzu Kubernetes Grid Service comprises several moving parts, all operated by independent but related controllers, working together to build and maintain a set of Kubernetes nodes. The `TanzuKubernetesCluster` object provides status conditions that give you with fine-grained information about cluster and machine health.

Check Cluster Health

To check the health of a Tanzu Kubernetes cluster:

- 1 Run the command `kubectl describe cluster`.

If the status is ready, it means that both the cluster infrastructure and the cluster control plane are ready. For example:

```
Status:
Conditions:
  Last Transition Time:   2020-11-24T21:37:32Z
  Status:                True
  Type:                 Ready
  Last Transition Time:   2020-11-24T21:37:32Z
  Status:                True
  Type:                 ControlPlaneReady
  Last Transition Time:   2020-11-24T21:31:34Z
  Status:                True
  Type:                 InfrastructureReady
```

But, if a cluster condition is false, the cluster is not ready, and a message field describes what is wrong. For example, here is the status is False and because the infrastructure is not ready:

```
Status:
  Conditions:
    Last Transition Time: 2020-11-24T21:37:32Z
    Status:              False
    Type:                Ready
    Last Transition Time: 2020-11-24T21:37:32Z
    Status:              True
    Type:                ControlPlaneReady
    Last Transition Time: 2020-11-24T21:31:34Z
    Status:              False
    Type:                InfrastructureReady
```

- 2 If the cluster is not ready, run the following command to determine what is wrong with the cluster infrastructure:

```
kubectl describe wcpcluster
```

List of Cluster Health Conditions

The table lists and defines the available health conditions for a Tanzu Kubernetes cluster.

Condition	Description
Ready	Summarizes the operational state of a Cluster API object.
Deleting	The Status is not True because the underlying object is currently being deleted.
DeletionFailed	The Status is not True because the underlying object encountered problems during deletion. This is a warning because the reconciler will retry deletion.
Deleted	The Status is not True because the underlying object was deleted.
InfrastructureReady	Reports a summary of current status of the infrastructure object defined for this cluster.
WaitingForInfrastructure	Reported when a cluster is waiting for the underlying infrastructure to be available. NOTE: This condition is used as a fallback when the infrastructure is not reporting a ready state.
ControlPlaneReady	Reported when the cluster control plane is ready.
WaitingForControlPlane	Reported when a cluster is waiting for the control plane to be available. NOTE: This condition is used as a fallback when the control plane is not reporting a ready state.

Condition Fields

Each condition may contain several fields.

Type	Describes the type of condition. For example, <code>ControlPlaneReady</code> . For the <code>Ready</code> condition, it is a summary of all the other conditions.
Status	Describes the status of the type. States can be <code>True</code> , <code>False</code> , or <code>Unknown</code> .
Severity	Classification of the <code>Reason</code> . <code>Info</code> means the reconciliation is happening. <code>Warning</code> means something might wrong and retry. <code>Error</code> means an error occurred and manual action is required to resolve.
Reason	Provides a reason why the status is <code>False</code> . It can be a waiting for ready or a failure reason. Usually is thrown when the status is <code>False</code> .
Message	Human readable information that explains the <code>Reason</code> .

Monitor Tanzu Kubernetes Cluster Status Using the vSphere Client

You can monitor the status of Tanzu Kubernetes clusters using the vSphere Client.

Procedure

- 1 Log in to the vCenter Server using the vSphere Client.
- 2 From the **Menu**, select the **Hosts and Clusters** view.
- 3 Expand the **Datacenter > Cluster** objects where the Supervisor Cluster is created.
- 4 Expand the **Namespaces** resource pool.
- 5 Select the vSphere Namespace where you have deployed the Tanzu Kubernetes cluster.
Each Tanzu Kubernetes cluster is listed as a folder within its namespace resource pool. Each Tanzu Kubernetes is graphically represented by the three hexagon icons beside its name.
- 6 Switch to the **Menu > VMs and Templates** view.
Within the cluster folder, you see the VMs that comprise the cluster nodes.
- 7 Select the vSphere Namespace, and then select the **Compute** tab.
- 8 Under **VMware Resources**, select **Tanzu Kubernetes**.
Each Tanzu Kubernetes cluster deployed in this vSphere Namespace is listed. For a description of each status field, see [Tanzu Kubernetes Cluster Lifecycle Status in vSphere](#).

Deploying Workloads and Packages on TKGS Clusters

14

Refer to the content in this section to install workloads and packages on Tanzu Kubernetes clusters.

Read the following topics next:

- [Deploy Workloads on Tanzu Kubernetes Clusters](#)
- [Install Packages on Tanzu Kubernetes Clusters](#)
- [Deploy AI/ML Workloads on Tanzu Kubernetes Clusters](#)

Deploy Workloads on Tanzu Kubernetes Clusters

You can deploy application workloads to Tanzu Kubernetes clusters using pods, services, persistent volumes, and higher-level resources such as Deployments and Replica Sets.

Deploy a Test Workload to a Tanzu Kubernetes Cluster

After you have provisioned a Tanzu Kubernetes cluster, it is good practice to deploy a test workload and validate cluster functionality.

Use the [kuard](#) demo app to verify that your Tanzu Kubernetes cluster is up and running.

Prerequisites

- Provision a Tanzu Kubernetes cluster. See [Workflow for Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API](#).
- [Connect to a Tanzu Kubernetes Cluster as a vCenter Single Sign-On User](#)

Procedure

- 1 Switch configuration context to the target Tanzu Kubernetes cluster.

```
kubectl config use-context TANZU-KUBERNETES-CLUSTER-NAME
```

For example:

```
kubectl config use-context tkgs-cluster-1  
Switched to context "tkgs-cluster-1".
```


2 Deploy the `kuard` demo app.

```
kubectl run --restart=Never --image=gcr.io/kuar-demo/kuard-amd64:blue kuard
```

Expected result:

```
pod/kuard created
```

3 Verify that the pod is running.

```
kubectl get pods
```

Expected result:

NAME	READY	STATUS	RESTARTS	AGE
kuard	1/1	Running	0	10d

4 Forward the pod container port 8080 to your local host port 8080.

```
kubectl port-forward kuard 8080:8080
```

Expected result:

```
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
Handling connection for 8080
```

5 Using a browser go to <http://localhost:8080>.

The `kuard` demo app web page appears which you can interact with and verify aspects of your cluster. For example, perform liveness and readiness probes.

6 Stop port forwarding by pressing `Ctrl+C` in the `kubectl` session.

7 Delete the `kuard` pod.

```
kubectl delete pod kuard
```

Expected result:

```
pod "kuard" deleted
```

8 Verify that the pod is deleted.

```
kubectl get pods
```

Install and Run Octant

You can install the Octant web interface to help you visualize Tanzu Kubernetes cluster workloads, namespaces, metadata, and more.

About Octant

[Octant](#) is an open source web interface for viewing Kubernetes clusters and their applications.

You install and run Octant on the same client where you run `kubectl`. Installation instructions for common platforms are provided below. For more information, see the [Octant site](#).

Once Octant is installed, to use it log in to your Tanzu Kubernetes cluster using `kubectl` and run the command `octant`.

Install Octant on Windows

Install the [Chocolatey](#) package manager for Windows PowerShell.

Run a PowerShell session as Administrator.

Install Octant using the following command:

```
choco install octant --confirm
```

Install Octant on Mac

Install the [Homebrew](#) package manager.

Install Octant with the following command:

```
brew install octant
```

Install Octant on Ubuntu

Download the `.deb` from the [releases page](#).

Install using the `dpkg -i` command.

Tanzu Kubernetes Service Load Balancer Example

To provision an external load balancer in a Tanzu Kubernetes cluster, you can create a Service of type `LoadBalancer`. The load balancer service exposes a public IP address. Traffic from the external load balancer can be directed at cluster pods.

You can provision an external load balancer for Kubernetes pods that are exposed as services. For example, you can deploy a Nginx container and expose it as a Kubernetes service of type `LoadBalancer`.

Prerequisites

- Review the [Service type LoadBalancer](#) in the Kubernetes documentation.
- Provision a Tanzu Kubernetes cluster. See [Workflow for Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API](#).
- Connect to the target Tanzu Kubernetes cluster. See [Connect to a Tanzu Kubernetes Cluster as a vCenter Single Sign-On User](#).

Procedure

- 1 Create an appropriate role binding to the default privileged PSP. See [Example Role Bindings for Pod Security Policy](#).
- 2 Create the following `nginx-lbsvc.yaml` YAML file.

This YAML file defines a Kubernetes service of type LoadBalancer and deploys a Nginx container as an external load balancer for the service.

```
kind: Service
apiVersion: v1
metadata:
  name: srvc1b-nginx
spec:
  selector:
    app: hello
    tier: frontend
  ports:
  - protocol: "TCP"
    port: 80
    targetPort: 80
  type: LoadBalancer
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: loadbalancer
spec:
  replicas: 2
  selector:
    matchLabels:
      app: hello
  template:
    metadata:
      labels:
        app: hello
        tier: frontend
    spec:
      containers:
      - name: nginx
        image: "nginxdemos/hello"
```

- 3 Apply the YAML.

```
kubectl apply -f nginx-lbsvc.yaml
```

- 4 Verify the deployment of the Nginx service.

```
kubectl get services
```

The `srvclb-nginx` is up with an external and internal IP address.

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
srvclb-nginx	LoadBalancer	10.11.12.19	10.19.15.89	80:30818/TCP	18m

- Using a browser, enter the external IP address for the Nginx LoadBalancer service.

You see the message `NGINX` banner and details of the load balancer.

Tanzu Kubernetes Service Load Balancer with Static IP Address Example

You can configure a Kubernetes service of type `LoadBalancer` to use a static IP address. Be aware of minimum component requirements, an important security consideration and cluster hardening guidance before implementing this feature.

Minimum Requirements

Static IP addressing for Kubernetes services of type `LoadBalancer` is supported on Tanzu Kubernetes clusters that meet the following requirements:

Component	Minimum Requirement	More Information
vCenter Server and ESXi	vSphere 7.0 Update 2	See the Release Notes .
Supervisor Cluster	v1.19.1+vmware.2- vsc0.0.8-17610687	See Update the Supervisor Cluster by Performing a vSphere Namespaces Update .
Load Balancer	NSX-T Data Center v3.1 Or NSX Advanced 20.1.x	See the Release Notes .
Tanzu Kubernetes Release	One of the latest Tanzu Kubernetes releases.	See Verify Tanzu Kubernetes Cluster Compatibility for Update .

Using a Static IP for a Service of Type LoadBalancer

Typically when you define a Kubernetes service of [Type LoadBalancer](#) you get an ephemeral IP address assigned by the load balancer. See [Tanzu Kubernetes Service Load Balancer Example](#).

Alternatively you can specify a static IP address for the load balancer. On creation of the service, the load balancer instance is provisioned with the static IP address you assigned.

The following example service demonstrates how to configure a supported load balancer with a static IP address. In the service specification you include the `loadBalancerIP` parameter and an IP address value, which is `10.11.12.49` in this example.

```
kind: Service
apiVersion: v1
metadata:
  name: load-balancer-service-with-static-ip
spec:
```

```

selector:
  app: hello-world
  tier: frontend
ports:
- protocol: "TCP"
  port: 80
  targetPort: 80
type: LoadBalancer
loadBalancerIP: 10.11.12.49

```

For the NSX Advanced load balancer, you use a IP address from the IPAM pool configured for the load balancer when it was installed. When the service is created and the static IP address is assigned, the load balancer marks it as allocated and manages the lifecycle of the IP address the same way it does an ephemeral IP address. That is, if the service is removed, the IP address is unassigned and made available for reallocation.

For the NSX-T load balancer, you have two options. The default mechanism is the same as the NSX Advanced load balancer: use an IP address taken from the IP pool configured for the load balancer when it was installed. When the static IP address is assigned, the load balancer automatically marks it as allocated and manages its lifecycle.

The second NSX-T option is to manually pre-allocate the static IP address. In this case you use an IP address that is not part of the external load balancer IP pool assigned to load balancer, but instead taken from a floating IP pool. In this case you manually administer the allocation and lifecycle of the IP address using the NSX Manager.

Important Security Consideration and Hardening Requirement

There is a potential security issue to be aware of when using this feature. If a developer is able to patch the `Service.status.loadBalancerIP` value, the developer may be able to hijack the traffic in the cluster destined for the patched IP address. Specifically, if a Role or ClusterRole with the `patch` permission is bound to a service or user account on a cluster where this feature is implemented, that account owner is able to use its own credentials to issue `kubectl` commands and change the static IP address assigned to the load balancer.

To avoid the potential security implications of using static IP allocation for a load balancer service, you must harden each cluster where you are implementing this feature. To do this, the Role or ClusterRole you define for any developer must not allow the `patch` verb for `apiGroups: ""` and `resources: services/status`. The example role snippet demonstrates what not to do when implementing this feature.

DO NOT ALLOW PATCH

```

- apiGroups:
  - ""
  resources:
  - services/status
  verbs:
  - patch

```

To check if a developer has patch permissions, run the following command as that user:

```
kubectl --kubeconfig <KUBECONFIG> auth can-i patch service/status
```

If the command returns `yes`, the user has patch permissions. See [Checking API Access](#) in the Kubernetes documentation for more information.

To grant developer access to a cluster, see [Grant Developer Access to Tanzu Kubernetes Clusters](#). For a sample Role template that you can customize, see [Example Role for Pod Security Policy](#). For an example on how to restrict cluster access, see <https://kubernetes.io/docs/reference/access-authn-authz/rbac/#role-example>.

Tanzu Kubernetes Ingress Example Using Nginx

A Kubernetes ingress resource provides HTTP or HTTPS routing from outside the cluster to one or more services within the cluster. Tanzu Kubernetes clusters support ingress through third-party controllers, such as Nginx.

This tutorial demonstrates how to deploy a Kubernetes ingress service based on NGINX for routing external traffic to services in your Tanzu Kubernetes cluster. An ingress service requires an ingress controller. We install the NGINX Ingress controller using Helm. Helm is a package manager for Kubernetes.

Note There several ways to accomplish this task. The steps here provide one approach. Other approaches may be more suitable for you in your given environment.

Prerequisites

- Review the [Ingress](#) resource in the Kubernetes documentation.
- Review the [Nginx](#) Ingress controller documentation.
- Provision a Tanzu Kubernetes cluster. See [Workflow for Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API](#).
- Enable pod security policy. See [Example Role for Pod Security Policy](#).
- Connect to the Tanzu Kubernetes cluster. See [Connect to a Tanzu Kubernetes Cluster as a vCenter Single Sign-On User](#).

Procedure

- 1 Install Helm by referring to the [documentation](#).
- 2 Install the NGINX Ingress controller using Helm.

```
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm install ingress-nginx ingress-nginx/ingress-nginx
```

3 Verify that the Nginx ingress controller is deployed as a service of type LoadBalancer.

```
kubectl get services
```

NAME	AGE	TYPE	CLUSTER-IP	EXTERNAL-IP
ingress-nginx-controller	59m	LoadBalancer	10.16.18.20	10.19.14.76 80:30635/TCP
ingress-nginx-controller-admission	59m	ClusterIP	10.87.41.25	<none> 443/TCP

4 Ping the load balancer using the external IP address.

```
ping 10.19.14.76
```

```
Pinging 10.19.14.76 with 32 bytes of data:
Reply from 10.19.14.76: bytes=32 time<1ms TTL=62
Reply from 10.19.14.76: bytes=32 time=1ms TTL=62
```

5 Verify that the Nginx Ingress controller is running.

```
kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
ingress-nginx-controller-7c6c46898c-v6blt	1/1	Running	0	76m

6 Create an ingress resource with an ingress rule and path named `ingress-hello.yaml`.

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: ingress-hello
spec:
  rules:
  - http:
      paths:
      - path: /hello
        backend:
          serviceName: hello
          servicePort: 80
```

Note The Kubernetes API `networking.k8s.io/v1beta1` is deprecated from Kubernetes v1.22 onwards. The new API to use is `networking.k8s.io/v1`, and it involves several [manifest changes](#).

7 Deploy the `ingress-hello` resource.

```
kubectl apply -f ingress-hello.yaml
```

```
ingress.networking.k8s.io/ingress-hello created
```

8 Verify that the ingress resource is deployed.

Note that the IP address maps to the external IP of the ingress controller.

```
kubectl get ingress
```

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress-hello	<none>	*	10.19.14.76	80	51m

9 Create a hello test app and service named `ingress-hello-test.yaml`.

```
kind: Service
apiVersion: v1
metadata:
  name: hello
spec:
  selector:
    app: hello
    tier: backend
  ports:
    - protocol: TCP
      port: 80
      targetPort: http
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello
spec:
  replicas: 3
  selector:
    matchLabels:
      app: hello
      tier: backend
      track: stable
  template:
    metadata:
      labels:
        app: hello
        tier: backend
        track: stable
    spec:
      containers:
        - name: hello
```



```
image: "gcr.io/google-samples/hello-go-gke:1.0"
ports:
  - name: http
    containerPort: 80
```

10 Deploy the `ingress-hello-test` resource.

```
kubectl apply -f ingress-hello-test.yaml
```

```
service/hello created
deployment.apps/hello created
```

11 Verify that the `hello` deployment is available.

```
kubectl get deployments
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
hello	3/3	3	3	4m59s
ingress-nginx-controller	1/1	1	1	3h39m

12 Get the public IP address of the load balancer used by the Nginx ingress controller.

```
kubectl get ingress
```

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress-hello	<none>	*	10.19.14.76	80	13m

13 Using a browser, navigate to the public IP and include the ingress path.

```
http://10.19.14.76/hello
```

The message "hello" is returned.

```
{"message": "Hello"}
```

Results

The backend app that is fronted by the service running inside the cluster is accessed externally by the browser through the ingress controller using the external IP address of the load balancer.

Tanzu Kubernetes Storage Class Example

For workloads requiring persistence, you can use the default storage class, or define your own storage class for use with persistent volumes. Tanzu Kubernetes clusters support the Container Storage Interface (CSI) provisioner.

Container Storage Interface (CSI) Is Supported

Tanzu Kubernetes clusters support the Container Storage Interface (CSI). In the `StorageClass` definition, this type of provisioner is identified as `csi.vsphere.vmware.com`.

The following YAML definition can be used as a template to define a storage class for a Tanzu Kubernetes cluster. Specify if you want the storage class to be the default ("true"), and provide the datastore URL for your storage environment.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: tkgs-storage-class
  annotations:
    storageclass.kubernetes.io/is-default-class: "true" or "false"
provisioner: csi.vsphere.vmware.com
parameters:
  datastoreurl: "ds:///vmfs/volumes/vsan:52d8eb4842dbf493-41523be9cd4ff7b7/"
```

Create the storage class:

```
kubectl apply -f tkgs-storage-class.yaml

storageclass.storage.k8s.io/tkgs-storage-class created
```

Verify that the storage class is created:

```
kubectl get storageclass
```

Or, using the shortcut:

```
kubectl get sc
```

VMware Cloud Provider (vCP) Is Not Supported

Tanzu Kubernetes clusters do not support the legacy VMware Cloud Provider (vCP) `StorageClass` as shown below. If you attempt to create a `StorageClass` using the vCP provisioner, the `StorageClass` is not created.

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: demo-sts-sc
provisioner: kubernetes.io/vsphere-volume
parameters:
  diskformat: thin
```

Tanzu Kubernetes Persistent Volume Claim Examples

To run stateful workloads on Tanzu Kubernetes clusters, you can create a persistent volume claim (PVC) to request persistent storage resources without knowing the details of the

underlying storage infrastructure. The storage used for the PVC is allocated out of the storage quota for the vSphere Namespace.

Containers by default are ephemeral and stateless. For stateful workloads, a common approach is to create a persistent volume claim (PVC). You can use a PVC to mount the persistent volumes and access storage. The request dynamically provisions a persistent volume object and a matching virtual disk. The claim is bound to the persistent volume. When you delete the claim, the corresponding persistent volume object and the provisioned virtual disk are also deleted.

Procedure

- 1 Log in to the target Tanzu Kubernetes cluster. See [Connect to a Tanzu Kubernetes Cluster as a vCenter Single Sign-On User](#).

- 2 Switch to the namespace where the cluster is running.

```
kubectl config use-context NAMESPACE
```

- 3 Verify the storage class, or create one.

To verify an existing storage class:

```
kubectl get storageclass
```

To create a storage class, see [Tanzu Kubernetes Storage Class Example](#).

- 4 Create a namespace.

```
kubectl create namespace guestbook
```

- 5 Create the Guestbook PVC YAML files.

- [Redis Leader PVC](#)
- [Redis Follower PVC](#)

- 6 Apply the Guestbook PVCs to the cluster.

```
kubectl apply -f redis-leader-pvc.yaml -n guestbook
```

```
kubectl apply -f redis-follower-pvc.yaml -n guestbook
```

- 7 Verify the status of the PVCs.

```
kubectl get pvc,pv -n guestbook
```

The PVCs and persistent volumes (PVs) are listed and available for use.

NAME	STATUS		
VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS
AGE			
persistentvolumeclaim/redis-follower-pvc	Bound	pvc-37b72f35-3de2-4f84-	

```

be7d-50d5dd968f62    2Gi    RWO    tkgs-storage-class    66s
persistentvolumeclaim/redis-leader-pvc    Bound    pvc-2ef51f31-dd4b-4fe2-bf4c-
f0149cb4f3da    2Gi    RWO    tkgs-storage-class    66s

```

NAME	CAPACITY	ACCESS	MODES	RECLAIM
POLICY STATUS CLAIM STORAGECLASS				
persistentvolume/pvc-2ef51f31-dd4b-4fe2-bf4c	2Gi	RWO		Delete
Bound guestbook/redis-leader-pvc	tkgs-storage-class			
persistentvolume/pvc-37b72f35-3de2-4f84-be7d	2Gi	RWO		Delete
Bound guestbook/redis-follower-pvc	tkgs-storage-class			

Tanzu Kubernetes Guestbook Tutorial

Deploy the Guestbook application to your Tanzu Kubernetes cluster to explore pod security policy for service accounts, and deployment and service creation.

Deploying the [Guestbook application](#) is a common way to explore Kubernetes. If you deploy all the Guestbook YAML files to a Tanzu Kubernetes cluster provisioned by the Tanzu Kubernetes Grid Service, the application pod is not created successfully. The following error message appears when you run the `kubectl describe pod` command:

```
"Error: container has runAsNonRoot and image will run as root"
```

The Guestbook application is using both `deployment` and `replicaset` resources to deploy privileged containers in the default namespace. Because the PodSecurityPolicy Controller is enabled for Tanzu Kubernetes clusters, when any cluster user attempts to create the Guestbook application pod, the service accounts for these controllers are checked against PodSecurityPolicy. If an appropriate PSP is not bound to these service accounts, the application is not deployed.

By default, Tanzu Kubernetes administrators can create privileged pods directly in any namespace using their user account. However, the Guestbook application deploys privileged containers using service accounts. A cluster administrator can create Deployments, StatefulSets, and DaemonSet in the `kube-system` namespace. However, the Guestbook application deploys these resources in the default namespace. In addition, non-administrative users cannot create privileged or unprivileged pods at all without the proper PSP and bindings.

One solution is to create bindings to the default privileged PSP to allow the Guestbook application to be deployed. The privileged PodSecurityPolicy permits run-as-root pods and privileged containers for bound accounts. You can create a ClusterRoleBinding that applies the `vmware-system-privileged` cluster-wide, but doing so might violate the principle of least privilege by granting more permission than is needed. A better approach is to create a RoleBinding that permits the system service accounts to use the privileged PodSecurityPolicy in the default namespace. For more information, see [Example Role Bindings for Pod Security Policy](#).

Prerequisites

Review the following topics:

- [Guestbook application tutorial](#) in the Kubernetes documentation

- [Using Pod Security Policies with Tanzu Kubernetes Clusters](#)
- [Example Role Bindings for Pod Security Policy](#)

Procedure

- 1 Log in to the Tanzu Kubernetes cluster. See [Connect to a Tanzu Kubernetes Cluster as a vCenter Single Sign-On User](#).
- 2 Create the Guestbook namespace.

```
kubectl create namespace guestbook
```

Verify:

```
kubectl get ns
```

- 3 Create role-based access control using the default privileged PSP.

```
kubectl create clusterrolebinding default-tkg-admin-privileged-binding --  
clusterrole=psp:vmware-system-privileged --group=system:authenticated
```

Note If tighter security is required, apply a RoleBinding on the Guestbook namespace. See [Example Role Bindings for Pod Security Policy](#)

- 4 Verify the storage class, or create one.

To verify an existing storage class:

```
kubectl get storageclass
```

To create a storage class, see [Tanzu Kubernetes Storage Class Example](#).

- 5 Create the persistent volume claims (PVCs) that use the storage class.

Use the following YAML files:

- [Redis Leader PVC](#)
- [Redis Follower PVC](#)

To create the PVCs, see [Tanzu Kubernetes Persistent Volume Claim Examples](#).

- 6 Create the Guestbook YAML files.

Use the following YAML files:

- [Redis Leader Deployment](#)
- [Redis Leader Service](#)
- [Redis Follower Deployment](#)
- [Redis Follower Service](#)
- [Guestbook Frontend Deployment](#)

■ Guestbook Frontend Service

7 Deploy the Guestbook application in its namespace.

```
kubectl apply -f . --namespace guestbook
```

8 Verify creation of the Guestbook resources.

```
kubectl get all -n guestbook
```

NAME	READY	STATUS
RESTARTS AGE		
pod/guestbook-frontend-deployment-56fc5b6b47-cd58r0	1/1	Running
pod/guestbook-frontend-deployment-56fc5b6b47-fh6dp0	1/1	Running
pod/guestbook-frontend-deployment-56fc5b6b47-hgd2b0	1/1	Running
pod/redis-follower-deployment-6fc9cf5759-99fgw0	1/1	Running
pod/redis-follower-deployment-6fc9cf5759-rhxf70	1/1	Running
pod/redis-leader-deployment-7d89bbdbcf-flt4q0	1/1	Running

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP
PORT(S) AGE			
service/guestbook-frontend	LoadBalancer	10.10.89.59	10.19.15.99
80:31513/TCP 65s			
service/redis-follower	ClusterIP	10.111.163.189	<none>
6379/TCP 65s			
service/redis-leader	ClusterIP	10.111.70.189	<none>
6379/TCP 65s			

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/guestbook-frontend-deployment	3/3	3	3	65s
deployment.apps/redis-follower-deployment	1/2	2	1	65s
deployment.apps/redis-leader-deployment	1/1	1	1	65s

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/guestbook-frontend-deployment-56fc5b6b47	3	3	3	65s
replicaset.apps/redis-follower-deployment-6fc9cf5759	2	2	1	65s
replicaset.apps/redis-leader-deployment-7d89bbdbcf	1	1	1	65s

9 Access the Guestbook web page using the External-IP address of the service/guestbook-frontend load balancer, which in this example is 10.19.15.99.

You see the Guestbook web interface, and you can enter values into the Guestbook database. If you restart the application, the data is persisted.

Guestbook Example YAML Files

Use the example YAML files to deploy the Guestbook application with persistent data.

Redis Leader PVC

The file `redis-leader-pvc.yaml` is an example persistent volume claim that references a named storage class. To use this example, enter the name of the storage class.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: redis-leader-pvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: tkgs-storage-class-name
  resources:
    requests:
      storage: 2Gi
```

Redis Follower PVC

The file `redis-follower-pvc.yaml` is an example persistent volume claim that references a named storage class. To use this example, enter the name of the storage class.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: redis-follower-pvc
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: tkgs-storage-class-name
  resources:
    requests:
      storage: 2Gi
```

Redis Leader Deployment

The file `redis-leader-deployment.yaml` is an example Redis leader deployment with a persistent volume.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-leader-deployment
spec:
  selector:
    matchLabels:
      app: redis
      role: leader
      tier: backend
```

```

replicas: 1
template:
  metadata:
    labels:
      app: redis
      role: leader
      tier: backend
  spec:
    containers:
      - name: leader
        image: redis:6.0.5
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        ports:
          - containerPort: 6379
        volumeMounts:
          - name: redis-leader-data
            mountPath: /data
    volumes:
      - name: redis-leader-data
        persistentVolumeClaim:
          claimName: redis-leader-pvc

```

Redis Follower Deployment

The file `redis-follower-deployment.yaml` is an example Redis follower deployment with a persistent volume.

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-follower-deployment
  labels:
    app: redis
spec:
  selector:
    matchLabels:
      app: redis
      role: follower
      tier: backend
  replicas: 1
  template:
    metadata:
      labels:
        app: redis
        role: follower
        tier: backend
    spec:
      containers:
        - name: follower
          image: gcr.io/google_samples/gb-redis-follower:v2
          resources:

```



```

    requests:
      cpu: 100m
      memory: 100Mi
    env:
      - name: GET_HOSTS_FROM
        value: dns
    ports:
      - containerPort: 6379
    volumeMounts:
      - name: redis-follower-data
        mountPath: /data
    volumes:
      - name: redis-follower-data
        persistentVolumeClaim:
          claimName: redis-follower-pvc

```

Redis Leader Service

The file `redis-leader-service.yaml` is an example Redis leader service.

```

apiVersion: v1
kind: Service
metadata:
  name: redis-leader
  labels:
    app: redis
    role: leader
    tier: backend
spec:
  ports:
    - port: 6379
      targetPort: 6379
  selector:
    app: redis
    role: leader
    tier: backend

```

Redis Follower Service

The file `redis-follower-service.yaml` is an example Redis follower service.

```

apiVersion: v1
kind: Service
metadata:
  name: redis-follower
  labels:
    app: redis
    role: follower
    tier: backend
spec:
  ports:
    - port: 6379

```

```
selector:
  app: redis
  role: follower
  tier: backend
```

Guestbook Frontend Deployment

The file `guestbook-frontend-deployment.yaml` is an example Guestbook frontend deployment.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: guestbook-frontend-deployment
spec:
  selector:
    matchLabels:
      app: guestbook
      tier: frontend
  replicas: 3
  template:
    metadata:
      labels:
        app: guestbook
        tier: frontend
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v5
        resources:
          requests:
            cpu: 100m
            memory: 100Mi
        env:
        - name: GET_HOSTS_FROM
          value: dns
        ports:
        - containerPort: 80
```

Guestbook Frontend Service

The file `guestbook-frontend-service.yaml` is an example Guestbook frontend load balancer service.

```
apiVersion: v1
kind: Service
metadata:
  name: guestbook-frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  type: LoadBalancer
  ports:
```

```
- port: 80
selector:
  app: guestbook
  tier: frontend
```

Using Pod Security Policies with Tanzu Kubernetes Clusters

Tanzu Kubernetes Grid Service provisions Tanzu Kubernetes clusters with the PodSecurityPolicy Admission Controller enabled. This means that pod security policy is required to deploy workloads. Cluster administrators can deploy pods from their user account to any namespace, and from service accounts to the kube-system namespace. For all other use cases, you must explicitly bind to a PodSecurityPolicy object. Clusters include default pod security policies that you can bind to, or create your own.

About Kubernetes Pod Security Policies

Kubernetes pod security policies (PSPs) are cluster-level resources that control the security of pods. Using PSPs gives you control over the types of pods that can be deployed and the types of accounts that can deploy them.

A PodSecurityPolicy resource defines a set of conditions that a pod must satisfy to be deployable. If the conditions are not met, the pod cannot be deployed. A single PodSecurityPolicy must validate a pod in its entirety. A pod cannot have some of its rules in one policy and some in another.

There are various ways to implement the use of pod security policies in Kubernetes. The typical approach is by using role-based access control (RBAC) objects. ClusterRole and ClusterRoleBinding apply cluster-wide; Role and RoleBinding apply to a specific namespace. If a RoleBinding is used, it only lets pods run in the same namespace as the binding.

There are two ways to create Kubernetes pods: directly or indirectly. You create a pod directly by deploying a pod spec with your user account. You create a pod indirectly by defining some higher-level resource, such as a Deployment or DaemonSet. In this case, a service account creates the underlying pod.

To use PSPs effectively, you must account for both pod creation workflows. If a user creates a pod directly, the PSP bound to the user account controls the operation. If a user creates a pod by way of a service account, the PSP must be bound to the service account used to create the pod. If no service account is specified in the pod spec, the default service account for the namespace is used.

For more information, see [Pod Security Policies](#), [RBAC](#), and [Service Accounts](#) in the Kubernetes documentation.

Default PodSecurityPolicy for Tanzu Kubernetes Clusters

The table lists and describes the privileged and restricted default pod security policies for Tanzu Kubernetes clusters, and the default ClusterRole associated with each policy.

Table 14-1. Default PodSecurityPolicy with Associated ClusterRole

Default PSP	Permission	Description	Associated Default ClusterRole
vmware-system-privileged	Run as any	Permissive PSP. Equivalent to running a cluster without the PSP Admission Controller enabled.	psp:vmware-system-privileged can use this PSP
vmware-system-restricted	Must run as non-root	Restrictive PSP. Does not permit privileged access to pod containers, blocks possible escalations to root, and requires use of several security mechanisms.	psp:vmware-system-restricted can use this PSP

No Default Bindings for Tanzu Kubernetes Clusters

The Tanzu Kubernetes Grid Service does not provide default RoleBinding and ClusterRoleBinding for Tanzu Kubernetes clusters.

A vCenter Single Sign-On user who is granted the **Edit** permission on a vSphere Namespace is assigned to the **cluster-admin** role for any Tanzu Kubernetes cluster deployed in that namespace. An authenticated cluster administrator can implicitly use the `vmware-system-privileged` PSP. Although not technically a ClusterRoleBinding, it has the same effect.

The cluster administrator must define any bindings to allow or restrict the types of pods users can deploy to a cluster. If a RoleBinding is used, the binding only allows pods to be run in the same namespace as the binding. This can be paired with system groups to grant access to all pods run in the namespace. Non-admin users who authenticate to the cluster are assigned to the `authenticated` role and can be bound to default PSP as such. See [Grant Developer Access to Tanzu Kubernetes Clusters](#).

Effect of Default PodSecurityPolicy on Tanzu Kubernetes Clusters

The following behavior is enforced for any Tanzu Kubernetes cluster:

- A cluster administrator can create privileged pods directly in any namespace by using his or her user account.
- A cluster administrator can create Deployments, StatefulSets, and DaemonSet (each of which creates privileged pods) in the kube-system namespace. If you want to use a different namespace, see [Tanzu Kubernetes Guestbook Tutorial](#).
- A cluster administrator can create his or her own PSPs (in addition to the two default PSPs) and bind these PSPs to any users. If you define your own PSP, see [Policy Order](#) in the Kubernetes documentation.
- No authenticated users can create privileged or unprivileged pods until the cluster administrator binds PSPs to the authenticated users. See [Tanzu Kubernetes Guestbook Tutorial](#).

Example Role Bindings for Pod Security Policy

Tanzu Kubernetes clusters include default PodSecurityPolicy that you can bind to for privileged and restricted workload deployment.

About Default Pod Security Policy

This section provides YAML and CLI commands for creating role binding objects to default pod security policy, including ClusterRoleBinding and RoleBinding. For more information, see [Using Pod Security Policies with Tanzu Kubernetes Clusters](#).

A RoleBinding grants permissions within a specific namespace whereas a ClusterRoleBinding grants permissions cluster-wide. The decision to use a RoleBindings or ClusterRoleBinding depends on your use case. For example, if you use a ClusterRoleBinding and configure subjects to use `system:serviceaccounts:<namespace>`, you can bind to a PSP before the namespace is created. For more information, see [RoleBinding and ClusterRoleBinding](#) in the Kubernetes documentation.

Example 1: ClusterRoleBinding to Run a Privileged Set of Workloads

The following kubectl command creates a ClusterRoleBinding that grants access to authenticated users run a privileged set of workloads using the default PSP `vmware-system-privileged`.

Warning Applying Example 1, declaratively or imperatively, allows the deployment of privileged workloads cluster-wide. In effect Example 1 disables native security controls and should be used with caution and full awareness of the implications. For tighter security, consider Examples 2, 3, and 4.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: psp:privileged
rules:
- apiGroups: ['policy']
  resources: ['podsecuritypolicies']
  verbs:     ['use']
  resourceNames:
  - vmware-system-privileged
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: all:psp:privileged
roleRef:
  kind: ClusterRole
  name: psp:privileged
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: Group
  name: system:serviceaccounts
  apiGroup: rbac.authorization.k8s.io
```

As an alternative to applying YAML, you can run the following `kubectl` command.

```
kubectl create clusterrolebinding default-tkg-admin-privileged-binding --
clusterrole=psp:vmware-system-privileged --group=system:authenticated
```

Example 2: RoleBinding to Run a Privileged Set of Workloads

The following `kubectl` command creates a `RoleBinding` that grants access to all service accounts within the default namespace to run a privileged set of workloads using the default PSP `vmware-system-privileged`.

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: rolebinding-default-privileged-sa-ns_default
  namespace: default
roleRef:
  kind: ClusterRole
  name: psp:vmware-system-privileged
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: Group
  apiGroup: rbac.authorization.k8s.io
  name: system:serviceaccounts
```

As an alternative to applying YAML, you can run the following `kubectl` command.

```
kubectl create rolebinding rolebinding-default-privileged-sa-ns_default --namespace=default --
clusterrole=psp:vmware-system-privileged --group=system:serviceaccounts
```

Example 3: ClusterRoleBinding to Run a Restricted Set of Workloads

The following YAML creates a `ClusterRoleBinding` that grants authenticated users cluster-wide access to run a restricted set of workloads using the default PSP `vmware-system-restricted`.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: psp:authenticated
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:authenticated
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: psp:vmware-system-restricted
```

As an alternative to applying YAML, you can run the following `kubectl` command.

```
kubectl create clusterrolebinding psp:authenticated --clusterrole=psp:vmware-system-restricted --group=system:authenticated
```

Example 4: RoleBinding to Run a Restricted Set of Workloads

The following YAML creates a RoleBinding that grants access to all service accounts within a specific namespace to run a restricted set of workloads using the default PSP `vmware-system-restricted`.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: psp:serviceaccounts
  namespace: some-namespace
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:serviceaccounts
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: psp:vmware-system-restricted
```

As an alternative to applying YAML, you can run the following `kubectl` command.

```
kubectl create rolebinding psp:serviceaccounts --clusterrole=psp:vmware-system-restricted --group=system:serviceaccounts
```

Example Role for Pod Security Policy

Tanzu Kubernetes clusters require pod security policy (PSP) to deploy workloads. If you define your own PSP, you must create a Role or ClusterRole that references the PSP.

Example Role for PodSecurityPolicy

The following example demonstrates a Role bound to PodSecurityPolicy. In the role definition, the `example-role` is granted the `use` verb to a custom PSP resource that you define. Alternatively, use one of the default PSPs. Then, create a [Tanzu Kubernetes Guestbook Tutorial](#).

```
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: Role
metadata:
  name: example-role
  namespace: tkgs-cluster-ns
rules:
- apiGroups:
```

```

- ""
resources:
- configmaps
verbs:
- create
- get
- list
- watch
- update
- apiGroups:
- ""
resources:
- events
verbs:
- create
- update
- patch
- apiGroups:
- extensions
resourceNames:
- CUSTOM-OR-DEFAULT-PSP
resources:
- podsecuritypolicies
verbs:
- use

```

Install Packages on Tanzu Kubernetes Clusters

VMware Tanzu provides a standard set of open source applications bundled as packages that you can install on your Tanzu Kubernetes clusters to operationalize them.

Tanzu Packages for TKG Clusters

The table lists the VMware Tanzu packages available for installation on Tanzu Kubernetes clusters provisioned on Supervisor Cluster.

For documentation, refer to the separate publication [Installing and Using VMware Tanzu Packages](#).

Package	Description	More Info
Cert Manager	Certificate management	https://cert-manager.io/
Contour with Envoy	Kubernetes ingress controller and reverse HTTP proxy	https://projectcontour.io/
ExternalDNS	DNS lookup of Kubernetes services	https://github.com/kubernetes-sigs/external-dns
Prometheus with Alertmanager	Monitoring and alerting	https://prometheus.io/
Grafana	Visualization	https://grafana.com/
Fluent Bit	Log forwarding	https://fluentbit.io/
Harbor	Container registry	https://goharbor.io/

Deploy AI/ML Workloads on Tanzu Kubernetes Clusters

You can deploy AI/ML workloads on Tanzu Kubernetes clusters provisioned by the Tanzu Kubernetes Grid Service. The deployment of AI/ML workloads requires some initial setup by the vSphere Administrator and some configuration by the Cluster Operator. Once the vSphere with Tanzu environment is vGPU-enabled, developers can deploy AI/ML workloads to their TKGS clusters same as they would any Kubernetes workload.

About Deploying AI/ML Workloads on TKGS Clusters

You can deploy AI/ML workloads on TKGS clusters using vSphere with Tanzu and NVIDIA vGPU technology.

Announcing TGKS Support for AI/ML Workloads

Beginning with the release of vSphere with Tanzu Version 7 Update 3 Monthly Patch 1, you can deploy compute intensive workloads to Tanzu Kubernetes clusters provisioned by the Tanzu Kubernetes Grid Service. In this context a compute intensive workload is an artificial intelligence (AI) or machine learning (ML) application that requires the use of a GPU accelerator device.

To facilitate running AI/ML workloads in a Kubernetes environment, VMware has partnered with NVIDIA to support the NVIDIA GPU Cloud platform on vSphere with Tanzu. This means that you can deploy container images from the [NGC Catalog](#) on Tanzu Kubernetes clusters provisioned by the Tanzu Kubernetes Grid Service.

To learn more about the joint NVIDIA and VMware architecture for the AI-Ready Enterprise, see [Accelerating Workloads on vSphere 7 with Tanzu - A Technical Preview of Kubernetes Clusters with GPUs](#).

Supported vGPU Modes

Deploying AI/ML workloads on TKGS requires the use of the Ubuntu OVA that is available through the vSphere with Tanzu content delivery network. TKGS supports two modes of GPU operations: vGPU and vGPU with NIC Passthrough. The table describes the two modes in more detail.

Mode	Configuration	Description
NVIDIA + TKGS + Ubuntu + vGPU	NVIDIA vGPU	The GPU device is virtualized by the NVIDIA Host Manager Driver installed on each ESXi host. The GPU device is then shared across multiple NVIDIA virtual GPUs (vGPUs). Each NVIDIA vGPU is defined by the amount of memory from the GPU device. For example, if the GPU device has a total amount of RAM of 32GB, you can create 8 vGPUs with approximately 4 GB of memory each.
NVIDIA + TKGS + Ubuntu + vGPU + NIC Passthrough	NVIDIA vGPU and Dynamic DirectPath IO	In the same VM Class where you configure the NVIDIA vGPU profile, you include support for a passthrough networking device using Dynamic DirectPath IO. In this case vSphere DRS determines VM placement.

Getting Started

To configure NVIDIA vGPU for TKGS, refer to the following topics:

- [vSphere Administrator Workflow for Deploying AI/ML Workloads on TKGS Clusters \(vGPU\)](#)
- [Cluster Operator Workflow for Deploying AI/ML Workloads on TKGS Clusters](#)

If you are using vGPU with NIC Passthrough, also refer to the following topic: [vSphere Administrator Addendum for Deploying AI/ML Workloads on TKGS Clusters \(vGPU and Dynamic DirectPath IO\)](#).

If you are using the NVIDIA Delegated Licensing Server (DLS) for your NVAIE account, also refer to the following topic: [Cluster Operator Addendum for Deploying AI/ML Workloads on TKGS Clusters \(DLS\)](#).

vSphere Administrator Workflow for Deploying AI/ML Workloads on TKGS Clusters (vGPU)

To enable developers to deploy AI/ML workloads on TKGS clusters, as a vSphere Administrator you set up the vSphere with Tanzu environment to support NVIDIA GPU hardware.

vSphere Administrator Workflow for Deploying AI/ML Workloads on TKGS Clusters

The high-level workflow for vSphere Administrators to enable the deployment of AI/ML workloads on TKGS clusters is listed in the table. Detailed instructions for each step follow.

Step	Action	Link
0	Review system requirements.	See Admin Step 0: Review System Requirements .
1	Install Supported NVIDIA GPU Device on ESXi Hosts.	See Admin Step 1: Install Supported NVIDIA GPU Device on ESXi Hosts .
2	Configure ESXi device graphics settings for vGPU operations.	See Admin Step 2: Configure Each ESXi Host for vGPU Operations .
3	Install the NVIDIA vGPU Manager (VIB) onto each ESXi host.	See Admin Step 3: Install the NVIDIA Host Manager Driver on Each ESXi Host .
4	Verify NVIDIA driver operation and GPU virtualization mode.	See Admin Step 4: Verify ESXi Hosts Are Ready for NVIDIA vGPU Operations .
5	Enable Workload Management on the GPU-configured cluster. The result is a Supervisor Cluster that is running on vGPU-enabled ESXi hosts.	See Admin Step 5: Enable Workload Management on the vGPU-configured vCenter Cluster .
6	Create* or Update a Content Library for Tanzu Kubernetes releases and populate the library with the supported Ubuntu OVA that is required for vGPU workloads.	See Admin Step 6: Create or Update a Content Library with the Tanzu Kubernetes Ubuntu Release . Note *If necessary. If you already have a content library for TKGS clusters Photon images, not create a new content library for Ubuntu images.
7	Create a custom VM Class with a certain vGPU profile selected.	See Admin Step 7: Create a Custom VM Class with the vGPU Profile

Step	Action	Link
8	Create and configure a vSphere Namespace for TKGS GPU clusters: add a user with Edit permissions and storage for persistent volumes.	See Admin Step 8: Create and Configure a vSphere Namespace for the TKGS GPU Cluster
9	Associate the Content Library with the Ubuntu OVA and the custom VM Class for vGPU with the vSphere Namespace you created for TKGS.	See Admin Step 9: Associate the Content Library and VM Class with the vSphere Namespace
10	Verify that the Supervisor Cluster is provisioned and accessible for the Cluster Operator.	See Admin Step 10: Verify that the Supervisor Cluster Is Accessible

Admin Step 0: Review System Requirements

Refer to the following system requirements to set up the environment for deploying AI/ML workloads on TKGS clusters.

Requirement	Description
vSphere infrastructure	vSphere 7 Update3 Monthly Patch 1 ESXi build 18778458 or later vCenter Server build 18644231 or later
Workload Management	vSphere Namespace version 0.0.11-18610518 or later
Supervisor Cluster	Supervisor Cluster version v1.21.0+vmware.1-vsc0.0.11-18610518 or later
TKR Ubuntu OVA	Tanzu Kubernetes release Ubuntu ob-18691651-tkgs-ova-ubuntu-2004-v1.20.8---vmware.1-tkg.2
NVIDIA vGPU Host Driver	Download the VIB from the NGC web site . For more information, see the vGPU Software Driver documentation . For example: NVIDIA-AIE_ESXi_7.0.2_Driver_470.51-10EM.702.0.0.17630552.vib
NVIDIA License Server for vGPU	FQDN provided by your organization

Admin Step 1: Install Supported NVIDIA GPU Device on ESXi Hosts

To deploy AI/ML workloads on TKGS, you install one or more supported NVIDIA GPU devices on each ESXi host comprising the vCenter Cluster where **Workload Management** will be enabled.

To view compatible NVIDIA GPU devices, refer to the [VMware Compatibility Guide](#).

What are you looking for: **Shared Pass-Through Graphics** Compatibility Guides Help Current Results: **5**

Product Release Version:
 All
 ESXi 7.0 U2
 ESXi 7.0 U1
 ESXi 7.0
 ESXi 6.7 U3
 ESXi 6.7 U2
 ESXi 6.7 U1
 ESXi 6.7
 ESXi 6.5 U3
 ESXi 6.5 U2
 ESXi 6.5 U1
 ESXi 6.5
 ESXi 6.0 U3

GPU Partners:
 All
 NVIDIA

GPU Device Model:
 All
 NVIDIA A10
 NVIDIA A100 40GB PCIe
 NVIDIA A40

GPU Technology:
 All
 Compute
 Virtual Desktop Interface (VDI)

Guest OS:
 All
 Windows
 Linux

Compute:
 All
 AI/ML

Features:
 All
 vMotion
 SuspendResume
 Multi-vGPU

Keyword:

Posted Date Range: All

[Update and View Results](#) [Reset](#)

[Click here to Read Important Support Information](#)

Click on the 'GPU Device Model' to view more details and to subscribe to RSS feeds.

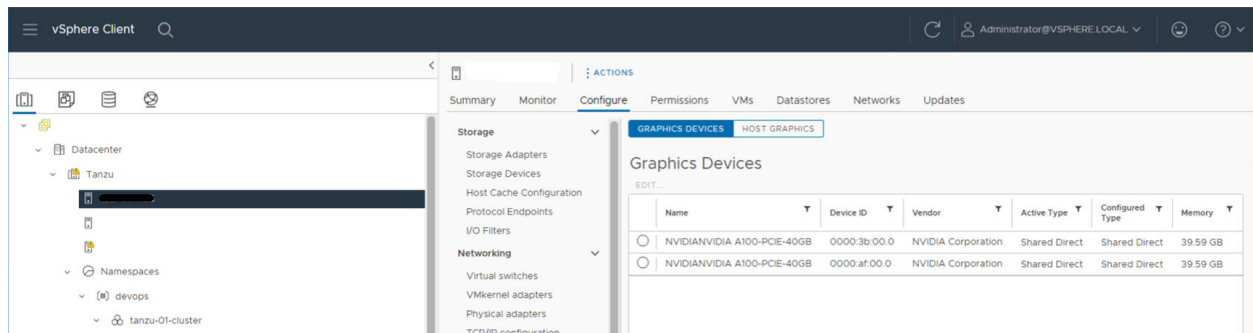
[Bookmark](#) | [Print](#) | [Export to CSV](#)

Search Results: Your search for "Shared Pass-Through Graphics" returned 5 results. [Back to Top](#) [Turn Off Auto Scroll](#) Display: 10

GPU Partner	GPU Device Model	ESX Version	Compute
NVIDIA	NVIDIA A10	ESXi 7.0 U2	AI/ML
NVIDIA	NVIDIA A100 40GB PCIe	ESXi 7.0 U2	AI/ML
NVIDIA	NVIDIA A100 80GB PCIe	ESXi 7.0 U2	AI/ML
NVIDIA	NVIDIA A30	ESXi 7.0 U2	AI/ML
NVIDIA	NVIDIA A40	ESXi 7.0 U2	AI/ML

The NVIDIA GPU device should support the latest [NVIDIA AI Enterprise \(NVAIE\)](#) vGPU profiles. Refer to the [NVIDIA Virtual GPU Software Supported GPUs](#) documentation for guidance.

For example, the following ESXi host has two NVIDIA GPU A100 devices installed on it.



Admin Step 2: Configure Each ESXi Host for vGPU Operations

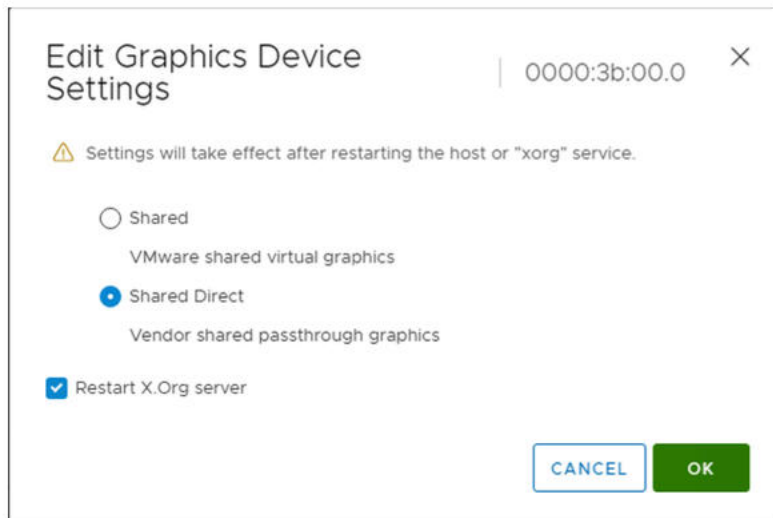
Configure each ESXi host for vGPU by enabling Shared Direct and SR-IOV.

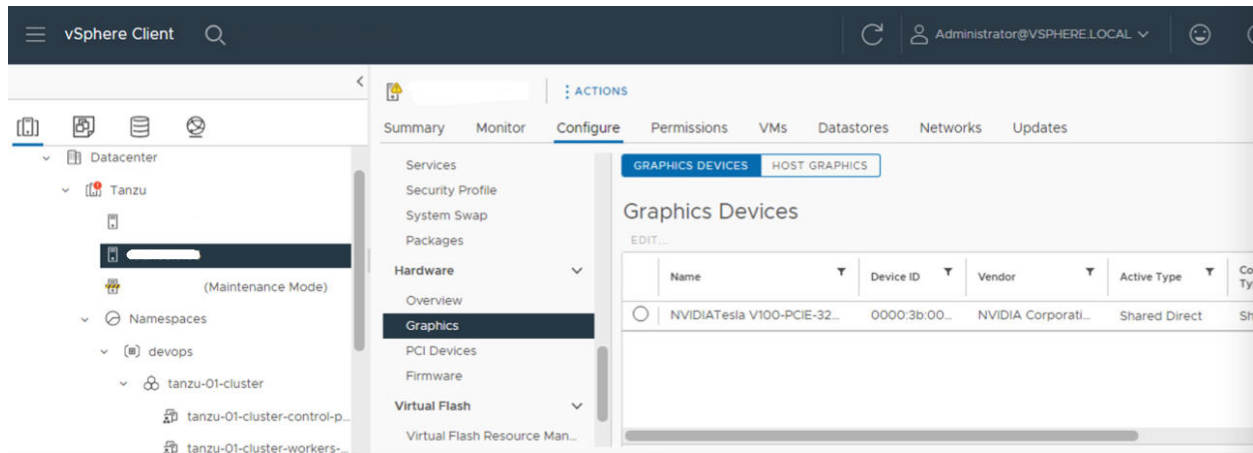
Enable Shared Direct on Each ESXi Host

For NVIDIA vGPU functionality to be unlocked, enable **Shared Direct** mode on each ESXi host comprising the vCenter Cluster where **Workload Management** will be enabled.

To enable **Shared Direct**, complete the following steps. For additional guidance, see [Configuring Graphics Devices](#) in the vSphere documentation.

- 1 Log on to the vCenter Server using the vSphere Client.
- 2 Select an ESXi host in the vCenter Cluster.
- 3 Select **Configure > Hardware > Graphics**.
- 4 Select the NVIDIA GPU accelerator device.
- 5 **Edit** the Graphics Device settings.
- 6 Select **Shared Direct**.
- 7 Select **Restart X.Org** server.
- 8 Click **OK** to save the configuration.
- 9 Right-click the ESXi host and put it into maintenance mode.
- 10 Reboot the host.
- 11 When the host is running again, take it out of maintenance mode.
- 12 Repeat this process for each ESXi host in the vCenter cluster where **Workload Management** will be enabled.





Turn On SR-IOV BIOS for NVIDIA GPU A30 and A100 Devices

If you are using the NVIDIA [A30](#) or [A100](#) GPU devices, which are required for Multi-Instance GPU (MIG mode), you must enable SR-IOV on the ESXi host. If SR-IOV is not enabled, Tanzu Kubernetes cluster node VMs cannot start. If this occurs, you see the following error message in the **Recent Tasks** pane of the vCenter Server where **Workload Management** is enabled.

```
Could not initialize plugin libnvidia-vgx.so for vGPU nvidia_aXXX-xx. Failed to start the virtual machine. Module DevicePowerOn power on failed.
```

To enable SR-IOV, log in to the ESXi host using the web console. Select **Manage > Hardware**. Select the NVIDIA GPU device and click **Configure SR-IOV**. From here you can turn on SR-IOV. For additional guidance, see [Single Root I/O Virtualization \(SR-IOV\)](#) in the vSphere documentation.

Note If you are using vGPU with NIC Passthrough, refer to the following topic for an additional ESXi configuration step: [vSphere Administrator Addendum for Deploying AI/ML Workloads on TKGS Clusters \(vGPU and Dynamic DirectPath IO\)](#).

Admin Step 3: Install the NVIDIA Host Manager Driver on Each ESXi Host

To run Tanzu Kubernetes cluster node VMs with NVIDIA vGPU graphics acceleration, you install the NVIDIA host manager driver on each ESXi host comprising the vCenter Cluster where **Workload Management** will be enabled.

The NVIDIA vGPU host manager driver components are packaged in a vSphere installation bundle (VIB). The NVAIE VIB is provided to you by your organization through its NVIDIA GRID licensing program. VMware does not provide NVAIE VIBs or make them available for download. As part of the NVIDIA licensing program your organization sets up a licensing server. Refer to the NVIDIA [Virtual GPU Software Quick Start Guide](#) for more information.

Once the NVIDIA environment is set up, run the following command on each ESXi host, replace the NVIDIA license server address and the NVAIE VIB version and with the appropriate values for your environment. For additional guidance, see [Installing and configuring the NVIDIA VIB on ESXi](#) at the VMware Support Knowledge Base.

Note The NVAIE VIB version installed on ESXi hosts must match the vGPU software version installed the node VMs. The version below is only an example.

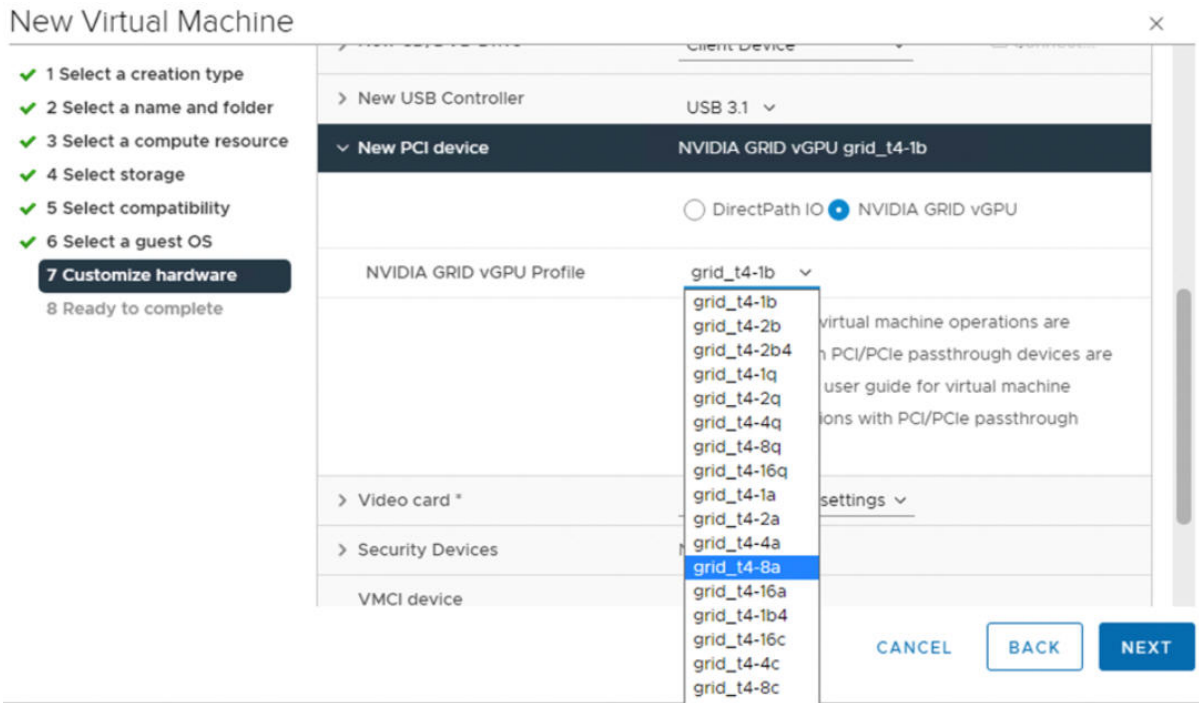
```
esxcli system maintenanceMode set --enable true
esxcli software vib install -v ftp://server.domain.example.com/nvidia/signed/
NVIDIA_bootbank_NVIDIA-VMware_ESXi_7.0_Host_Driver_460.73.02-1OEM.700.0.0.15525992.vib
esxcli system maintenanceMode set --enable false
/etc/init.d/xorg restart
```

Admin Step 4: Verify ESXi Hosts Are Ready for NVIDIA vGPU Operations

To verify that each ESXi host is ready for NVIDIA vGPU operations, perform the following checks on each ESXi host in the vCenter Cluster where **Workload Management** will be enabled:

- SSH into the ESXi host, enter shell mode and run the command `nvidia-smi`. The NVIDIA System Management Interface is a command line utility provided by the NVIDIA vGPU host manager. Running this command returns the GPUs and drivers on the host.
- Run the following command to verify that the NVIDIA driver is properly installed: `esxcli software vib list | grep NVIDIA`.
- Verify that host is configured with GPU shared direct and that SR-IOV is turned on (if you are using NVIDIA A30 or A100 devices).

- Using the vSphere Client, on the ESXi host that is configured for GPU, create a new virtual machine with a PCI device included. The NVIDIA vGPU profile should appear and be selectable.



Admin Step 5: Enable Workload Management on the vGPU-configured vCenter Cluster

Now that ESXi hosts are configured to support NVIDIA vGPU, create a vCenter Cluster comprising these hosts. To support **Workload Management**, the vCenter Cluster must meet specific requirements, including shared storage, high-availability, fully-automated DRS.

Enabling **Workload Management** also requires the selection of a networking stack, either native vSphere vDS networking or NSX-T Data Center networking. If you use vDS networking, you need to install a load balancer, either NSX Advanced or HAProxy.

The result of enabling **Workload Management** is a Supervisor Cluster that is running on vGPU-enabled ESXi hosts. Refer to the following tasks and documentation to enable **Workload Management**.

Note Skip this step if you already have a vCenter Cluster with **Workload Management** enabled, assuming that cluster is using the ESXi hosts you have configured for vGPU.

Task	Instructions
Create a vCenter Cluster that meets the requirements for enabling Workload Management	Prerequisites for Configuring vSphere with Tanzu on a vSphere Cluster
Configure the networking for the Supervisor Cluster, either NSX-T or vDS with a load balancer.	Configuring NSX for vSphere with Tanzu. Configuring vSphere Networking and NSX Advanced Load Balancer for vSphere with Tanzu. Configuring vSphere Networking and HAProxy Load Balancer for vSphere with Tanzu.
Enable Workload Management	Enable Workload Management with NSX Networking. Enable Workload Management with vSphere Networking.

Admin Step 6: Create or Update a Content Library with the Tanzu Kubernetes Ubuntu Release

Once **Workload Management** is enabled on a GPU-configured vCenter Cluster, the next step is to create a Content Library for the Tanzu Kubernetes release OVA image.

Warning If you already have a Content Library with Tanzu Kubernetes releases consisting of Photon images, you only have to synchronize the existing content library with the required Ubuntu image(s). Do not create a second content library for TKGS clusters. Doing so can cause system instability.

NVIDIA vGPU requires the Ubuntu operating system. VMware provides an Ubuntu OVA for such purposes. You cannot use the PhotonOS Tanzu Kubernetes release for vGPU clusters.

To import this image into your vSphere with Tanzu environment, choose one of the methods listed in the table and follow the corresponding instructions.

Content Library Type	Description
Create a Subscribed Content Library and automatically synchronize the Ubuntu OVA with your environment.	Create, Secure, and Synchronize a Subscribed Content Library for Tanzu Kubernetes releases
Create a Local Content Library and manually upload the Ubuntu OVA to your environment.	Create, Secure, and Synchronize a Local Content Library for Tanzu Kubernetes releases

When you have completed this task, you should see the Ubuntu OVA available in your content library.

ubuntu

Other Types

OVF & OVA Templates

VM Templates

Summary

Templates

Other Types

VM Templates

OVF & OVA Templates

Filter

☒	Name	Guest OS	Stored Locally	Security Compliant	Size	Last Modified Date	Last Sync Date	Content Library	UUID	Content Version	Description
☒	ubuntu-2004-efi-v1.20.8--vmware.1		Yes	Yes	17.29 GB	09/23/20...		ubuntu	urn:vapic...	2	Imported

Admin Step 7: Create a Custom VM Class with the vGPU Profile

To next step is to create a custom VM Class with a vGPU profile. The system will use this class definition when it creates the Tanzu Kubernetes cluster nodes.

Follow the instructions below to create a custom VM Class with a vGPU profile. For additional guidance, see [Add PCI Devices to a VM Class in vSphere with Tanzu](#).

Note If you are using vGPU with NIC Passthrough, refer to the following topic for an additional step: [vSphere Administrator Addendum for Deploying AI/ML Workloads on TKGS Clusters \(vGPU and Dynamic DirectPath IO\)](#).

- 1 Log on to the vCenter Server using the vSphere Client.
- 2 Select **Workload Management**.
- 3 Select **Services**.
- 4 Select **VM Classes**.
- 5 Click **Create VM Class**.
- 6 At the **Configuration** tab, configure the custom VM Class.

Configuration Field	Description
Name	Enter a self-descriptive name for the custom VM class, such as vmclass-vgpu-1 .
vCPU Count	2
CPU Resource Reservation	Optional, OK to leave blank
Memory	80 GB , for example
Memory Resource Reservation	100% (mandatory when PCI devices are configured in a VM Class)
PCI Devices	Yes Note Selecting Yes for PCI Devices tells the system you are using a GPU device and changes the VM Class configuration to support vGPU configuration.

For example:

Create VM Class

1 Configuration

2 PCI Devices

3 Review and Confirm

Configuration

below.

i Memory Resource Reservation must be set to 100% when PCI devices are configured in a VM Class.

Name *i*

vmclass-vgpu-01

vCPU Count

2

CPU Resource Reservation *i*

Optional %

Memory

80 GB

Memory Resource Reservation *i*

100 %

PCI Devices *i*

Yes

CANCEL

NEXT

- Click **Next**.
- At the **PCI Devices** tab, select the **Add PCI Device > NVIDIA vGPU** option.
- Configure the NVIDIA vGPU model.

NVIDIA vGPU Field	Description
Model	Select the NVIDIA GPU hardware device model from those available in the NVIDIA vGPU > Model menu. If the system does not show any profiles, none of the hosts in the cluster have supported PCI devices.
GPU Sharing	This setting defines how the GPU device is shared across GPU-enabled VMs. There are two types of vGPU implementations: Time Sharing and Multi-Instance GPU Sharing. In Time Sharing mode, the vGPU scheduler instructs the GPU to perform the work for each vGPU-enabled VM serially for a duration of time with the best effort goal of balancing performance across vGPUs. MIG mode allows multiple vGPU-enabled VMs to run in parallel on a single GPU device. MIG mode is based on a newer GPU architecture and is only supported on NVIDIA A100 and A30 devices. If you do not see the MIG option, the PCI device you selected does not support it.
GPU Mode	Compute
GPU Memory	8 GB, for example
Number of vGPUs	1, for example

For example, here is a NVIDIA vGPU profile configured in Time Sharing mode:

The screenshot shows the 'Create VM Class' wizard with three steps: 1 Configuration, 2 PCI Devices (selected), and 3 Review and Confirm. The 'PCI Devices' section is titled 'PCI Devices' and includes a close button (X). Below the title, it says 'with this class.' and has an 'ADD PCI DEVICE' button. A table lists the configuration for the 'NVIDIA vGPU' device:

		REMOVE
Model ⓘ	NVIDIATesla T4	▼
GPU Sharing ⓘ	Time Sharing	▼
GPU Mode ⓘ	Compute	▼
GPU Memory ⓘ	16	GB
	Max. 16 GB	
Number of vGPUs ⓘ	1	
	Max. 4 GPUs	

At the bottom of the wizard, there are three buttons: CANCEL, BACK, and NEXT.

For example, here is a NVIDIA vGPU profile configured in MIG mode with supported GPU device:

Edit VM Class

- 1 Configuration
- 2 PCI Devices**
- 3 Review and Confirm

PCI Devices

Adding PCI devices to the VM class will make them available to VMs created with this class.

ADD PCI DEVICE ▾

NVIDIA vGPU
REMOVE

Model ⓘ

GPU Sharing ⓘ

GPU Mode ⓘ

GPU Memory ⓘ

Number of vGPUs ⓘ

NVIDIA A100-PCIE-40GB ▾

Select sharing mode
Time Sharing
☒ Multi-Instance GPU Sharing

Compute ▾

20 GB

Max. 40 GB

1

Max. 4 GPUs

CANCEL

BACK

NEXT

- 10 Click **Next**.
- 11 Review and confirm your selections.
- 12 Click **Finish**.
- 13 Verify that the new custom VM Class is available in the list of VM Classes.

Admin Step 8: Create and Configure a vSphere Namespace for the TKGS GPU Cluster

Create a vSphere Namespace for each TKGS GPU cluster you plan to provision. Configure the namespace by adding a vSphere SSO user with Edit permissions, and attach a storage policy for persistent volumes.

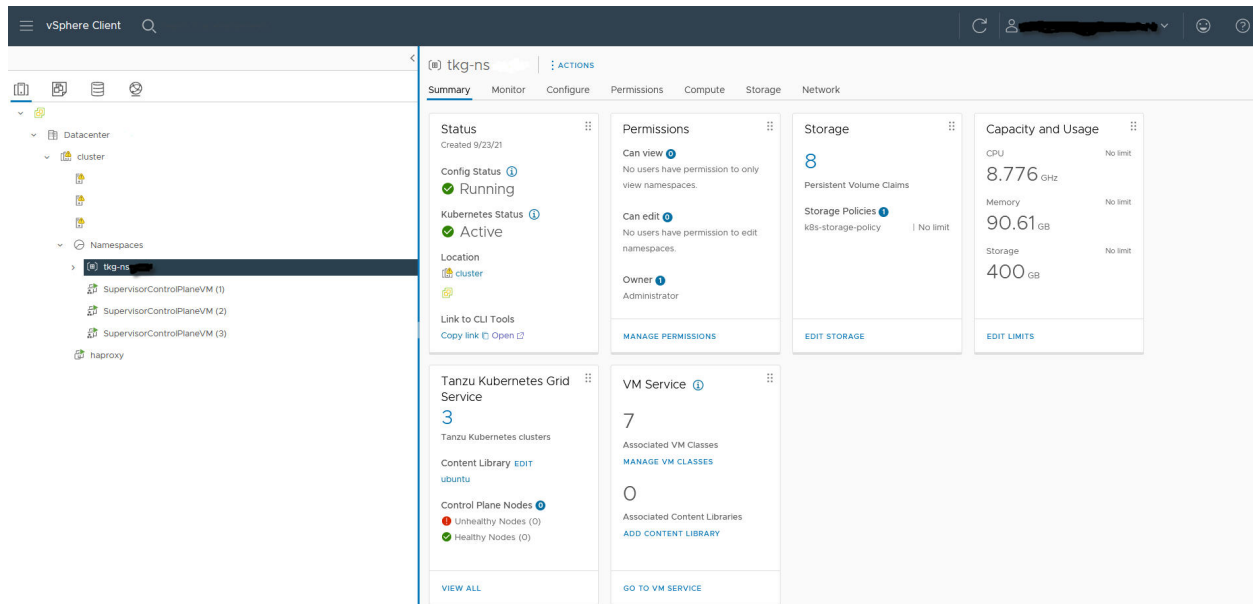
To do this, see [Create and Configure a vSphere Namespace](#).

Admin Step 9: Associate the Content Library and VM Class with the vSphere Namespace

After you have created and configured the vSphere Namespace, associate the Content Library that includes the Ubuntu OVA with the vSphere Namespace, and associate the custom VM Class with the vGPU profile with the same vSphere Namespace.

Task	Description
Associate the Content Library with the Ubuntu OVA for vGPU with the vSphere Namespace where you will provision the TKGS cluster.	See Configure a vSphere Namespace for Tanzu Kubernetes releases .
Associate the custom VM Class with the vGPU profile with the vSphere Namespace where you will provision the TKGS cluster.	See Associate a VM Class with a Namespace in vSphere with Tanzu .

The following example shows a configured vSphere Namespace with an associated Content Library and custom VM Class for use with vGPU clusters.



Admin Step 10: Verify that the Supervisor Cluster Is Accessible

The last administration task is to verify that the Supervisor Cluster is provisioned and available for use by the Cluster Operator to provision a TKGS cluster for AI/ML workloads.

- Download and install the Kubernetes CLI Tools for vSphere.
See [Download and Install the Kubernetes CLI Tools for vSphere](#).
- Connect to the Supervisor Cluster.
See [Connect to the Supervisor Cluster as a vCenter Single Sign-On User](#).
- Provide the Cluster Operator with the link to download the Kubernetes CLI Tools for vSphere and the name of the vSphere Namespace.
See [Cluster Operator Workflow for Deploying AI/ML Workloads on TKGS Clusters](#).

Cluster Operator Workflow for Deploying AI/ML Workloads on TKGS Clusters

To enable developers to deploy AI/ML workloads on TKGS clusters, as a Cluster Operator you configure the Kubernetes environment to support NVIDIA vGPU operations.

Cluster Operator Workflow for Deploying AI/ML Workloads on TKGS Clusters

The high-level steps to deploy AI/ML workloads on TKGS clusters are as follows:

Step	Action	Link
0	Review system requirements.	See Operator Step 0: Review System Requirements .
1	Download kubectl and vSphere Plugin for Kubectl to local workstation.	See Operator Step 1: Install the Kubernetes CLI Tools for vSphere on Your Workstation .
2	Use kubectl to log in to the Supervisor Cluster, which populates .kube/config with the context for the new Supervisor Cluster.	See Operator Step 2: Log in to the Supervisor Cluster .
3	Use kubectl to switch context to the vSphere Namespace.	See Operator Step 3: Switch Context to the vSphere Namespace .
4	Use kubectl to list VM classes and verify that the NVIDIA vGPU-enabled class is present.	See Operator Step 4: Get the Custom VM Class for vGPU Workloads .
5	Use kubectl to list the available Tanzu Kubernetes releases and verify that the Ubuntu image is present.	See Operator Step 5: Get the Ubuntu Tanzu Kubernetes Release for GPU Nodes .
6	Craft the YAML specification for provisioning the GPU-enabled TKGS cluster; specify the TKR version and the VM class.	See Operator Step 6: Craft the YAML for Provisioning the vGPU-enabled TKGS Cluster .
7	Provision the TKGS cluster.	See Operator Step 7: Provision the TKGS Cluster .
8	Log in to the cluster and verify provisioning.	See Operator Step 8: Log In to the TKGS Cluster and Verify Provisioning .
9	Prepare to install the NVAIE GPU Operator by creating some prerequisite objects in the TKGS cluster, including a namespace, role bindings, image secret, and license configmap.	See Operator Step 9: Prepare to Install the NVAIE GPU Operator .
10	Install the NVAIE GPU Operator in the cluster.	See Operator Step 10: Install the NVIDIA GPU Operator in the Cluster .
11	Deploy AI/ML workloads to the vGPU-enabled TKGS cluster.	See Operator Step 11: Deploy an AI/ML Workload .

Operator Step 0: Review System Requirements

Refer to the following system requirements to set up the environment for deploying AI/ML workloads on TKGS clusters.

Requirement	Description
vSphere Administrator has set up the environment for NVIDIA vGPU	See vSphere Administrator Workflow for Deploying AI/ML Workloads on TKGS Clusters (vGPU)
TKR Ubuntu OVA	Tanzu Kubernetes release Ubuntu ob-18691651-tkgs-ova-ubuntu-2004-v1.20.8---vmware.1-tkg.2
TKG Cluster Provisioner	Tanzu Kubernetes Grid Service API version: <code>run.tanzu.vmware.com/v1alpha2</code>
NVIDIA GPU Operator	GPU Operator v1.8.0
NVIDIA GPU Driver Container	<code>nvcr.io/nvstating/cnt-ea/driver:470.51-ubuntu20.04</code>

Operator Step 1: Install the Kubernetes CLI Tools for vSphere on Your Workstation

Download and install the Kubernetes CLI Tools for vSphere.

If you are using Linux you can run the following command to download the tools.

```
curl -LOk https://${SC_IP}/wcp/plugin/linux-amd64/vsphere-plugin.zip
unzip vsphere-plugin.zip
mv -v bin/* /usr/local/bin/
```

For additional guidance, see [Download and Install the Kubernetes CLI Tools for vSphere](#).

Operator Step 2: Log in to the Supervisor Cluster

Using the vSphere Plugin for kubectl, authenticate with the Supervisor Cluster.

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME
```

Operator Step 3: Switch Context to the vSphere Namespace

Using kubectl, switch context to the vSphere Namespace that the vSphere administrator created for the TKGS GPU cluster.

```
kubectl config get-contexts
```

```
kubectl config use-context TKGS-GPU-CLUSTER-NAMESPACE
```

Operator Step 4: Get the Custom VM Class for vGPU Workloads

Verify that the custom VM Class with the vGPU profile that the vSphere Administrator created is available in the target vSphere Namespace.

```
kubectl get virtualmachineclassbindings
```

Note The VM class must be bound to the target vSphere Namespace. If you do not see the custom VM class for vGPU workloads, check with the vSphere Administrator.

Operator Step 5: Get the Ubuntu Tanzu Kubernetes Release for GPU Nodes

Verify that the required Ubuntu Tanzu Kubernetes release that the vSphere Administrator synchronized from the Content Library is available in the vSphere Namespace.

```
kubectl get tanzukubernetesreleases
```

Or, using the shortcut:

```
kubectl get tkr
```

Operator Step 6: Craft the YAML for Provisioning the vGPU-enabled TKGS Cluster

Construct the YAML file for provisioning a Tanzu Kubernetes cluster.

Start with one of the of the examples below. Use the information you gleaned from the output of the preceding commands to customize the cluster specification. Refer to the full list of configuration parameters: [TKGS v1alpha2 API for Provisioning Tanzu Kubernetes Clusters](#).

Example 1 specifies two worker node pools.

```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  #cluster name
  name: tkgs-cluster-gpu-a100
  #target vsphere namespace
  namespace: tkgs-gpu-operator
spec:
  topology:
    controlPlane:
      replicas: 3
      #storage class for control plane nodes
      #use `kubectl describe storageclasses`
      #to get available pvcs
      storageClass: vwt-storage-policy
      vmClass: guaranteed-medium
      #TKR NAME for Ubuntu ova supporting GPU
      tkr:
        reference:
          name: 1.20.8---vmware.1-tkg.1
    nodePools:
      - name: nodepool-a100-primary
        replicas: 3
        storageClass: vwt-storage-policy
        #custom VM class for vGPU
        vmClass: class-vgpu-a100
        #TKR NAME for Ubuntu ova supporting GPU
        tkr:
          reference:
            name: 1.20.8---vmware.1-tkg.1
      - name: nodepool-a100-secondary
        replicas: 3
```

```

vmClass: class-vgpu-a100
storageClass: vwt-storage-policy
#TKR NAME for Ubuntu ova supporting GPU
tkr:
  reference:
    name: 1.20.8---vmware.1-tkg.1
settings:
  storage:
    defaultClass: vwt-storage-policy
  network:
    cni:
      name: antrea
    services:
      cidrBlocks: ["198.51.100.0/12"]
    pods:
      cidrBlocks: ["192.0.2.0/16"]
  serviceDomain: managedcluster.local

```

Example 2 specifies a separate volume on worker nodes for the containerd runtime with a capacity of 50 GiB. This setting is configurable. Providing a separate volume of good size is recommended for container-based AI/ML workloads.

```

apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkc
  namespace: tkg-ns-auto
spec:
  distribution:
    fullVersion: v1.20.8+vmware.1-tkg.1
  topology:
    controlPlane:
      replicas: 3
      storageClass: vwt-storage-policy
      tkr:
        reference:
          name: v1.20.8---vmware.1-tkg.1
      vmClass: best-effort-medium
    nodePools:
      - name: workers
        replicas: 3
        storageClass: k8s-storage-policy
        tkr:
          reference:
            name: v1.20.8---vmware.1-tkg.1
        vmClass: vmclass-vgpu
        volumes:
          - capacity:
              storage: 50Gi
              mountPath: /var/lib/containerd
              name: containerd
          - capacity:
              storage: 50Gi
              mountPath: /var/lib/kubelet

```

```

      name: kubelet
-   name: nodepool-1
      replicas: 1
      storageClass: vwt-storage-policy
      vmClass: best-effort-medium

```

Example 3 includes cluster additional metadata such as a label.

```

apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  annotations:
  labels:
    run.tanzu.vmware.com/tkr: v1.20.8---vmware.1-tkg.1
  name: tkgs-gpu-direct-rdma
  namespace: tkgs-ns
spec:
  settings:
    network:
      cni:
        name: antrea
    pods:
      cidrBlocks:
        - 192.168.0.0/16
      serviceDomain: cluster.local
    services:
      cidrBlocks:
        - 10.96.0.0/12
  topology:
    controlPlane:
      replicas: 3
      storageClass: tkgs-storage-policy
      vmClass: guaranteed-medium
      tkr:
        reference:
          name: v1.20.8---vmware.1-tkg.1
    nodePools:
      - name: workers
        replicas: 5
        storageClass: tkgs-storage-policy
        vmClass: claire-gpu-direct-rdma
        volumes:
          - capacity:
              storage: 50Gi
            mountPath: /var/lib/containerd
            name: containerd
          - capacity:
              storage: 50Gi
            mountPath: /var/lib/kubelet
            name: kubelet
      tkr:
        reference:
          name: v1.20.8---vmware.1-tkg.1

```

Operator Step 7: Provision the TKGS Cluster

Provision the cluster by running the following `kubectl` command.

```
kubectl apply -f CLUSTER-NAME.yaml
```

For example:

```
kubectl apply -f tkgs-gpu-cluster-1.yaml
```

Monitor the deployment of cluster nodes using `kubectl`.

```
kubectl get tanzukubernetesclusters -n NAMESPACE
```

Operator Step 8: Log In to the TKGS Cluster and Verify Provisioning

Using the vSphere Plugin for `kubectl`, log in to the TKGS cluster.

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME \
--tanzu-kubernetes-cluster-name CLUSTER-NAME --tanzu-kubernetes-cluster-namespace NAMESPACE-NAME
```

Use the following commands to verify the cluster:

```
kubectl cluster-info
```

```
kubectl get nodes
```

```
kubectl get namespaces
```

```
kubectl api-resources
```

Operator Step 9: Prepare to Install the NVAIE GPU Operator

Prior to installing the GPU Operator with NVIDIA AI Enterprise, complete the following tasks for the TKGS cluster you provisioned. For additional guidance, see [Prerequisite Tasks](#) in the NVAIE documentation.

Note If you are using the NVIDIA Delegated Licensing Server (DLS), refer to the following topic for instructions: [Cluster Operator Addendum for Deploying AI/ML Workloads on TKGS Clusters \(DLS\)](#)

- 1 Create the Kubernetes namespace `gpu-operator-resources`. As a best practice, always deploy everything in this namespace.

```
kubectl create ns gpu-operator-resources
```

- 2 Create role bindings.

Tanzu Kubernetes clusters have pod security policy enabled.

Create `rolebindings.yaml`.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: psp:vmware-system-privileged:default
  namespace: default
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: psp:vmware-system-privileged
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:nodes
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:serviceaccounts
```

Apply the role binding.

```
kubectl apply -f rolebindings.yaml
```

Create `post-rolebindings.yaml`.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: psp:vmware-system-privileged:gpu-operator-resources
  namespace: gpu-operator-resources
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: psp:vmware-system-privileged
subjects:
- kind: Group
  apiGroup: rbac.authorization.k8s.io
  name: system:serviceaccounts
```

Apply the role binding:

```
kubectl apply -f post-rolebindings.yaml
```

- 3 Create an image secret with NGC credentials that can be used by Docker to pull container images from the [NVIDIA GPU Cloud Catalog](#).

```
kubectl create secret docker-registry registry-secret \
  --docker-server=server-name --docker-username='$oauthtoken' \
  --docker-password=<place_holder> \
  --docker-email=email-name -n gpu-operator-resources
```

4 Create a configmap for the NVIDIA license server.

```
kubectl create configmap licensing-config -n gpu-operator-resources --from-file=gridd.conf
```

The `gridd.conf` reference the NVIDIA license server address, for example:

```
# Description: Set License Server Address
# Data type: string
# Format: "<address>"
ServerAddress=<place_holder>
```

Operator Step 10: Install the NVIDIA GPU Operator in the Cluster

Install the [NVAIE GPU Operator](#) version 1.8.0 in the TKGS cluster. For additional guidance, refer to the GPU Operator [documentation](#).

Note If you are using the NVIDIA Delegated Licensing Server (DLS), refer to the following topic for instructions: [Cluster Operator Addendum for Deploying AI/ML Workloads on TKGS Clusters \(DLS\)](#)

- 1 Install Helm by referring to the [Helm documentation](#).
- 2 Add the `gpu-operator` Helm repository.

```
helm repo add nvidia https://nvidia.github.io/gpu-operator
```

- 3 Install the NVAIE GPU Operator by running the following command.

Where necessary, substitute environment variable values with those that match your environment.

```
export PRIVATE_REGISTRY="private/registry/path"
export OS_TAG=ubuntu20.04
export VERSION=460.73.01
export VGPU_DRIVER_VERSION=460.73.01-grid
export NGC_API_KEY=ZmJjMHZya...LWExNTRi
export REGISTRY_SECRET_NAME=registry-secret

helm install nvidia/gpu-operator \
  --set driver.repository=$PRIVATE_REGISTRY \
  --set driver.version=$VERSION \
  --set driver.imagePullSecrets=${REGISTRY_SECRET_NAME} \
  --set operator.defaultRuntime=containerd \
  --set driver.licensingConfig.configMapName=licensing-config
```

Operator Step 11: Deploy an AI/ML Workload

The [NVIDIA GPU Cloud Catalog](#) offers several off-the-shelf container images you can use to run AI/ML workloads on your vGPU-enabled Tanzu Kubernetes clusters. For more information on the images available, see the [NGC documentation](#).

vSphere Administrator Addendum for Deploying AI/ML Workloads on TKGS Clusters (vGPU and Dynamic DirectPath IO)

Refer to this delta topic if you are configuring TKGS to support AI/ML workloads using vGPU and Dynamic DirectPath IO.

vSphere Administrator Workflow Adjustments for vGPU with Dynamic DirectPath IO

To use vGPU and Dynamic DirectPath IO, follow the same [vSphere Administrator Workflow for Deploying AI/ML Workloads on TKGS Clusters \(vGPU\)](#) with the following changes.

Admin Step 2: Enable Passthrough for the PCI Device

To use vGPU and Dynamic DirectPath IO, configure each ESXi host for vGPU as described here: [Admin Step 2: Configure Each ESXi Host for vGPU Operations](#).

In addition, configure the GPU device as follows.

- 1 Log on to the vCenter Server using the vSphere Client.
- 2 Select the target ESXi host in the vCenter Cluster.
- 3 Select **Configure > Hardware > PCI Devices**.
- 4 Select the **All PCI Devices** tab.
- 5 Select the target NVIDIA GPU accelerator device.
- 6 Click **Toggle Passthrough**.
- 7 Right-click the ESXi host and put it into maintenance mode.
- 8 Reboot the host.
- 9 When the host is running again, take it out of maintenance mode.

Admin Step 7: Create a Custom VM Class with a vGPU and Dynamic DirectPath IO

To use vGPU and Dynamic DirectPath IO, first configure a custom VM Class with a **NVIDIA vGPU** profile as described here: [Admin Step 7: Create a Custom VM Class with the vGPU Profile](#).

Then, to this VM Class you add a second PCI Device configuration with **Dynamic DirectPath IO** specified and the supported PCI Device selected. When a VM Class of this type is instantiated, the vSphere Distributed Resource Scheduler (DRS) determines VM placement.

Refer to the following instructions to create a custom VM Class supporting vGPU and Dynamic DirectPath IO. For additional guidance, see [Add PCI Devices to a VM Class in vSphere with Tanzu](#).

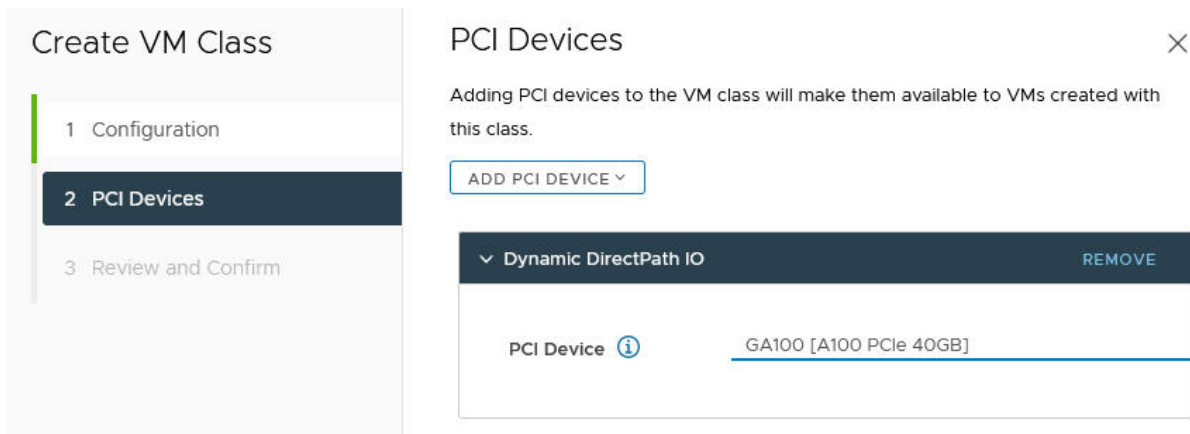
- 1 Log on to the vCenter Server using the vSphere Client.
- 2 Select **Workload Management**.
- 3 Select **Services**.
- 4 Select **VM Classes**.

- 5 Edit the custom VM Class that is already configured with a **NVIDIA vGPU** profile.
- 6 Select the **PCI Devices** tab.
- 7 Click **Add PCI Device**.
- 8 Select the **Dynamic DirectPath IO** option.



- 9 Select the **PCI Device**.

For example:



- 10 Click **Next**.
- 11 Review and confirm your selections.
- 12 Click **Finish**.
- 13 Verify that the new custom VM Class is available in the list of VM Classes.

Cluster Operator Addendum for Deploying AI/ML Workloads on TKGS Clusters (DLS)

Refer to this delta topic if you are using the NVIDIA Delegated Licensing Server (DLS) for your NVIDIA AI Enterprise account.

Cluster Operator Addendum for Deploying AI/ML Workloads on TKGS Clusters

NVIDIA provides a new NVIDIA Licensing Server (NLS) system called DLS which stands for Delegated Licensing Server. For more information, refer to the NVIDIA [documentation](#).

If you are using DLS for your NVAIE account, the steps for preparing to and deploying the NVAIE GPU Operator are different than what is documented here: [Cluster Operator Workflow for Deploying AI/ML Workloads on TKGS Clusters](#). Specifically, Steps 9 and 10 are modified as follows.

Operator Step 9: Prepare to Install the NVAIE GPU Operator

Complete the following steps to prepare to install the GPU Operator using a DLS.

- 1 Create a Secret.

```
kubectl create secret docker-registry registry-secret \
  --docker-server=<users private NGC registry name>
  --docker-username='$oauthtoken' \
  --docker-password=ZmJj.....Ri \
  --docker-email=<user-email-address> -n gpu-operator-resources
```

Note The password is the user API Key that was previously created on the NVIDIA GPU Cloud (NGC) Portal.

- 2 Get a Client Token from the DLS Server.

A user who wishes to use a vGPU license will need to get a token from that DLS license server called a “Client token. The mechanism for doing this is in the NVIDIA [documentation](#).

- 3 Create a ConfigMap object in the TKGS cluster using the Client Token.

Place the Client Token file into a file at <path>/client_configuration_token.tok.

Then, run the following command:

```
kubectl delete configmap licensing-config -n gpu-operator-resources; > gridd.conf
kubectl create configmap licensing-config \
  -n gpu-operator-resources --from-file=./gridd.conf --from-file=./
  client_configuration_token.tok
```

Note The grid.conf file used by the DLS is empty. However, both the “--from-file” parameters are required.

Operator Step 10: Install the NVAIE GPU Operator

Complete the following steps to install the NVAIE GPU Operator using a DLS. For additional guidance, refer to the GPU Operator [documentation](#).

- 1 Install the [NVAIE GPU Operator](#) in the TKGS cluster.
 - Install Helm by referring to the [Helm documentation](#).

- Add the `gpu-operator` Helm repository.

```
helm repo add nvidia https://nvidia.github.io/gpu-operator
```

- Install the GPU Operator using Helm.

```
export PRIVATE_REGISTRY="<user's private registry name>"
export OS_TAG=ubuntu20.04
export VERSION=470.63.01
export VGPU_DRIVER_VERSION=470.63.01-grid
export NGC_API_KEY=Zm.....Ri  <- The user's NGC AP Key
export REGISTRY_SECRET_NAME=registry-secret

helm show chart .
kubectl delete crd clusterpolicies.nvidia.com
helm install gpu-operator . -n gpu-operator-resources \
  --set psp.enabled=true \
  --set driver.licensingConfig.configMapName=licensing-config \
  --set operator.defaultRuntime=containerd \
  --set driver.imagePullSecrets={$REGISTRY_SECRET_NAME} \
  --set driver.version=$VERSION \
  --set driver.repository=$PRIVATE_REGISTRY \
  --set driver.licensingConfig.nlsEnabled=true
```

2 Verify that DLS has worked.

From within a NVIDIA Driver DaemonSet pod that was deployed by the GPU Operator, execute the `nvidia-smi` command to verify that DLS is working.

First, run the following command to get into the pod and bring up a shell session:

```
kubectl exec -it nvidia-driver-daemonset-cvxx6 nvidia-driver-ctr -n gpu-operator-resources
- bash
```

Now you can run the command to verify the DLS setup.

```
nvidia-smi
```

If DLS is setup correctly, this command should return "Licensed" in the output.

Using a Container Registry for vSphere with Tanzu Workloads

15

Container registries provide Kubernetes operators with a convenient repository for storing and sharing container images. vSphere with Tanzu includes an embedded Harbor Registry that you can enable on the Supervisor Cluster. You can also use an external private container registry with Tanzu Kubernetes clusters.

You can enable the embedded Harbor Registry on the Supervisor Cluster to serve as the private container registry for the deployment of vSphere Pods and Tanzu Kubernetes cluster workloads. You provide the registry URL to developers who can use the vSphere Docker Credential Helper to securely access the registry and push and pull container images.

Note The embedded Harbor Registry requires that the Supervisor Cluster is configured to use NSX-T networking.

As an alternative to the embedded Harbor Registry, or in addition to, you can configure Tanzu Kubernetes clusters to use an external private container registry.

Read the following topics next:

- [Enable the Embedded Harbor Registry on the Supervisor Cluster](#)
- [Log In to the Embedded Harbor Registry Console](#)
- [Download and Install the Embedded Harbor Registry Certificate](#)
- [Configure a Docker Client with the Embedded Harbor Registry Certificate](#)
- [Install the vSphere Docker Credential Helper and Connect to the Registry](#)
- [Push Images to the Embedded Harbor Registry](#)
- [Purge Images from the Embedded Harbor Registry](#)
- [Use the Embedded Harbor Registry with Tanzu Kubernetes Clusters](#)
- [Use an External Container Registry with Tanzu Kubernetes Clusters](#)

Enable the Embedded Harbor Registry on the Supervisor Cluster

As a vSphere administrator, you can enable the Harbor Registry that is embedded with vSphere with Tanzu. You can push and pull container images from the registry as well as deploy containers using these images.

Once the Harbor Registry is enabled, every namespace on the Supervisor Cluster has a matching project with the same name as the private image registry. Every user or group that has edit or view permission to a namespace becomes a corresponding role member on the matching project with the same name in the private image registry. The lifecycle of projects and members of the private image registry is automatically managed and is linked to the lifecycle of namespaces and user or group permissions in namespaces.

Prerequisites

To enable the embedded Harbor Registry, you must have enabled **Workload Management** and deployed a Supervisor Cluster. In addition, create a storage policy for placement of container images. This storage policy is used to provision persistent volumes to use as the backing store for the container images in the registry.

Note To use the embedded Harbor Registry, you must deploy the Supervisor Cluster with NSX-T Data Center as the networking solution. See [Configuring NSX for vSphere with Tanzu](#).

Procedure

- 1 In the vSphere Client, browse to the vCenter cluster where **Workload Management** is enabled.
- 2 Select **Configure**.
- 3 Select **Supervisor Cluster**.
- 4 Select **Image Registry**.
- 5 Click **Enable Harbor**.
- 6 Select the **Storage Policy** for placement of container images.
- 7 Click **OK** to complete the process.

Results

A private image registry becomes enabled after a few minutes. A special namespace is created for that instance of the private image registry. You cannot perform any operations on that namespace, it is read only for vSphere users.

What to do next

[Log In to the Embedded Harbor Registry Console](#).

Log In to the Embedded Harbor Registry Console

Use the embedded Harbor Registry Admin Console to manage and operate the private registry.

As a vSphere Administrator, you can use the embedded Harbor Registry Admin Console to create and manage projects, view logs, and explore the Harbor API.

Prerequisites

[Enable the Embedded Harbor Registry on the Supervisor Cluster](#)

Procedure

- 1 In the vSphere Client, browse to the vCenter cluster where **Workload Management** is enabled.
- 2 Select **Configure**.
- 3 Under **Namespaces**, select **Image Registry**.
- 4 Click the **Link to Harbor UI**.

The embedded Harbor Registry Console log in page appears.

- 5 Log in using your vSphere administrator credentials.

Download and Install the Embedded Harbor Registry Certificate

Download the embedded Harbor Registry root CA certificate so you can use it to connect clients to the registry.

To log in to the embedded Harbor Registry using a Docker client, you must install the root CA certificate on that client.

Prerequisites

This task assumes that you have installed and configured Docker on a client host machine. In addition, the embedded Harbor Registry must be enabled. See [Enable the Embedded Harbor Registry on the Supervisor Cluster](#).

Procedure

- 1 Download the embedded Harbor Registry certificate. There are two ways to do this.
 - Using the embedded Harbor Registry Console interface.
 - Log in to the embedded Harbor Registry Console using the URL. See [Log In to the Embedded Harbor Registry Console](#).
 - Click the project link at the **Projects > Project Name** page.
 - Select the **Repositories** tab.
 - Click **Registry Certificate**.

- Save the `ca.crt` certificate file to your local machine.
 - Using the vSphere Client.
 - Select the vCenter cluster where **Workload Management** and the embedded Harbor Registry are enabled.
 - Select **Configure > Namespaces > Image Registry**.
 - In the **Root certificate** field, click the link **Download SSL Root Certificate**.
 - Save the `root-certificate.txt` file to your local machine.
 - Rename the file to be `ca.crt`.
- 2 Copy the embedded Harbor Registry `ca.crt` file that you downloaded to the appropriate directory on a host where Docker is installed. The default certificate location differs depending on the type of OS that the Docker client is running.

- Linux

```
/etc/docker/certs.d/ca.crt
```

- Mac OS

```
security add-trusted-cert -d -r trustRoot -k ~/Library/Keychains/login.keychain ca.crt
```

Note If you do not install the `ca.crt` in the default location, you can pass the `--tlscacert /path/to/ca.crt` flag when you log in using the vSphere Docker Credential Helper.

- 3 Once the import is complete, restart the Docker daemon.

- Linux

```
sudo systemctl restart docker.service
```

- Mac

Use the Docker Desktop menu option **Restart Docker**, or the `command R` keyboard shortcut.

What to do next

[Install the vSphere Docker Credential Helper and Connect to the Registry](#)

Configure a Docker Client with the Embedded Harbor Registry Certificate

To work with container images in the embedded Harbor Registry using Docker, you must add the registry certificate to your Docker client. The certificate is used to authenticate to Docker during login.

Configure your Docker client to interact with the embedded Harbor Registry. This task is required in preparation for using the Docker Credential Helper that vSphere provides to connect and interact with the embedded Harbor Registry.

Prerequisites

This task assumes that the embedded Harbor Registry is enabled and that you can log in:

- [Enable the Embedded Harbor Registry on the Supervisor Cluster](#)
- [Log In to the Embedded Harbor Registry Console](#)

In addition, the instructions assume that you are using a Linux host (Ubuntu) on which the Docker daemon is installed. To verify that Docker is installed and that you can pull images from the Docker hub, run the following command:

```
docker run hello-world
```

Expected result:

```
Hello from Docker!
This message shows that your installation appears to be working correctly.
```

Note These instructions are verified using Ubuntu 20.04 and Docker 19.03.

Procedure

- 1 Download the embedded Harbor Registry certificate `root-certificate.txt`. See [Download and Install the Embedded Harbor Registry Certificate](#).
- 2 Change the name of the certificate to `ca.crt`.
- 3 Securely copy the `ca.crt` file to your Docker host.
- 4 On the Docker host, create a directory path for the private registry using the Harbor IP address.

```
/etc/docker/certs.d/IP-address-of-harbor/
```

For example:

```
mkdir /etc/docker/certs.d/10.179.145.77
```

- 5 Move the `ca.crt` to this directory.

For example:

```
mv ca.crt /etc/docker/certs.d/10.179.145.77/ca.crt
```

- 6 Restart the Docker daemon.

```
sudo systemctl restart docker.service
```

7 Log in to the embedded Harbor Registry using your Docker client.

```
docker login https://10.179.145.77
```

You should see the following message:

```
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

What to do next

As indicated by the message, for security purposes, download and install the vSphere Docker Credential Helper. See [Install the vSphere Docker Credential Helper and Connect to the Registry](#).

Install the vSphere Docker Credential Helper and Connect to the Registry

Use the vSphere Docker Credential Helper CLI to securely push container images to and pull container images from the embedded Harbor Registry.

The Kubernetes CLI Tools download page includes a link to download the vSphere Docker Credential Helper. Use the vSphere Docker Credential Helper to securely connect your Docker client to the embedded Harbor Registry.

Prerequisites

- [Enable the Embedded Harbor Registry on the Supervisor Cluster](#)
-
- Get the link to the Kubernetes CLI Tools for vSphere download page from your vSphere administrator.
- Alternatively, if you have access to the vCenter Server, get the link as follows:
 - Log in to the vCenter Server using the vSphere Client.
 - Navigate to **vSphere Cluster > Namespaces** and select the vSphere Namespace where you are working.
 - Select the **Summary** tab and locate the **Status** tile.
 - Select **Open** beneath the **Link to CLI Tools** heading to open the download page. Or, you can **Copy** the link.
- Configure a Docker client. See [Configure a Docker Client with the Embedded Harbor Registry Certificate](#).

Procedure

- 1 Using a browser, navigate to the **Kubernetes CLI Tools** download URL for your environment.
- 2 Scroll down to the vSphere Docker Credential Helper section.
- 3 Select the operating system.
- 4 Download the `vsphere-docker-credential-helper.zip` file.
- 5 Extract the contents of the ZIP file to a working directory.

The `docker-credential-vsphere` binary executable is available.

- 6 Copy the `docker-credential-vsphere` binary to your Docker client host.
- 7 Add the location of the binary to your system PATH.

For example, on Linux:

```
mv docker-credential-vsphere /usr/local/bin/docker-credential-vsphere
```

- 8 Verify the installation of the vSphere Docker Credential Helper by running the command `docker-credential-vsphere` in a shell or terminal session.

You see the banner message, and the list of command-line options for the CLI.

```
vSphere login manager is responsible for vSphere authentication.
It allows vSphere users to securely login and logout to access Harbor images.

Usage:
  docker-credential-vsphere [command]

Available Commands:
  help          Help about any command
  login         Login into specific harbor server and get authentication
  logout        Logout from Harbor server and erase user token

Flags:
  -h, --help    help for docker-credential-vsphere

Use "docker-credential-vsphere [command] --help" for more information about a command.
```

- 9 Log in to the embedded Harbor Registry.

First, check the usage:

```
docker-credential-vsphere login -help
Usage:
  docker-credential-vsphere login [harbor-registry] [flags]

Flags:
```

```
-h, --help            help for login
-s, --service string  credential store service
    --tlscacert string location to CA certificate (default "/etc/docker/certs.d/*.crt")
-u, --user string     vSphere username and password
```

Then, log in using the following command:

```
docker-credential-vsphere login <container-registry-IP>
```

The authentication token is fetched and saved, and you are logged in.

```
docker-credential-vsphere login 10.179.145.77
Username: administrator@vsphere.local
Password: INFO[0017] Fetched username and password
INFO[0017] Fetched auth token
INFO[0017] Saved auth token
```

10 Log out of the embedded Harbor Registry.

```
docker-credential-vsphere logout 10.179.145.77
```

What to do next

[Push Images to the Embedded Harbor Registry](#) .

Push Images to the Embedded Harbor Registry

You can push images from Docker to a project on the embedded Harbor Registry. Projects in the embedded Harbor Registry correspond to vSphere namespaces on a Supervisor Cluster.

Prerequisites

It is assumed that the following tasks are completed:

- [Enable the Embedded Harbor Registry on the Supervisor Cluster](#)
- [Install the vSphere Docker Credential Helper and Connect to the Registry](#)

In addition, obtain your user account for which you have write permissions on the namespace that corresponds to the project on Harbor Registry where you want to push images.

Lastly, you need a local image that you can push to the registry. The following command pulls the hello-world image from Docker Hub. You will need an account there to pull the image.

```
docker run hello-world
```

Expected result:

```
Hello from Docker!
This message shows that your installation appears to be working correctly.
```

Verify the image using the `docker images` command.

```
docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello-world	latest	bf756fb1ae65	10 months ago	13.3kB

Procedure

- 1 Login to Harbor Registry with the vSphere Docker Credential Helper.

```
docker-credential-vsphere login <container-registry-IP> --user username@domain.com
```

Note While providing `--user username` is acceptable for login, you should use the UserPrincipalName (UPN) syntax (`--user username@domain.com`) to login and use `docker push` commands.

- 2 Tag the image that you want to push to the project in Harbor Registry with same name as the namespace, where you want to use it:

```
docker tag <image-name>[:TAG] <container-registry-IP>/<project-name>/<image-name>[:TAG]
```

For example:

```
docker tag hello-world:latest 10.179.145.77/tkgs-cluster-ns/hello-world:latest
```

```
docker images
```

REPOSITORY	TAG	IMAGE ID
10.179.145.77/tkgs-cluster-ns/hello-world	latest	bf756fb1ae65
hello-world	latest	bf756fb1ae65

- 3 To push an image to a project in Harbor, run the following command:

Syntax:

```
docker push <container-registry-IP>/<namespace-name>/<image_name>
```

For example:

```
docker push 10.179.145.77/tkgs-cluster-ns/hello-world:latest
```

Expected result.

```
The push refers to repository [10.179.145.77/tkgs-cluster-ns/hello-world]
9c27e219663c: Pushed
latest: digest: sha256:90659bf80b44ce6be8234e6ff90a1ac34acbeb826903b02cfa0da11c82cbc042
size: 525
```

4 Verify that the image is now available in the embedded Harbor Registry.

- [Log In to the Embedded Harbor Registry Console](#)
- Click the project link at the **Projects > Project Name**.
- Select the **Repositories** tab.
- You should see the image that you pushed to the registry is present, in the form namespace/image-name, such as tkgs-cluster-ns/hello-world.
- Select this image and you see the `latest` tag and other metadata.

5 Navigate back to the **Repositories** tab.

6 Select the **Push Image Docker Command** drop down menu. The commands to tag and push images to this repository are provided to you.

Example

Here is another example image push to the embedded Harbor Registry:

```
docker tag busybox:latest <container-registry-IP>/<namespace-name>/busybox:latest
docker push <container-registry-IP>/busybox/busybox:latest
```

What to do next

Deploy vSphere Pods by using images from the Harbor registry. See [Deploy an Application to a vSphere Pod Using the Embedded Harbor Registry](#).

Purge Images from the Embedded Harbor Registry

As a vSphere administrator, you can purge the images for a project in the private image registry by request from DevOps engineers. Purging images from the private image registry deletes all references to the images made by pods, but it does not remove the images from the image registry.

In case DevOps engineers report that too many images are stored for a project, as a vSphere administrator you can purge the images for that project in the private image registry. Purging the images from a project deletes all references to these images, but does not delete them from the vSphere datastore. Every Saturday at 2 AM, the garbage collector service deletes all images from the entire private image registry that are not referenced by applications.

Procedure

- 1 In the vSphere Client, navigate to the namespace.
- 2 Select **Configure** and select **General**.
- 3 Next to **Embedded Registry**, click **Purge**.

Use the Embedded Harbor Registry with Tanzu Kubernetes Clusters

You can use the embedded Harbor Registry to serve as the private container registry for images that you deploy to Tanzu Kubernetes clusters provisioned by the Tanzu Kubernetes Grid Service.

vSphere with Tanzu embeds a Harbor Registry instance that you can enable on the Supervisor Cluster and use to deploy container-based workloads to Tanzu Kubernetes clusters.

Once the embedded Harbor Registry is enabled on the Supervisor Cluster, the Tanzu Kubernetes Grid Service will install onto the Tanzu Kubernetes cluster nodes the root CA certificate for the registry instance. This certificate is installed on both new clusters and on existing clusters (by way of a reconciliation loop). From there you can run images on the cluster by specifying the private registry in the workload YAML.

Workflow

Use the following workflow to securely access the private registry from Tanzu Kubernetes cluster nodes and pull container images.

Step	Action	Instructions
0	Review the workflow for using the embedded Harbor Registry with Tanzu Kubernetes clusters.	See Chapter 15 Using a Container Registry for vSphere with Tanzu Workloads .
1	Enable the embedded Harbor Registry on the Supervisor Cluster.	See Enable the Embedded Harbor Registry on the Supervisor Cluster .
2	Configure the kubeconfig for each cluster with the Registry Service Secret.	See the instructions below: Configure a Tanzu Kubernetes cluster with the image pull secret for the embedded Harbor Registry.
3	Configure the workload YAML to specify the private container registry.	See the instructions below: Configure a Tanzu Kubernetes cluster with the image pull secret for the embedded Harbor Registry.
4	To push images to the embedded Harbor Registry, configure a Docker client and install the vSphere Docker Credential Helper.	See Configure a Docker Client with the Embedded Harbor Registry Certificate and Push Images to the Embedded Harbor Registry .

Configure a Tanzu Kubernetes Cluster with the Image Pull Secret for the Embedded Harbor Registry

Configure your kubeconfig with the image pull secret to connect a Tanzu Kubernetes cluster to a private container registry, either the embedded Harbor Registry or an external private registry.

- 1 Connect to the Supervisor Cluster. See [Connect to the Supervisor Cluster as a vCenter Single Sign-On User](#).
- 2 Switch context to the vSphere Namespace where the target Tanzu Kubernetes cluster is provisioned.

```
kubectl config use-context tkgs-cluster-ns
```

- 3 Get the image pull secret for the vSphere Namespace and store it in a file.

```
kubectl get secret -n <vsphere-namespace> <vsphere-namespace>-default-image-pull-secret -o yml > <path>/image-pull-secret.yaml
```

For example:

```
kubectl get secret -n tkgs-cluster-ns tkgs-cluster-ns-default-image-pull-secret -o yml > tanzu/image-pull-secret.yaml
```

- 4 Open `image-pull-secret.yaml` with a text editor. At a minimum make the required change and save the file when you are done.

```
apiVersion: v1
data:
  .dockerconfigjson: ewoJCQkJImFldGhzJUV2s1ZVZwWVFuWmp...
kind: Secret
metadata:
  creationTimestamp: "2020-11-12T02:41:08Z"
  managedFields:
    - apiVersion: v1
      ...
  name: harbor-registry-secret #OPTIONAL: Change if desired
  namespace: default #REQUIRED: Enter the Kubernetes namespace
  ownerReferences:
    - apiVersion: registryagent.vmware.com/v1alpha1
      ...
  resourceVersion: "675868"
```

```
selfLink: /api/v1/namespaces/tkgs-cluster-ns/secrets/tkgs-cluster-ns-default-image-pull-secret
uid: 66606b41-7363-4b74-a3f2-4436f83f
type: kubernetes.io/dockerconfigjson
```

- **REQUIRED:** Change the value for `namespace` to match an appropriate Kubernetes namespace in the cluster, such as **default**.

Note To configure the image pull secret, specify a Kubernetes namespace. If the Tanzu Kubernetes cluster already exists, switch context to it and run `kubectl get namespaces` to list available Kubernetes namespaces. If necessary create the target namespace before proceeding. If the Tanzu Kubernetes cluster does not exist, you can use the `default` namespace.

- **OPTIONAL:** Change the value for `name` to something meaningful, such as **harbor-registry-secret** Or **private-registry-secret**.

- 5 Create a kubeconfig file that can be used to access the Tanzu Kubernetes cluster.

```
kubectl get secret -n <vsphere-namespace> <cluster-name>-kubeconfig -o
jsonpath='{.data.value}' | base64 -d > <path>/cluster-kubeconfig
```

Replace `<vsphere-namespace>` with the name of the vSphere Namespace where the target Tanzu Kubernetes cluster is provisioned. Replace `<cluster-name>` with the name of the Tanzu Kubernetes cluster.

```
kubectl get secret -n tkgs-cluster-ns tkgs-cluster-5-kubeconfig -o
jsonpath='{.data.value}' | base64 -d > tanzu/cluster-kubeconfig
```

- 6 Create the Registry Service secret in the Tanzu Kubernetes cluster. Reference the image pull secret file that you saved and updated locally.

```
kubectl --kubeconfig=<path>/cluster-kubeconfig apply -f <path>/image-pull-secret.yaml
```

For example:

```
kubectl --kubeconfig=tanzu/cluster-kubeconfig apply -f tanzu/image-pull-secret.yaml
```

You should see that the Registry Service secret is successfully created.

```
secret/harbor-registry-secret created
```

Configure a Tanzu Kubernetes Workload to Pull Images from a Private Container Registry

To pull images from a private container registry for a Tanzu Kubernetes cluster workload, configure the workload YAML with the private registry details.

This procedure can be used to pull images from a private container registry, or the embedded Harbor Registry. In this example, we create a pod specification that will use an image stored in the embedded Harbor Registry and utilize the image pull secret previously configured.

- 1 Create an example pod spec with the details about the private registry.

```
apiVersion: v1
kind: Pod
metadata:
  name: <workload-name>
  namespace: <kubernetes-namespace>
spec:
  containers:
  - name: private-reg-container
    image: <Registry-IP-Address>/<vsphere-namespace>/<image-name>:<version>
    imagePullSecrets:
    - name: <registry-secret-name>
```

- Replace <workload-name> with the name of the pod workload.
- Replace <kubernetes-namespace> with the Kubernetes namespace in the cluster where the pod will be created. This must be the same Kubernetes namespace where the Registry Service image pull secret is stored in the Tanzu Kubernetes cluster (such as the default namespace).
- Replace <Registry-IP-Address> with the IP address for the embedded Harbor Registry instance running on the Supervisor Cluster.
- Replace <vsphere-namespace> with the vSphere Namespace where the target Tanzu Kubernetes is provisioned.
- Replace <image-name> with an image name of your choice.
- Replace <version> with an appropriate version of the image, such as "latest".
- Replace <registry-secret-name> with the name of the Registry Service image pull secret that you created previously.

- 2 Create a workload in the Tanzu Kubernetes cluster based on the pod specification you defined.

```
kubect1 --kubeconfig=<path>/cluster-kubeconfig apply -f <pod.yaml>
```

The pod should be created from the image pulled from the registry.

Use an External Container Registry with Tanzu Kubernetes Clusters

You can use an external container registry with Tanzu Kubernetes cluster pods. This is an alternative to using the embedded Harbor Registry.

External Private Registry Use Case

Container registries provide a critical function for Kubernetes deployments, serving as a centralized repository for storing and sharing container images. The most commonly used public container registry is [DockerHub](#). There are many private container registry offerings. VMware [Harbor](#) is an open-source, cloud native, private container registry. vSphere with Tanzu embeds an instance of Harbor that you can use as the private container registry for vSphere Pods and for pods running on Tanzu Kubernetes clusters. For more information, see [Enable the Embedded Harbor Registry on the Supervisor Cluster](#).

The embedded Harbor Registry that ships with vSphere with Tanzu requires NSX-T networking. If you are using vSphere networking, you cannot use it. In addition, you may already be running your own private container registry that you want to integrate with your Tanzu Kubernetes clusters. In this case you can configure the Tanzu Kubernetes Grid Service to trust private registries with self-signed certificates, thereby allowing Kubernetes pods running on Tanzu Kubernetes clusters to use the external registry.

External Private Registry Requirements

To use an external private registry with Tanzu Kubernetes clusters, you must use vSphere with Tanzu version 7 U2 or later.

You can only use your own private registry with Kubernetes pods that run on Tanzu Kubernetes clusters and Tanzu Kubernetes release node VMs. You cannot use your own private registry with vSphere Pods that run natively on ESXi hosts. The supported registry for vSphere Pods is the Harbor Registry that is embedded in the vSphere with Tanzu platform.

Once you configure the Tanzu Kubernetes Grid Service for a private registry, any new cluster that is provisioned will support the private registry. For existing clusters to support the private registry, a rolling update is required to apply the `TkgServiceConfiguration`. See [Update Tanzu Kubernetes Clusters](#). In addition, the first time you create a custom `TkgServiceConfiguration`, the system will initiate a rolling update.

External Private Registry Configuration

To use your own private registry with Tanzu Kubernetes clusters, you configure the Tanzu Kubernetes Grid Service with one or more self-signed certificates to serve private registry content over HTTPS.

The `TkgServiceConfiguration` is updated to support self-signed certificates for the private registry. Specifically, a new `trust` section with the `additionalTrustedCAs` field is added, allowing you to define any number of self-signed certificates that Tanzu Kubernetes clusters should trust. This functionality lets you easily define a list of certificates, and update those certificates should they need rotation.

Once the `trust.additionalTrustedCAs` fields are added, the TLS certificates will be applied for each new cluster that is created. In other words, applying an update to `TkgServiceConfiguration.trust.additionalTrustedCAs` does not trigger an automatic rolling update of existing Tanzu Kubernetes clusters.

```
apiVersion: run.tanzu.vmware.com/v1alpha1
kind: TkgServiceConfiguration
metadata:
  name: tkg-service-configuration
spec:
  defaultCNI: antrea
  trust:
    additionalTrustedCAs:
      - name: first-cert-name
        data: base64-encoded string of a PEM encoded public cert 1
      - name: second-cert-name
        data: base64-encoded string of a PEM encoded public cert 2
```

To apply the update, run the following command.

```
kubectl apply -f tkgserviceconfiguration.yaml
```

Because the Tanzu Kubernetes Grid Service specification is being updated with the private registry certificates, you do not need to add the public key to the Tanzu Kubernetes cluster kubeconfig as you do when using the embedded Harbor Registry with Tanzu Kubernetes clusters.

Configure a Tanzu Kubernetes Workload to Pull Images from a Private Container Registry

To pull images from a private container registry for a Tanzu Kubernetes cluster workload, configure the workload YAML with the private registry details.

This procedure can be used to pull images from a private container registry, or the embedded Harbor Registry. In this example, we create a pod specification that will use an image stored in the embedded Harbor Registry and utilize the image pull secret previously configured.

- 1 Create an example pod spec with the details about the private registry.

```
apiVersion: v1
kind: Pod
metadata:
  name: <workload-name>
  namespace: <kubernetes-namespace>
spec:
  containers:
```

```

- name: private-reg-container
  image: <Registry-IP-Address>/<vsphere-namespace>/<image-name>:<version>
imagePullSecrets:
- name: <registry-secret-name>

```

- Replace `<workload-name>` with the name of the pod workload.
 - Replace `<kubernetes-namespace>` with the Kubernetes namespace in the cluster where the pod will be created. This must be the same Kubernetes namespace where the Registry Service image pull secret is stored in the Tanzu Kubernetes cluster (such as the default namespace).
 - Replace `<Registry-IP-Address>` with the IP address for the embedded Harbor Registry instance running on the Supervisor Cluster.
 - Replace `<vsphere-namespace>` with the vSphere Namespace where the target Tanzu Kubernetes is provisioned.
 - Replace `<image-name>` with an image name of your choice.
 - Replace `<version>` with an appropriate version of the image, such as "latest".
 - Replace `<registry-secret-name>` with the name of the Registry Service image pull secret that you created previously.
- 2 Create a workload in the Tanzu Kubernetes cluster based on the pod specification you defined.

```
kubectl --kubeconfig=<path>/cluster-kubeconfig apply -f <pod.yaml>
```

The pod should be created from the image pulled from the registry.

Trust Fields for External Private Registries

Add a certificate entry (base64-encoded string of a PEM-encoded public certificate) to the `additionalTrustedCAs` section in the `TkgServiceConfiguration`. The data is public certificates stored in plain text in the `TkgServiceConfiguration`.

Table 15-1. Trust Fields for Private Registries

Field	Description
<code>trust</code>	Section marker. Accepts no data.
<code>additionalTrustedCAs</code>	Section marker. Includes a array of certificates with name and data for each.
<code>name</code>	The name of the TLS certificate.
<code>data</code>	The base64-encoded string of a PEM encoded public certificate.

Removing External Private Registry Certificates

Remove a certificate from the list of certificates in the `additionalTrustedCAs` section in the `TkgServiceConfiguration` and apply the `TkgServiceConfiguration` to the Tanzu Kubernetes Grid Service.

Rotating External Private Registry Certificates

Changing the `TkgServiceConfiguration` does not trigger an automatic rolling update of existing Tanzu Kubernetes clusters. But, when a rolling update of a cluster is triggered (by manually changing the cluster specification or as a byproduct of an upgrade), then changes made to the `TkgServiceConfiguration` will be applied.

Thus, to rotate the TLS certificates, a VI Admin or DevOps Engineer can change the contents of the certificate fields in the `additionalTrustedCAs` section of the `TkgServiceConfiguration`. The next time a rolling update is triggered, the changes will be applied. Alternatively, the TLS certificates can be edited directly in the Tanzu Kubernetes cluster specification itself and applied, thereby triggering a rolling update.

Troubleshooting External Private Registry Certificates

If you configure the Tanzu Kubernetes Grid Service with the certificates to trust, and you add the self-signed certificate to the cluster kubeconfig, you should be able to successfully pull a container image from a private registry that uses that self-signed certificate.

The following command can help you determine if the container image has been successfully pulled for a Pod workload:

```
kubectl describe pod PODNAME
```

This command shows detailed status and error messages for a given pod. An example of attempting to pull an image before adding custom certificates to the cluster:

```
Events:
  Type      Reason              Age             From              Message
  ----      -
  Normal    Scheduled           33s            default-scheduler ...
  Normal    Image              32s            image-controller  ...
  Normal    Image              15s            image-controller  ...
  Normal    SuccessfulRealizeNSXResource 7s (x4 over 31s) nsx-container-ncp ...
  Normal    Pulling            7s            kubelet           Waiting test-gc-
e2e-demo-ns/testimage-8862e32f68d66f727d1baf13f7eddef5a5e64bbd-v10612
  Warning   Failed              4s            kubelet           failed to get
images: ... Error: ... x509: certificate signed by unknown authority
```

And, when running the following command:

```
kubectl get pods
```

The `ErrImagePull` error is also visible in the overall Pod status view:

NAME	READY	STATUS	RESTARTS	AGE
testimage-nginx-deployment-89d4fcff8-2d9pz	0/1	Pending	0	17s
testimage-nginx-deployment-89d4fcff8-7kp9d	0/1	ErrImagePull	0	79s
testimage-nginx-deployment-89d4fcff8-7mpkj	0/1	Pending	0	21s
testimage-nginx-deployment-89d4fcff8-fszth	0/1	ErrImagePull	0	50s
testimage-nginx-deployment-89d4fcff8-sjnjw	0/1	ErrImagePull	0	48s
testimage-nginx-deployment-89d4fcff8-xr5kg	0/1	ErrImagePull	0	79s

The errors “x509: certificate signed by unknown authority” and “ErrImagePull” indicate that cluster is not configured with the correct certificate to connect to the private container registry. Either the certificate is missing, or it is misconfigured.

If you are experiencing errors connecting to a private registry after configuring the certificates, you can check if certificates applied in the configuration are applied to the cluster. You can check whether the certificates were applied in their configuration have been applied properly by using SSH.

Two investigative steps can be done by connecting to a worker node over SSH.

- 1 Check the folder `/etc/ssl/certs/` for files named `tkg-<cert_name>.pem`, where `<cert_name>` is the "name" property of the certificate added in the `TkgServiceConfiguration`. If the certificates match what is in the `TkgServiceConfiguration`, and using a private registry still does not work, diagnose further by completing the next step.
- 2 Run the following openssl connection test to the target server using self-signed certificates by executing the command `openssl s_client -connect hostname:port_num`, where `hostname` is the host name/DNS name of the private registry that is using self-signed certificates, and `port_num` is the port number that the service is running on (usually 443 for HTTPS). You can check exactly what error is being returned by openssl when attempting to connect to the endpoint that is using self-signed certificates, and remedy the situation from there, for example, by adding the right certificates to the `TkgServiceConfiguration`. If the Tanzu Kubernetes cluster is embedded with the wrong certificate, you will need to update the Tanzu Kubernetes Grid Service configuration with the correct certificates, delete the Tanzu Kubernetes cluster, then recreate it using the configuration that contains the correct certificates.

Working with vSphere Lifecycle Manager

16

As a vSphere administrator, you can enable vSphere with Tanzu on vSphere clusters that you manage with a single VMware vSphere Lifecycle Manager image. You can then use the Supervisor Cluster while it is managed by vSphere Lifecycle Manager.

vSphere Lifecycle Manager allows you to manage the ESXi hosts and clusters in your environment. You can upgrade a Supervisor Cluster to the latest version of vSphere with Tanzu. You can also upgrade the ESXi version of the hosts in the Supervisor Cluster.

vSphere Lifecycle Manager is a service that runs in vCenter Server. When you deploy vCenter Server, the vSphere Lifecycle Manager user interface is enabled in the HTML5-based vSphere Client.

For more information about the vSphere Lifecycle Manager, see the *Managing Host and Cluster Lifecycle* documentation.

Read the following topics next:

- [Requirements](#)
- [Enable vSphere with Tanzu on a Cluster Managed by vSphere Lifecycle Manager](#)
- [Upgrade a Supervisor Cluster](#)
- [Add Hosts to a Supervisor Cluster](#)
- [Remove Hosts from a Supervisor Cluster](#)
- [Disable a Supervisor Cluster](#)

Requirements

To configure vSphere with Tanzu on a vSphere cluster that is managed by vSphere Lifecycle Manager, your environment must meet specific requirements.

System Requirements

To enable vSphere with Tanzu, verify that the components on the vSphere cluster meet the following requirements:

- Verify that vCenter Server and ESXi are of version 7.0 Update 2 and later if you use NSX.

- Verify that vCenter Server and ESXi are of at least version 7.0 Update 1 and later if you use vSphere Networking.
 - Verify that all ESXi hosts that you want to use as part of a Supervisor Cluster are assigned a VMware vSphere 7 Enterprise Plus with Add-on for Kubernetes license.
 - Verify that HA and DRS are enabled on the vSphere cluster.
 - Verify that vSphere Distributed Switch version 7.0 Update 2 and later is configured.
 - Verify that vSphere Networking or NSX 3.1 or a later version is configured on the cluster. You cannot use vSphere Lifecycle Manager image to manage a cluster that is configured with earlier versions of NSX.
-
- **Note** You cannot transition Supervisor Clusters that use vSphere Update Manager to vSphere vSphere Lifecycle Manager. Transition from vSphere Update Manager to vSphere Lifecycle Manager. is only supported for clusters that do not have vSphere with Tanzu enabled.
-

Enable vSphere with Tanzu on a Cluster Managed by vSphere Lifecycle Manager

To run Kubernetes workloads, you can enable vSphere with Tanzu on a cluster that you manage with a single vSphere Lifecycle Manager image. After it is enabled, you can manage the Supervisor Cluster using vSphere Lifecycle Manager.

When you enable the cluster with vSphere with Tanzu that uses NSX, the vSphere Lifecycle Manager installs the Spherelet vSphere Installation Bundle (VIB) on every ESXi host in the cluster. Enabling the cluster assigns the version of the Kubernetes version that ships with vCenter. After the installation is complete, the WCP service performs the post installation tasks such as, starting and configuring the Spherelet.

For the steps to enable the cluster, see [Enable Workload Management with vSphere Networking](#).

Upgrade a Supervisor Cluster

You can update to the latest version of vSphere with Tanzu, including the vSphere infrastructure supporting vSphere with Tanzu clusters, the Kubernetes versions, and the Kubernetes CLI Tools for vSphere on clusters that use a single vSphere Lifecycle Manager image.

You upgrade the ESXi version of the hosts in the Supervisor Cluster. During the upgrade the Spherelet VIB on every ESXi host is upgraded.

vSphere Lifecycle Manager uses DRS and puts the hosts in maintenance mode before remediation. DRS first attempts to migrate the virtual machine running vCenter Server to another host, such as VMs that are affinitized to the host or running on the local storage of the host, and workloads, including vSphere Pods to other hosts, so that the remediation succeeds.

Note You can use vSphere Lifecycle Manager to upgrade a Supervisor Cluster only on clusters that use a single vSphere Lifecycle Manager image.

Procedure

- 1 From the vSphere Client menu, select **Workload Management**.
- 2 Select the **Updates** tab.
- 3 Select the **Available Version** that you want to update to.
For example, select the version `v1.17.4-vsc0.0.2-16293900`.
- 4 Select the Supervisor Cluster to apply the update to.
- 5 To initiate the update, click **Apply Updates**.
- 6 Use the **Recent Tasks** pane to monitor the status of the update.

Add Hosts to a Supervisor Cluster

As a vSphere administrator, you might need to scale out the Supervisor Cluster to run more workloads. To add capacity to a cluster, you can add ESXi hosts to the cluster that uses a single vSphere Lifecycle Manager image.

When you add a host to the Supervisor Cluster that is configured with NSX, vSphere Lifecycle Manager installs the Spherelet VIB and the image on the host. After it is installed, vSphere with Tanzu configures the Spherelet process on the newly added host, which allows running containers natively on ESXi.

Prerequisites

- Obtain the user name and password of the root user account for the host.
- Verify that hosts behind a firewall might communicate with the vCenter Server.

Procedure

- 1 From the vSphere Client menu, select **Workload Management**.
- 2 Right-click the data center, cluster, or folder and select **Add Host**.
- 3 Enter the IP address or the name of the host and click **Next**.
- 4 Enter the administrator credentials and click **Next**.
- 5 Review the host summary and click **Next**.
- 6 Assign a license to the host and click **Finish**.

- 7 In the **Add Host** wizard, click **Next**.
- 8 Review the summary and click **Finish**.

Note If a host is part of the same data center, you can move it into the Supervisor Cluster. To move the host, put it in maintenance mode and drag it into the cluster.

Remove Hosts from a Supervisor Cluster

As a vSphere administrator, you might need to scale in the Supervisor Cluster to save cost. To reduce the capacity of a Supervisor Cluster, you can remove ESXi hosts from a cluster that uses a single vSphere Lifecycle Manager image.

When you remove a host from the Supervisor Cluster that is configured with NSX, vSphere with Tanzu clears the configuration of the Spherelet and stops the Spherelet process on the ESXi host. vSphere Lifecycle Manager then uninstalls the Spherelet VIB and image from the host and vSphere with Tanzu removes the host metadata from the cluster control plane.

Prerequisites

Before you can remove a host from a cluster, you must power off all virtual machines that are running on the host or migrate the virtual machines to a new host.

Procedure

- 1 In the vSphere Client, navigate to the cluster from which you want to remove the host.
- 2 Right-click the host and select **Enter Maintenance Mode** from the pop-up menu.
- 3 In the confirmation dialog box that appears, click **Yes**.

The confirmation dialog box also asks if you want to move virtual machines that are powered off to other hosts. Select this option if you want those virtual machines to remain registered to a host within the cluster.

The host icon changes and the term “maintenance mode” is added to the name in parentheses.

- 4 Select the host icon from the inventory, and drag it to the new location.

The host can be moved to another cluster or another data center.

vCenter Server moves the host to the new location.

- 5 Right-click the host, and select **Exit Maintenance Mode** from the pop-up menu.
- 6 (Optional) Restart any virtual machines, if needed.

Disable a Supervisor Cluster

You can disable vSphere with Tanzu from a vSphere cluster that uses a single vSphere Lifecycle Manager image, to make it available for traditional workloads.

When you disable a vSphere with Tanzu on a cluster, vSphere Lifecycle Manager uninstalls the Spherelet VIB and image from each ESXi host and the `WCP` service stops and deletes all workloads from the cluster.

Procedure

- 1 From the vSphere Client menu, select **Workload Management**.
- 2 Select the **Clusters** tab.
- 3 Select the cluster on which you want to disable vSphere with Tanzu.
- 4 Click **Disable**.

The **Disable Cluster** dialog box appears with a message that all Kubernetes workloads and NSX configuration will be disabled on the cluster.

- 5 Click **Disable**.

Updating the vSphere with Tanzu Environment

17

You can update to the latest version of vSphere with Tanzu, including the vSphere infrastructure supporting vSphere with Tanzu clusters, the Kubernetes versions, and the Kubernetes CLI Tools for vSphere.

Read the following topics next:

- [About vSphere with Tanzu Updates](#)
- [Network Topology Upgrade](#)
- [Update the Supervisor Cluster by Performing a vSphere Namespaces Update](#)
- [Supervisor Cluster Auto Upgrade](#)
- [Update the vSphere Plugin for kubectl](#)
- [Verify Tanzu Kubernetes Cluster Compatibility for Update](#)
- [Update Tanzu Kubernetes Clusters](#)

About vSphere with Tanzu Updates

vSphere with Tanzu supports rolling updates for Supervisor Clusters and Tanzu Kubernetes Grid Service clusters, and for the infrastructure supporting these clusters.

Note You cannot transition Supervisor Clusters that use vSphere Update Manager to vSphere Lifecycle Manager. Transition from vSphere Update Manager to vSphere Lifecycle Manager. is only supported for clusters that do not have vSphere with Tanzu enabled.

How Supervisor Clusters and Tanzu Kubernetes Grid Service Clusters Are Updated

vSphere with Tanzu uses a rolling update model for Supervisor Clusters and Tanzu Kubernetes Grid Service clusters. The rolling update model ensures that there is minimal downtime for cluster workloads during the update process. Rolling updates include upgrading the Kubernetes software versions and the infrastructure and services supporting the Kubernetes clusters, such as virtual machine configurations and resources, vSphere services and namespaces, and custom resources.

For the update to succeed, your configuration must meet several compatibility requirements, so the system enforces recheck conditions to ensure that clusters are ready for updates, and supports rollback if cluster upgrade is not successful.

Note A vSphere with Tanzu update involves more than just an upgrade of the Kubernetes software version. We use the term "update" to describe this process instead of the term "upgrade," which is a limited form of update that increments the software version.

Dependency Between Supervisor Cluster Updates and Tanzu Kubernetes Grid Service Cluster Updates

You update the Supervisor Cluster and the Tanzu Kubernetes Grid Service clusters separately. Note, however, that there are dependencies between the two.

Updating a Supervisor Cluster will likely trigger a rolling update of the Tanzu Kubernetes Grid Service clusters deployed there. See [Update the Supervisor Cluster by Performing a vSphere Namespaces Update](#).

You may need to update one or more Tanzu Kubernetes Grid Service clusters before updating a Supervisor Cluster if the Tanzu Kubernetes Grid Service cluster is not compliant with the target Supervisor Cluster version. See [Verify Tanzu Kubernetes Cluster Compatibility for Update](#).

About Supervisor Cluster Updates

When you initiate an update for a Supervisor Cluster, the system creates a new control plane node and joins it to the existing control plane. The vSphere inventory shows four control plane nodes during this phase of the update as the system adds a new updated node and then removes the older out-of-date node. Objects are migrated from one of the old control plane nodes to the new one, and the old control plane node is removed. This process repeats one-by-one until all control plane nodes are updated. Once the control plane is updated, the worker nodes are updated in a similar rolling update fashion. The worker nodes are the ESXi hosts, and each spherelet process on each ESXi host is updated one-by-one.

You can choose between the following updates:

- Update the vSphere Namespaces.
- Update everything, including VMware versions and Kubernetes versions.

You use the vSphere Namespaces update workflow to update the Kubernetes version that the Supervisor Cluster is running, such as from Kubernetes 1.16.7 to Kubernetes 1.17.4, and the infrastructure supporting the Supervisor Cluster and Tanzu Kubernetes Grid Service clusters. This type of update is more frequent and is used to maintain pace with the Kubernetes release cadence. The following is the vSphere Namespaces update sequence.

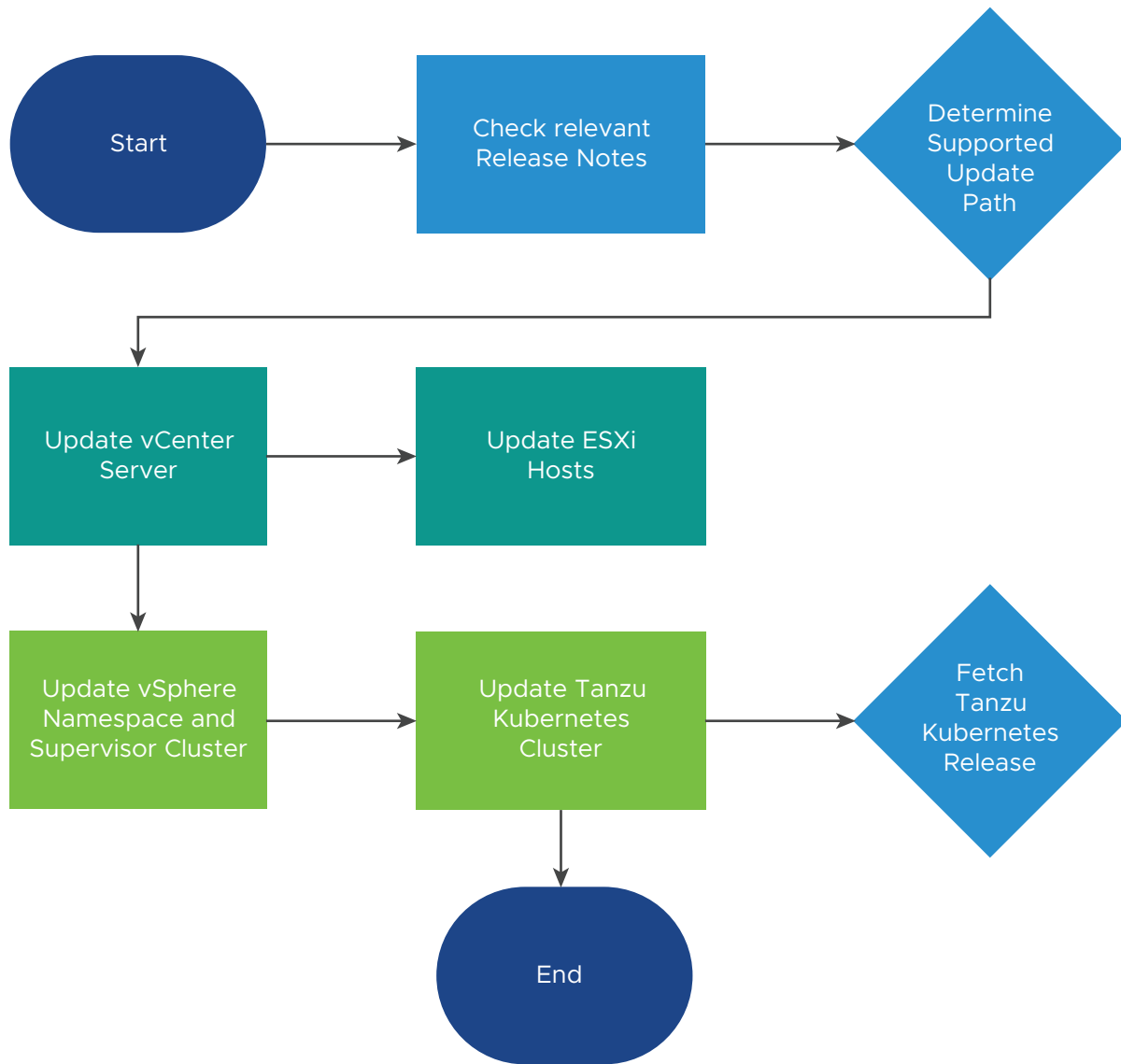
- 1 Upgrade vCenter Server.
- 2 Perform a vSphere Namespaces update (including Kubernetes upgrade).

To perform a vSphere Namespaces update, see [Update the Supervisor Cluster by Performing a vSphere Namespaces Update](#).

You use the update everything workflow to update all vSphere with Tanzu components. This type of update is required when you are updating major releases, for example such as from NSX-T 3.X to 4 and from vSphere 7.X to 8. This update workflow is infrequent depending on when there are new VMware product releases. This is the update everything sequence:

- 1 Check the VMware Interoperability matrix <https://interopmatrix.vmware.com/Interoperability> for the vCenter Server and NSX to determine compatibility. vSphere with Tanzu functionality is delivered by Workload Control Plane (WCP) software which ships with vCenter Server.
- 2 Upgrade NSX, if compatible.
- 3 Upgrade vCenter Server.
- 4 Upgrade vSphere Distributed Switch.
- 5 Upgrade ESXi hosts.
- 6 Check compatibility of any provisioned Tanzu Kubernetes Grid Service clusters with the target Supervisor Cluster version.
- 7 Update vSphere Namespaces (including the Supervisor Cluster Kubernetes version).
- 8 Update Tanzu Kubernetes Grid Service clusters.

The diagram illustrates the general workflow for vSphere with Tanzu updates.



About Tanzu Kubernetes Grid Service Cluster Updates

When you update a Supervisor Cluster, the infrastructure components supporting the Tanzu Kubernetes Grid Service clusters deployed to that Supervisor Cluster, such as the Tanzu Kubernetes Grid Service, are likewise updated. Each infrastructure update can include updates for services supporting the Tanzu Kubernetes Grid Service (CNI, CSI, CPI), and updated configuration settings for the control plane and worker nodes that can be applied to existing Tanzu Kubernetes Grid Service clusters. To ensure that your configuration meets compatibility requirements, vSphere with Tanzu performs pre-checks during rolling update and enforces compliance.

To perform a rolling update of a Tanzu Kubernetes Grid Service cluster, you update the cluster manifest. See [Update Tanzu Kubernetes Clusters](#). Note, however, that when a vSphere Namespaces update is performed, the system immediately propagates updated configurations to all Tanzu Kubernetes Grid Service clusters. These updates can automatically trigger a rolling update of the Tanzu Kubernetes Grid Service control plane and worker nodes.

The rolling update process for replacing the cluster nodes is similar to the [rolling update of pods](#) in a Kubernetes Deployment. There are two distinct controllers responsible for performing a rolling update of Tanzu Kubernetes Grid Service clusters: the Add-ons Controller and the TanzuKubernetesCluster controller. Within those two controllers there are three key stages to a rolling update: updating add-ons, updating the control plane, and updating the worker nodes. These stages occur in order, with pre-checks that prevent a step from beginning until the preceding step has sufficiently progressed. These steps might be skipped if they are determined to be unnecessary. For example, an update might only affect worker nodes and therefore not require any add-on or control plane updates.

During the update process, the system adds a new cluster node, and waits for the node to come online with the target Kubernetes version. The system then marks the old node for deletion, moves to the next node, and repeats the process. The old node is not deleted until all pods are removed. For example, if a pod is defined with PodDisruptionBudgets that prevent a node from being fully drained, the node is cordoned off but is not removed until those pods can be evicted. The system upgrades all control plane nodes first, then worker nodes. During an update, the Tanzu Kubernetes Grid Service cluster status changes to "updating". After the rolling update process completes, the Tanzu Kubernetes Grid Service cluster status changes to "running".

Note Worker nodepools are updated at the same time when you update a TKG with multiple nodepools. For more information, see [Understanding the Rolling Update Model for TKG Clusters on Supervisor](#) in the vSphere 8.x documentation for TKGS.

Pods running on a Tanzu Kubernetes Grid Service cluster that are not governed by a replication controller will be deleted during a Kubernetes version upgrade as part of the worker node drain during the Tanzu Kubernetes Grid Service cluster update. This is true if the cluster update is triggered manually or automatically by a vSphere Namespaces update. Pods not governed by a replication controller include pods that are not created as part of a Deployment or ReplicaSet spec. Refer to the topic [Pod Lifecycle: Pod lifetime](#) in the Kubernetes documentation for more information.

Network Topology Upgrade

When you install vSphere with Tanzu version 7.0 Update 1c or upgrade the Supervisor Cluster from version 7.0 Update 1 to version 7.0 Update 1c, you upgrade the NSX Container Plug-in (NCP). This in turn migrates the networking topology of the Supervisor Cluster, namespaces, and Tanzu Kubernetes clusters. After the upgrade, the networking topology is upgraded from a single tier-1 gateway topology to a topology that has a tier-1 gateway for each namespace within the Supervisor Cluster.

During the upgrade, the NCP configures the NSX resources to support the new topology. The NCP provides a shared network infrastructure for namespaces that have less of layer-4 and layer-7 load balancing services. This reduces the resources on NSX and makes more available for Tanzu Kubernetes clusters.

System namespaces are namespaces that are used by the core components that are integral to functioning of the Supervisor Cluster and Tanzu Kubernetes clusters. The shared network resources that include the tier-1 gateway, load balancer, and SNAT IP are grouped in a system namespace.

The NCP creates one shared tier-1 gateway for system namespaces and a tier-1 gateway and load balancer for each namespace, by default. The tier-1 gateway is connected to the tier-0 gateway and a default segment.

The NSX-T load balancer provides load balancing services in the form of virtual servers.

After migration, the networking topology has the following characteristics:

- Each vSphere Namespace has a separate network and set of networking resources shared by applications inside the namespace such as, tier-1 gateway, load balancer service, and SNAT IP address.
- Workloads running in vSphere Pods, regular VMs, or Tanzu Kubernetes clusters, that are in the same namespace, share a same SNAT IP for North-South connectivity.
- Workloads running in vSphere Pods or Tanzu Kubernetes clusters will have the same isolation rule that is implemented by the default firewall.
- A separate SNAT IP is not required for each Kubernetes namespace. East west connectivity between namespaces will be no SNAT.

The maximum number of namespaces that can be run depends on Edge Node size (Medium, Large, or X-Large) and the number of Edge nodes in the NSX Edge Cluster. The number of namespaces that can be run is less than 20 times the number of Edge nodes. For example, if the NSX Edge cluster has 10 Edge nodes of Large size, the maximum number of Supervisor Namespaces that can be created is 199.

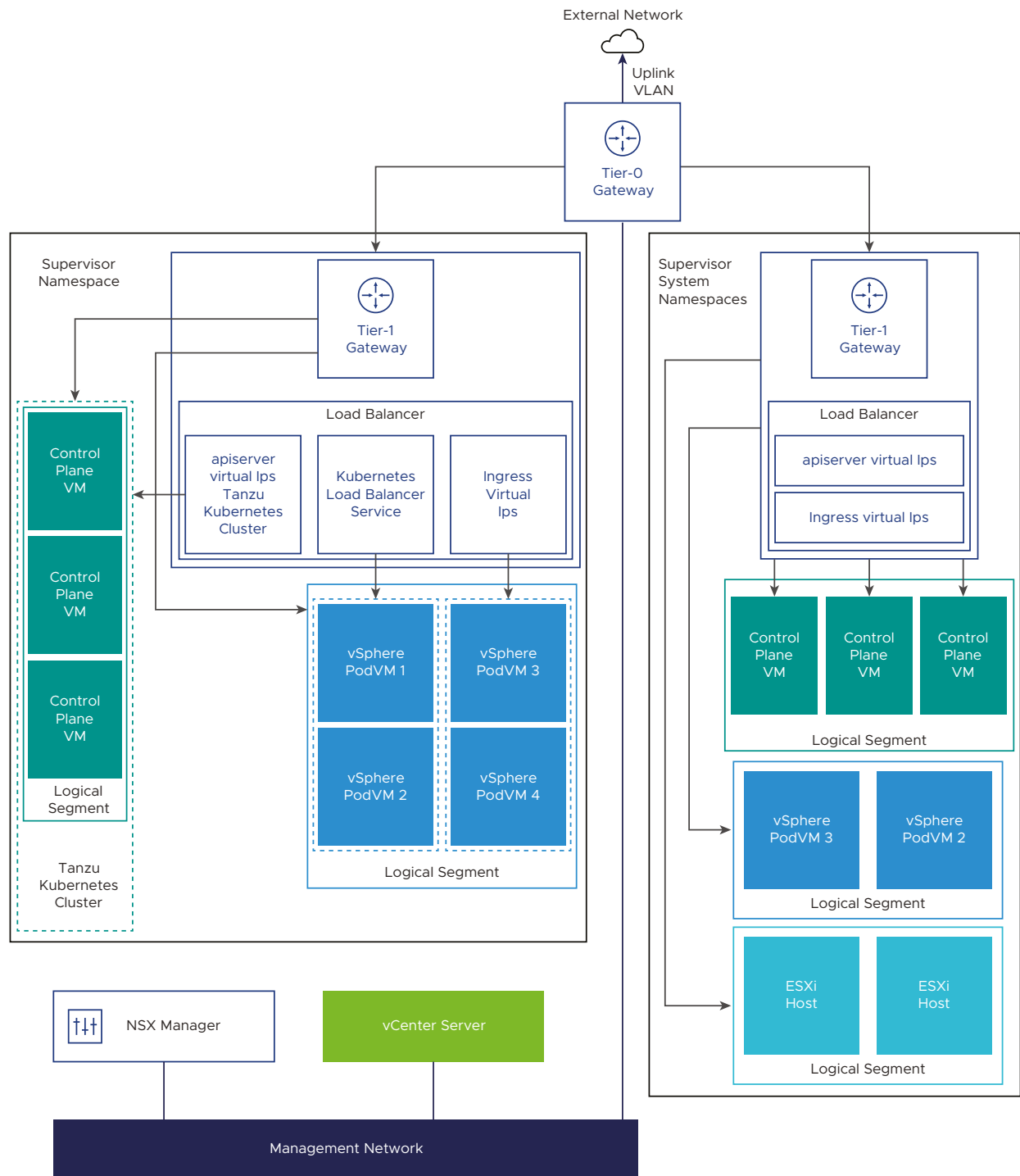
For more information on the Edge Node size, see the *NSX-T Data Center Installation Guide*.

Supervisor Cluster Networking

Supervisor clusters have separate segments within the shared tier-1 gateway. For each Tanzu Kubernetes cluster, segments are defined within the tier-1 gateway of the namespace.

Workloads, including vSphere Pods and Tanzu Kubernetes clusters, that are within the same namespace, will share a SNAT IP for north-south connectivity. East-west connectivity between namespaces will be no SNAT.

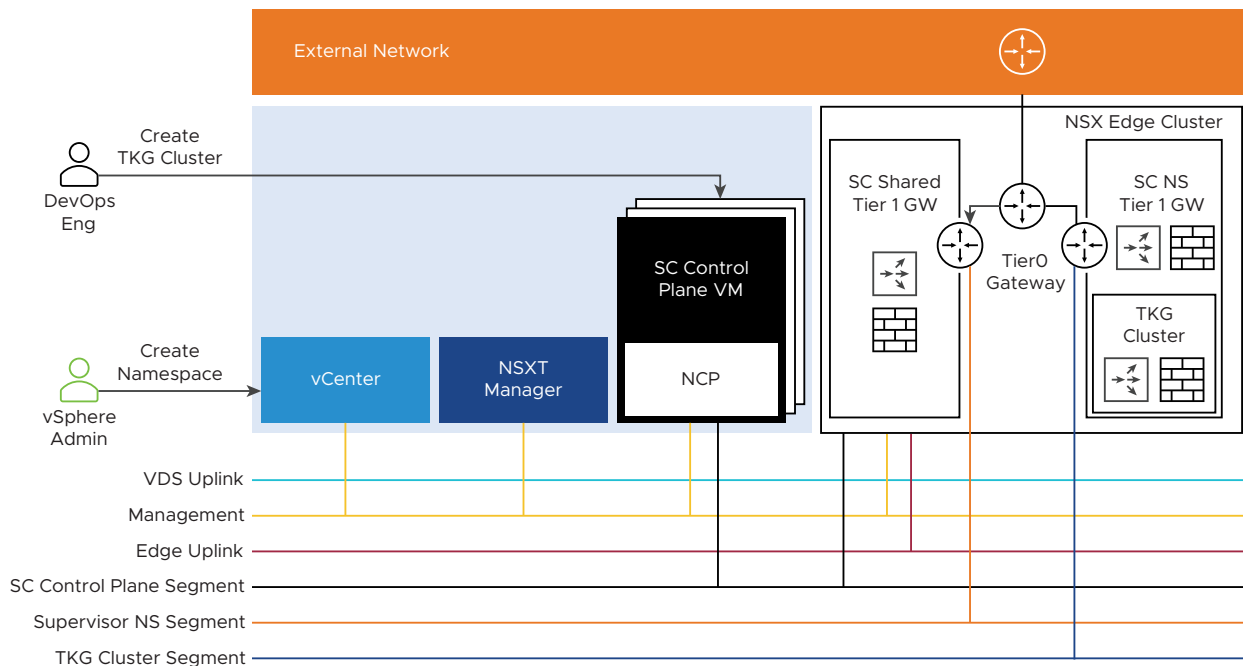
Figure 17-1. Supervisor Cluster Networking



Tanzu Kubernetes Clusters Networking

After the Supervisor Cluster upgrade, when DevOps engineers provision the first Tanzu Kubernetes cluster in a Supervisor Namespace, the cluster will share the same tier-1 gateway and load balancer as the namespace. For each Tanzu Kubernetes cluster that is provisioned in that namespace, a segment is created for that cluster and it is connected to the shared tier-1 gateway in its Supervisor Namespace.

When a Tanzu Kubernetes cluster is provisioned by the Tanzu Kubernetes Grid Service, a single virtual server is created that provides layer-4 load balancing for the Kubernetes API. This virtual server is hosted on the shared load balancer with the namespace and is responsible for routing kubectl traffic to the control plane. In addition, for each Kubernetes service load balancer that is resourced on the cluster, a virtual server is created that provides layer-4 load balancing for that service.



Upgrade the NSX-T Network Topology

To upgrade the network topology, you upgrade NSX, vCenter Server, and all vSphere with Tanzu components.

You can upgrade NSX-T Data Center version. When you perform the upgrade, the NSX-T Data Center components, which include the data plane, control plane, and management plane are upgraded with minimum system downtime.

For step-by-step information about upgrading the NSX components, see the [NSX Upgrade Guide](#).

Note Refer to the vSphere with Tanzu [Release Notes](#) for supported versions.

Prerequisites

- Verify that your environment meets the system requirements. For information about requirements, see [System Requirements for Setting Up vSphere with Tanzu with NSX](#).
- For configuration limits specific to vSphere with Tanzu, see [vSphere Configuration Limits](#) in VMware Configuration Maximums tool.

Procedure

- 1 Upgrade NSX from NSX 3.0.x to NSX 3.1.
- 2 Upgrade vCenter Server.
- 3 Upgrade ESXi hosts.
- 4 Update the Supervisor Namespace.

To perform an update, see [Update the Supervisor Cluster by Performing a vSphere Namespaces Update](#).

Upgrade vSphere Distributed Switch

You can upgrade vSphere Distributed Switch version 7.0 to a later version. The upgrade of a distributed switch causes the hosts and virtual machines attached to the switch to experience a brief downtime.

Prerequisites

- Upgrade vCenter Server to version 7.0.
- Upgrade all hosts connected to the distributed switch to ESXi 7.0.

Procedure

- 1 From the vSphere Client home menu, select **Hosts and Clusters**.
- 2 Select **Networking** and navigate to the distributed switch.
For example, **DSwitch**.
The vSphere Client UI lists the versions that are available to upgrade to.
- 3 From the **Actions** menu, select **Upgrade > Upgrade Distributed Switch**.
- 4 Select the vSphere Distributed Switch version that you want to upgrade the switch to and click **Next**.
- 5 Review host compatibility and click **Next**.
- 6 Complete the upgrade configuration and click **Finish**.

Update the Supervisor Cluster by Performing a vSphere Namespaces Update

To update one or more Supervisor Clusters, including the Kubernetes version that the Supervisor Cluster is running and the infrastructure supporting Tanzu Kubernetes clusters, including the Tanzu Kubernetes Grid Service, you perform a vSphere Namespaces update.

There is a version entity for vSphere with Tanzu. The version entity is a semantic version string in the form `v1.19.1+vmware.2-vsc0.0.8-17610687`, where the prefix is the Kubernetes version (`v1.19.1`) and the suffix is the vSphere Namespaces version (`vsc0.0.8-17610687`).

The tables lists the available Supervisor Cluster releases:

Version	Description
<code>v1.19.1+vmware.2-vsc0.0.8-17610687</code>	Most recent Supervisor Cluster release; Supports vSphere 7.0 Update 2.
<code>v1.18.2-vsc0.0.5-16762486</code>	Supports vSphere 7.0 Update 1.
<code>v1.17.4-vsc0.0.5-16762486</code>	Minimum Supervisor Cluster release that includes a Tanzu Kubernetes Grid Servicethat supports the Antrea CNI.
<code>v1.16.7-vsc0.0.5-16762486</code>	If you are running this release, you should update it to 1.17 at a minimum.

Prerequisites

Read the release notes before you perform a vSphere Namespaces update.

To install the necessary binaries, upgrade the vCenter Server appliance and ESXi hosts to the supported version. See [Upgrading the vCenter Server Appliance](#) in the vCenter Server documentation.

Note When you perform a vSphere Namespaces update, all provisioned Tanzu Kubernetes clusters must be in a running state. If a Tanzu Kubernetes is in a creating or deleting state, wait until the process finishes before updating a Supervisor Cluster, otherwise might not succeed.

Note Updating a Supervisor Cluster will likely trigger a rolling update of the Tanzu Kubernetes clusters deployed there. See [Update Tanzu Kubernetes Clusters](#).

Procedure

- 1 Log in to the vCenter Server as a vSphere administrator.
- 2 Select **Menu > Workload Management**.
- 3 Select the **Namespaces > Updates** tab.

- 4 Select the **Available Version** that you want to update to.

For example, select the version `v1.18.2-vsc0.0.5-16762486`.

Note You must update incrementally. Do not skip updates, such as from 1.16 to 1.18. The path should be 1.16, 1.17, 1.18.

- 5 Select one or more Supervisor Clusters to apply the update to.
- 6 To initiate the update, click **Apply Updates**.
- 7 Use the **Recent Tasks** pane to monitor the status of the update.

Supervisor Cluster Auto Upgrade

When you upgrade the vCenter Server appliance, a Supervisor Cluster upgrade is automatically triggered.

vSphere with Tanzu components include components in the vCenter Server, Kubernetes components, and ESXi components. When you upgrade vCenter Server, only the vSphere with Tanzu components on the vCenter Server are upgraded. You must manually upgrade the Kubernetes components, and ESXi components.

With the auto upgrade feature, when you upgrade vCenter Server, an upgrade of the Supervisor Cluster is triggered.

Note You cannot configure the auto upgrade. The upgrade is triggered automatically depending on the compatibility between the Kubernetes versions of the Supervisor Cluster and vCenter Server.

When you attempt to upgrade the vCenter Server, prechecks are run to check the compatibility between the Kubernetes versions of the Supervisor Cluster and vCenter Server. Warnings are displayed in the following scenarios:

- The Kubernetes components of the target vCenter Server are one version higher than the Kubernetes version of the Supervisor Cluster. In this scenario, when you proceed with the vCenter Server upgrade, the Kubernetes version of the Supervisor Cluster is automatically upgraded to match the Kubernetes version of vCenter Server.

For example, the Kubernetes version of the active Supervisor Cluster version is 1.22 and the supported Kubernetes versions on the target vCenter Server are 1.23, 1.24, and 1.25. In this case, upon vCenter Server upgrade, the Supervisor Cluster is also upgraded to 1.23.
- The Kubernetes components of the target vCenter Server are higher than the Kubernetes version of the Supervisor Cluster by two versions or more. In this scenario, you cannot upgrade the vCenter Server. To be able to upgrade vCenter Server, first you must manually upgrade the Kubernetes version of the Supervisor Cluster to at least one version lower of the Kubernetes version of vCenter Server.

For example, the Kubernetes version of the active Supervisor Cluster is lower than or equal to 1.22 and supported Kubernetes versions on the target vCenter Server are 1.24, 1.25, and 1.26. To upgrade vCenter Server, you must first upgrade the Supervisor Cluster to at least 1.23, or ideally 1.24.

- The Supervisor Cluster license has expired. If the Supervisor Cluster has a license other than an evaluation license, irrespective of the license status (either expired or active), you can upgrade the Supervisor Cluster and all operations are intact. If the Supervisor Cluster has an evaluation license and it has expired, you cannot upgrade or update the Supervisor Cluster.

For a list of vCenter Server releases, see the following KB article: <https://kb.vmware.com/s/article/2143838>.

Update the vSphere Plugin for kubectl

Once you have performed a vSphere Namespace update and upgraded the Supervisor Cluster, update the vSphere Plugin for kubectl.

The most recent version of the vSphere Plugin for kubectl downloads and installs the Tanzu Kubernetes cluster root CA certificate in the Kubernetes secret named `TANZU-KUBERNETES-CLUSTER-NAME-ca`. The plugin uses this certificate to populate the CA information in the corresponding cluster's certificate authority datastore.

To download and install the vSphere Plugin for kubectl, see [Download and Install the Kubernetes CLI Tools for vSphere](#). For more information about the `TANZU-KUBERNETES-CLUSTER-NAME-ca` secret, see [Get Tanzu Kubernetes Cluster Secrets](#).

Verify Tanzu Kubernetes Cluster Compatibility for Update

A Tanzu Kubernetes release provides the Kubernetes software distribution signed and supported by VMware for Tanzu Kubernetes clusters provisioned by the Tanzu Kubernetes Grid Service.

Tanzu Kubernetes releases Release Notes

For a list of all Tanzu Kubernetes releases, refer to the [release notes](#).

Verify Tanzu Kubernetes Cluster Compatibility for Update

Before upgrading the vSphere with Tanzu infrastructure, including the vCenter Server and the Supervisor Cluster, refer to the Knowledge Base article <https://kb.vmware.com/s/article/82592> for guidance on how to check if any Tanzu Kubernetes clusters are incompatible with the upgrade. If a Tanzu Kubernetes cluster is incompatible with the target infrastructure, upgrade the cluster before proceeding with the system upgrade.

Use Kubectl to List Available Tanzu Kubernetes releases

You can list Tanzu Kubernetes releases and view the compatibility and updatability for each release using following command.

```
kubectl get tanzukubernetesreleases
```

The `COMPATIBLE` column indicates if that Tanzu Kubernetes release is compatible with the current Supervisor Cluster. The `UPDATES AVAILABLE` column indicates if there is a Kubernetes upgrade available and the recommended next Tanzu Kubernetes releases to be used.

NAME	VERSION	READY	COMPATIBLE
CREATED UPDATES AVAILABLE			
v1.18.15---vmware.1-tkg.1.600e412 21h [1.19.7+vmware.1-tkg.1.fc82c41]	1.18.15+vmware.1-tkg.1.600e412	True	True
v1.19.7---vmware.1-tkg.1.fc82c41 21h [1.20.2+vmware.1-tkg.1.1d4f79a]	1.19.7+vmware.1-tkg.1.fc82c41	True	True
v1.20.2---vmware.1-tkg.1.1d4f79a	1.20.2+vmware.1-tkg.1.1d4f79a	True	True 21h

This same information is also available using `kubectl get tkc <tkgs-cluster-name>`.

Update Tanzu Kubernetes Clusters

You can initiate a rolling update a Tanzu Kubernetes cluster, including the Kubernetes version, by updating the Tanzu Kubernetes release, the virtual machine class, or the storage class.

Tanzu Kubernetes Cluster Update Preparation Checklist

Complete the following list of prerequisite tasks before performing a Tanzu Kubernetes cluster update.

Step	Action
1	Read the vSphere with Tanzu release notes .
2	Read the Tanzu Kubernetes releases release notes .
3	Review the vSphere with Tanzu About vSphere with Tanzu Updates .
4	Verify Tanzu Kubernetes Verify Tanzu Kubernetes Cluster Compatibility for Update with the target upgrade version(s).
5	Review the features of the Tanzu Kubernetes Grid Service API target version, such as Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API , and the current version, such as Provisioning Tanzu Kubernetes Clusters Using the Tanzu Kubernetes Grid Service v1alpha1 API . (See Important note below.)
6	Verify that all provisioned Tanzu Kubernetes clusters are in a View Tanzu Kubernetes Cluster Lifecycle Status state.
7	Perform a vSphere Namespaces Update the Supervisor Cluster by Performing a vSphere Namespaces Update which upgrades the Supervisor Cluster and Tanzu Kubernetes Grid Service.

Step	Action
8	Review the options for Initiating a Rolling Update of a Tanzu Kubernetes Cluster of a Tanzu Kubernetes cluster.
9	Review the Methods for Editing the Cluster Manifest for updating the cluster manifest.

Important vSphere with Tanzu version 7 Update 3, specifically the Supervisor Cluster version `v1.21.0+vmware.wcp.2`, includes an automatic upgrade to the Tanzu Kubernetes Grid Service `v1alpha2` API. Some fields in the Tanzu Kubernetes cluster specification are deprecated and the cluster manifest may need to be manually edited before you upgrade the Kubernetes version. See [Updating a Tanzu Kubernetes Release After the Cluster Spec Is Converted to the TKGS v1alpha2 API](#).

Initiating a Rolling Update of a Tanzu Kubernetes Cluster

You initiate a rolling update by making one or more of the following modifications to the `TanzuKubernetesCluster` specification:

- [Update a Tanzu Kubernetes Cluster by Upgrading the Tanzu Kubernetes Release Version](#)
- [Update a Tanzu Kubernetes Cluster by Changing the VirtualMachineClass](#)
- [Update a Tanzu Kubernetes Cluster by Changing the Storage Class](#)

Note While these are the most common ways to initiate a rolling update, they are not the only ones. A change to any of the configuration elements can also initiate a rolling update. For example, renaming or replacing the `VirtualMachineImage` that corresponds to a distribution version initiates a rolling update as the system attempts to get all nodes running on the new image. Also, updating a Supervisor Cluster will likely trigger a rolling update of the Tanzu Kubernetes clusters deployed there. For example, when the `vmware-system-tkg-controller-manager` is updated, the system introduces new values into the manifest generator and the controller initiates a rolling update to deploy those values.

Methods for Editing the Cluster Manifest

Updating a cluster requires updating the cluster manifest. There are various ways to do this, including:

- By using the `kubectl edit tanzukubernetescluster/CLUSTER-NAME` command. This command opens the entire cluster manifest in the text editor defined by your `KUBE_EDITOR` or `EDITOR` environment variable. When you save the file, the cluster is updated with the changes. For more information on the `kubectl edit` command, see [edit command](#) in the Kubernetes documentation. To use the `kubectl edit` approach, refer to the following topics:
 - [Update a Tanzu Kubernetes Cluster by Upgrading the Tanzu Kubernetes Release Version](#)
 - [Update a Tanzu Kubernetes Cluster by Changing the VirtualMachineClass](#)
 - [Update a Tanzu Kubernetes Cluster by Changing the Storage Class](#)

- By using the `kubectl patch` command. This command performs an "in-place" update of a cluster. The purpose of the this command is to provide a method for upgrading Kubernetes versions, and is the approach documented here. For more information on the `kubectl patch` command, see [Update API Objects in Place Using kubectl patch](#) in the Kubernetes documentation. To use the `kubectl patch` approach, refer to the following topic:
 - [Update Tanzu Kubernetes Clusters Using the Patch Method](#)
- By using the `kubectl apply` command with a local YAML file that you manually update. While this approach has the advantage of being similar to the way you [Workflow for Provisioning Tanzu Kubernetes Clusters Using the TKGS v1alpha2 API](#), if you do not have access to the current cluster YAML this approach can be destructive and is therefore not recommended.

Update a Tanzu Kubernetes Cluster by Upgrading the Tanzu Kubernetes Release Version

Update a Tanzu Kubernetes cluster by upgrading the Tanzu Kubernetes release version.

You can initiate a rolling update of a Tanzu Kubernetes cluster by upgrading the Tanzu Kubernetes release version. How you do this differs depending on which version of the Tanzu Kubernetes Grid Service API you are using.

TKGS API Version	Version Update Method
v1alpha2 API	Update the TKR NAME string in the <code>spec.topology.controlPlane.tkr.reference.name</code> and the <code>spec.topology.nodePools[*].tkr.reference.name</code> properties of the cluster manifest. See Updating a Tanzu Kubernetes Release After the Cluster Spec Is Converted to the TKGS v1alpha2 API .
v1alpha1 API	Update the DISTRIBUTION version in the <code>spec.distribution.version</code> and <code>spec.distribution.fullVersion</code> properties of the cluster manifest. See below.

Prerequisites

Verify completion of the prequisites for updating Tanzu Kubernetes clusters. See [Update Tanzu Kubernetes Clusters](#).

This task uses the command `kubectl edit tanzukubernetescluster/CLUSTER-NAME` to update the cluster manifest. The `kubectl edit` command opens the cluster manifest in the text editor defined by your `KUBE_EDITOR` or `EDITOR` environment variable. When you save the file, the cluster is updated with the changes. See [Specify a Default Text Editor for Kubectl](#).

Procedure

- 1 Authenticate with the Supervisor Cluster. See [Connect to the Supervisor Cluster as a vCenter Single Sign-On User](#).

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME
```

- 2 Switch context to the vSphere Namespace where the target Tanzu Kubernetes cluster is provisioned.

```
kubectl config use-context SUPERVISOR-NAMESPACE
```

- 3 Get the target Tanzu Kubernetes cluster and version.

```
kubectl get tanzukubernetescluster
```

For example, the result using the TKGS v1alpha2 API:

```
kubectl get tanzukubernetescluster
```

NAMESPACE	NAME	CONTROL PLANE	WORKER	TKR
tkgs-cluster-1	test-cluster	3	3	v1.21.2---vmware.1-tkg.1.ee25d55
tkg.1.ee25d55	38h	True	True	[1.21.2+vmware.1-tkg.1.ee25d55]

For example, the result using the TKGS v1alpha1 API:

```
kubectl get tanzukubernetescluster
```

NAME	CONTROL PLANE	WORKER	DISTRIBUTION	AGE	PHASE
tkgs-cluster-1	3	3	v1.19.16+vmware.1-tkg.1.df910e2	19h	running

- 4 List the available Tanzu Kubernetes releases.

```
kubectl get tanzukubernetesreleases
```

- 5 Run the following command to edit the cluster manifest.

```
kubectl edit tanzukubernetescluster/CLUSTER-NAME
```

- 6 Edit the manifest by updating Tanzu Kubernetes release. How you do this differs depending on which version of the TKGS API you are using.

If you are using the TKGS v1alpha2 API, update the TKR NAME string. See [Updating a Tanzu Kubernetes Release After the Cluster Spec Is Converted to the TKGS v1alpha2 API](#).

If you are using the TKGS v1alpha1 API, update the manifest from, for example:

```
spec:
  distribution:
    fullVersion: v1.19.16+vmware.1-tkg.1.df910e2
    version: v1.19.16
```

To, for example:

```
spec:
  distribution:
    fullVersion: null
    version: v1.20.12
```

Note If you are using the TKGS v1alpha1 API, you can specify the fully qualified version, or use a version shortcut, such as `version: v1.20.12`, which is resolved to the most recent image matching that patch version, or `version: v1.20`, which is resolved to the most recent matching patch version. The resolved version displays as the `fullVersion` in the cluster manifest. To perform a version upgrade using a shortcut, you must unset or null out the `fullVersion` to avoid a potential version mismatch during discovery.

- 7 Save the changes you made to the manifest file.

When you save the file, `kubectl` applies the changes to the cluster. In the background, the Virtual Machine Service on the Supervisor Cluster provisions the new worker node.

- 8 Verify that `kubectl` reports that the manifest edits were successfully recorded.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 edited
```

Note If you receive an error, or `kubectl` does not report that the cluster manifest was successfully edited, make sure you have properly configured your default text editor using the `KUBE_EDITOR` environment variable. See [Specify a Default Text Editor for Kubectl](#).

- 9 Verify that the cluster is updating.

```
kubectl get tanzukubernetescluster
```

NAME	CONTROL PLANE	WORKER	DISTRIBUTION	AGE	PHASE
tkgs-cluster-1	3	3	v1.20.12+vmware.1-tkg.1.b9a42f3	21h	updating

- 10 Verify that the cluster is updated.

```
kubectl get tanzukubernetescluster
```

NAME	CONTROL PLANE	WORKER	DISTRIBUTION	AGE	PHASE
tkgs-cluster-1	3	3	v1.20.12+vmware.1-tkg.1.b9a42f3	22h	running

Update a Tanzu Kubernetes Cluster by Changing the VirtualMachineClass

You can update a Tanzu Kubernetes cluster by changing the virtual machine class used to host the cluster nodes.

The Tanzu Kubernetes Grid Service supports updating a cluster by changing the `VirtualMachineClass` definition. If you do this, the service rolls out new nodes with that new class and spins down the old nodes. See [About Tanzu Kubernetes Grid Service Cluster Updates](#).

Note The `VirtualMachineClass` must be bound to the vSphere Namespace where the Tanzu Kubernetes cluster is provisioned. See [Virtual Machine Classes for Tanzu Kubernetes Clusters](#).

Prerequisites

This task uses the command `kubectl edit tanzukubernetescluster/CLUSTER-NAME` to update the cluster manifest. The `kubectl edit` command opens the cluster manifest in the text editor defined by your `KUBE_EDITOR` or `EDITOR` environment variable. When you save the file, the cluster is updated with the changes. See [Specify a Default Text Editor for Kubectl](#).

Procedure

- 1 Authenticate with the Supervisor Cluster. See [Connect to the Supervisor Cluster as a vCenter Single Sign-On User](#).

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME
```

- 2 Switch context to the vSphere Namespace where the target Tanzu Kubernetes cluster is provisioned.

```
kubectl config use-context SUPERVISOR-NAMESPACE
```

- 3 Describe the target Tanzu Kubernetes cluster and check the VM class.

```
kubectl describe tanzukubernetescluster CLUSTER-NAME
```

For example, this cluster is using the best-effort-medium VM class:

```
Spec:
  ...
  Topology:
    Control Plane:
      Class:          best-effort-medium
      ...
    Workers:
      Class:          best-effort-medium
      ...
```

- 4 List and describe the available VM classes in the namespace.

```
kubectl get virtualmachineclassbindings
```

Note The command `kubectl get virtualmachineclasses` lists all the VM classes present on the Supervisor Cluster. Because you must associate VM classes with the vSphere Namespace, you can only use those VM classes that are bound to the target namespace.

- Run the following command to edit the cluster manifest.

```
kubectl edit tanzukubernetescluster/CLUSTER-NAME
```

- Edit the manifest by changing the `version` string and unsetting or nulling out the `fullVersion` to avoid a potential version mismatch during discovery.

For example, change the cluster manifest from using the `best-effort-medium` VM class for control plane and worker nodes:

```
spec:
  topology:
    controlPlane:
      class: best-effort-medium
      ...
    workers:
      class: best-effort-medium
      ...
```

To using the `guaranteed-large` VM class for control plane and worker nodes:

```
spec:
  topology:
    controlPlane:
      class: guaranteed-large
      ...
    workers:
      class: guaranteed-large
      ...
```

- Save the changes you made to the manifest file.

When you save the file, `kubectl` applies the changes to the cluster. In the background, the Tanzu Kubernetes Grid Service provisions the new node VMs and spins down the old ones.

- Verify that `kubectl` reports that the manifest edits were successfully recorded.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 edited
```

Note If you receive an error, or `kubectl` does not report that the cluster manifest was successfully edited, make sure you have properly configured your default text editor using the `KUBE_EDITOR` environment variable. See [Specify a Default Text Editor for Kubectl](#).

- Verify that the cluster is updating.

```
kubectl get tanzukubernetescluster
```

NAME	CONTROL PLANE	WORKER	DISTRIBUTION	AGE	PHASE
tkgs-cluster-1	3	3	v1.18.5+vmware.1-tkg.1.c40d30d	21h	updating

10 Verify that the cluster is updated.

```
kubectl get tanzukubernetescluster
```

NAME	CONTROL PLANE	WORKER	DISTRIBUTION	AGE	PHASE
tkgs-cluster-1	3	3	v1.18.5+vmware.1-tkg.1.c40d30d	22h	running

Update a Tanzu Kubernetes Cluster by Changing the Storage Class

You can update a Tanzu Kubernetes cluster by changing the storage class used by the cluster nodes.

The Tanzu Kubernetes Grid Service supports updating a cluster by changing the `StorageClass` for the node pools, that is, by changing the property `.spec.topology.controlPlane.storageClass` or the property `.spec.topology.workers.storageClass`. See [About Tanzu Kubernetes Grid Service Cluster Updates](#).

Prerequisites

This task uses the command `kubectl edit tanzukubernetescluster/CLUSTER-NAME` to update the cluster manifest. The `kubectl edit` command opens the cluster manifest in the text editor defined by your `KUBE_EDITOR` or `EDITOR` environment variable. When you save the file, the cluster is updated with the changes. See [Specify a Default Text Editor for Kubectl](#).

Procedure

- 1 Authenticate with the Supervisor Cluster. See [Connect to the Supervisor Cluster as a vCenter Single Sign-On User](#).

```
kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME
```

- 2 Switch context to the vSphere Namespace where the target Tanzu Kubernetes cluster is provisioned.

```
kubectl config use-context SUPERVISOR-NAMESPACE
```

- 3 To determine available storage classes and decide which to use, run the following command.

```
kubectl describe tanzukubernetescluster CLUSTER-NAME
```

- 4 Run the following command to edit the cluster manifest.

```
kubectl edit tanzukubernetescluster/CLUSTER-NAME
```

5 Edit the manifest by changing the `storageClass` value.

For example, change the cluster manifest from the `silver-storage-class` class for control plane and worker nodes:

```
spec:
  topology:
    controlPlane:
      ...
      storageClass: silver-storage-class
    workers:
      ...
      storageClass: silver-storage-class
```

To the `gold-storage-class` class for control plane and worker nodes:

```
spec:
  topology:
    controlPlane:
      ...
      storageClass: gold-storage-class
    workers:
      ...
      storageClass: gold-storage-class
```

6 Save the changes you made to the manifest file.

When you save the file, `kubectl` applies the changes to the cluster. In the background, the Tanzu Kubernetes Grid Service provisions the new node VMs and spins down the old ones.

7 Verify that `kubectl` reports that the manifest edits were successfully recorded.

```
kubectl edit tanzukubernetescluster/tkgs-cluster-1
tanzukubernetescluster.run.tanzu.vmware.com/tkgs-cluster-1 edited
```

Note If you receive an error, or `kubectl` does not report that the cluster manifest was successfully edited, make sure you have properly configured your default text editor using the `KUBE_EDITOR` environment variable. See [Specify a Default Text Editor for Kubectl](#).

8 Verify that the cluster is updating.

```
kubectl get tanzukubernetescluster
```

NAME	CONTROL PLANE	WORKER	DISTRIBUTION	AGE	PHASE
tkgs-cluster-1	3	3	v1.18.5+vmware.1-tkg.1.c40d30d	21h	updating

9 Verify that the cluster is updated.

```
kubectl get tanzukubernetescluster
```

NAME	CONTROL PLANE	WORKER	DISTRIBUTION	AGE	PHASE
tkgs-cluster-1	3	3	v1.18.5+vmware.1-tkg.1.c40d30d	22h	running

Update Tanzu Kubernetes Clusters Using the Patch Method

You can use the `kubectl patch` method to perform an "in-place" update of a Tanzu Kubernetes cluster. The `kubectl patch` method is an alternative to using the `kubectl edit` command to perform one of the supported cluster update operations.

About the Kubectl Patch Command

Restriction Do not attempt to use `kubectl patch` as described in this topic to update a cluster specification so that it conforms with the TKGS v1alpha2 API. For this type of update you should use `kubectl edit`. See [Updating a Tanzu Kubernetes Release After the Cluster Spec Is Converted to the TKGS v1alpha2 API](#).

The `kubectl patch` command performs an "in-place" update of a cluster. The purpose of this command is to provide a method for upgrading Kubernetes versions, and is the approach documented here. For more information on the `kubectl patch` command, see [Update API Objects in Place Using kubectl patch](#) in the Kubernetes documentation.

The approach demonstrated here uses the UNIX shell command `read` to take input from the keyboard and assign it to a variable named `$PATCH`. The `kubectl patch` command invokes the Kubernetes API to update the cluster manifest. The `--type=merge` flag indicates that the data contains only those properties that are different from the existing manifest.

Upgrade the Kubernetes Version Using the Patch Method

The most common method to trigger a rolling update is by changing the Kubernetes distribution version for the cluster, using the `.spec.distribution.version` and `.spec.distribution.fullVersion` properties. You update the `version` hint and unset or null out the `fullVersion` to avoid a potential version mismatch during discovery.

```
$ read -r -d '' PATCH <<'EOF'
spec:
  distribution:
    fullVersion: null    # NOTE: Must set to null when updating just the version field
    version: v1.18.5
EOF
```


Apply the update using the `kubectl patch` command. You must include the quotes around the `"$PATCH"` variable to preserve newline characters in the cluster manifest. Replace the `TKG-CLUSTER-NAME` value with the actual name of your cluster.

```
kubectl patch --type=merge tanzukubernetescluster TKG-CLUSTER-NAME --patch "$PATCH"
```

Expected result:

```
tanzukubernetescluster.run.tanzu.vmware.com/TKG-CLUSTER-NAME patched
```

Update by the Cluster Changing the VirtualMachineClass for the Nodes Using the Patch Method

Another way to trigger a rolling update of a Tanzu Kubernetes cluster is by changing the `VirtualMachineClass` for the node pools, that is, by changing the property `.spec.topology.controlPlane.class` or the property `.spec.topology.workers.class`.

```
read -r -d '' PATCH <<'EOF'
spec:
  topology:
    controlPlane:
      class: best-effort-xlarge
    workers:
      class: best-effort-xlarge
EOF
```

Apply the update using the `kubectl patch` command, replacing the variable with the cluster name.

```
kubectl patch --type=merge tanzukubernetescluster TKG-CLUSTER-NAME --patch "$PATCH"
```

Expected result:

```
tanzukubernetescluster.run.tanzu.vmware.com/TKG-CLUSTER-NAME patched
```

Update the Cluster by Changing the StorageClass for the Nodes Using the Patch Method

Another way to trigger a rolling update of a Tanzu Kubernetes cluster is by changing the `StorageClass` for the node pools, that is, by changing the property `.spec.topology.controlPlane.storageClass` or the property `.spec.topology.workers.storageClass`.

```
$ read -r -d '' PATCH <<'EOF'
spec:
  topology:
    controlPlane:
      storageClass: gc-storage-profile
    workers:
      storageClass: gc-storage-profile
EOF
```

Apply the update using the `kubectl patch` command, replacing the variable with the cluster name.

```
kubectl patch --type=merge tanzukubernetescluster TKG-CLUSTER-NAME --patch "$PATCH"
```

Expected result:

```
tanzukubernetescluster.run.tanzu.vmware.com/TKG-CLUSTER-NAME patched
```

Backing Up and Restoring vSphere with Tanzu

18

You can backup and restore the workloads running on vSphere Pods and Tanzu Kubernetes clusters, as well as the vCenter Server and NSX-T infrastructure supporting your vSphere with Tanzu installation.

Read the following topics next:

- [Considerations for Backing Up and Restoring vSphere with Tanzu](#)
- [Install and Configure the Velero Plugin for vSphere on the Supervisor Cluster](#)
- [Backup and Restore vSphere Pods Using the Velero Plugin for vSphere](#)
- [Install and Configure the Velero Plugin for vSphere on a Tanzu Kubernetes Cluster](#)
- [Backup and Restore Tanzu Kubernetes Cluster Workloads Using the Velero Plugin for vSphere](#)
- [Install and Configure Standalone Velero and Restic on a Tanzu Kubernetes Cluster](#)
- [Backup and Restore Tanzu Kubernetes Cluster Workloads Using Standalone Velero and Restic](#)
- [Backup and Restore vCenter Server](#)
- [Backup and Restore NSX-T Data Center](#)

Considerations for Backing Up and Restoring vSphere with Tanzu

This topic provides an overview of the backup and restore process for vSphere with Tanzu, and provides high-level considerations for implementing your backup and restore strategy for vSphere with Tanzu.

vSphere with Tanzu backup and restore comprises various layers and tools.

The table summarizes these layers and tools from the top-down, workload to infrastructure. Refer to the individual sections for details on performing backup and restore for that layer.

Scenario	Tools	Comments
Backup and restore vSphere Pods	Velero Plugin for vSphere	Install and configure the plugin on the Supervisor Cluster. Note The plugin is not backing up Supervisor Cluster state. See Install and Configure the Velero Plugin for vSphere on the Supervisor Cluster. See Backup and Restore vSphere Pods Using the Velero Plugin for vSphere.
Backup stateless and stateful workloads on a Tanzu Kubernetes cluster and restore to a cluster provisioned by the Tanzu Kubernetes Grid Service	Velero Plugin for vSphere	Both Kubernetes metadata and persistent volumes can be backed up and restored. Velero snapshotting (not Restic) is used for persistent volumes. See Install and Configure the Velero Plugin for vSphere on a Tanzu Kubernetes Cluster. See Backup and Restore Tanzu Kubernetes Cluster Workloads Using the Velero Plugin for vSphere.
Backup stateless and stateful workloads on a Tanzu Kubernetes cluster and restore to a conformant Kubernetes cluster not provisioned by the Tanzu Kubernetes Grid Service	Standalone Velero and Restic	If you require portability, use standalone Velero. Must include Restic for stateful applications. See Install and Configure Standalone Velero and Restic on a Tanzu Kubernetes Cluster. See Backup and Restore Tanzu Kubernetes Cluster Workloads Using Standalone Velero and Restic.
Supervisor Cluster After a Supervisor Cluster upgrade, you must do a new backup. Restoring a vCenter Server to a backup where it expects an older version of Supervisor Cluster is not supported.	vCenter Server Velero Plugin for vSphere Standalone Velero and Restic	Restore vCenter Server from backup. vCenter will recreate all three Supervisor Cluster control plane VMs. Restore cluster workloads from backup using the plugin or standalone Velero and Restic.
vCenter Configuration	vCenter Server	If vCenter is lost, use vCenter Server to backup and restore vCenter objects. See Backup and Restore vCenter Server.
NSX-T Data Center	NSX-T Manager	Load balancer and ingress services depend on NSX-T backup. Use NSX-T Manager to backup and restore the NSX-T database. See Backup and Restore NSX-T Data Center.

Install and Configure the Velero Plugin for vSphere on the Supervisor Cluster

You can use the Velero Plugin for vSphere to backup and restore workloads running on vSphere Pods by installing the Velero Plugin for vSphere on the Supervisor Cluster.

Overview

The Velero Plugin for vSphere provides a solution for backing up and restoring vSphere with Tanzu workloads. The solution requires the installation and configuration of several components. Once you have the Velero Plugin for vSphere installed and configured on the Supervisor Cluster, you can backup and restore vSphere Pods. For persistent workloads, the Velero Plugin for vSphere lets you take snapshots of the persistent volumes.

Installing Velero Plugin for vSphere on the Supervisor Cluster is also a prerequisite for using the Velero Plugin for vSphere to backup and restore Tanzu Kubernetes cluster workloads.

Note The Velero Plugin for vSphere cannot be used to backup and restore Supervisor Cluster state. See [Considerations for Backing Up and Restoring vSphere with Tanzu](#).

Note The Velero Plugin for vSphere by itself does not perform incremental backups. Dell EMC [PowerProtect](#) supports incremental backup and leverages Velero and the Velero Plugin for vSphere.

Prerequisites

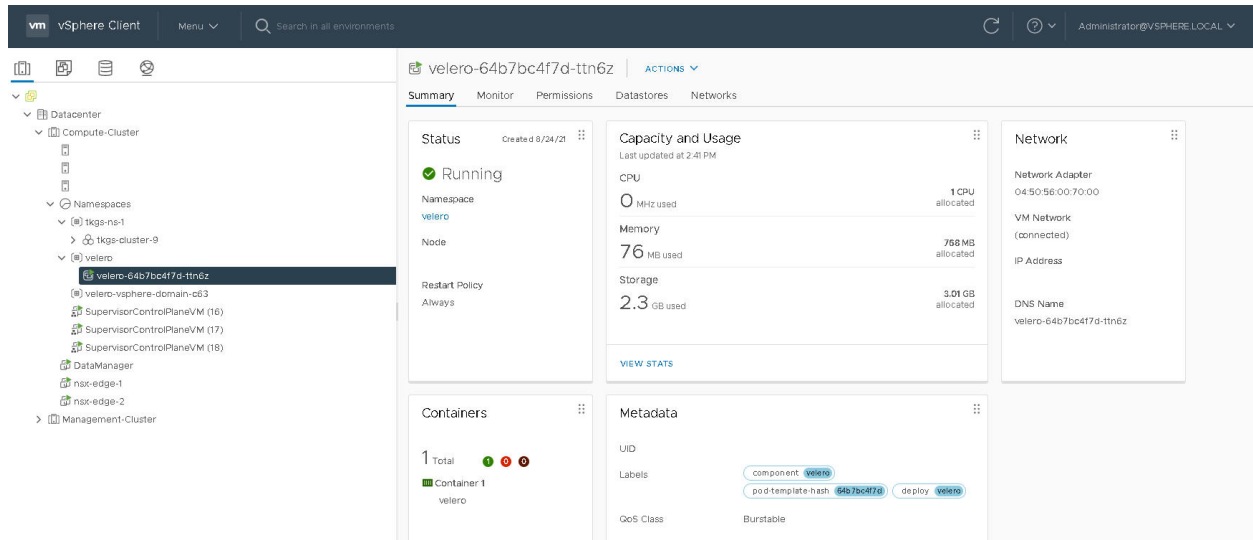
Before installing the Velero Plugin for vSphere, adhere to the following prerequisites:

- Workload Management is enabled with NSX-T Data Center networking. See [Enable Workload Management with NSX Networking](#)
- Supervisor Cluster is version 1.21.1 or later.
- vSphere Namespace is created and configured.
- You must be a member of the vSphere Administrator role, or have following vSphere privileges:
 - **SupervisorServices.Manage**
 - **Namespaces.Manage**
 - **Namespaces.Configure**

The screenshot shows the end state of a Velero Plugin for vSphere installation.

- NSX-T networking is used to support the deployment of vSphere Pods
- A Data Manager VM is deployed
- The Velero Operator is enabled and running in the `velero-vsphere-domain-cXX` namespace
- A namespace called `velero` is configured

- The Velero Plugin for vSphere is running as a vSphere Pod in the `velero` namespace



Upgrades

These instructions assume that you are running vSphere 7 U3. If you previously installed the Velero Plugin for vSphere on a vSphere 7 U2 P3 environment, on upgrade the Data Manager VM and **Velero vSphere Operator** are migrated to the new framework. The **Velero vSphere Operator** is converted to the new vSphere Services format. No action is required.

Create a Dedicated Network for Backup and Restore Traffic (Optional)

Although not required, it is recommended that for production environments you separate the backup and restore traffic from the vSphere with Tanzu management network traffic. There are two aspects to this:

- Tag the ESXi hosts to support network file copy (NFC)
- Configure the backup and restore network using NSX-T Data Center

To configure vSphere 7.x ESXi hosts to support a dedicated network block device (NBD) transport, you add a VMkernel NIC on each ESXi host in the vCenter Server cluster where Workload Management is enabled and set the `vSphereBackupNFC` on that NIC. When the tag `vSphereBackupNFC` is applied to the NIC type for a VMkernel adapter, backup and restore traffic goes through the chosen virtual NIC.

To perform this configuration, you use the Virtual Disk Development Kit. See the [NBD documentation](#).

Note If the `vSphereBackupNFC` is not enabled on the VMkernel NIC, the backup and restore traffic will not be sent on the backup and restore network, even if you configure one. If `vSphereBackupNFC` is not enabled, the traffic will travel on the vSphere management network.

Once the `vSphereBackupNFC` tag is enabled, configure the backup and restore network using NSX-T by updating the existing vSphere Distributed Switch (vDS) for the cluster as follows:

- In the vSphere Client, select **Menu > Networking**.
- Select the existing vDS for the cluster.
- Right-click the vDS and select **Distributed Port Group > New Distributed Port Group**.
- Create a new Distributed Port Group named **BackupRestoreNetwork**.
- Add a VMkernel adapter to the **BackupRestoreNetwork** Distributed Port Group.
- Attach all the ESXi hosts in the vCenter cluster where Workload Management is enabled to the **BackupRestoreNetwork** Distributed Port Group.
- Enable the `vSphereBackupNFC` tag.

For guidance on creating an NSX-T network on the existing vDS, see [Install and Configure NSX for vSphere with Tanzu](#).

Create an S3-Compatible Object Store

For backup and restore of persistent volumes, you need to provide an S3-compatible object store. Velero supports a number of [object store providers](#).

To install the Velero Plugin for vSphere, you will need to provide the following information about your S3-compatible object store:

Data Item	Example Value
s3Url	http://my-s3-store.example.com
aws_access_key_id	ACCESS-KEY-ID-STRING
aws_secret_access_key	SECRET-ACCESS-KEY-STRING

Create a secrets file name `s3-credentials` with the following information. You will reference this file when you install the Velero Plugin for vSphere.

```
[default]
aws_access_key_id = ACCESS-KEY-ID-STRING
aws_secret_access_key = SECRET-ACCESS-KEY-STRING
```

MinIO is an S3-compatible object store that is easy to install and use. vSphere with Tanzu ships with a MinIO Supervisor Service that you can enable. For more information, see [Enable Stateful Services in vSphere with Tanzu](#).

Alternatively, you can manually install a MinIO server on a Linux VM. For instructions, see [Install and Configure Standalone Velero and Restic on a Tanzu Kubernetes Cluster](#).

Install and Configure Data Manager

To facilitate backup and restore using the Velero Plugin for vSphere, you deploy one or more Data Manager VMs to move persistent volume backup data into and out of S3-compatible object storage. The Data Manager moves the volume snapshot data from the vSphere volume to the remote durable S3-compatible storage on backup, and from remote S3-compatible storage to a vSphere volume during restore.

Warning The Data Manager is only functionally tested and it is not meant to work at scale and does not promise any performance expectations. It is not meant to backup business critical applications in production

In a vSphere with Tanzu environment, install the Data Manager as a virtual machine.

Note Do not power on the Data Manager VM until after you have enabled the Velero vSphere Operator.

- 1 Download the Data Manager OVA:
<https://vsphere-velero-datamgr.s3-us-west-1.amazonaws.com/datamgr-ob-17253392-photon-3-release-1.1.ova>
- 2 Using the vSphere Client, right-click the **Datacenter** where Workload Management is enabled and select **Deploy OVF Template**.
- 3 Select the Data Manager OVA file that you downloaded and upload it to the vCenter Server.
- 4 Name the virtual machine **DataManager**, for example.
- 5 Select the compute resource which is the vCenter cluster where the Supervisor Cluster is configured.
- 6 Review the VM deployment details and click **Next**.
- 7 Accept the license agreements and click **Next**.
- 8 Select the storage and click **Next**.
- 9 Select the destination network for the Data Manager VM.
 - Select the **BackupRestoreNetwork** if you configured a dedicated backup and restore network. See [Create a Dedicated Network for Backup and Restore Traffic \(Optional\)](#).
 - Select the **Management Network** if you did not configure a dedicated backup and restore network.
- 10 Confirm the selections and click **Finish** to complete the process.
- 11 Use the Recent Tasks panel to monitor the progress of the deployment.

Note If you receive an error that the "OVF descriptor is not available," use Chrome browser.

- 12 Once the Data Manager VM is deployed, configure the input parameters for the VM.

- 13 Right-click the VM and select **Edit Settings**.
- 14 In the Virtual Hardware tab, for CD/DVD Drive, change from **Host Device** to **Client Device**.

Note If you do not do this, you cannot save the required advanced configuration settings.

- 15 In the **Edit Settings > VM Options** tab, select **Advanced > Edit Configuration Parameters**.
- 16 Configure the input parameters for each of the following settings:

Parameter	Value
guestinfo.cnsdp.vcUser	Enter the vCenter Server user name with sufficient privileges to deploy VMs.
guestinfo.cnsdp.vcAddress	Enter the vCenter Server IP address or FQDN.
guestinfo.cnsdp.vcPasswd	Enter the vCenter Server user password.
guestinfo.cnsdp.vcPort	The default is 443 . Do not change this value.
guestinfo.cnsdp.wcpControlPlaneIP	Enter the Supervisor Cluster IP address. Get this value by navigating to the vCenter cluster where Workload Management is enabled and selecting Configure > Namespaces > Network > Management Network > Starting IP Address
guestinfo.cnsdp.updateKubect1	The default is false . Do not change this value.
guestinfo.cnsdp.veleroNamespace	The default is velero and you should leave it as such unless you have a compelling reason to change it. Later in the process you will create a vSphere Namespace on the Supervisor Cluster with the name velero . These names must match.
guestinfo.cnsdp.datamgrImage	If not configured (unset), the system by default will pull the container image from Docker Hub at vsphereveleroplugin/data-manager-for-plugin:1.1.0

- 17 Click OK to save the configuration and OK again to save the VM settings.

Note If you did not change the CD/DVD Drive from **Host Device** to **Client Device**, you cannot save the settings. If this is the case, cancel the operation, change the drive and repeat the advanced configuration settings.

- 18 Do not power on the Data Manager VM until after you enable the Velero vSphere Operator (next section).

Install the Velero vSphere Operator Service on the Supervisor Cluster

vSphere with Tanzu provides the Velero vSphere Operator as a Supervisor Service. The Velero vSphere Operator service works with the Velero Plugin for vSphere to support backup and restore of Kubernetes workloads, including snapshotting persistent volumes. For more information about Supervisor Services, see [Chapter 8 Managing Supervisor Services with vSphere with Tanzu](#).

Complete the following operation to register the **Velero vSphere Operator** specification with the vCenter Server where **Workload Management** is enabled, and to install the **Velero vSphere Operator** as a service on the Supervisor Cluster.

Note The **Velero vSphere Operator** runs as a vSphere Pod and therefore requires NSX-T networking.

- 1 Refer to the [Compatibility Chart](#) to determine which version of the **Velero vSphere Operator** is appropriate for the version of vCenter Server.

- 2 Download the YAML for the **Velero vSphere Operator** from the following location:

<http://vmware.com/go/supervisor-service>

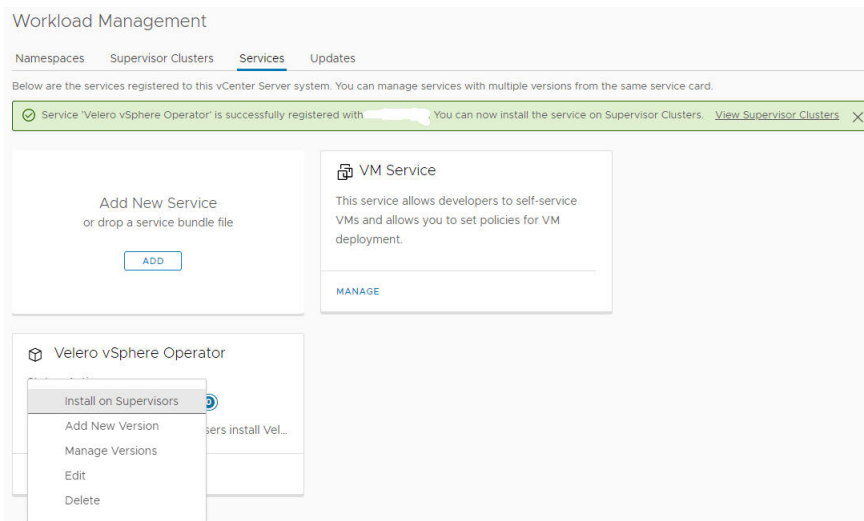
- 3 From the vSphere Client home menu, select **Workload Management**.
- 4 Select the **Services** tab.
- 5 Select the target vCenter Server from the drop-down menu at the top.
- 6 Drag and drop the service specification file that you downloaded to the **Add New Service** card.

Alternatively you can click **Add** and browse to and select the service specification file.

- 7 Click **Next** and accept the licence agreement.
- 8 Click **Finish**.

The **Velero vSphere Operator** is registered with vCenter Server. Verify that the service is in an **Active** state. You cannot install the service if it is Deactivated.

- 9 Locate the **Velero vSphere Operator** specification in the **Services** tab.
- 10 Click **Actions > Install on**



Supervisors.

- 11 Select the target Supervisor Cluster where you want to install the service.

Note If you do not see your Supervisor Cluster, verify that you are using NSX-T networking.

12 Configure the **Velero vSphere Operator** service installation as follows:

- a Select the version from the drop-down: **1.1.0**.
- b Do not specify a **Repository endpoint**.
- c Do not enter a username or password.
- d Click **Next**.

13 Click **Finish** to complete the installation of the service.

Verify the **Velero vSphere Operator** service on the Supervisor Cluster and start the Data Manager VM.

- 1 From the vSphere Client home menu, select **Inventory**.
- 2 Select the vCenter cluster where **Workload Management** is enabled.
- 3 Select **Configure > vSphere Services > Overview**.
- 4 Verify that you see the **Velero vSphere Operator** installed and its status is **Configured**.
- 5 In the **Namespaces** resource pool, verify that you see a new namespace named `svc-velero-vsphere-domain-xxx`, where `xxx` is a unique alphanumeric token. This is the namespace created by the system for the **Velero vSphere Operator**.

Note You do not need to configure this namespace and you should not edit it.

6 In the **Hosts and Clusters** view, select the **Data Manager** VM.

7 Right-click the **Data Manager** VM and power it on.

Create a vSphere Namespace for the Velero Plugin for vSphere

Using the vSphere Client, manually create a vSphere Namespace on the Supervisor Cluster. This namespace is required for the Velero Plugin for vSphere. For guidance, see [Create and Configure a vSphere Namespace](#).

- Name the namespace **velero**.
- Select the **velero** namespace and configure it.
- Specify the Storage for the **velero** namespace.
- Grant a user with appropriate privileges the Edit permission on the **velero** namespace.

Install the Velero Plugin for vSphere

Now you are ready to install the Velero Plugin for vSphere. To do this, download and run the **velero-vsphere** CLI.

Note This procedure requires a Linux VM. Download and run the **velero-vsphere** to the Linux jump host where you run the `kubectl-vsphere` and `kubectl` CLIs.

- 1 Refer to the [Compatibility Chart](#) to determine which version of the Velero Plugin for vSphere is appropriate your environment.
- 2 Download the Velero Plugin for vSphere CLI from the following location:
<https://github.com/vmware-tanzu/velero-plugin-for-vsphere/releases>
- 3 Create a Linux VM where you can run the CLI. Or use an existing Linux jump host where you access the Supervisor Cluster.
- 4 Securely copy the CLI to the Linux jump host. For example:

```
pscp -P 22 C:\temp\velero-vsphere-x.y.z-linux-amd64.tar.gz ubuntu@10.117.29.131:/home/
ubuntu/tanzu
```

- 5 Extract the `velero-vsphere` CLI and make it writable.

```
tar -xf velero-vsphere-x.y.z-linux-amd64.tar.gz
chmod +x velero-vsphere
```

- 6 Create the `s3-credentials` file with the following contents.

```
aws_access_key_id = ACCESS-KEY-ID-STRING
aws_secret_access_key = SECRET-ACCESS-KEY-STRING
```

- 7 Obtain the region, URL, and bucket name for your S3-compatible object store.
- 8 Log in to the Supervisor Cluster using the vSphere Plugin for `kubectl`.
- 9 Switch context to the Supervisor Cluster.

```
kubectl config use-context SUPERVISOR-CLUSTER-IP-ADDRESS
```

- 10 In the **velero** namespace that you created, install a config map file that specifies **SUPERVISOR** as the flavor of the cluster where the Velero Plugin for vSphere will be installed.

```
% cat <<EOF | kubectl -n velero apply -f -
apiVersion: v1
kind: ConfigMap
metadata:
  name: velero-vsphere-plugin-config
data:
  cluster_flavor: SUPERVISOR
EOF
```

- 11 Run the following `velero-vsphere` CLI command to install the Velero Plugin for vSphere into the **velero** namespace.

Replace the placeholder values for the **BUCKET-NAME**, **REGION** (two instances), and **s3Url** fields with the appropriate values. If you deviated from any of the preceding instructions, adjust those values as well, such as the name or location of the secrets file, the name of the manually created `velero` namespace, and so on.

```
./velero-vsphere install \
  --namespace velero \
  --image velero/velero:vX.Y.Z \
  --provider aws \
  --plugins velero/velero-plugin-for-aws:vX.Y.Z,vsphereveleroplugin/velero-plugin-for-
vsphere:X.Y.Z \
  --bucket BUCKET-NAME \
  --secret-file s3-credentials \
  --snapshot-location-config region=REGION \
  --backup-location-config region=REGION,s3ForcePathStyle="true",s3Url=http://my-s3-
store.example.com
```

Note Refer to the [Compatibility Chart](#) to determine which version of the Velero Plugin for vSphere is appropriate your environment.

- 12 Verify successful installation of the Velero Plugin for vSphere.

On successful installation, you see the following message:

```
Send the request to the operator about installing Velero in namespace velero
```

Run the following command to further verify that you see "Completed" and the version.

```
kubectl -n velero get veleroservice default -o json | jq '.status'
```

Expected result:

```
{
  "enabled": true,
  "installphase": "Completed",
  "version": "vX.Y.Z"
}
```

Note The above command assumes you have the `jq` utility installed, which formats JSON output sent to the terminal. If you do not have `jq` installed, either install it or remove that part of the command (everything after `json`).

- 13 Troubleshoot as necessary.

If the installation is not successful, remove the installation and try again. To remove the installation, complete the steps in the next section in the order listed.

Install the Velero Plug-in in an Air-gapped Environment

If you plan to install the Velero Plugin for vSphere in an air-gapped environment, you must install it with customized images. You must make sure that the matching images of `backup-driver` and `data-manager-for-plugin` of the customized images are available in the expected private registry and are accessible from the Kubernetes cluster.

In an air-gapped environment, customized images from private registry are expected since the released images in docker hub are not accessible. You can use the embedded Harbor Registry to serve as the private container registry for the customized images. For more information, see [Use the Embedded Harbor Registry with Tanzu Kubernetes Clusters](#).

To install the plug-in, perform the following steps:

- 1 Download the released images of `velero-plugin-for-vsphere`, `backup-driver`, and `data-manager-for-plugin`.
- 2 Rename the images, that is, tag them with matching `<Registry endpoint and path>` and `<Version tag>` and upload them in customized repositories.
- 3 You can deploy the Velero Plugin for vSphere with images in the following scenarios:
 - a When images are from the same private registry. To deploy the Velero Plugin for vSphere with images from a private registry, verify that you have provided the credentials of the registry while enabling the Velero vSphere Operator. You must also specify the `--use-private-registry` option in the install command. Install the plug-in using the `velero-plugin-for-vsphere` image you customized.

For example, you can use the following `velero-vsphere install` command:

```
BUCKET=velero-vX.Y.Z-backups
REGION=minio
NAMESPACE=velero
S3URL=http://csm-minio.eng.vmware.com

VSPHERE_PLUGIN_IMAGE=harbor-stg-repo.vmware.com/velerovsphereoperator/velero-plugin-for-vsphere:vX.Y.Z
AWS_PLUGIN_IMAGE=harbor-stg-repo.vmware.com/velerovsphereoperator/velero-plugin-for-aws:vX.Y.Z
VELERO_IMAGE=harbor-stg-repo.vmware.com/velerovsphereoperator/velero:vX.Y.Z

# install velero in air-gapped environment
velero-vsphere install \
  --namespace $NAMESPACE \
  --image $VELERO_IMAGE \
  --provider aws \
  --plugins $AWS_PLUGIN_IMAGE,$VSPHERE_PLUGIN_IMAGE \
  --bucket $BUCKET \
  --secret-file ~/.minio/credentials \
```

```
--snapshot-location-config region=$REGION \
--backup-location-config region=$REGION,s3ForcePathStyle="true",s3Url=$S3URL \
--use-private-registry # <===== Key Option to deploy
Velero with images from private registry
```

Note Deploying the plug-in from a different private registry or not using the `--use-private-registry` in the install command results in a **ImagePullBackOff** error.

- b When images are in any public registry. You can deploy the Velero Plugin for vSphere instance with images from any other public registry, even if the Velero vSphere Operator is deployed with images from a private registry. Install the plug-in using the `velero-plugin-for-vsphere` image you customized.

For example, you can use the following `velero-vsphere install` command:

```
BUCKET=velero-vX.Y.Z-backups
REGION=minio
NAMESPACE=velero
S3URL=http://csm-minio.eng.vmware.com

VSPHERE_PLUGIN_IMAGE=harbor-repo.vmware.com/velero/velero-plugin-for-vsphere:vX.Y.Z
AWS_PLUGIN_IMAGE=harbor-repo.vmware.com/velero/velero-plugin-for-aws:vX.Y.Z
VELERO_IMAGE=harbor-repo.vmware.com/velero/velero:vX.Y.Z

# install velero in air-gapped environment
velero-vsphere install \
  --namespace $NAMESPACE \
  --image $VELERO_IMAGE \
  --provider aws \
  --plugins $AWS_PLUGIN_IMAGE,$VSPHERE_PLUGIN_IMAGE \
  --bucket $BUCKET \
  --secret-file ~/.minio/credentials \
  --snapshot-location-config region=$REGION \
  --backup-location-config region=$REGION,s3ForcePathStyle="true",s3Url=$S3URL
```

- 4 When you install Velero Plugin for vSphere in a vanilla cluster, it deploys two additional components, that is, a `backup-driver` deployment and a `data-manager-for-plugin` DaemonSet in the background. In the Supervisor Cluster and Tanzu Kubernetes clusters, only a `backup-driver` deployment is deployed.

When you provide the container image of `velero-plugin-for-vsphere`, the matching `backup-driver` and `data-manager-for-plugin` images are parsed using an image parsing mechanism.

Container images are formalized into the following pattern:

```
<Registry endpoint and path>/<Container name>:<Version tag>
```

When you provide the `velero-plugin-for-vsphere` container image, the corresponding images of `backup-driver` and `data-manager-for-plugin` with matching `<Registry endpoint and path>` and `<Version tag>` are parsed.

For example, consider the following `velero-plugin-for-vsphere` container image:

```
abc.io:8989/x/y/.../z/velero-plugin-for-vsphere:vX.Y.Z
```

The following matching images of `backup-driver` and `data-manager-for-plugin` are expected to be pulled:

```
abc.io:8989/x/y/.../z/backup-driver:vX.Y.Z
abc.io:8989/x/y/.../z/data-manager-for-plugin:vX.Y.Z
```

5 Troubleshoot the installation.

If there are any issues or errors in parsing the matching images of `backup-driver` and `data-manager-for-plugin`, the installation falls back to corresponding images from the official `velerovsphereplugin` repositories in Docker hub. The following issues trigger the fall-back mechanism:

- a An unexpected container name is used in the customized `velero-plugin-for-vsphere` image in the user input.

For example, `x/y/velero-velero-plugin-for-vsphere:vX.Y.Z` is used.

- b The Velero deployment name is customized to anything other than `velero`. For example, an issue is triggered if the Velero deployment name is updated to `velero-server` in the Velero `manifests` file before deploying Velero.

The existing image parsing mechanism in `velero-plugin-for-vsphere` can only recognize the Velero deployment with the fixed name, `velero`.

Uninstall the Velero Plugin for vSphere

Complete these steps to uninstall the Velero Plugin for vSphere. Complete these steps in the order listed.

- 1 Run the `velero-vsphere` CLI to uninstall the Velero Plugin for vSphere.

```
./velero-vsphere uninstall -n velero
```

- 2 Verify that the vSphere Pod named `velero` is removed.

```
kubectl get pods -n velero
```

If you see that the pod is "Terminating," wait for it to be removed before proceeding.

- 3 Using the vSphere Client, delete the vSphere Namespace named `velero` that you manually created.

Note Do not proceed with the next step until the namespace deletion is complete. You can use `kubectl` to verify that the `velero` namespace is removed (but do not use `kubectl` to remove the `velero` namespace).

- 4 Using the vSphere Client, uninstall the **Velero vSphere Operator** from the Supervisor Cluster.
 - a Select the vCenter cluster where **Workload Management** is enabled.
 - b Select **Configure > vSphere Services > Overview**.
 - c Select the **Velero vSphere Operator**.
 - d Click **Uninstall**.

This action uninstalls the **Velero vSphere Operator** from the Supervisor Cluster. The operator remains available for re-installation at the **Workload Management > Services** page. To remove the service entirely, select **Actions > Delete**.

Backup and Restore vSphere Pods Using the Velero Plugin for vSphere

You can use the Velero Plugin for vSphere to backup and restore workloads running on vSphere Pods.

Overview

You can use the Velero Plugin for vSphere to backup and restore workloads running on vSphere Pods in the Supervisor Cluster. You can backup and restore both stateless and stateful applications running on vSphere Pods. For stateful applications, you use the Velero Plugin for vSphere to take snapshots of the persistent volumes (PVs).

Note You cannot use Velero standalone with Restic to backup and restore vSphere Pods. You must use the Velero Plugin for vSphere installed on the Supervisor Cluster.

Prerequisites

Before you can backup and restore vSphere Pods, you need to install and configure the Velero Plugin for vSphere. See [Install and Configure the Velero Plugin for vSphere on the Supervisor Cluster](#).

Note The Velero Plugin for vSphere is not backing up and restoring the state of the Supervisor Cluster.

Backup a vSphere Pod

To backup a stateless vSphere Pod, run the following command:

```
velero backup create <backup name> --include-namespaces=my-namespace
```

The backup is marked `Completed` after all local snapshots have been taken and Kubernetes metadata is uploaded to the object store. However, the backup of volume snapshots occurs asynchronously and may still be occurring in the background and take some time to complete.

You can check the status of volume snapshots by monitoring Snapshot and Upload custom resources.

Snapshot CRD

For each volume snapshot, a Snapshot custom resource is created in the same namespace as the PVC that is snapshotted. You can get all Snapshots in PVC namespace by running the following command.

```
kubectl get -n <pvc namespace> snapshot
```

The Snapshot CRD has several phases for the `status.phase` field, including:

State	Description
New	Not processed yet
Snapshotted	Local snapshot was taken
SnapshotFailed	Local snapshot failed
Uploading	Snapshot is being uploaded
Uploaded	Snapshot is uploaded
UploadFailed	Snapshot failed to be uploaded
Canceling	Upload of snapshot is being cancelled
Canceled	Upload of snapshot is cancelled
CleanupAfterUploadFailed	Cleanup of local snapshot after the upload of snapshot failed

Upload CRD

For each volume snapshot to be uploaded to object store, an Upload CR will be created in the same namespace as Velero. You can get all Uploads in Velero namespace by running the following command.

```
kubectl get -n <velero namespace> upload
```

Upload CRD has several phases for the `status.phase` field, including:

State	Description
New	Not processed yet
InProgress	Upload in progress
UploadError	Upload failed
CleanupFailed	Delete local snapshot failed after the upload Will be retried
Canceling	Upload is being cancelled Can occur if <code>velero backup delete</code> is called while snapshot upload is in progress
Canceled	Upload is cancelled

UploadError uploads will be periodically retried. At that point their phase will return to InProgress. After an upload has been successfully completed, its record will remain for a period of time and eventually be removed.

Restore a vSphere Pod

To restore a vSphere Pod workload that has been backed up using the Velero Plugin for vSphere, complete the following steps.

- 1 Create a vSphere Namespace for the workload you will restore.
- 2 Configure the storage policy for the namespace.
- 3 Run the following Velero command to restore the workload:

```
velero restore create --from-backup backup-name
```

Velero restore will be marked as `Completed` when volume snapshots and other Kubernetes metadata have been successfully restored to the current cluster. At this point, all tasks of vSphere plugin related to this restore are completed as well. There are no asynchronous data movement tasks behind the scene as that in the case of Velero backup.

Before Velero restore is `Completed`, you can check the status of volume restore by monitoring `CloneFromSnapshot/Download` CRs as below.

CloneFromSnapshots CRD

For restore from each volume snapshot, a `CloneFromSnapshot` CR will be created in the same namespace as the PVC that is originally snapshotted. We can get all `CloneFromSnapshots` in PVC namespace by running the following command.

```
kubectl -n <pvc namespace> get clonefromsnapshot
```

`CloneFromSnapshot` CRD has several phases for the `status.phase` field, including:

State	Description
New	Clone from snapshot is not completed
Completed	Clone from snapshot is completed
Failed	Clone from snapshot failed

Download CRD

From each restore of volume snapshot to be downloaded from object store, a `Download` CR will be created in the same namespace as Velero. We can get all `Downloads` in Velero namespace by running the following command.

```
kubectl -n <velero namespace> get download
```

The `Download` CRD has several phases for the `status.phase` field, including:

Status	Description
New	Not processed yet
InProgress	Download in progress
Completed	Download is completed
Retry	Download is retried. When there is any failure during the download of backup data, the download is retried
Failed	Download is failed

Install and Configure the Velero Plugin for vSphere on a Tanzu Kubernetes Cluster

You can use the Velero Plugin for vSphere to backup and restore workloads running on a Tanzu Kubernetes cluster by installing the Velero Plugin for vSphere on that cluster.

Overview

The Velero Plugin for vSphere provides a solution for backing up and restoring Tanzu Kubernetes cluster workloads for clusters provisioned by the Tanzu Kubernetes Grid Service. For persistent workloads, the Velero Plugin for vSphere lets you take snapshots of the persistent volumes.

Note If you require portability for the Tanzu Kubernetes cluster workloads you want to backup and restore, do not use the Velero Plugin for vSphere. For portability across Kubernetes clusters, use standalone Velero with Restic. See [Install and Configure Standalone Velero and Restic on a Tanzu Kubernetes Cluster](#).

Prerequisite: Install the Velero Plugin for vSphere on the Supervisor Cluster

Installing the Velero Plugin for vSphere on a Tanzu Kubernetes cluster requires the Supervisor Cluster to have the Velero Plugin for vSphere installed. In addition, the Supervisor Cluster must be configured with NSX-T networking.

Prior to the installation of the Velero Plugin for vSphere on a Tanzu Kubernetes cluster, you must first install the Velero Plugin for vSphere on the Supervisor Cluster. See [Install and Configure the Velero Plugin for vSphere on the Supervisor Cluster](#).

Install the Velero CLI on a Linux Workstation

The Velero CLI is the standard tool for interfacing with Velero. The Velero CLI provides more functionality than the Velero Plugin for vSphere CLI (`velero-vsphere`) and is required for backing up and restoring Tanzu Kubernetes cluster workloads.

Install the Velero CLI on a Linux workstation. Ideally this is the same jump host where you run associated CLIs for your vSphere with Tanzu environment, including `kubectl`, `kubectl-vsphere`, and `velero-vsphere`.

Complete the following steps to install the Velero CLI.

- 1 Download the supported version of the Velero CLI from the VMware product downloads page. For more information about the supported Velero version, see the [Release Notes](#).
- 2 Open a command line and change directory to the Velero CLI download.

```
gunzip velero-linux-v1.x.x_vmware.1.gz
```

- 3 Check for the Velero binary.

```
ls -l

-rw-r--r-- 1 root root 7142128 Aug 14 14:14 velero-linux-v1.x.x_vmware.1
```

- 4 Grant execute permissions to the Velero CLI.

```
chmod +x velero-linux-v1.x.x_vmware.1
```

- 5 Make the Velero CLI globally available by moving it to the system path.

```
cp velero-linux-v1.x.x_vmware.1 /usr/local/bin/velero
```

- 6 Verify the installation of the Velero CLI.

```
velero version

Client:
  Version: v1.x.x
```

Get the S3-Compatible Bucket Details

For convenience, the steps assume that you are using the same S3-compatible object store that you configured when you installed the Velero Plugin for vSphere on the Supervisor Cluster. In production you may want to create a separate object store.

To install the Velero Plugin for vSphere, you will need to provide the following information about your S3-compatible object store.

Data Item	Example Value
s3Url	http://my-s3-store.example.com
aws_access_key_id	ACCESS-KEY-ID-STRING
aws_secret_access_key	SECRET-ACCESS-KEY-STRING

Create a secrets file name `s3-credentials` with the following information. You will reference this file when you install the Velero Plugin for vSphere.

```
aws_access_key_id = ACCESS-KEY-ID-STRING
aws_secret_access_key = SECRET-ACCESS-KEY-STRING
```

Create the Velero Plugin Configmap

On the TKG cluster, create a configmap for the Velero plugin named `velero-vsphere-plugin-config.yaml`.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: velero-vsphere-plugin-config
data:
  cluster_flavor: GUEST
```

Apply the configmap on the TKG cluster.

```
kubectl apply -n <velero-namespace> -f velero-vsphere-plugin-config.yaml
```

If you do not install the configmap, you receive the following error when you try to install the Velero Plugin for vSphere.

```
Error received while retrieving cluster flavor from config, err: configmaps "velero-vsphere-
plugin-config" not found
Falling back to retrieving cluster flavor from vSphere CSI Driver Deployment
```

Install the Velero Plugin for vSphere on the Tanzu Kubernetes Cluster

You are going to use the Velero CLI to install the Velero Plugin for vSphere on the target Tanzu Kubernetes cluster that you want to backup and restore.

The Velero CLI context will automatically follow the `kubectl` context. Before running Velero CLI commands to install Velero and the Velero Plugin for vSphere on the target cluster, be sure to set the `kubectl` context to the target cluster.

- 1 Using the vSphere Plugin for `kubectl`, authenticate with the Supervisor Cluster. See [Connect to the Supervisor Cluster as a vCenter Single Sign-On User](#).
- 2 Set the `kubectl` context to the target Tanzu Kubernetes cluster.

```
kubectl config use-context TARGET-TANZU-KUBERNETES-CLUSTER
```

- 3 Run the following Velero CLI command to install Velero on the target cluster.

Replace the placeholder values for the **BUCKET-NAME**, **REGION** (two instances), and **s3Url** fields with the appropriate values. If you deviated from any of the preceding instructions, adjust those values as well, such as the name or location of the secrets file, the name of the manually created `velero` namespace, etc.

```
./velero install --provider aws \
--bucket BUCKET-NAME \
--secret-file ./s3-credentials \
--features=EnableVSPHEREItemActionPlugin \
--plugins velero/velero-plugin-for-aws:v1.1.0 \
--snapshot-location-config region=REGION \
--backup-location-config region=REGION,s3ForcePathStyle="true",s3Url=http://my-s3-
store.example.com
```

- 4 Install the Velero Plugin for vSphere on the target cluster. The installed Velero will communicate with Kubernetes API server to install the plugin.

```
velero plugin add vsphereveleroplugin/velero-plugin-for-vsphere:1.1.0
```

Uninstall the Velero Plugin for vSphere from the Cluster

Complete these steps to uninstall the Velero Plugin for vSphere.

- 1 Set the `kubectl` context to the target Tanzu Kubernetes cluster.

```
kubectl config use-context TARGET-TANZU-KUBERNETES-CLUSTER
```

- 2 To uninstall the plugin, run the following command to remove the `InitContainer` of `velero-plugin-for-vsphere` from the Velero deployment.

```
velero plugin remove vsphereveleroplugin/velero-plugin-for-vsphere:1.1.0
```

- 3 To complete the un-installation, delete the Backup Driver deployment and related CRDs.

```
kubectl -n velero delete deployment.apps/backup-driver
```

```
kubectl delete crds \
backuprepositories.backupdriver.cnsdp.vmware.com \
backuprepositoryclaims.backupdriver.cnsdp.vmware.com \
clonefromsnapshots.backupdriver.cnsdp.vmware.com \
deletesnapshots.backupdriver.cnsdp.vmware.com \
snapshots.backupdriver.cnsdp.vmware.com
```

```
kubectl delete crds uploads.datamover.cnsdp.vmware.com downloads.datamover.cnsdp.vmware.com
```

Backup and Restore Tanzu Kubernetes Cluster Workloads Using the Velero Plugin for vSphere

You can backup and restore Tanzu Kubernetes cluster workloads using the Velero Plugin for vSphere. However, if you require portability, use standalone Velero.

Prerequisites

To backup and restore Tanzu Kubernetes clusters workloads using the Velero Plugin for vSphere, you must first install Velero and the Velero Plugin for vSphere on the target cluster. See [Install and Configure the Velero Plugin for vSphere on a Tanzu Kubernetes Cluster](#).

Backup a Workload

Below is an example command for creating a Velero backup.

```
velero backup create <backup name> --include-namespaces=my-namespace
```

Velero backup will be marked as `Completed` after all local snapshots have been taken and Kubernetes metadata, except volume snapshots, has been uploaded to the object store. At this point, asynchronous data movement tasks, i.e., the upload of volume snapshot, are still happening in the background and may take some time to complete. We can check the status of volume snapshot by monitoring [Snapshot Custom Resources \(CRs\)](#).

Snapshots

Snapshots are used for backing up persistent volumes. For each volume snapshot, a Snapshot CR is created in the same namespace as the persistent volume claim (PVC) that is snapshotted.

You can get all Snapshots in PVC namespace by running the following command.

```
kubectl get -n <pvc namespace> snapshot
```

The Snapshot Custom Resource Definition (CRD) has a number of phases for the `.status.phase` field, including:

Snapshot Phase	Description
New	Not processed yet
Snapshotted	Local snapshot was taken
SnapshotFailed	Local snapshot was failed
Uploading	The snapshot is being uploaded
Uploaded	The snapshot is uploaded
UploadFailed	The snapshot is failed to be uploaded
Canceling	The upload of snapshot is being cancelled

Snapshot Phase	Description
Canceled	The upload of snapshot is cancelled
CleanupAfterUploadFailed	The Cleanup of local snapshot after the upload of snapshot was failed

Restore a Workload

Below is an example command of Velero restore.

```
velero restore create --from-backup <velero-backup-name>
```

Velero restore will be marked as `Completed` when volume snapshots and other Kubernetes metadata have been successfully restored to the current cluster. At this point, all tasks of vSphere plugin related to this restore are completed as well. There are no any asynchronous data movement tasks behind the scene as that in the case of Velero backup.

CloneFromSnapshots

To restore from each volume snapshot, a `CloneFromSnapshot` Custom Resource (CR) will be created in the same namespace as the PVC that is originally snapshotted. We can get all `CloneFromSnapshots` in PVC namespace by running the following command.

```
kubectl -n <pvc namespace> get clonefromsnapshot
```

`CloneFromSnapshot` CRD has some key phases for the `.status.phase` field:

Snapshot Phase	Description
New	Clone from snapshot is not completed
InProgress	The vSphere volume snapshot is being downloaded from remote repository
Completed	Clone from snapshot is completed
Failed	Clone from snapshot is failed

Install and Configure Standalone Velero and Restic on a Tanzu Kubernetes Cluster

To backup and restore workloads on Tanzu Kubernetes, create a data store and install Velero with Restic on the Kubernetes cluster.

Overview

Tanzu Kubernetes clusters run on virtual machine nodes. To backup and restore Tanzu Kubernetes clusters, you install Velero and Restic on the cluster.

Prerequisites

Ensure that your environment meets the following prerequisites for installing Velero and Restic to back up and restore workloads running on Tanzu Kubernetes clusters.

- A Linux VM with sufficient storage to store several workload backups. You will install MinIO on this VM.
- A Linux VM where the Kubernetes CLI Tools for vSphere are installed, including the vSphere Plugin for kubectl and kubectl. You will install the Velero CLI on this client VM. If you do not have such a VM, you can install the Velero CLI locally, but you must adjust the installation steps accordingly.
- The Kubernetes environment has internet access and can be reached by the client VM.

Install and Configure MinIO Object Store

Velero requires an S3-compatible object store as the destination for Kubernetes workload backups. Velero supports several such [object store providers](#). For simplicity, these instructions use [MinIO](#), an S3-compatible storage service that runs locally on the object store VM.

- 1 Install MinIO.

```
wget https://dl.min.io/server/minio/release/linux-amd64/minio
```

- 2 Grant execute permissions to MinIO.

```
chmod +x minio
```

- 3 Create a directory on the file system for MinIO.

```
mkdir /DATA-MINIO
```

- 4 Start the MinIO server.

```
./minio server /DATA-MINIO
```

- 5 After the MinIO server starts, you are provided with important data store instance details, including the Endpoint URL, AccessKey, and SecretKey. Record the Endpoint URL, AccessKey, and SecretKey in the table.

Data Store Metadata	Value
Endpoint URL	
AccessKey	
SecretKey	

- 6 Browse to the MinIO data store by opening a browser to the MinIO server endpoint URL.
- 7 Log in to the MinIO server and provide the AccessKey and SecretKey.

- 8 To enable MinIO as a service, configure MinIO for automatic start-up by download the `minio.service` script.

```
curl -O https://raw.githubusercontent.com/minio/minio-service/master/linux-systemd/minio.service
```

- 9 Edit the `minio.service` script and add the following value for `ExecStart`.

```
ExecStart=/usr/local/bin/minio server /DATA-MINIO path
```

- 10 Save the revised script.
- 11 Configure the MinIO service by running the following commands.

```
cp minio.service /etc/systemd/system
cp minio /usr/local/bin/
systemctl daemon-reload
systemctl start minio
systemctl status minio
systemctl enable minio
```

- 12 Create a MinIO bucket for backup and restore by launching the MinIO browser and logging in to your object store.
- 13 Click the Create Bucket icon.
- 14 Enter the bucket name, for example: `my-cluster-backups`.
- 15 Verify that the bucket was created.
- 16 By default, a new MinIO bucket is read-only. For Velero standalone backup and restore, the MinIO bucket must be read-write. To set the bucket to read-write, select the bucket and click on the ellipses (dots) link.
- 17 Select **Edit Policy**.
- 18 Change the policy to **Read and Write**.
- 19 Click **Add**.
- 20 To close the dialog box, click X.

Install the Velero CLI

Install the Velero CLI on the VM client or on your local machine.

- 1 Download the supported version of the signed Velero binary for vSphere with Tanzu from the VMware product downloads page.

Note You must use the Velero binary signed by VMware to be eligible for support from VMware.

- 2 Open a command line and change directory to the Velero CLI download.

- 3 Unzip the download file. For example:

```
gunzip velero-linux-vX.X.X_vmware.1.gz
```

- 4 Check for the Velero binary.

```
ls -l
```

- 5 Grant execute permissions to the Velero CLI.

```
chmod +x velero-linux-vX.X.X_vmware.1
```

- 6 Make the Velero CLI globally available by moving it to the system path:

```
cp velero-linux-vX.X.X_vmware.1 /usr/local/bin/velero
```

- 7 Verify the installation.

```
velero version
```

Install Velero and Restic on the Tanzu Kubernetes Cluster

The Velero CLI context will automatically follow the kubectl context. Before running Velero CLI commands to install Velero and Restic on the target cluster, set the kubectl context.

- 1 Retrieve the name of the MinIO bucket. For example, `my-cluster-backups`.
- 2 Get the AccessKey and SecretKey for the MinIO bucket.
- 3 Set the context for the target Kubernetes cluster so that the Velero CLI knows which cluster to work on.

```
kubectl config use-context tkgs-cluster-name
```

- 4 Create a secrets file named `credentials-minio`. Update the file with the MinIO server access credentials that you collected. For example:

```
aws_access_key_id = 0XXN08JCCGV41QZBV0RQ
aws_secret_access_key = c1Z1bf8Ljkvkmq7fHucrKCkxV39BRbcycGeXQDfx
```

Note If you receive an error message "Error getting a backup store" with the description "NoCredentialProviders: no valid providers in chain," prepend the line `[default]` to the beginning of the credentials file. For example:

```
[default]
aws_access_key_id = 0XXN08JCCGV41QZBV0RQ
aws_secret_access_key = c1Z1bf8Ljkvkmq7fHucrKCkxV39BRbcycGeXQDfx
```

- 5 Save the file and verify that the file is in place.

```
ls
```

- 6 Run the following command to install Velero and Restic on the target Kubernetes cluster. Replace both URLs with the URL of your MinIO instance.

```
velero install \
--provider aws \
--plugins velero/velero-plugin-for-aws:v1.0.0 \
--bucket tkgs-velero \
--secret-file ./credentials-minio \
--use-volume-snapshots=false \
--use-restic \
--backup-location-config \
region=minio,s3ForcePathStyle="true",s3Url=http://10.199.17.63:9000,publicUrl=http://
10.199.17.63:9000
```

- 7 Verify the installation of Velero and Restic.

```
kubectl logs deployment/velero -n velero
```

- 8 Verify the `velero` namespace.

```
kubectl get ns
```

- 9 Verify the `velero` and `restic` pods.

```
kubectl get all -n velero
```

Troubleshoot Restic DaemonSet (If Necessary)

To run the three-pod Restic DaemonSet on a Kubernetes cluster, you may have to update the Restic DaemonSet spec and modify the `hostPath`. For more information about this issue, see [Restic Integration](#) in the Velero documentation.

- 1 Verify the three-pod Restic DaemonSet.

```
kubectl get pod -n velero
```

If the pods are in a `CrashLoopBackOff` status, edit them as follows.

- 2 Run the `edit` command.

```
kubectl edit daemonset restic -n velero
```

- 3 Change `hostPath` from `/var/lib/kubelet/pods` to `/var/vcap/data/kubelet/pods`.

```
- hostPath:
  path: /var/vcap/data/kubelet/pods
```

- 4 Save the file.
- 5 Verify the three-pod Restic DaemonSet.

```
kubectl get pod -n velero
```

NAME	READY	STATUS	RESTARTS	AGE
restic-5jln8	1/1	Running	0	73s
restic-bpvtq	1/1	Running	0	73s
restic-vg8j7	1/1	Running	0	73s
velero-72c84322d9-1e7bd	1/1	Running	0	10m

Adjust Velero Memory Limits (If Necessary)

If your Velero backup returns `status=InProgress` for many hours, increase the limits and requests memory settings.

- 1 Run the following command.

```
kubectl edit deployment/velero -n velero
```

- 2 Change the limits and request memory settings from the default of 256Mi and 128Mi to 512Mi and 256Mi.

```
ports:
- containerPort: 8085
  name: metrics
  protocol: TCP
resources:
  limits:
    cpu: "1"
    memory: 512Mi
  requests:
    cpu: 500m
    memory: 256Mi
terminationMessagePath: /dev/termination-log
terminationMessagePolicy: File
```

Backup and Restore Tanzu Kubernetes Cluster Workloads Using Standalone Velero and Restic

You can backup and restore Tanzu Kubernetes cluster workloads using standalone Velero and Restic. This method is an alternative to using the Velero Plugin for vSphere. The primary reason for using standalone Velero instead of the Velero Plugin for vSphere is if you require portability. Restic is required for stateful workloads.

Prerequisites

To backup and restore Tanzu Kubernetes clusters workloads using standalone Velero and Restic, you must install the standalone version of Velero and Restic on the target cluster. See [Install and Configure Standalone Velero and Restic on a Tanzu Kubernetes Cluster](#).

Note Backing up and restoring a Kubernetes cluster using standalone Velero with Restic gives you portability. This means if you want to be able to restore cluster workloads to a Kubernetes cluster not provisioned by Tanzu Kubernetes Grid Service, you need to use standalone Velero.

Backup a Stateless Application Running on a Tanzu Kubernetes Cluster

Backing up a stateless application running on a Tanzu Kubernetes cluster requires the use of Velero.

This example shows how to backup and restore an example stateless application using the `--include namespaces` tag where all application components are in that namespace.

```
velero backup create example-backup --include-namespaces example-backup
```

You should see the following:

```
Backup request "example-backup" submitted successfully.
Run `velero backup describe example-backup` or `velero backup logs example-backup` for more details.
```

Verify the backup that was created.

```
velero backup get
```

```
velero backup describe example-backup
```

Check the Velero bucket on the S3-compatible object store such as the MinIO server.

Velero writes some metadata in Kubernetes custom resource definitions (CRDs).

```
kubectl get crd
```

The Velero CRDs let you run certain commands, such as the following:

```
kubectl get backups.velero.io -n velero
```

```
kubectl describe backups.velero.io guestbook-backup -n velero
```

Restore a Stateless Application Running on a Tanzu Kubernetes Cluster

Restoring a stateless application running on a Tanzu Kubernetes cluster requires the use of Velero.

To test the restoration of an example application, delete it.

Delete the namespace:

```
kubectrl delete ns guestbook
namespace "guestbook" deleted
```

Restore the app:

```
velero restore create --from-backup example-backup
```

You should see the following:

```
Restore request "example-backup-20200721145620" submitted successfully.
Run `velero restore describe example-backup-20200721145620` or `velero restore logs example-backup-20200721145620` for more details.
```

Verify that the app is restored:

```
velero restore describe example-backup-20200721145620
```

Run the following commands to verify:

```
velero restore get
```

```
kubectrl get ns
```

```
kubectrl get pod -n example
```

```
kubectrl get svc -n example
```

Backup a Stateful Application Running on a Tanzu Kubernetes Cluster

Backing up a stateful application running on a Tanzu Kubernetes cluster involves backing up both the application metadata and the application data stored on a persistent volume. To do this you need both Velero and Restic.

For this example we are going to use the Guestbook application. It is assumed that you have deployed the Guestbook application to a Tanzu Kubernetes cluster. See [Tanzu Kubernetes Guestbook Tutorial](#) for guidance.

So that we can demonstrate stateful backup and restore, submit some message to the Guestbook application using the frontend web page so that the messages are persisted. For example:

← → ↻
🛡️ 🔒 10.1 . 4.7

Guestbook

Messages

Submit

message 1

message 2

message 3

This example shows how to backup and restore the Guestbook app using the `--include namespace` tag as well as pod annotations.

Note This example uses annotations. However, annotations are no longer needed for Velero version 1.5 and later. To not use annotations, you can use the `--default-volumes-to-restic` option when creating the backup. This will automatically backup all PVs using Restic. See <https://velero.io/docs/v1.5/restic/> for more information.

To begin the backup procedure, get the names of the pods:

```
kubectl get pod -n guestbook
```

For example:

```
kubectl get pod -n guestbook
```

NAME	READY	STATUS	RESTARTS	AGE
guestbook-frontend-deployment-85595f5bf9-h8cff	1/1	Running	0	55m
guestbook-frontend-deployment-85595f5bf9-lw6tg	1/1	Running	0	55m

guestbook-frontend-deployment-85595f5bf9-wpgc8	1/1	Running	0	55m
redis-leader-deployment-64fb8775bf-kbs6s	1/1	Running	0	55m
redis-follower-deployment-84cd76b975-jrn8v	1/1	Running	0	55m
redis-follower-deployment-69df9b5688-zml4f	1/1	Running	0	55m

The persistent volumes are attached to the Redis pods. Because we are backing up these stateful pods with Restic, we need to add annotations to the stateful pods with the `volumeMount` name.

You need to know the `volumeMount` to annotate the stateful pod. To get the `mountName`, run the following command.

```
kubectl describe pod redis-leader-deployment-64fb8775bf-kbs6s -n guestbook
```

In the results you see `Containers.leader.Mounts: /data from redis-leader-data`. This last token is the `volumeMount` name to use for the leader pod annotation. For the follower it will be `redis-follower-data`. You can also obtain the `volumeMount` name from the source YAML.

Annotate each of the Redis pods, for example:

```
kubectl -n guestbook annotate pod redis-leader-64fb8775bf-kbs6s backup.velero.io/backup-volumes=redis-leader-data
```

You should see the following:

```
pod/redis-leader-64fb8775bf-kbs6s annotated
```

Verify the annotations:

```
kubectl -n guestbook describe pod redis-leader-64fb8775bf-kbs6s | grep Annotations
Annotations:  backup.velero.io/backup-volumes: redis-leader-data
```

```
kubectl -n guestbook describe pod redis-follower-779b6d8f79-5dphr | grep Annotations
Annotations:  backup.velero.io/backup-volumes: redis-follower-data
```

Perform the Velero backup:

```
velero backup create guestbook-backup --include-namespaces guestbook
```

You should see the following:

```
Backup request "guestbook-backup" submitted successfully.
Run `velero backup describe guestbook-pv-backup` or `velero backup logs guestbook-pv-backup`
for more details.
```

Verify the backup that was created.

```
velero backup get
```

NAME	STATUS	ERRORS	WARNINGS	CREATED
EXPIRES	STORAGE	LOCATION	SELECTOR	
guestbook-backup	Completed	0	0	2020-07-23 16:13:46 -0700 PDT
29d	default	<none>		

Verify backup details.

```
velero backup describe guestbook-backup --details
```

Note that Velero lets you run other commands, such as:

```
kubectl get backups.velero.io -n velero
```

NAME	AGE
guestbook-backup	4m58s

And:

```
kubectl describe backups.velero.io guestbook-backup -n velero
```

Restore a Stateful Application Running on a Tanzu Kubernetes Cluster

Restoring a stateful application running on a Tanzu Kubernetes cluster involves restoring both the application metadata and the application data stored to a persistent volume. To do this you need both Velero and Restic.

This example assumes that you backed up the stateful Guestbook application as described in the previous section.

To test the restoration of the stateful application, delete its namespace:

```
kubectl delete ns guestbook
namespace "guestbook" deleted
```

Verify application deletion:

```
kubectl get ns
kubectl get pvc,pv --all-namespaces
```

Restore the application:

```
Restore request "guestbook-backup-20200723161841" submitted successfully.
Run `velero restore describe guestbook-backup-20200723161841` or `velero restore logs
guestbook-backup-20200723161841` for more details.
```

Verify that the stateful Guestbook application is restored:

```
velero restore describe guestbook-backup-20200723161841

Name:          guestbook-backup-20200723161841
Namespace:     velero
Labels:        <none>
Annotations:   <none>

Phase:  Completed

Backup:  guestbook-backup

Namespaces:
  Included:  all namespaces found in the backup
  Excluded:  <none>

Resources:
  Included:  *
  Excluded:  nodes, events, events.events.k8s.io, backups.velero.io,
restores.velero.io, resticrepositories.velero.io
  Cluster-scoped:  auto

Namespace mappings:  <none>

Label selector:  <none>

Restore PVs:  auto

Restic Restores (specify --details for more information):
  Completed:  3
```

Run the following additional command to verify restoration:

```
velero restore get
```

NAME	BACKUP	STATUS	ERRORS	WARNINGS
CREATED	SELECTOR			
guestbook-backup-20200723161841	guestbook-backup	Completed	0	0
2021-08-11 16:18:41 -0700 PDT	<none>			

Check that the namespace is restored:

```
kubectl get ns
```

NAME	STATUS	AGE
default	Active	16d
guestbook	Active	76s
...		
velero	Active	2d2h

Check that the application is restored:

```
vkubectl get all -n guestbook
```

NAME	READY	STATUS	RESTARTS	AGE
pod/frontend-6cb7f8bd65-h2pnb	1/1	Running	0	6m27s
pod/frontend-6cb7f8bd65-kwlpr	1/1	Running	0	6m27s
pod/frontend-6cb7f8bd65-snw14	1/1	Running	0	6m27s
pod/redis-leader-64fb8775bf-kbs6s	1/1	Running	0	6m28s
pod/redis-follower-779b6d8f79-5dphr	1/1	Running	0	6m28s
pod/redis-follower-899c7e2z65-8apnk	1/1	Running	0	6m28s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP
service/guestbook-frontend	LoadBalancer	10.10.89.59	10.19.15.99
service/redis-follower	ClusterIP	10.111.163.189	<none>
service/redis-leader	ClusterIP	10.111.70.189	<none>

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/guestbook-frontend-deployment	3/3	3	3	65s
deployment.apps/redis-follower-deployment	1/2	2	1	65s
deployment.apps/redis-leader-deployment	1/1	1	1	65s

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/guestbook-frontend-deployment-56fc5b6b47	3	3	3	65s
replicaset.apps/redis-follower-deployment-6fc9cf5759	2	2	1	65s
replicaset.apps/redis-leader-deployment-7d89bbdbcf	1	1	1	65s

Check that the persistent volumes are restored:

```
kubectl get pvc,pv -n guestbook
```

NAME	STATUS			
VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
persistentvolumeclaim/redis-leader-claim	Bound	pvc-a2f6e6d4-42db-4fb8-a198-5379a2552509	2Gi	RWO
		thin-disk	2m40s	
persistentvolumeclaim/redis-follower-claim	Bound	pvc-55591938-921f-452a-b418-2cc680c0560b	2Gi	RWO
		thin-disk	2m40s	

NAME	CAPACITY	ACCESS MODES	RECLAIM

POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
persistentvolume/pvc-55591938-921f-452a-b418-2cc680c0560b			2Gi	RWO	
Delete	Bound	guestbook/redis-follower-claim	thin-disk		2m40s
persistentvolume/pvc-a2f6e6d4-42db-4fb8-a198-5379a2552509			2Gi	RWO	
Delete	Bound	guestbook/redis-leader-claim	thin-disk		2m40s

Lastly, access the Guestbook frontend using the external IP of the guestbook-frontend service and verify that the messages you submitted at the beginning of the tutorial are restored. For example:

← → ↻
🛡️ 🔒 10.1.1.4.7

Guestbook

Messages

Submit

message 1
message 2
message 3

Backup and Restore vCenter Server

This topic describes how to back up and restore the vCenter Server in context of a vSphere with Tanzu deployment.

vCenter Cluster HA

To support a highly available Supervisor Cluster, vSphere with Tanzu requires that the vCenter cluster where the vSphere with Tanzu is enabled be configured with high-availability. For more information, see [vSphere Availability](#) in the VMware vSphere documentation.

vCenter Server Backup and Recover

vCenter Server supports file backup to network attached storage. To backup and restore the vCenter Server, create a backup of the primary vCenter Server. For more information, see [File-Based Backup and Restore of vCenter Server](#) in the vCenter documentation.

Backup and Restore NSX-T Data Center

To support network functionality in the case of a vSphere with Tanzu workload outage, back up and restore NSX-T Data Center.

Requirements

To backup and restore NSX-T Data Center, it is assumed that 3 NSX Manager Nodes are deployed, and there is an HA VIP configured for access to the NSX Management Plane. In addition, there are at least 2 Edge Nodes deployed with an HA VIP for the Edge Nodes. For more information, see [Configuring NSX for vSphere with Tanzu](#).

NSX-T Backup and Restore

NSX-T Data Center provides in-product backup and recovery that supports backup and restore of the NSX Manager Nodes and objects. For more information, see [Backing Up and Restoring NSX Manager](#) in the NSX-T documentation.

Troubleshooting vSphere with Tanzu

19

Use the following troubleshooting techniques and best practices for your infrastructure in vSphere with Tanzu.

Read the following topics next:

- [Storage Best Practices and Troubleshooting](#)
- [Troubleshooting Networking](#)
- [Troubleshooting the NSX Advanced Load Balancer](#)
- [Troubleshooting Network Topology Upgrade](#)
- [Troubleshooting Tanzu Kubernetes Clusters](#)
- [Troubleshooting Workload Management](#)

Storage Best Practices and Troubleshooting

You can use best practices and troubleshooting techniques for your storage environment in vSphere with Tanzu.

Use Anti-Affinity Rules for Control Plane VMs on Non-vSAN Datastores

When you use datastores other than vSAN in your cluster with vSphere with Tanzu, place the three control plane VMs on different datastores for availability reasons.

Because the control plane VMs are system managed, you cannot manually migrate them. Use a combination of a Datastore Cluster and Storage DRS to rebalance the control plane VMs and place them on separate datastores.

Procedure

- 1 In the vSphere Client, create a datastore cluster.
 - a Navigate to data centers.
 - b Right-click the data center object and select **New Datastore Cluster**.
 - c Name your datastore cluster and make sure **Turn ON Storage DRS** is enabled.
 - d Set the cluster automation level to **No Automation (Manual Mode)**.

- e Leave the Storage DRS runtime settings as default.
 - f Select the ESXi cluster enabled with vSphere with Tanzu.
 - g Select all shared datastores to add to the datastore cluster.
 - h Click **Finish**.
- 2 Define Storage DRS rules for control Plane VMs.
- a Navigate to the datastore cluster.
 - b Click the **Configure** tab and click **Rules** under **Configuration**.
 - c Click the **Add** icon and enter a name for the rule.
 - d Make sure **Enable rule** is enabled.
 - e Set **Rule type** to **VM anti-affinity**.
 - f Click the **Add** icon and select the three supervisor control plane VMs.
 - g Click **OK** to finish your configuration.
- 3 Create VM overrides.
- a Navigate to the datastore cluster.
 - b Click the **Configure** tab and click **VM Overrides** under **Configuration**.
 - c Click the **Add** icon and select the three control plane VMs.
 - d To enable the Storage DRS automation level, select the **Override** check box and set the value to **Fully Automated**.
 - e Click **Finish**.

Results

This task enables Storage DRS for the control plane VMs only and rebalances the VMs to be on different datastores.

Once the Storage vMotion takes place, you can remove the SDRS rules and overrides, disable Storage DRS, and remove the datastore cluster.

Storage Policy Removed from vSphere Continues to Appear as Kubernetes Storage Class

When you use the vSphere Client to remove the storage policy from VMware vCenter or a namespace in the Supervisor Cluster, its matching storage class remains in the Kubernetes environment, but cannot be used.

Problem

If you run the `kubectl get sc` command, the output continues to list the storage class as available in the namespace. However, the storage class cannot be used. For example, your attempts to use the storage class for a new persistent volume claim fail.

If the storage class is already used by a Kubernetes deployment, the deployment might behave unpredictably.

Solution

- 1 To verify which storage classes exist in the namespace, run the **kubectl describe namespace *namespace_name*** command.

The output for this command does not list the storage class if its matching storage policy is removed.

- 2 If the storage class is already used by a deployment, restore the storage class.
 - a Use the vSphere Client to create a new storage policy with the same name as the policy you removed.

For example, if you deleted the *Gold* policy, name the new policy *Gold*. See [Create Storage Policies for vSphere with Tanzu](#).

- b Assign the policy to the namespace.

See [Change Storage Settings on a Namespace](#).

After you assign the policy to the namespace, vSphere with Tanzu deletes the old storage class and creates a matching storage class with the same name.

Use External Storage with vSAN Direct

When you use vSAN Direct in the vSphere with Tanzu environment, you can use external shared storage to store management internal VMs and other metadata.

Problem

When you deploy a homogeneous vSAN Direct cluster, you must create a replicated vSAN datastore on each ESXi host in the cluster to store vSphere with Tanzu management VMs and other metadata. The vSAN datastore consumes space, requires an additional I/O controller on each host, and limits hardware configuration on which vSAN Direct can be supported.

Instead of configuring the vSAN datastore, you can use external shared storage for storing management internal VMs and other metadata.

Solution

- 1 If vSAN or vSAN Direct was deployed on the ESXi hosts in the cluster, clear the hosts from any configuration.
 - a Remove any disks assigned to vSAN or vSAN Direct. See [Remove Disk Groups or Devices from vSAN](#).
 - b (Optional) Use the script to tag disks on the hosts for vSAN Direct. See [Using Script to Tag Storage Devices for vSAN Direct](#).

2 Use VMware Cloud Foundation to create a workload domain with external storage.

Make sure to select one of the storage options, such as NFS, vVols, or FC. Only one of these options can be selected.

For more information, see *Working with Workload Domains* in the [VMware Cloud Foundation Operations and Administration Guide](#).

This step deploys a workload domain with vCenter Server and specified ESXi hosts. The external storage is mounted on all hosts and added to the default cluster.

3 Enable vSAN.

Make sure that no disks are claimed for vSAN.

For more information, see [Enable vSAN on an Existing Cluster](#).

This step creates a zero byte vSAN datastore with vSAN network. No local disks are used for vSAN.

4 Claim local disks on the hosts for vSAN Direct.

For information, see [Set Up vSAN Direct for vSphere with Tanzu](#).

For each device you claim, vSAN Direct creates a separate datastore.

5 Create storage policies for vSAN Direct.

For information, see [Create vSAN Direct Storage Policy](#).

6 Configure and enable Workload Management.

For information, see [Chapter 5 Configuring and Managing a Supervisor Cluster](#).

Example

In this example, a setup includes external NFS storage and a vSAN Direct datastore. Control plane VMs and vSphere Pods are running on external NFS storage. Persistent volume claims run on vSAN Direct.

The screenshot shows the vSphere Client interface. The top bar has a warning: "There are expired or expiring licenses in your inventory. MANAGE YOUR LICENSES". The main view is for a workload domain named "sample-1-0". The "Datastores" tab is selected, showing a table with two datastores:

Name	Status	Type	Datastore Cl...	Capacity	Free
nfs0-1	✓ Normal	NFS 3		295.17 GB	263.03 GB
vSANDirect	✓ Normal	vSAN Direct		199.75 GB	197.34 GB

The left sidebar shows the navigation tree with "sample-1-0" selected. The bottom right of the table has an "Export" button and "2 items" text.

Troubleshooting Networking

You can troubleshoot networking problems that you might encounter when you configure vSphere with Tanzu with NSX.

Register vCenter Server with NSX Manager

You might need to re-register vCenter Server OIDC with NSX Manager in certain situations, for example when the FQDN/PNID of vCenter Server changes.

Procedure

- 1 Connect to the vCenter Server Appliance through SSH.
- 2 Run the command `shell`.
- 3 To get the vCenter Server thumbprint, run the following command:

```
- openssl s_client -connect vcenterserver-FQDN:443 </dev/null 2>/dev/null | openssl x509
-fingerprint -sha256 -noout -in /dev/stdin
```

The thumbprint is displayed. For example,

```
08:77:43:29:E4:D1:6F:29:96:78:5F:BF:D6:45:21:F4:0E:3B:2A:68:05:99:C3:A4:89:8F:F2:0B
:EA:3A:BE:9D
```

- 4 Copy the SHA256 thumbprint and remove colons.

```
08774329E4D16F2996785FBFD64521F40E3B2A680599C3A4898FF20BEA3ABE9D
```

- 5 To update the OIDC of vCenter Server, run the following command:

```
curl --location --request POST 'https://<NSX-T_ADDRESS>/api/v1/trust-management/oidc-uris' \
  --header 'Content-Type: application/json' \
  --header 'Authorization: Basic <AUTH_CODE>' \
  --data-raw '{
    "oidc_type": "vcenter",
    "oidc_uri": "https://<VC_ADDRESS>/openidconnect/vsphere.local/.well-known/openid-configuration",
    "thumbprint": "<VC_THUMBPRINT>"
  }'
```

Unable to Change NSX Appliance Password

You might be unable to change the NSX appliance password for the `root`, `admin`, or `audit` users.

Problem

Attempts to change the NSX appliance password for the `root`, `admin`, or `audit` users through the vSphere Client may fail.

Cause

During the installation of the NSX Manager, the procedure only accepts one password for all three roles. Attempts to change this password later may fail.

Solution

- ◆ Use the NSX APIs to change the passwords.

For more information, see <https://kb.vmware.com/s/article/70691> and the *NSX Administration Guide*.

Troubleshooting Failed Workflows and Unstable NSX Edges

If your workflows fail or the NSX Edges are unstable, you can perform troubleshooting steps.

Problem

When you change the distributed port group configuration on the vSphere Client, workflows might fail and the NSX Edges might become unstable.

Cause

Removal or modification of the distributed port groups for overlay and uplink that were created during the NSX Edge cluster setup of cluster configuration, is not allowed by design.

Solution

If you require to change the VLAN or IP Pool configuration of NSX Edges, you must first remove elements of NSX and the vSphere with Tanzu configuration from the cluster.

For information about removing elements of NSX, see the *NSX Installation Guide*.

Collect Support Bundles for Troubleshooting NSX-T

You can collect support bundles on registered cluster and fabric nodes for troubleshooting and download the bundles to your machine or upload them to a file server.

If you choose to download the bundles to your machine, you get a single archive file consisting of a manifest file and support bundles for each node. If you choose to upload the bundles to a file server, the manifest file and the individual bundles are uploaded to the file server separately.

Procedure

- 1 From your browser, log in with admin privileges to an NSX Manager.
- 2 Select **System > Support Bundle**.
- 3 Select the target nodes.

The available types of nodes are **Management Nodes**, **Edges**, **Hosts**, and **Public Cloud Gateways**.

- 4 (Optional) Specify log age in days to exclude logs that are older than the specified number of days.
- 5 (Optional) Toggle the switch that indicates whether to include or exclude core files and audit logs.

Note Core files and audit logs might contain sensitive information such as passwords or encryption keys.

- 6 (Optional) Select the check box to upload the bundles to a file server.
- 7 Click **Start Bundle Collection** to start collecting support bundles.
The number of log files for each node determines the time taken for collecting support bundles.
- 8 Monitor the status of the collection process.
The **Status** tab shows the progress of collecting support bundles.
- 9 Click **Download** to download the bundle if the option to send the bundle to a file server was not set.

Collect Log Files for NSX-T

You can collect logs that are in the vSphere with Tanzu and NSX components to detect and troubleshoot errors. The log files might be requested by VMware Support.

Procedure

- 1 Log in to the vCenter Server using the vSphere Client .
- 2 Collect the following log files.

Log File	Description
<code>/var/log/vmware/wcp/wcpsvc.log</code>	Contains information related to vSphere with Tanzu enablement.
<code>/var/log/vmware/wcp/nsxd.log</code>	Contains information related to the NSX components configuration.

- 3 Log in to NSX Manager.
- 4 Collect the `/var/log/proton/nsxapi.log` for information on the error that the NSX Manager returns when a specific vSphere with Tanzu operation has failed.

Restart the WCP Service If the NSX-T Management Certificate, Thumbprint, or IP Address Changes

If the NSX-T Management certificate, thumbprint or IP address changes after you have installed vSphere with Tanzu, you must restart the `WCP` service.

Restart the vSphere with Tanzu Service If the NSX-T Certificate Changes

Currently, vSphere with Tanzu requires that if the NSX-T certificate or thumbprint, or if the NSX-T IP address changes, you must restart the `wcp` service for the change to take effect. If either change occurs without a restart of the service, communication between vSphere with Tanzu and NSX-T fails and certain symptoms can arise, such as NCP entering into CrashLoopBackoff stage or Supervisor Cluster resources becoming undeployable.

To restart the `wcp` service, use the `vmon-cli`.

- 1 SSH to the vCenter Server and log in as the root user.
- 2 Run the command `shell`.
- 3 Run the command `vmon-cli -h` to view usage syntax and options.
- 4 Run the command `vmon-cli -l` to view the `wcp` process.
You see the `wcp` service at the bottom of the list.
- 5 Run the command `vmon-cli --restart wcp` to restart the `wcp` service.
You see the message `Completed Restart service request`.
- 6 Run the command `vmon-cli -s wcp` and verify that the `wcp` service is started.

For example:

```
root@localhost [ ~ ]# vmon-cli -s wcp
Name: wcp
Starttype: AUTOMATIC
RunState: STARTED
RunAsUser: root
CurrentRunStateDuration(ms): 22158
HealthState: HEALTHY
FailStop: N/A
MainProcessId: 34372
```

VDS Required for Host Transport Node Traffic

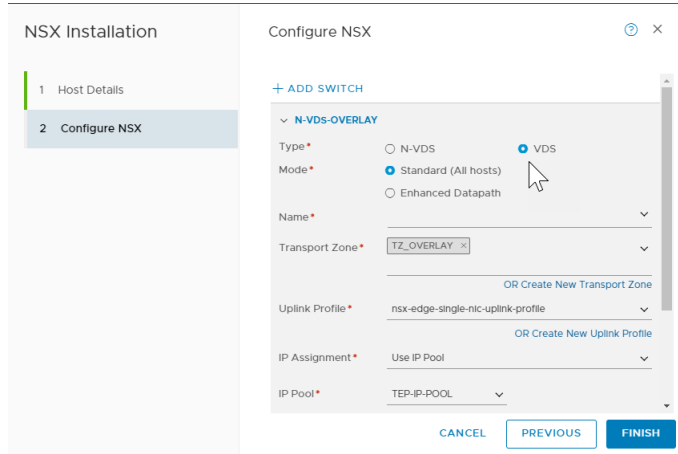
vSphere with Tanzu requires the use of a vSphere 7 Virtual Distributed Switch (VDS) for host transport node traffic. You cannot use the NSX-T VDS (N-VDS) for host transport node traffic with vSphere with Tanzu.

VDS Is Required

vSphere with Tanzu requires a converged VDS that supports both vSphere 7 traffic and NSX-T 3 traffic on the same VDS. With previous releases of vSphere and NSX-T, you have one VDS (or VSS) for vSphere traffic and one N-VDS for NSX-T traffic. This configuration is not supported by vSphere with Tanzu. If you attempt to enable Workload Management using an N-VDS, the system reports that the vCenter cluster is not compatible. For more information, see [Troubleshoot Workload Management Enablement Cluster Compatibility Errors](#).

To use a converged VDS, create a vSphere 7 VDS using vCenter and in NSX-T specify this VDS when preparing the ESXi hosts as transport nodes. Just having VDS-DSwitch on the vCenter side is not sufficient. The VDS-DSwitch 7.0 has to be configured with NSX-T transport node profile as documented in the topic [Create a Transport Node Profile](#) and shown below.

Figure 19-1. VDS Configuration in NSX-T



If you have upgraded to vSphere 7 and NSX-T 3 from previous versions, you must uninstall the N-VDS from each ESXi transport node and reconfigure each host with a VDS. Contact VMware Global Support Service for guidance.

Troubleshooting the NSX Advanced Load Balancer

You can troubleshoot NSX Advanced Load Balancer problems that you might encounter by collecting support bundles.

Collect Support Bundles for Troubleshooting

To troubleshoot NSX Advanced Load Balancer problems, you can collect support bundles. The support bundles might be requested by VMware Support.

When you generate the support bundle, you get a single file for the debug logs that you can download.

Procedure

- 1 In the Avi Controller dashboard, click the menu in the upper-left hand corner and select **Administration**.
- 2 In the **Administration** section, select **System**.
- 3 In the **System** screen, select **Tech Support**.
- 4 To generate a diagnostics bundle, click **Generate Tech Support**.
- 5 In the **Generate Tech Support** window, select **Debug Logs** type and click **Generate**.

- 6 Once the bundle is generated, click the download icon to download it to your machine.

Troubleshooting Network Topology Upgrade

When you install vSphere with Tanzu version 7.0 Update 1c or upgrade the Supervisor Cluster from version 7.0 Update 1 to version 7.0 Update 1c, the networking topology is upgraded from a single tier-1 gateway topology to a topology that has a tier-1 gateway for each namespace within the Supervisor Cluster.

You can troubleshoot problems that you might encounter during the upgrade.

Upgrade Precheck Fails Due to Insufficient Edge Load Balancer Capacity

The upgrade precheck fails and the error message indicates insufficient load balancer capacity.

Problem

The upgrade precheck process fails with an error message that indicates that the load balancer capacity is less than the capacity required by the Supervisor Cluster.

Solution

Perform one of the following steps to resolve the problem:

- Force the upgrade by clicking on the **Force Upgrade** button in the error message or use the vCenter Server command line with the `--ignore-precheck-warnings true` flag.

Note This solution is recommended only if the Edge cluster can support the existing namespace workloads. Otherwise, these workloads might be skipped during upgrade.

- Delete unused workloads.
- Add additional Edge nodes to the cluster.

Supervisor Cluster Workload Namespaces Skipped During Upgrade

During the Supervisor Cluster upgrade some namespace workloads are not upgraded.

Problem

The Supervisor Cluster upgrade succeeds but some namespace workloads are skipped during the upgrade. The Kubernetes resources indicate insufficient resources and the newly created tier-1 gateway is in `ERROR` state.

Cause

The load balancer capacity is insufficient to support the workloads.

Solution

Perform one of the following steps to resolve the problem:

- Delete unused workloads, restart NCP, and run the upgrade again.
- Add additional Edge nodes to the cluster and trigger a re-allocation to the tier-1 gateway. Restart NCP and run the upgrade again.

Load Balancer Service Skipped During Upgrade

During the Supervisor Cluster upgrade some load balancer services are not upgraded.

Problem

The Supervisor Cluster upgrade succeeds but some Kubernetes load balancer services are skipped during the upgrade.

Cause

The number of Kubernetes load balancer type services in the Supervisor Cluster workloads and associated Tanzu Kubernetes cluster exceed the NSX Edge virtual server limit.

Solution

Delete unused workloads, restart NCP, and run the upgrade again.

Troubleshooting Tanzu Kubernetes Clusters

You can troubleshoot Tanzu Kubernetes clusters by connecting to any cluster node, viewing the cluster resource hierarchy, and collecting log files.

Collect a Support Bundle for Tanzu Kubernetes Clusters

To troubleshoot Tanzu Kubernetes cluster errors, you can run a utility to collect a diagnostic log bundle.

VMware provides the TKC Support Bundler utility that you can use to collect Tanzu Kubernetes cluster log files and troubleshoot problems.

To obtain and use the utility, refer to the article [Gathering Logs for vSphere with Tanzu](#) at the VMware Support Knowledge Base.

Troubleshoot vCenter Single Sign-On Connection Errors

If you do not have sufficient permissions on the vSphere Namespace, you cannot connect to the Supervisor Cluster or to a Tanzu Kubernetes cluster as a vCenter Single Sign-On user.

Problem

The vSphere Plugin for kubectl returns the error message `Error from server (Forbidden)` when you attempt to connect to a Supervisor Cluster or a Tanzu Kubernetes cluster as a vCenter Single Sign-On user.

Cause

You do not have sufficient permissions on the vSphere Namespace, or do not have cluster access.

Solution

If you are a DevOps engineer who operates the cluster, verify with your vSphere administrator that you have been granted **Edit** permissions for the vSphere Namespace. See [Create and Configure a vSphere Namespace](#).

If you are a developer who is using the cluster to deploy workloads, verify with your cluster administrator that you have been granted cluster access. See [Grant Developer Access to Tanzu Kubernetes Clusters](#).

Troubleshoot Subscribed Content Library Errors

If the Subscribed Content Library reaches storage capacity limits, you cannot provision Tanzu Kubernetes clusters.

Problem

When attempting to provision a Tanzu Kubernetes cluster, you are unable to pull items from the Subscribed Content Library, and the following error appears:

```
Internal error occurred: get library items failed for.
```

Cause

The Subscribed Content Library has reached capacity. The Content Library is backed by attached storage. Over time, as more Kubernetes versions are released and OVA files are pulled into the library, the storage might fill to capacity.

Solution

Migrate to a new Content Library. See [Migrate Tanzu Kubernetes Clusters to a New Content Library](#).

Troubleshoot Local Content Library Errors

Refer to this topic to troubleshoot common local content library errors.

TKR Cannot Be Found in the Supervisor Cluster

Local content libraries can be used in internet-restricted environments. To create a local content library, see [Create, Secure, and Synchronize a Local Content Library for Tanzu Kubernetes releases](#).

On applying the Security Policy to the Content Library, the Tanzu Kubernetes release (TKR) in the Supervisor Cluster will not show up if one of the following conditions exist.

- The OVF package in the content library are not signed.
- The OVF package is signed using an invalid certificate.
- The OVF package is signed using a certificate not trusted by the vCenter Server where the Local Content Library is configured.

Troubleshoot Cluster Provisioning Errors

If you have provisioned a Tanzu Kubernetes cluster, but the control plane virtual machines do not start, you might need to change the virtual machine size or class.

Problem

You have provisioned a Tanzu Kubernetes cluster. The system is attempting to power on the control plane virtual machine(s), but it errors out with the following message:

```
The host does not have sufficient CPU resources to satisfy the reservation.
```

Cause

The virtual machine size or class is not sufficient for the cluster deployment.

Solution

Change the virtual machine type or class. See [Virtual Machine Classes for Tanzu Kubernetes Clusters](#).

Troubleshoot Workload Deployment Errors

Workload deployment errors might occur if PodSecurityPolicy and bindings are not configured for authenticated users.

Problem

You deploy a container workload to a Tanzu Kubernetes cluster but the workload does not start. You see an error similar to the following:

```
Error: container has runAsNonRoot and image will run as root.
```

Cause

Tanzu Kubernetes clusters are provisioned with the PodSecurityPolicy Admission Controller enabled. No authenticated users can create privileged or unprivileged pods until the cluster administrator binds PodSecurityPolicy to the authenticated users.

Solution

Create an appropriate binding to default PodSecurityPolicy, or define custom PodSecurityPolicy. See [Using Pod Security Policies with Tanzu Kubernetes Clusters](#) and [Tanzu Kubernetes Guestbook Tutorial](#).

Troubleshoot Virtual Machine Class Errors

Virtual machine classes must be bound to the vSphere Namespace for Tanzu Kubernetes clusters and VM classes.

Virtual Machine Class Binding Error

Virtual machine classes must be bound to the vSphere Namespace. See [Associate a VM Class with a Namespace in vSphere with Tanzu](#). If you do not associate the VM class with the namespace, you receive the error `VirtualMachineClassBindingNotFound` similar to the following:

```
conditions:
- lastTransitionTime: "2021-04-25T02:50:58Z"
  message: 1 of 2 completed
  reason: VirtualMachineClassBindingNotFound @ Machine/test-cluster
  severity: Error
  status: "False"
  type: ControlPlaneReady
- lastTransitionTime: "2021-04-25T02:49:21Z"
  message: 0/1 Control Plane Node(s) healthy. 0/2 Worker Node(s) healthy
  reason: WaitingForNodesHealthy
  severity: Info
  status: "False"
  type: NodesHealthy
```

Restart a Failed Tanzu Kubernetes Cluster Update Job

If the update of a Tanzu Kubernetes cluster fails, you can restart the update job and try the update again.

Problem

The update of a Tanzu Kubernetes cluster fails and results in cluster status of `upgradefailed`.

Cause

There can be a number of reasons for a failed cluster update, such as insufficient storage. To restart a failed update job and try the cluster update again, complete the following procedure.

Solution

- 1 Log in to the Supervisor Cluster as an administrator. See [Connect to the Supervisor Cluster as a vCenter Single Sign-On User](#).
- 2 Look up the `update_job_name`.

```
kubectl get jobs -n vmware-system-tkg -l "run.tanzu.vmware.com/cluster-namespace=${cluster_namespace},cluster.x-k8s.io/cluster-name=${cluster_name}"
```

- 3 Run `kubectl proxy` so that `curl` can be used to issue requests.

```
kubectl proxy &
```

You should see `Starting to serve on 127.0.0.1:8001`.

Note You cannot use `kubectl` to patch or update the `.status` of a resource.

- 4 Using `curl` issue the following patch command to raise the `.spec.backoffLimit`.

```
curl -H "Accept: application/json" -H "Content-Type: application/json-patch+json"
--request PATCH --data ' [{"op": "replace", "path": "/spec/backoffLimit", "value": 8}] '
http://127.0.0.1:8001/apis/batch/v1/namespaces/vmware-system-tkg/jobs/${update_job_name}
```

- 5 Using `curl` issue the following patch command to clear the `.status.conditions` so that the Job controller will create new pods.

```
$ curl -H "Accept: application/json" -H "Content-Type: application/json-patch+json"
--request PATCH --data ' [{"op": "remove", "path": "/status/conditions"}] '
http://127.0.0.1:8001/apis/batch/v1/namespaces/vmware-system-tkg/jobs/${update_job_name}/
status
```

Troubleshooting Workload Management

Refer to this section for troubleshooting Workload Management or if you need to collect a support bundle for the .

Collect the Support Bundle for Workload Management

Checkout how to collect a support bundle for a Supervisor Cluster. You can collect a support bundle even in the Supervisor Cluster is in an error or configuring state.

Prerequisites

- Your user account must have the **Global.Diagnostics** privilege.

Procedure

- 1 Log in to your vSphere with Tanzu environment using the vSphere Client.
- 2 Select **Menu > Workload Management**.

- 3 Select the **Clusters** tab.
- 4 Select the target Supervisor Cluster
- 5 Click **Export Logs**.

Results

Once you have collected the support bundle, refer to the following KB article: Uploading diagnostic information for VMware through the Secure FTP portal: <http://kb.vmware.com/kb/2069559>.

Tail the Workload Management Log File

Tailing the Workload Management log file can help troubleshoot enablement problems and Supervisor Cluster deployment errors.

Solution

- 1 Establish an SSH connection to the vCenter Server Appliance.
- 2 Log in as the `root` user.
- 3 Run the command `shell`.

You see the following:

```
Shell access is granted to root
root@localhost [ ~ ]#
```

- 4 Run the following command to tail the log.

```
tail -f /var/log/vmware/wcp/wcpsvc.log
```

Troubleshoot Workload Management Enablement Cluster Compatibility Errors

Follow these troubleshooting tips if the system indicates that your vSphere cluster is incompatible for enabling Workload Management.

Problem

The **Workload Management** page indicates that your vCenter cluster is incompatible when you try to enable Workload Management.

Cause

There can be multiple reasons for this. First, make sure that your environment meets the minimum requirements for enabling Workload Management:

- Valid license: VMware vSphere 7 Enterprise Plus with Add-on for Kubernetes
- At least two ESXi hosts

- Fully automated DRS
- vSphere HA
- vSphere Distributed Switch 7.0
- Sufficient storage capacity

If your environment meets these prerequisites, but the target vCenter cluster is not compatible, use the VMware Datacenter CLI (DCLI) to help identify the problems.

Solution

- 1 SSH to vCenter Server.
- 2 Log in as the root user.
- 3 Run the command `dcli` to list the VMware Datacenter CLI help.
- 4 List the available vCenter clusters by running the following DCLI command.

```
dcli com vmware vcenter cluster list
```

For example:

```
dcli +username VI-ADMIN-USER-NAME +password VI-ADMIN-PASSWORD com vmware vcenter cluster list
```

Example result:

```
|-----|-----|-----|-----|
|drs_enabled|cluster |name      |ha_enabled|
|-----|-----|-----|-----|
|True       |domain-d7|vSAN Cluster|True      |
|-----|-----|-----|-----|
```

- 5 Verify vCenter cluster compatibility by running the following DCLI command.

```
dcli com vmware vcenter namespacemanagement clustercompatibility list
```

For example:

```
dcli +username VI-ADMIN-USER-NAME +password VI-ADMIN-PASSWORD com vmware vcenter namespacemanagement clustercompatibility list
```

The following example result indicates that the environment is missing a compatible NSX-T VDS switch.

```
|-----|-----|-----|-----|
|-----|
|cluster |compatible|
|incompatibility_reasons                                     |
|-----|-----|-----|-----|
```



```

-----|
|domain-d7|False      |Failed to list all distributed switches in vCenter 2b1c1fa5-
e9d4-45d7-824c-fa4176da96b8.|
|          |          |Cluster domain-d7 is missing compatible NSX-T
VDS.                                     |
|-----|-----|-----|-----|-----|-----|-----|-----|
-----|

```

- 6 Run additional DCLI commands as necessary to determine additional compatibility issues. In addition to NSX-T errors, other common reasons for incompatibility are DNS and NTP connectivity problems.
- 7 To troubleshoot further, complete the following steps.
 - a Tail the `wcpsvc.log` file. See [Tail the Workload Management Log File](#).
 - b Navigate to the **Workload Management** page and click **Enable**.

Shut Down and Start Up the vSphere with Tanzu Workload Domain

To avoid data loss and maintain the components and workloads of your vSphere with Tanzu environment operational, you must follow the specified order when shutting down or starting up the components.

Typically, you perform the shutdown and startup operations after you apply a patch, upgrade, or restore your vSphere with Tanzu environment.

The vSphere with Tanzu solution, including Tanzu Kubernetes clusters provisioned by the Tanzu Kubernetes Grid Service, is part of the vSphere software defined data center (SDDC). As such, you must take into consideration the entire vSphere infrastructure stack when performing a shut down and start up of your vSphere with Tanzu environment. Refer to the following validated set of procedures for shutdown and start up of the vSphere SDDC including vSphere with Tanzu:

- vSphere SDDC including vSphere with Tanzu [shut down procedure](#)
- vSphere SDDC including vSphere with Tanzu [start up procedure](#)