Tanzu Kubernetes Grid Air-Gapped Reference Design and Deployment

Tanzu Kubernetes Grid Air-Gapped Reference Design and Deployment 2.3

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

https://techdocs.broadcom.com/

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

Contents

VMware Tanzu Kubernetes Grid 2.3 Air-Gapped Reference Design and Deployment

You can deploy Tanzu Kubernetes Grid management clusters and Tanzu Kubernetes Grid workload clusters in air-gapped environments that do not have a physical connection to the Internet.

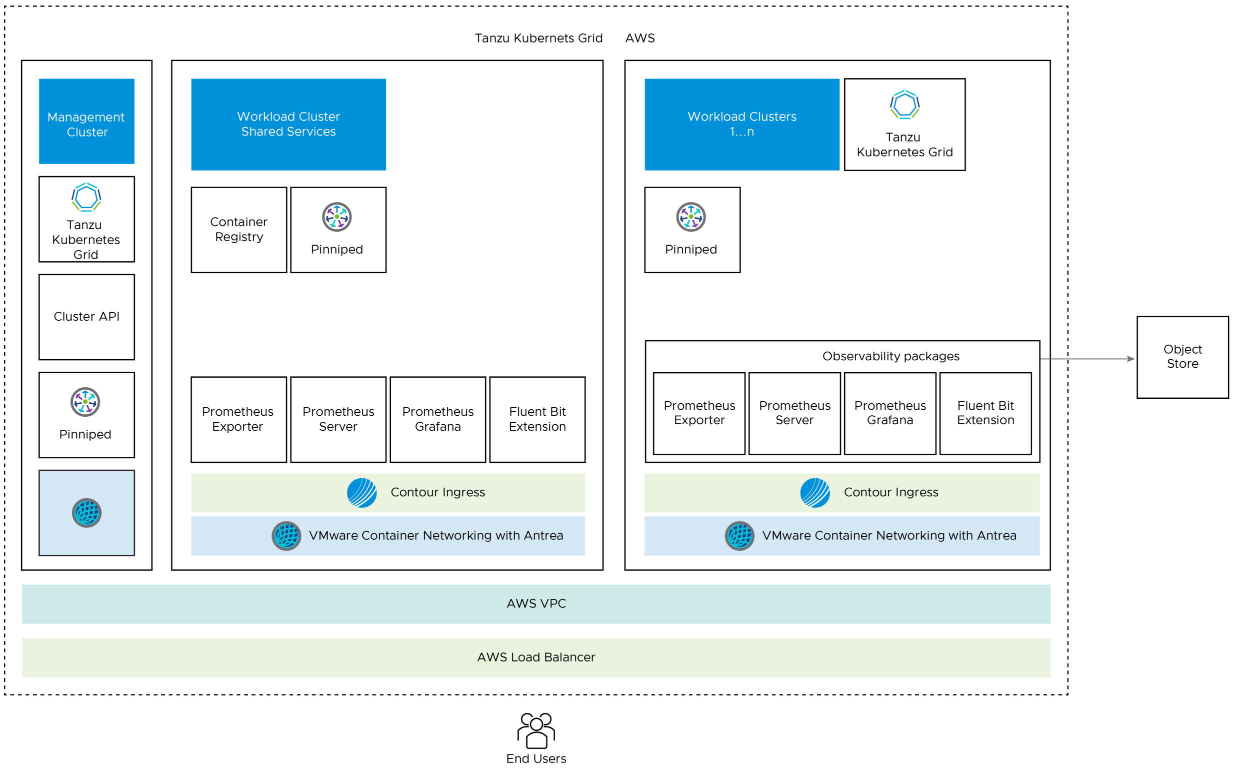
This documentation provides reference designs for deploying VMware Tanzu Kubernetes Grid (informally known as TKG) in an air-gapped environment. The documentation also includes the steps to deploy VMware Tanzu Kubernetes Grid in an air-gapped environment on AWS.

The VMware Tanzu Kubernetes Grid Air-Gapped Reference Design document provides steps to deploy the Tanzu Kubernetes Grid management and workload clusters in a specific and validated configuration in air-gapped environments. For the generic steps to deploy the management and workload clusters on air-gapped environments, see VMware Tanzu Kubernetes Grid Documentation.

VMware Tanzu Kubernetes Grid on AWS Air-Gapped Reference Design

VMware Tanzu Kubernetes Grid (multi-cloud) provides a consistent, upstream-compatible, and regional Kubernetes substrate that is ready for end-user workloads and ecosystem integrations.

This document lays out a reference design for deploying VMware Tanzu for Kubernetes Grid on AWS Networking in an air-gapped environment with Tanzu components on AWS. An air-gapped environment is a network security measure employed to ensure that a computer or computer network is secure by physically isolating it from unsecured networks, such as the public Internet or an unsecured local area network. This reference design is based on the architecture and components described in VMware Tanzu Kubernetes Grid Reference Architecture 2.3.

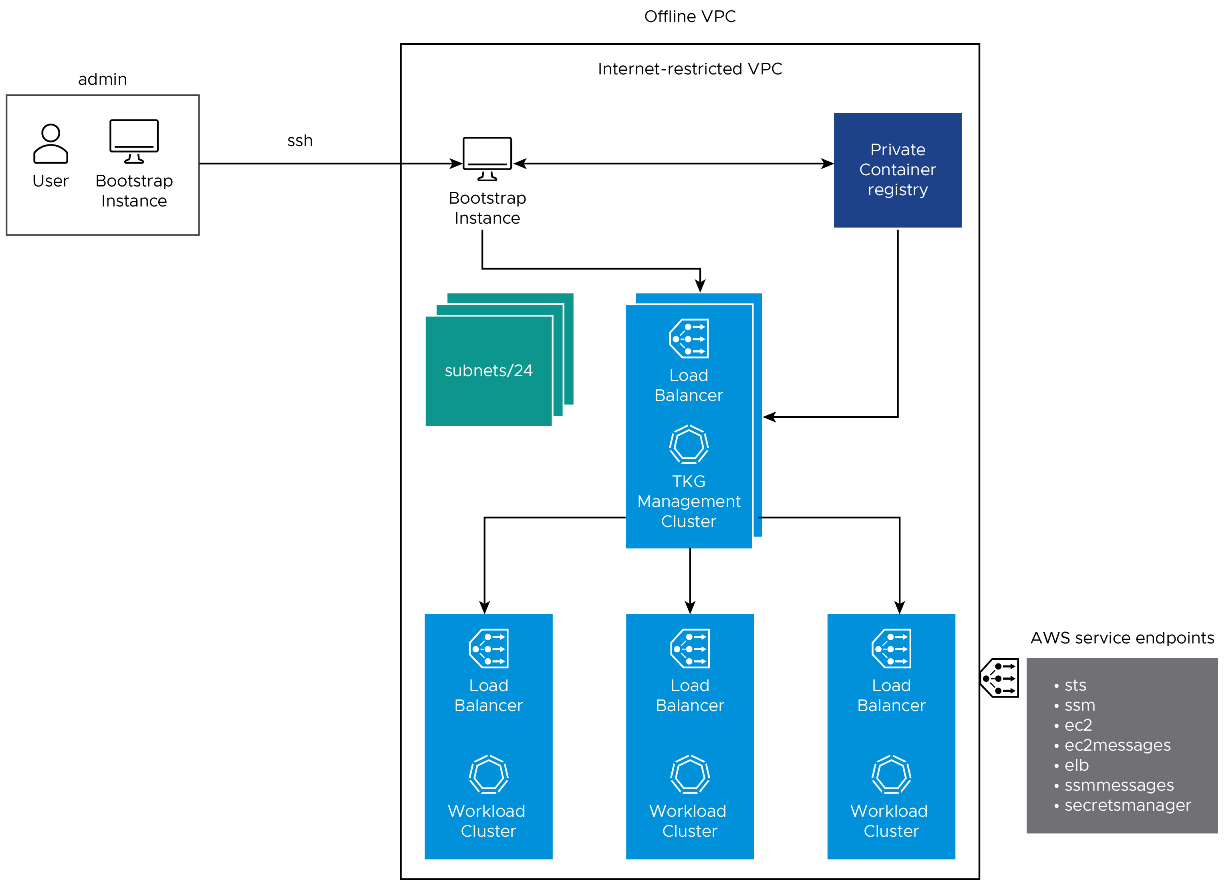


Tanzu Kubernetes Grid Infrastructure Network Overview

The following network diagram shows the network layout used with this reference design. It shows the layout for a single virtual private cloud (VPC). The network layout uses the following:

- One private subnet for each AWS availability zone (AZ). Each subnet is allocated a private IP address.

- A bootstrap VM running within your Internet-restricted (offline) VPC to install Tanzu Kubernetes Grid.
- A private Docker-compatible container registry such as Harbor, Docker, or Artifactory installed and configured. This registry runs outside of Tanzu Kubernetes Grid and is separate from any registry deployed as a shared service for clusters.



Network Recommendations

This reference design uses Tanzu Kubernetes Grid to manage the lifecycle of multiple Kubernetes workload clusters by bootstrapping a Kubernetes management cluster with the Tanzu command-line tool. Consider the following when configuring the network for Tanzu Kubernetes Grid:

- Create Internet-restricted VPCs with no Internet gateway (offline VPCs) for Tanzu Kubernetes Grid management and workload clusters. The Administrator/Operator must be able to access/ssh into Internet-restricted (offline) VPCs.
- Create an AWS Transit Gateway for a network architecture with multiple VPCs with multiple Availability Zones. The AWS Transit Gateway connects all your VPCs and on-premises networks through a central hub. This simplifies your network and avoids complex peering relationships. The AWS Transit Gateway acts as a cloud router – each new connection is made only once.
- Use an internal load balancer scheme. A best practice is to create an internal load balancer to avoid exposing the Kubernetes API server to the public Internet. To use an internal load balancer, include the following setting in the cluster configuration file: `AWS_LOAD_BALANCER_SCHEME_INTERNAL: true`

If you use an internal load balancer, run Tanzu Kubernetes Grid from a machine with access to the target VPC private IP space.

- Beware that 172.17.0.0/16 is the default Docker subnet. If you are going to use that subnet for a VPC deployment, you must change the subnet for your Docker container.

Storage

Tanzu Kubernetes Grid ships with the AWS cloud storage driver, which allows you to provision stateful storage volumes in your Tanzu Kubernetes Grid cluster. The following storage classes are available:

- gp2 - General Purpose SSD (default storage class)
- io1 - IOPS provisioned SSD
- st1 - Throughput Optimized HHD
- sc1 - Cold HDD

For more information on available storage options, see Amazon EBS volume types.

VPC Architectures

In a production deployment, Tanzu Kubernetes Grid creates multiple Availability Zones (AZs).

VMware recommends that you create the VPCs before you deploy Tanzu Kubernetes Grid. Also, make sure that you tag a private subnet in each AZ, including the control plane cluster, with a key of `kubernetes.io/cluster/<cluster_name>`. As a best practice, ensure that the value you use for the private subnets for an AZ can easily identify the subnets as belonging to the same AZ. For example:

```
aws ec2 create-subnet --vpc-id $vpcId --cidr-block <ip_address> --availability-zone
${AWS_REGION}b --tag-specifications ‘ResourceType=subnet, Tags=[{Key=Name,Value=priv-b}]’
--output json > $WORKING_DIR/subnet-priv-b
```

All Internet-restricted VPCs must add the following endpoints to enable private connections between the VPCs and supported AWS services.

Service endpoints:

- sts
- ssm
- ec2
- ec2messages
- elasticloadbalancing
- secretsmanager
- ssmmessages
- s3 (optional; recommended)

Based on your application needs and desired outcomes, you can organize your workloads by using one of the following VPC architectures.

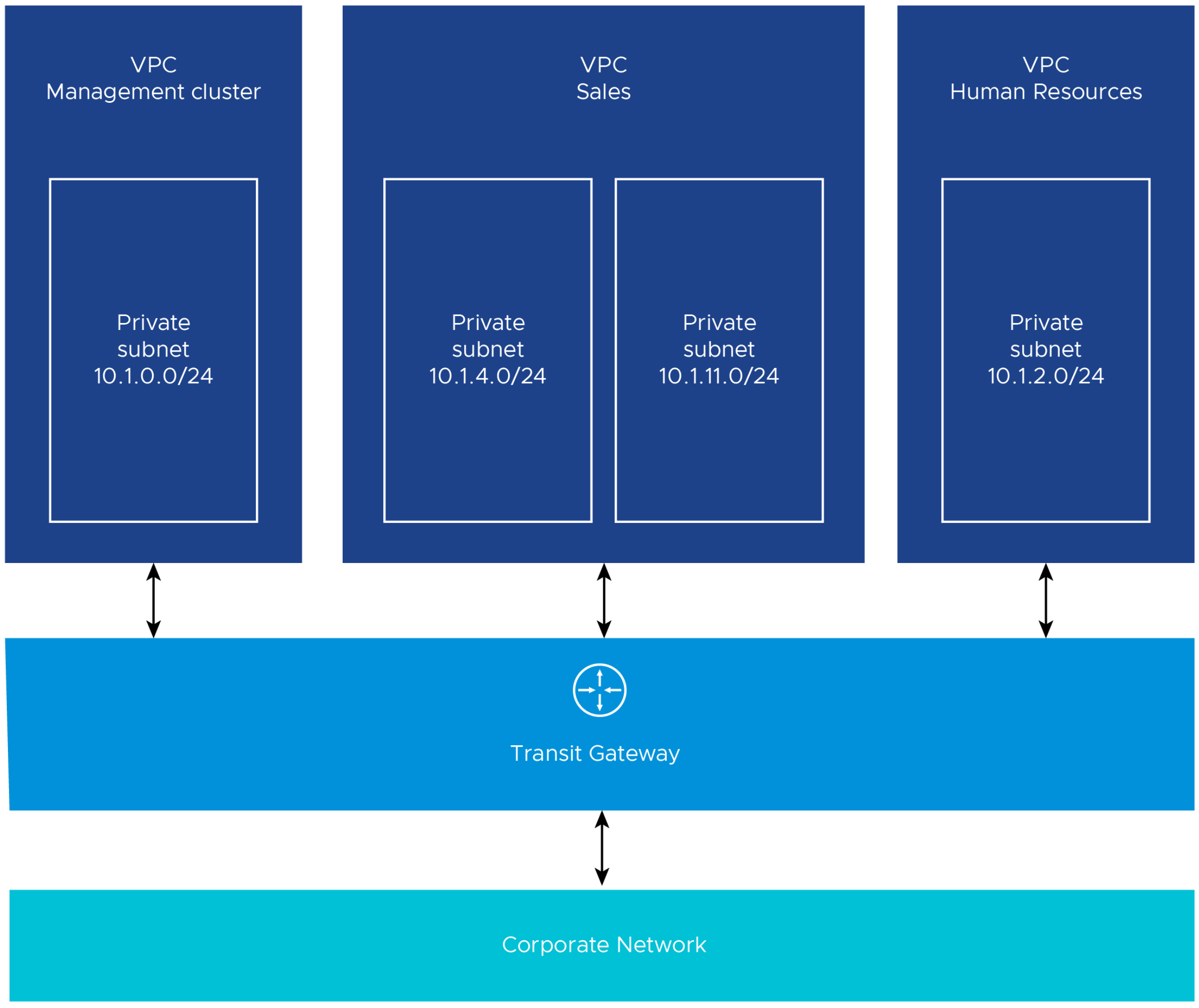
Single VPC with Multiple Availability Zones

For most use cases, a single VPC spread across multiple AZs is sufficient. If more separation is needed within one VPC, more subnets can be used to provide better IP-based visibility to corporate firewalls. The network diagram above depicts this architecture.

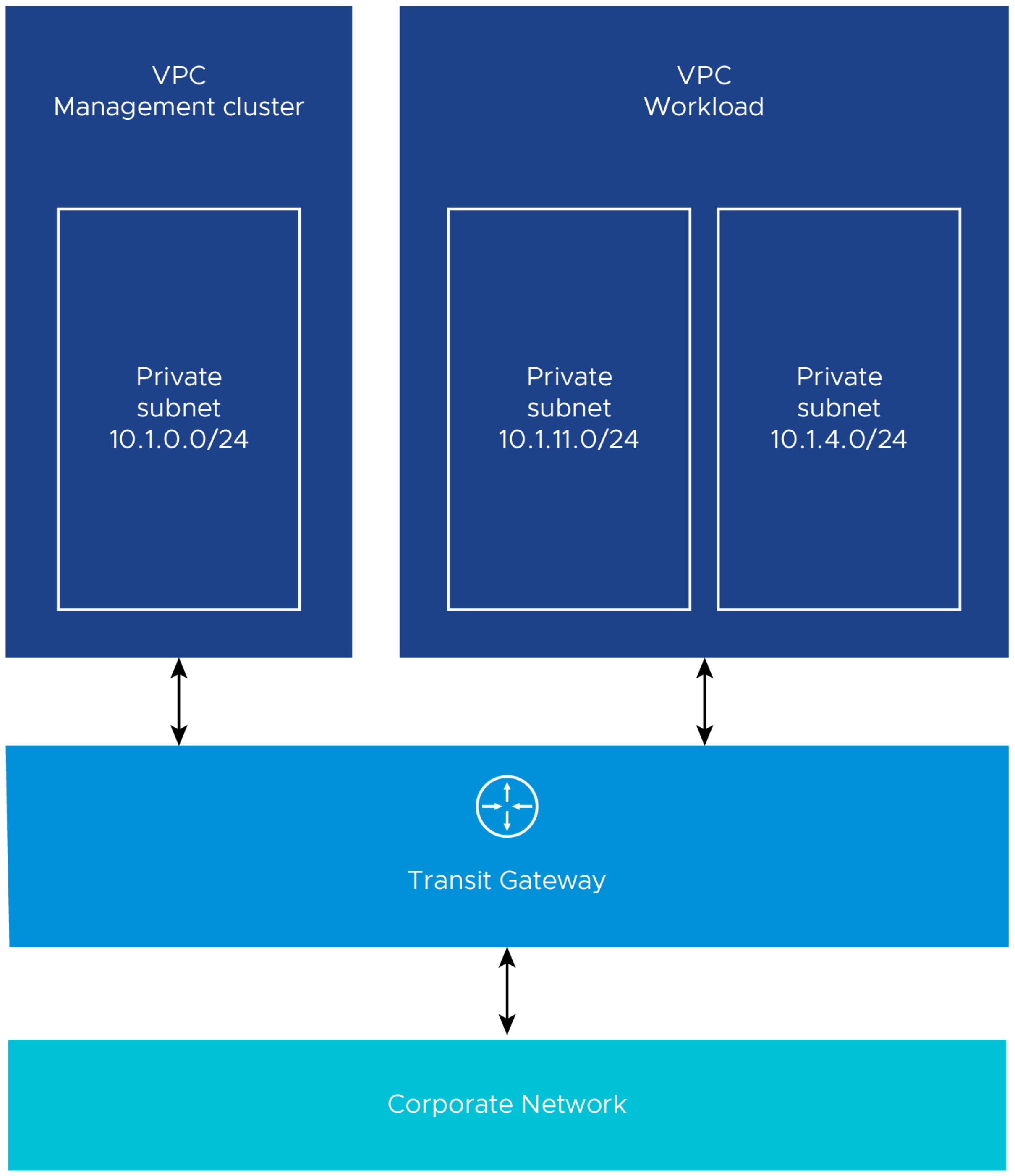
Multiple VPC with Multiple Availability Zones

For more separation of application workloads on AWS, you can deploy separate Kubernetes clusters to independent private VPCs. This separation might be desirable for workloads with different compliance requirements across different business units. By default, Tanzu Kubernetes Grid creates one VPC per cluster.

The following diagram shows an example architecture with **multiple offline VPCs**. The control plane load balancers in the example architecture are configured as internal load balancers.


VPC
Management cluster
Private subnet 10.1.0.0/24
VPC
Sales
Private subnet 10.1.4.0/24
Private subnet 10.1.11.0/24
VPC
Human Resources
Private subnet 10.1.2.0/24
Transit Gateway
Corporate Network

Another variant of multiple VPC and multiple AZ design is to have one VPC for the management cluster and another for workload clusters. The following diagram illustrates such a design.


VPC
Management cluster
Private subnet 10.1.0.0/24
VPC
Workload
Private subnet 10.1.11.0/24
Private subnet 10.1.4.0/24
Transit Gateway
Corporate Network

Consider the following design implications when designing your network architecture.

Decision ID	Design Decision	Design Justification	Design Implications
TKG-AG-001	Use separate networks/VPCs for the management cluster and workload clusters	Better isolation and security policies between environments isolate production Kubernetes clusters from dev/test clusters	Sharing the same network for multiple clusters can cause shortage of IP addresses

Decision ID	Design Decision	Design Justification	Design Implications
TKG-AG-002	Use separate networks for workload clusters based on their usage	Isolate production Kubernetes clusters from dev/test clusters	A separate set of Service Engines can be used for separating dev/test workload clusters from prod clusters

Availability

VMware recommends deploying your Tanzu Kubernetes Grid cluster in an odd number of AZs to ensure high availability of components that require consensus to operate in failure modes.

The Tanzu Kubernetes Grid management cluster performs Machine Health Checks on all Kubernetes worker VMs. This ensures that workloads remain in a functional state, and can remediate issues such as:

- a worker VM is accidentally deleted or corrupted.
- the Kubelet process on a worker VM is accidentally stopped or corrupted.

These health checks ensure that your worker capacity remains stable and can be scheduled for workloads. These health checks, however, do not apply to the control plane or to the load balancer VMs.

Quotas

It is essential to provide sufficient quotas to support both the management cluster and the workload clusters in your deployment. Otherwise, the cluster deployments will fail. Depending on the number of workload clusters you deploy, you may need to increase the AWS services quotas from their default values in every region in which you deploy Tanzu Kubernetes Grid.

The number of VPCs depends on the VPC architecture you select. The following table indicates the number of VPCs for the network architectures in the network diagrams shown above.

VPC Architecture	Number of VPCs
Single VPC	1
Multiple VPCs - one for each Kubernetes cluster	3
Multiple VPCs - one for the management cluster and one for workload cluster	2

See AWS service quotas for more information on AWS services default quotas.

Private Registry for Tanzu Kubernetes Grid

Before installing Tanzu Kubernetes grid into an air-gapped environment, a private Docker-compatible container registry such as Harbor, Docker, or Artifactory must be installed and configured as follows:

- Should run outside of Tanzu Kubernetes Grid and should be separate from any registry deployed as a shared service for clusters.
- Should use an RFC 1918 (private) address and remain routable to the Tanzu Kubernetes Grid clusters.
- Should be configured with SSL certificates signed by a trusted CA.

- Must not implement user authentication. For example, if you use a Harbor registry, the project must be public, not private.
- Must have all the Tanzu Kubernetes Grid images uploaded before you start installing Tanzu Kubernetes grid. See Prepare an Internet-Restricted Environment for more details.

Compliance and Security

VMware-published Tanzu Kubernetes releases (TKrs), along with compatible versions of Kubernetes and supporting components, use the latest stable and generally-available update of the OS version that it packages. They contain all current CVE and USN fixes as of the day that the image is built. The image files are signed by VMware and have file names that contain a unique hash identifier. The VMware-published Tanzu Kubernetes releases (TKrs) contain a BOM (bill of materials) for every component that appears in each Tanzu Kubernetes releases. This can be combined with the Tanzu Kubernetes grid BOM to provide a holistic view of the containers and contents of every combination of a Tanzu Kubernetes releases and Tanzu Kubernetes Grid release. These BOM files are securely served from VMware and stored as imgpkg generated Open Container Initiative (OCI) compliant images that have immutable hashes associated with the BOM file itself.

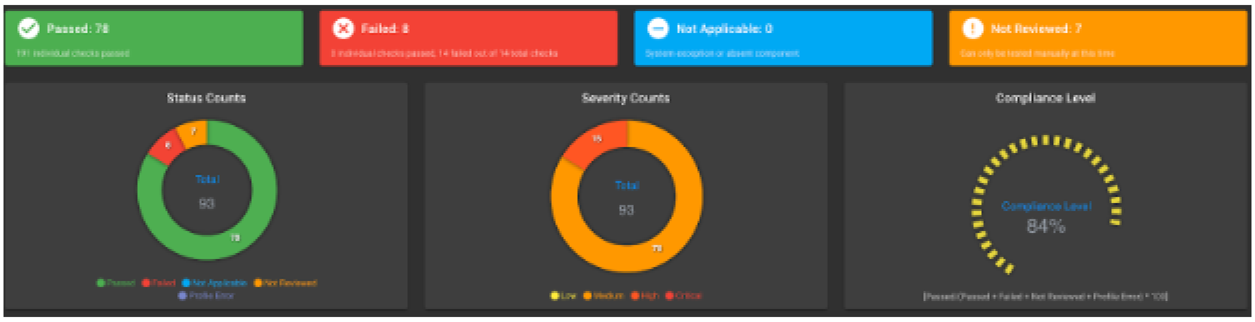
VMware provides FIPS (Federal Information Processing Standards)-capable version of Tanzu Kubernetes Grid. You can install and run a FIPS-capable version of Tanzu Kubernetes Grid, in which core components use cryptographic primitives provided by a FIPS-compliant library that provides FIPS 140-2 compliance based on the BoringCrypto / Boring SSL module. These core components include components of Kubernetes, Containerd and CRI, CNI plugins, CoreDNS, and etcd.

Kubernetes Hardening

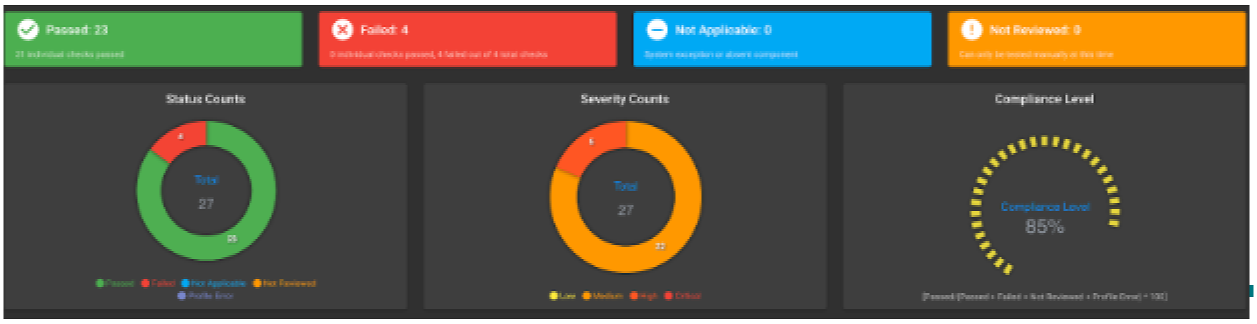
VMware has a robust process of following U.S. Department of Defense security standards for Tanzu Kubernetes Grid which includes scanning against an official Security Technical Implementation Guide (STIG) provided by the Defense Information Systems Agency (DISA).

Refer to the following sample post hardening results snapshots for Tanzu Kubernetes Grid.

Kubernetes Hardening - Control Plane



Kubernetes Hardening - Worker Node



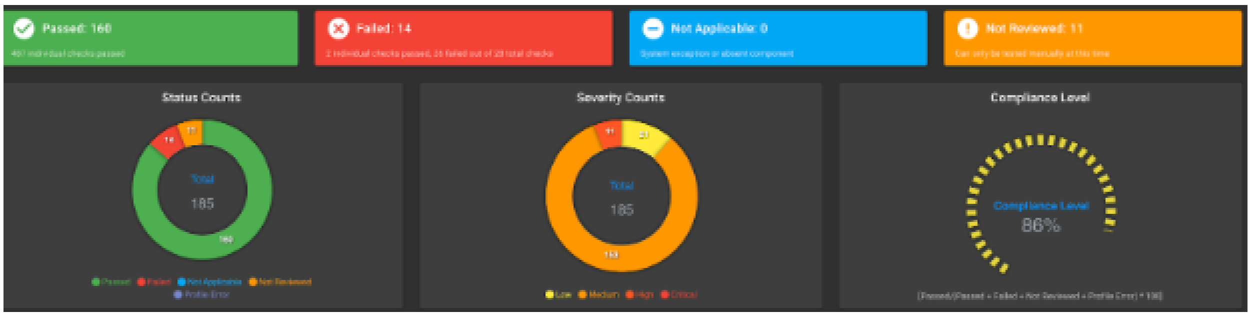
You can download the sample test results output.

Node OS Hardening

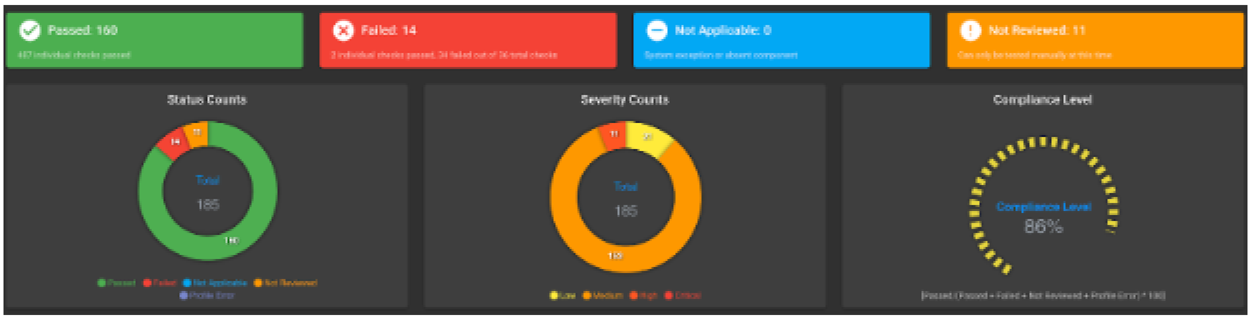
Tanzu Kubernetes Grid is layered on top of VMs using the Ubuntu operating system. Ubuntu has an official Security Technical Implementation Guide (STIG) provided by the Defense Information Systems Agency (DISA). To comply with STIG guidelines and to enable consistent and fast machine deployments, VMs are deployed from images using Ubuntu as the base operating system. VMware publishes AMI, OVA, or VHD that are FIPS enabled and STIG hardened.

Refer to the following OS hardening sample results snapshots.

Node OS Hardening - Control Plane



Node OS Hardening - Worker Node



You can download the sample test results output.

Service Installer for VMware Tanzu allows you to deploy a working Tanzu Kubernetes Grid cluster that already has the DISA Kubernetes STIG applied and it also enables FIPS 140-2 compatible algorithms.

Ports, Protocols, and Services Management (PPSM)

The Cloud Computing SRG V1R4 states in Section 5.15 that `mission owners using CSOs of any service type (I/P/SaaS) must comply with DoDI 8551.01: Ports, Protocols, and Services Management (PPSM) when implementing and operating their systems/applications in an IaaS/PaaS CSO or when using a SaaS offering.`

This requirement is to ensure that a standardized process is in place to catalog, regulate, and control the use and management of protocols in the Internet protocol suite, and associated ports on government networks including interconnected systems and platforms.

To further this mission, and ensure that this information is readily available, refer to the VMware public repository for Tanzu Kubernetes Grid PPSM.

National Institute of Standards and Technology (NIST)

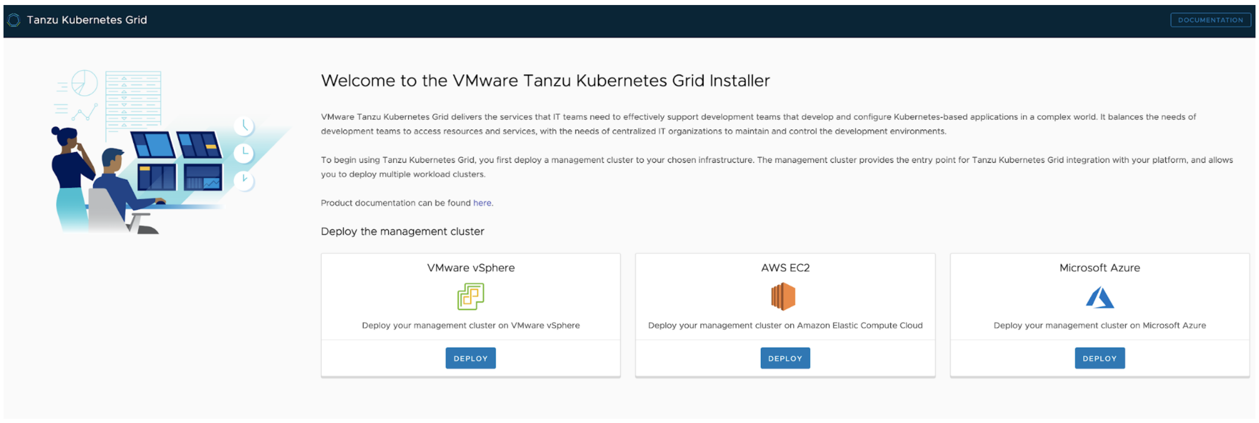
Since 2014, the public sector has been required to develop, document, implement, and maintain information security of government information systems through a standardized approach or framework. A major component of how this strategy is implemented relies on the security controls documented in NIST Special Publication 800-53, and the Risk Management Framework guidelines established in NIST SP 800-37. VMware maintains a partnership with the NIST Cybersecurity Center of Excellence (NCCoE) which includes validation of core VMware products including NSX, vSphere, vRealize, and Tanzu Kubernetes Grid. Refer to Security for more information.

Tanzu Kubernetes Grid Security Overview

For in depth information on the VMware security process and the current state of the art of Tanzu Kubernetes Grid security standards, see Tanzu Kubernetes Grid Security Overview Whitepaper.

Cluster Creation and Management

In this reference design, Tanzu Kubernetes Grid creates and manages ubiquitous Kubernetes clusters on AWS using Kubernetes Cluster API. Tanzu Kubernetes Grid functions through the creation of a management cluster that hosts the Cluster API. The Cluster API then interacts with the infrastructure provider to service workload Kubernetes cluster lifecycle requests.



Welcome to the VMware Tanzu Kubernetes Grid Installer

VMware Tanzu Kubernetes Grid delivers the services that IT teams need to effectively support development teams that develop and configure Kubernetes-based applications in a complex world. It balances the needs of development teams to access resources and services, with the needs of centralized IT organizations to maintain and control the development environments.

To begin using Tanzu Kubernetes Grid, you first deploy a management cluster to your chosen infrastructure. The management cluster provides the entry point for Tanzu Kubernetes Grid integration with your platform, and allows you to deploy multiple workload clusters.

Product documentation can be found here.

Deploy the management cluster

VMware vSphere
Deploy your management cluster on VMware vSphere
DEPLOY
AWS EC2
Deploy your management cluster on Amazon Elastic Compute Cloud
DEPLOY
Microsoft Azure
Deploy your management cluster on Microsoft Azure
DEPLOY

Tanzu for Kubernetes Operations includes observability components as well as a container registry. VMware recommends installing the necessary components into a centralized shared services cluster.

When making design decisions for your Tanzu Kubernetes Grid clusters, consider the design implications listed in the following table.

Decision ID	Design Decision	Design Justification	Design Implications
TKG-CLS-001	Deploy TKG Management cluster from CLI	UI doesn’t provide an option to specify an internal registry to use for TKG installation	Additional parameters are required to be passed in the cluster deployment file. Using UI, you can’t pass these additional parameters.
TKG-CLS-002	Use AWS internal Load Balancer scheme for your Control Plane Endpoints	Don’t expose Kubernetes API endpoints to Internet in Tanzu Kubernetes Grid clusters.	Create additional AWS load balancers in your AWS account which may increase AWS infrastructure cost.
TKG-CLS-003	Deploy Tanzu Kubernetes clusters in large and above sizes ec2 instances(example t2.large or ++)	Allow TKG clusters to have enough resources for all Tanzu packages	Create bigger AWS ec2 instances into your aws account which may increase AWS infrastructure cost .
TKG-CLS-004	Deploy Tanzu Kubernetes clusters with Prod plan	This deploys multiple control plane nodes and provides High Availability for the control plane	TKG infrastructure is not impacted by single node failure
TKG-CLS-005	Deploy Tanzu Kubernetes clusters with an odd number of AWS AZs for HA	This deploys multiple control plane nodes and provides High Availability for the control plane	TKG infrastructure is not impacted by single zone failure.
TKG-CLS-006	Enable identity management for Tanzu Kubernetes Grid clusters	To avoid usage of administrator credentials and ensure that required users with the right roles have access to Tanzu Kubernetes Grid clusters	Pinniped package helps with integrating the TKG Management cluster with LDAPS Authentication and Workload cluster inherits the authentication configuration from the management cluster
TKG-CLS-007	Enable Machine Health Checks for TKG clusters	The Tanzu Kubernetes Grid management cluster performs Machine Health Checks on all Kubernetes worker VMs and HA, Machine Health Checks interoperably work together to enhance workload resiliency	A MachineHealthCheck is a resource within the Cluster API that allows users to define conditions under which Machines within a Cluster should be considered unhealthy. Remediation actions can be taken when MachineHealthCheck has identified a node as unhealthy.

Bring Your Own Images for the Tanzu Kubernetes Grid Deployment

You can build custom machine images for Tanzu Kubernetes Grid to use as a VM template for the management and Tanzu Kubernetes (workload) cluster nodes that it creates. Each custom machine image packages a base operating system (OS) version and a Kubernetes version, along with any additional customizations, into an image that runs on vSphere, Microsoft Azure infrastructure and AWS (EC2) environments.

A custom image must be based on the OS versions that are supported by Tanzu Kubernetes Grid. The table below provides a list of operating systems that are supported for building custom images for the Tanzu Kubernetes Grid.

vSphere	AWS	Azure
- Ubuntu 20.04 - Ubuntu 18.04 - RHEL 7 - Photon OS 3	- Ubuntu 20.04 - Ubuntu 18.04 - Amazon Linux 2	- Ubuntu 20.04 - Ubuntu 18.04

For additional information on building custom images for TKG, see the Tanzu Kubernetes Grid Build Machine Images documentation for the applicable operating system:

- Linux Custom Machine Images
- Windows Custom Machine Images

Tanzu Kubernetes Clusters Networking

A Tanzu Kubernetes cluster provisioned by the Tanzu Kubernetes Grid supports two Container Network Interface (CNI) options:

- Antrea
- Calico

Both are open-source software that provide networking for cluster pods, services, and ingress.

When you deploy a Tanzu Kubernetes cluster using Tanzu CLI using the default configuration, Antrea CNI is automatically enabled in the cluster. While Kubernetes does have in-built network policies, Antrea builds on those native network policies to provide more fine-grained network policies of its own.

Antrea has a ClusterNetworkPolicy which operates at the Kubernetes cluster level. It also has a NetworkPolicy which limits the scope of a policy to a Kubernetes namespace. The ClusterNetworkPolicy can be thought of as a means for a Kubernetes Cluster Admin to create a security policy for the cluster as a whole. The NetworkPolicy can be thought of as a means for a developer to secure applications in a particular namespace. See Tanzu Kubernetes Grid Security and Compliance for more details.

To provision a Tanzu Kubernetes cluster using a non-default CNI, see the following instructions:

- Deploy Tanzu Kubernetes clusters with calico
- Implement Multiple Pod Network Interfaces with Multus

Each CNI is suitable for a different use case. The following table lists some common use cases for the two CNIs that Tanzu Kubernetes Grid supports. The information in this table will help you select the right CNI in your Tanzu Kubernetes Grid implementation.

CNI	Use Case	Pros and Cons
Antrea	Enable Kubernetes pod networking with IP overlay networks using VXLAN or Geneve for encapsulation. Optionally encrypt node-to-node communication using IPSec packet encryption. Antrea supports advanced network use cases like kernel bypass and network service mesh	**Pros**: Provide an option to configure egress IP pool or static egress IP for the Kubernetes workloads. **Cons**: More complicated for network troubleshooting because of the additional overlay network

CNI	Use Case	Pros and Cons
Calico	Calico is used in environments where factors like network performance, flexibility, and power are essential. For routing packets between nodes, Calico leverages the BGP routing protocol instead of an overlay network. This eliminates the need to wrap packets with an encapsulation layer resulting in increased network performance for Kubernetes workloads.	**Pros**: - Support for network policies - High network performance - SCTP Support **Cons**: - No multicast support.
Multus	Multus CNI provides multiple interfaces per each Kubernetes pod. Using Multus CRDs, you can specify which pods get which interfaces and allow different interfaces depending on the use case.	Pros - Separation of data/control planes. - Separate security policies can be used for separate interfaces. - Supports SR-IOV, DPDK, OVS-DPDK, and VPP workloads in Kubernetes with both cloud native and NFV based applications in Kubernetes.

Consider the following design implications on Tanzu Kubernetes Clusters Networking.

Decision ID	Design Decision	Design Justification	Design Implications
TKG-AG-00N	Use Antrea for CNI	Antrea is the preferred go forward network stack for Tanzu Kubernetes Clusters and is being actively developed within VMware for performance and advanced network functionality	Some specific networking features such as BGP external IP advertisement might require the Calico CNI.

Ingress and Load Balancing

Tanzu Kubernetes Grid requires load balancing for both the control plane and the workload clusters. Tanzu Kubernetes Grid for AWS uses elastic load balancers for both.

A default installation of Tanzu Kubernetes Grid does not deploy an ingress controller. Users can use Contour (available for installation through Tanzu Packages) or any third-party ingress controller of their choice. Contour is an open-source controller for Kubernetes ingress routing and can be used for layer 7 load balancing. Contour can be installed in the Shared Services cluster on any Tanzu Kubernetes Cluster. Deploying Contour is a prerequisite for deploying the Prometheus, Grafana, or Harbor packages on a workload cluster. For more information about Contour, see the Implementing Ingress Control with Contour.

To use a private load balancer, set `service.beta.kubernetes.io/aws-load-balancer-internal: "true"` in the annotations for the service. This setting also applies to the Contour ingress and controls.

Example :

```
annotations:
   service.beta.kubernetes.io/aws-load-balancer-internal: "true"
```

The following table provides general recommendations for when you should use the Contour ingress controller for your Kubernetes environment.

Ingress Controller	Use Case
Contour	Use Contour when north-south traffic is needed in a Kubernetes cluster. You can apply security policies for north-south traffic by defining the policies in the applications manifest file. Contour is a reliable solution for simple Kubernetes workloads.

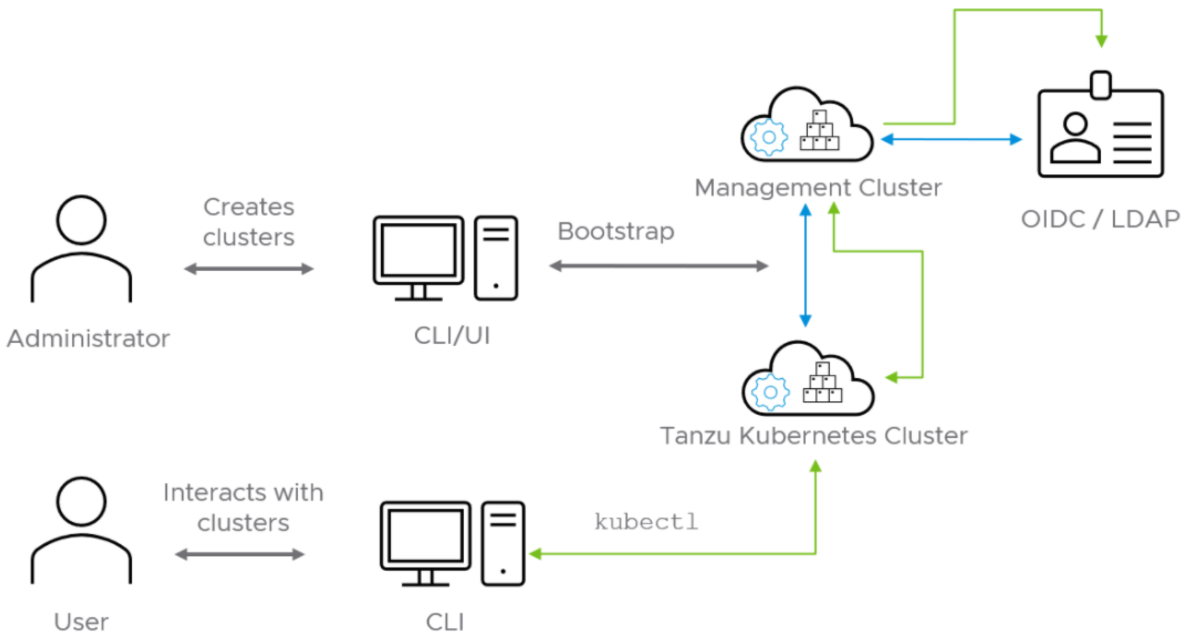
Authentication with Pinniped

The Pinniped authentication and authorization service components are deployed into the management cluster. Pinniped uses the LDAP identity provider (IDP) configurations specified during the management cluster deployment. The workload cluster inherits its authentication configurations from its management cluster. With authentication in place, a Kubernetes administrator can enforce role-based access control (RBAC) with Kubernetes RoleBinding resources. These resources associate an identity provider user with a given Kubernetes role on the workload cluster.

Pinniped consists of following components:

- **The Pinniped Supervisor:** It is an OIDC server that authenticates users through an external identity provider (IDP)/LDAP, and then issues its own federation ID tokens to be passed on to clusters based on the user information from the IDP.
- **The Pinniped Concierge:** It is a credential exchange API which takes as input a credential from an identity source (e.g., Pinniped Supervisor, proprietary IDP), authenticates the user via that credential, and returns another credential which is understood by the host Kubernetes cluster or by an impersonation proxy which acts on behalf of the user.
- **Dex:** Pinniped uses Dex as a broker for your upstream LDAP identity provider. Dex is only deployed when LDAP is selected as the OIDC backend during Tanzu Kubernetes Grid management cluster creation.

The following diagram shows the Pinniped authentication flow with an LDAP. In the diagram, the blue arrows represent the authentication flow between the workload cluster, the management cluster and the LDAP. The green arrows represent Tanzu CLI and `kubectl` traffic between the workload cluster, the management cluster and the external IDP.



See the Pinniped docs for more information on how to integrate Pinniped into Tanzu Kubernetes Grid with LDAP.

VMware recommends the following best practices for managing identities in Tanzu Kubernetes Grid-provisioned clusters:

- Configure Pinniped services during management cluster creation.
- Limit access to cluster resources following the least privilege principle.
- Limit access to management clusters to the appropriate set of users. For example, provide access only to users who are responsible for managing infrastructure and cloud resources but not to application developers. This is especially important because access to the management cluster provides access to all workload clusters.

Observability

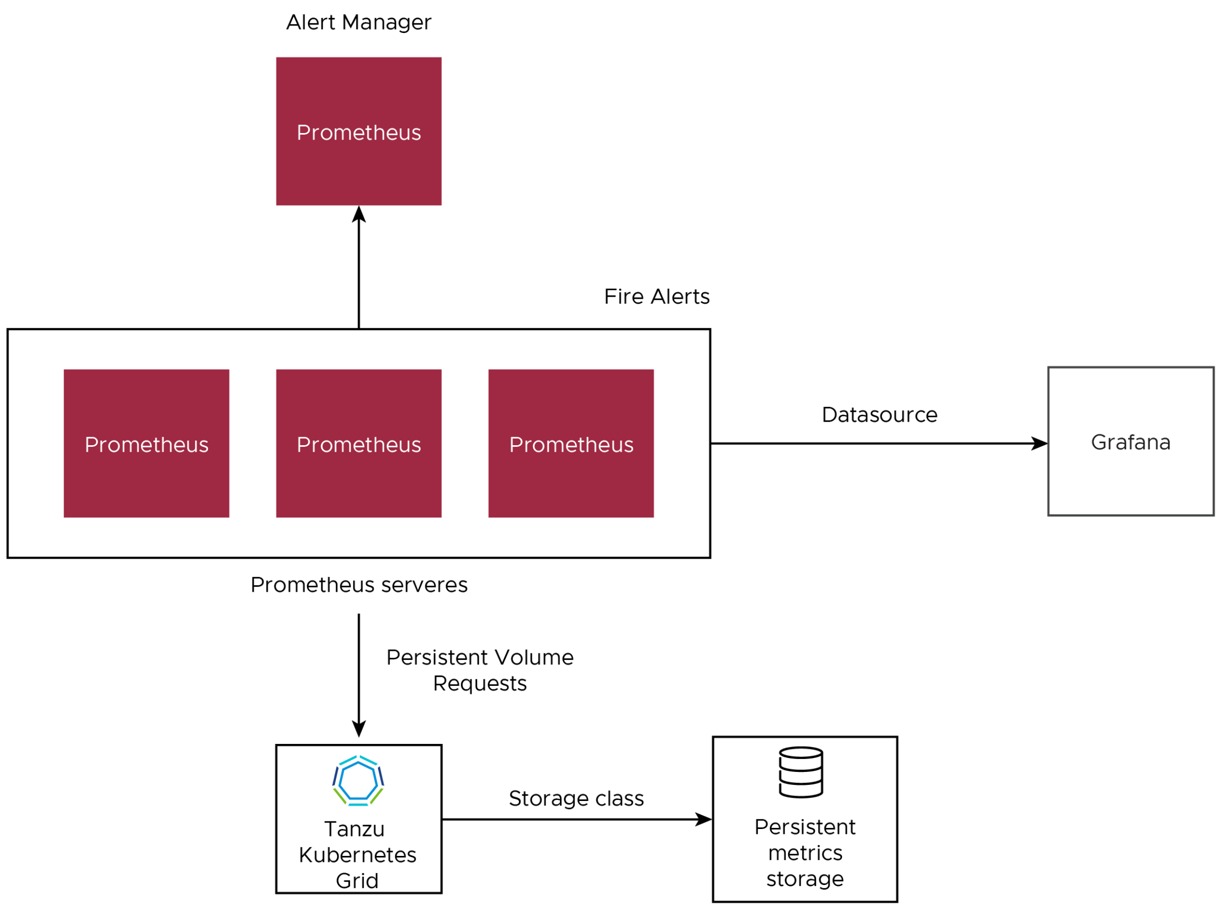
Tanzu Kubernetes Grid Monitoring

In an air-gapped environment, monitoring for the Tanzu Kubernetes clusters is provided via Prometheus and Grafana.

- **Prometheus** is an open-source system monitoring and alerting toolkit. It can collect metrics from target clusters at specified intervals, evaluate rule expressions, display the results, and trigger alerts if certain conditions arise. The Tanzu Kubernetes Grid implementation of Prometheus includes Alert Manager, which you can configure to notify you when certain events occur. Prometheus exposes scrapable metrics endpoints for various monitoring targets throughout your cluster. Metrics are ingested by polling the endpoints at a set interval. The metrics are then stored in a time-series database. You use the Prometheus Query Language interface to explore the metrics.
- **Grafana** is an open-source visualization and analytics software. It allows you to query, visualize, alert on, and explore your metrics no matter where they are stored. Grafana is responsible for visualizing Prometheus metrics without the need to manually write the PromQL queries. You can create custom charts and graphs in addition to the pre-packaged options.

Both Prometheus and Grafana are installed with CLI-managed Tanzu packages by creating the deployment manifests and invoking the kubectl command to deploy the packages in the Tanzu Kubernetes clusters.

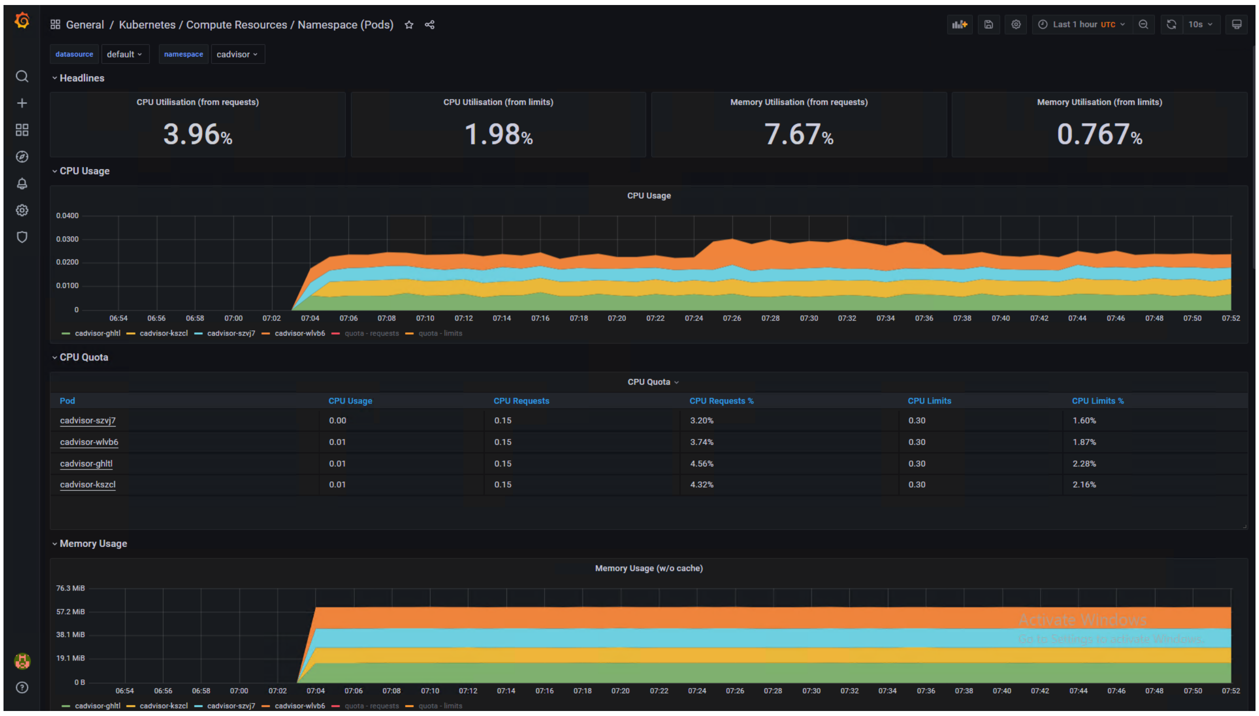
The following diagram shows how the monitoring components on a cluster interact.



You can use out-of-the-box Kubernetes dashboards or you can create new dashboards to monitor compute/network/storage utilization of Kubernetes objects such as Clusters, Namespaces, and Pods.

The following pictures show some sample dashboards.

Namespace (Pods) Compute Resources Utilization Dashboard

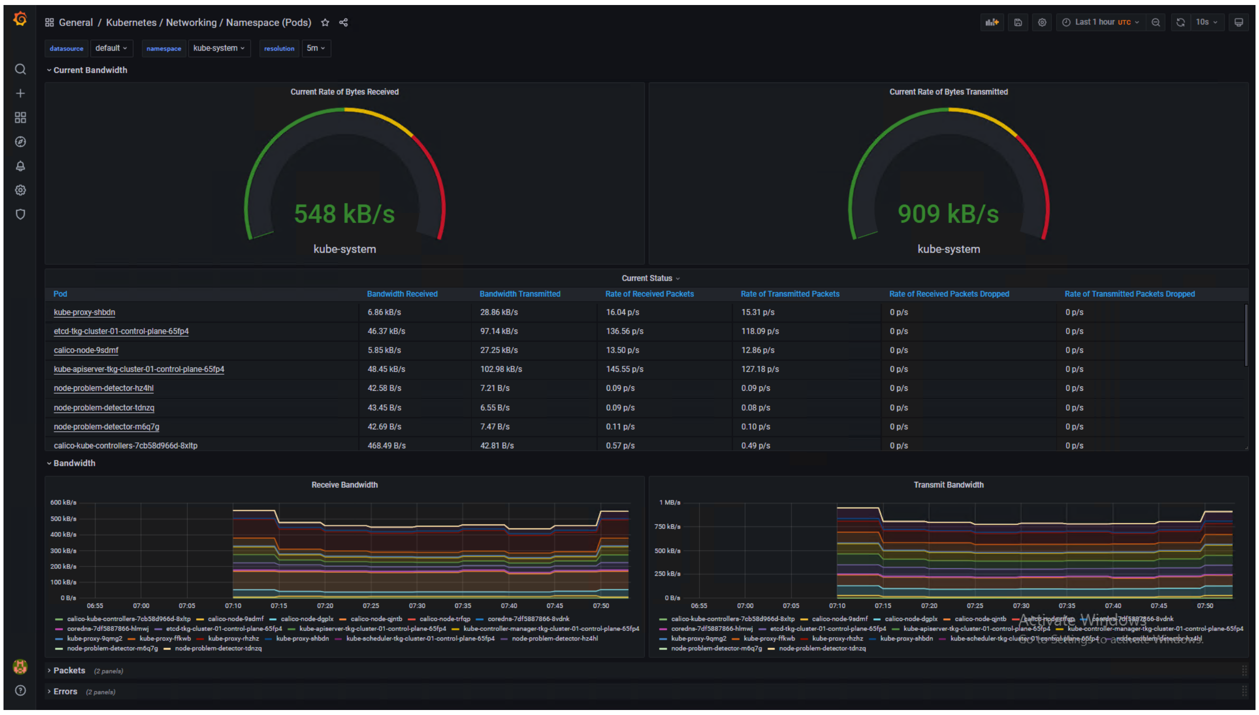


CPU Utilisation (from requests)	CPU Utilisation (from limits)	Memory Utilisation (from requests)	Memory Utilisation (from limits)
3.96%	1.98%	7.67%	0.767%

CPU Quota

Pod	CPU Usage	CPU Requests	CPU Requests %	CPU Limits	CPU Limits %
cadvisor-szvj7	0.00	0.15	3.20%	0.30	1.60%
cadvisor-wlvb6	0.01	0.15	3.74%	0.30	1.87%
cadvisor-ghltl	0.01	0.15	4.56%	0.30	2.28%
cadvisor-kszcl	0.01	0.15	4.32%	0.30	2.16%

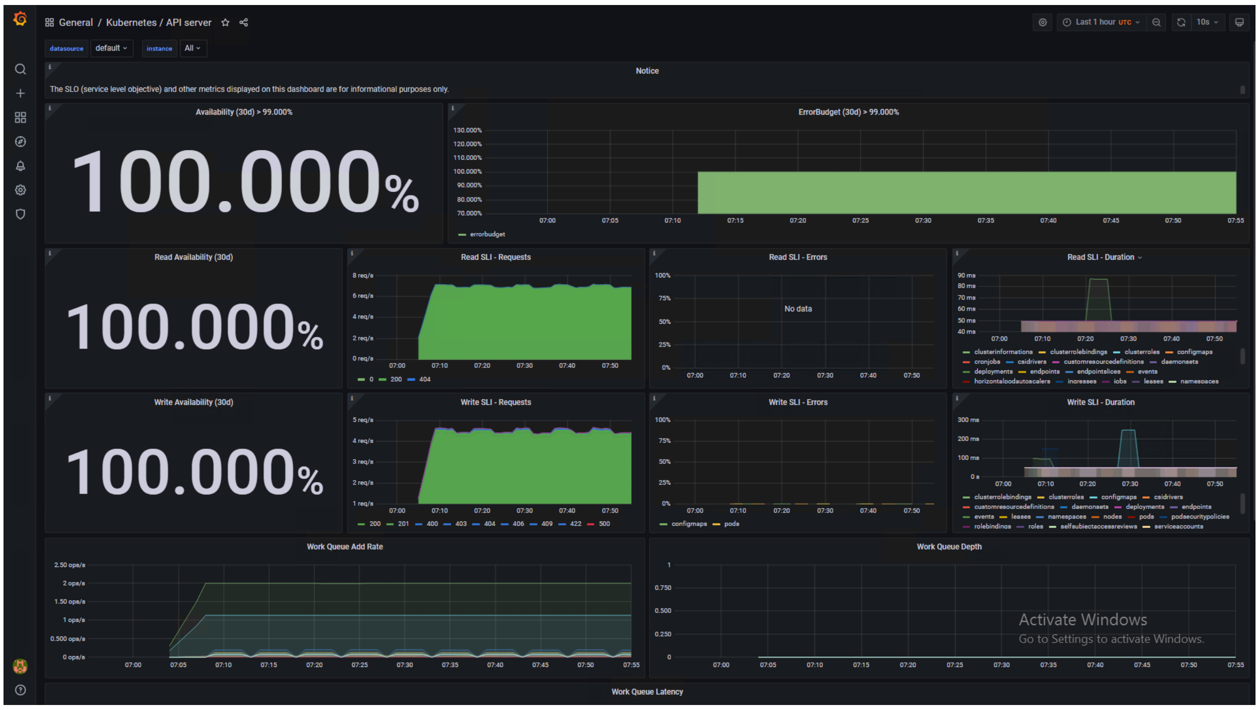
Namespace (Pods) Networking Utilization Dashboard



Current Status

Pod	Bandwidth Received	Bandwidth Transmitted	Rate of Received Packets	Rate of Transmitted Packets	Rate of Received Packets Dropped	Rate of Transmitted Packets Dropped
kube-proxy-shbdn	6.86 kB/s	28.86 kB/s	16.04 p/s	15.31 p/s	0 p/s	0 p/s
etcd-tkg-cluster-01-control-plane-65fp4	46.37 kB/s	97.14 kB/s	136.56 p/s	118.09 p/s	0 p/s	0 p/s
calico-node-9sdmf	5.85 kB/s	27.25 kB/s	13.50 p/s	12.86 p/s	0 p/s	0 p/s
kube-apiserver-tkg-cluster-01-control-plane-65fp4	48.45 kB/s	102.98 kB/s	145.55 p/s	127.18 p/s	0 p/s	0 p/s
node-problem-detector-hz4hl	42.58 B/s	7.21 B/s	0.09 p/s	0.09 p/s	0 p/s	0 p/s
node-problem-detector-tdnzq	43.45 B/s	6.55 B/s	0.09 p/s	0.08 p/s	0 p/s	0 p/s
node-problem-detector-m6q7g	42.69 B/s	7.47 B/s	0.11 p/s	0.10 p/s	0 p/s	0 p/s
calico-kube-controllers-7cb58d966d-8xltp	468.49 B/s	42.81 B/s	0.57 p/s	0.49 p/s	0 p/s	0 p/s

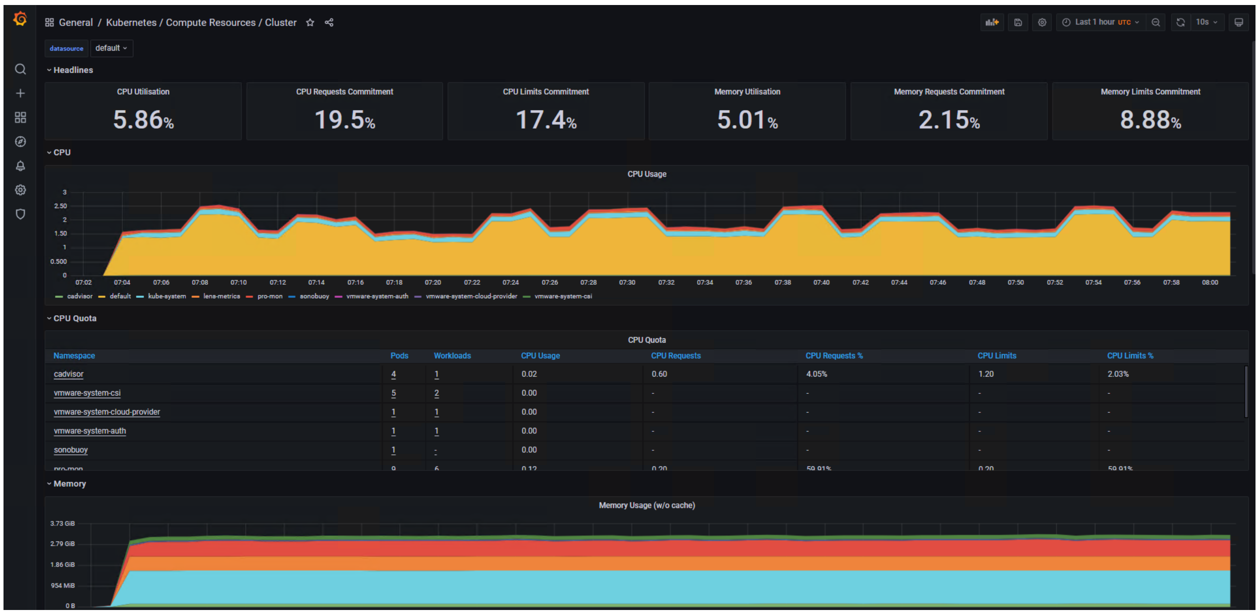
API Server Availability Dashboard



The SLO (service level objective) and other metrics displayed on this dashboard are for informational purposes only.

Panel	Value
Availability (30d) > 99.000%	100.000%
Read Availability (30d)	100.000%
Write Availability (30d)	100.000%

Cluster Compute Resources Utilization Dashboard



CPU Utilisation	CPU Requests Commitment	CPU Limits Commitment	Memory Utilisation	Memory Requests Commitment	Memory Limits Commitment
5.86%	19.5%	17.4%	5.01%	2.15%	8.88%

CPU Quota

Namespace	Pods	Workloads	CPU Usage	CPU Requests	CPU Requests %	CPU Limits	CPU Limits %
cadvisor	4	1	0.02	0.60	4.05%	1.20	2.03%
vmware-system-csi	5	2	0.00	-	-	-	-
vmware-system-cloud-provider	1	1	0.00	-	-	-	-
vmware-system-auth	1	1	0.00	-	-	-	-
sonobuoy	1	-	0.00	-	-	-	-
pro-mon	9	6	0.12	0.20	59.91%	0.20	59.91%

Log Forwarding

Tanzu also includes Fluent Bit for integration with logging platforms such as vRealize, Log Insight Cloud, and Elasticsearch. See Fluent Bit Documentation for various logging providers.

You can deploy Fluent Bit on any management cluster or Tanzu Kubernetes clusters from which you want to collect logs. First, configure an output plugin on the cluster from which you want to gather logs, depending on the endpoint that you use. Then deploy Fluent Bit on the cluster as a package.

Tanzu Kubernetes Grid Upgrade

To upgrade the previous version of Tanzu Kubernetes Grid into your environment, see Tanzu Kubernetes Grid Upgrade instructions.

Summary

Tanzu Kubernetes Grid on AWS offers high-performance potential, convenience, and addresses the challenges of creating, testing, and updating cloud-based Kubernetes platforms in a consolidated production environment. This validated approach will result in a production quality installation with all the application services needed to serve combined or uniquely separated workload types via a combined infrastructure solution.

This plan meets many Day-0 needs for quickly aligning product capabilities to full stack infrastructure, including networking, configuring your firewall, load balancing, workload compute alignment and other capabilities. Observability and Metrics Monitoring can be quickly implemented with Prometheus and Grafana.

Deploy Tanzu Kubernetes Grid on AWS in an Air-Gapped Environment

This document outlines the steps for deploying VMware Tanzu for Kubernetes Operations (informally known as TKO) on AWS in an air-gapped (Internet-restricted) environment. The deployment is based on the reference design provided in VMware Tanzu Kubernetes Grid on AWS Airgap Reference Design.

Deploying with VMware Service Installer for Tanzu

You can use VMware Service Installer for VMware Tanzu to automate this deployment.

VMware Service Installer for Tanzu automates the deployment of the reference designs for Tanzu for Kubernetes Operations. It uses best practices for deploying and configuring the required Tanzu for Kubernetes Operations components.

To use Service Installer to automate this deployment, see Deploying Tanzu Kubernetes Grid on Federal Air-gapped AWS VPC Using Service Installer for VMware Tanzu.

Alternatively, if you decide to manually deploy each component, follow the steps provided in this document.

Prerequisites

Before deploying VMware Tanzu for Kubernetes Operations in an AWS air-gapped environment, ensure that the following are set up.

- **AWS Account**: An IAM user account with **administrative privileges**.
- **AWS Resource Quotas**: Sufficient quotas to support both the management cluster and the workload clusters in your deployment. Otherwise, the cluster deployments will fail. Depending on the number of workload clusters you plan to deploy, you may need to increase the AWS services quotas from their default values. You will need to increase the quota in every region in which you deploy Tanzu Kubernetes Grid.
    - AWS service quotas in the AWS.

- **An Internet-connected Linux bootstrap machine** The bootstrap machine can be a local device such as a laptop or a virtual machine running in, for example, VMware Workstation or Fusion. You will use the bootstrap machine to create the AWS VPC and jumpbox. The bootstrap machine:
    - Is not inside the Internet-restricted environment or is able to access the domains listed in Proxy Server Allowlist.
    - Has the Docker client app installed.
    - Has imgpkg installed.
    - Has the latest version of yq installed.
    - Has the latest version of jq installed.
    - Has AWS CLI installed.
    - Has the Carvel Tools installed, if you intend to install one or more of the optional packages provided by Tanzu Kubernetes Grid, such as Harbor.
- **VMware Cloud**: Access to VMware Cloud to download Tanzu CLI.

Overview of the Deployment Steps

The main steps to deploy Tanzu for Kubernetes Operations on AWS EC2 are as follows. Each step links to more detailed instructions.

1. Set Up AWS Infrastructure.
2. Create an Offline JumpBox.
3. Create and Set Up a Private Container Registry.
4. Copy the Container Images Required to Deploy Tanzu Kubernetes Grid.
5. Tanzu Kubernetes Grid Build Machine Image.
6. Prepare an Internet-Restricted Environment.
7. Install Tanzu Kubernetes Grid Management Cluster.
8. Examine the Management Cluster Deployment.
9. Deploy Workload Clusters.
10. Install and Configure Packages into Workload Clusters.
11. Logs and Troubleshooting.
12. Delete Clusters.
13. Air-Gapped STIG/FIPS Deployment on AWS.
14. Tanzu Kubernetes Grid Upgrade.

Set Up AWS Infrastructure

The following describes the steps to create your AWS environment and configure your network. Follow the steps in the order provided.

Create an Internet-Restricted Virtual Private Cloud (VPC)

Follow these steps to create an Internet-restricted (offline) AWS VPC or see Work with VPCs in AWS documentation. The offline VPC must not have a NAT or Internet gateway. Create your offline VPC with private subnets only.

1. Create a VPC.
    1. Log in to your AWS Console and select VPC service.
    2. Create your VPC with Valid CIDR and name.
2. Create 3 Private Subnets. Click **Subnet** and create your subnet with:
    1. Private Subnet 1, Private Subnet 2 and Private Subnet 3 valid Name & VPC.
    2. Valid Subnet range which is valid IPv4 CIDR Block.
3. Create Private Route Table.
    1. Create a Route table in the same VPC.
    2. Make sure you selected the right VPC and gave a proper tag.
4. Add Private Subnet in Private Route Table.
    1. Edit the Subnet Association.
    2. Select the PrivateSubnet checkbox.
    3. Click **Save**.
5. Edit DNS Resolution and Hostname.
    1. Click **Action** and Edit DNS hostname.
    2. Select DNS Hostname and click **Save**.
    3. Refer to AWS Documentation for more information.

If you create multiple offline VPCs, also see Getting started with transit gateways in AWS documentation to create an AWS transit gateway.

1. Create a VPC peering connection between offline and Internet-connected VPC. If you have created the transit gateway, you can skip this step.

Add VPC Endpoints into Offline VPC

After you create the offline VPC, you must add the following endpoints to the offline VPC. VPC endpoints enable private connections between your VPC and supported AWS services.

Service endpoints:

- sts
- ssm
- ec2
- ec2messages
- elasticloadbalancing

- secretsmanager
- ssmmessages
- s3 (gateway type)

For more information about creating an endpoint service, see Create a service powered by AWS PrivateLink in AWS documentation.

Create an Offline JumpBox

After configuring the network, complete the steps described in this section to set up your jumpbox. You will download the Tanzu CLI to the jumpbox, which you will use to deploy the management cluster and workload clusters. You also keep the Tanzu and Kubernetes configuration files for your deployments on your jumpbox.

1. Create a jumpbox.

```
#Set up AWS credentials
export AWS_ACCESS_KEY_ID=xx
export AWS_SECRET_ACCESS_KEY=xx
# Should be a region with at least 3 available AZs
export AWS_REGION=us-east-1
export AWS_PAGER=""

#Set up AWS profile
aws ec2 describe-instances --profile <profile name>
export AWS_PROFILE=<profile name>

# Set offline VPC and private subnet ID
export vpcId=<offline vpc id>
export prisubnetId=<private subnet id>
export WORKING_DIR=<local working dir>

aws ec2 create-security-group --group-name "jumpbox-ssh" --description "To Jump
box" --vpc-id "$vpcId" --output json > $WORKING_DIR/sg_jumpbox_ssh
aws ec2 create-tags --resources $(jq -r .GroupId $WORKING_DIR/sg_jumpbox_ssh) -
-tags Key=Name,Value="jumpbox-ssh"

# Allow SSH access to jumpbox
aws ec2 authorize-security-group-ingress --group-id  $(jq -r .GroupId $WORKING_
DIR/sg_jumpbox_ssh) --protocol tcp --port 22 --cidr "0.0.0.0/0"

# Save this file (or use an existing team keypair)
aws ec2 create-key-pair --key-name tkg-kp --query 'KeyMaterial' --output text >
tkgkp.pem
chmod 400 tkgkp.pem

# Find an Amazon Machine Image (AMI) for your region https://cloud-images.ubunt
u.com/locator/ec2/ (20.04)<_Correct?_>
aws ec2 run-instances --image-id ami-036d46416a34a611c --count 1 --instance-typ
e t2.xlarge --key-name tkg-kp --security-group-ids  $(jq -r .GroupId $WORKING_D
IR/sg_jumpbox_ssh)   --subnet-id $prisubnetId  --tag-specifications 'ResourceTy
pe=instance,Tags=[{Key=Name,Value=tkg-jumpbox}]' --block-device-mappings 'Devic
eName=/dev/sda1,Ebs={VolumeSize=64}' > $WORKING_DIR/instance_jb_starting
```

2. Wait a few minutes for the instance to start. Then SSH to the jumpbox.

```
aws ec2 describe-instances --instance-id $(jq -r '.Instances[0].InstanceId' $WO
RKING_DIR/instance_jb_starting) > $WORKING_DIR/instance_jb_started

echo j IP: $(jq -r '.Reservations[0].Instances[0].PrivateIpAddress' $WORKING_DI
R/instance_jb_started)

ssh ubuntu@$(jq -r '.Reservations[0].Instances[0].PrivateIpAddress' $WORKING_DI
R/instance_jb_started) -i tkgkp.pem
```

3. Log in to the jumpbox to install the necessary packages and configurations.

    1. Download Docker Ubuntu binaries and transfer to the jumpbox.

    ```
    #transfer docker package file
    scp -i tkgkp.pem docker-ce-cli_20.10.9_3-0_ubuntu-focal_amd64.deb ubuntu@
    $(jq -r '.Reservations[0].Instances[0].PrivateIpAddress' $WORKING_DIR/ins
    tance_jb_started):/home/ubuntu
    ```

    2. Add `ubuntu` user to Docker and reboot the jumpbox.

    ```
    #login to jumpbox
    ssh ubuntu@<jumpbox-ip> -i tkgkp.pem
    #install docker
    dpkg --install <docker-ce-cli_20.10.9_3-0_ubuntu-focal_amd64.deb
    #add ubuntu user to docker
    sudo adduser ubuntu docker
    #reboot
    sudo reboot
    ```

4. Download the Tanzu CLI and other Linux utilities from the Tanzu Kubernetes Grid Download Product site.

5. Copy the files and binaries to the jumpbox.

```
scp -i tkgkp.pem tanzu-cli-bundle-linux-amd64.tar kubectl-linux-v1.23.8+vmware.
gz ubuntu@$(jq -r '.Reservations[0].Instances[0].PublicIpAddress' $WORKING_DIR/
instance_jb_started):/home/ubuntu
```

6. Connect to the jumpbox.

```
ssh ubuntu@<jumpbox-ip> -i tkgkp.pem
```

7. Install the Tanzu CLI.

Run the session in `screen` in case your SSH connection is terminated. If your connection is terminated, you can reattach to the screen session with `screen -r` once you have reconnected.

```
screen
tar -xzvf tanzu-cli-bundle-linux-amd64.tar.gz
gunzip kubectl-*.gz
sudo install kubectl-linux-* /usr/local/bin/kubectl
cd cli/
sudo install core/*/tanzu-core-linux_amd64 /usr/local/bin/tanzu
gunzip *.gz
sudo install imgpkg-linux-amd64-* /usr/local/bin/imgpkg
sudo install kapp-linux-amd64-* /usr/local/bin/kapp
```

```
sudo install kbld-linux-amd64-* /usr/local/bin/kbld
sudo install vendir-linux-amd64-* /usr/local/bin/vendir
sudo install ytt-linux-amd64-* /usr/local/bin/ytt
cd ..
tanzu plugin sync
tanzu config init
```

Running the `tanzu config init` command for the first time creates the `~/.config/tanzu/tkg` subdirectory, which contains the Tanzu Kubernetes Grid configuration files.

Create and Set Up a Private Container Registry

This registry should run outside of Tanzu Kubernetes Grid and is separate from any registry deployed as a shared service for clusters.

- You can configure the container registry with SSL certificates signed by a trusted CA, or with self-signed certificates.
- The registry must not implement user authentication. For example, if you use a Harbor registry, the project must be public, not private.
- You can set up this private registry on an offline jumpbox machine (should be large enough to set up a private registry) or set it up on another ec2 instance inside an offline VPC.

Install Harbor

- Download the binaries for the latest Harbor release.
- Follow the Harbor Installation and Configuration instructions in the Harbor documentation.

Copy the Container Images Required to Deploy Tanzu Kubernetes Grid

Copy the container images required to deploy Tanzu Kubernetes Grid on AWS to a private registry in a physically air-gapped, offline environment. This procedure uses the scripts `download-images.sh`, `gen-publish-images-totar.sh`, and `gen-publish-images-fromtar.sh` to:

- Copy the images from the Tanzu Kubernetes Grid public registry and save them locally in tar format on an offline jumpbox.
- Extract the images from the tar files and copy them to a private registry.

See Copy the container images required to deploy Tanzu Kubernetes Grid for more detailed instructions.

Tanzu Kubernetes Grid Build Machine Image

If you have a requirement to build custom images, follow the steps in Tanzu Kubernetes Grid Build Machine Images.

For compliance and security requirements VMware has published security overview whitepaper. Refer to Tanzu Kubernetes Grid security overview whitepaper for more information.

Prepare an Internet-Restricted Environment

Before you can deploy management clusters and Tanzu Kubernetes clusters in an Internet-restricted environment, you need to prepare the environment.

Set the IP address or FQDN of your local private registry as an environment variable:

`export TKG_CUSTOM_IMAGE_REPOSITORY="PRIVATE-REGISTRY"` Where PRIVATE-REGISTRY is the IP address or FQDN of your private registry and the name of the project. For example, `custom-image-repository.io/yourproject`.

Follow the instructions in Prepare an Internet-Restricted Environment.

Deploy a Tanzu Kubernetes Grid Management Cluster

Create and edit a YAML configuration file. Then use the configuration file with CLI commands to deploy a management cluster.

This section describes how to deploy a Tanzu Kubernetes Grid management cluster from a configuration file using the Tanzu CLI.

Before creating a management cluster using the Tanzu CLI, define the base configuration for the cluster in a YAML file. Specify this file by using the `tanzu management-cluster create` command with the `--file` option.

In the configuration file for the management cluster, enable the AWS internal load balancer as follows:

`AWS_LOAD_BALANCER_SCHEME_INTERNAL: "true"` Using an internal load balancer scheme prevents the Kubernetes API server for the cluster from being accessed and routed over the Internet.

To create a new Tanzu Kubernetes Grid management cluster, run the following command:

```
tanzu management-cluster create --file path/to/cluster-config-file.yaml
```

For more information about deploying a management cluster from a configuration file, see Deploy Management Clusters from a Configuration File.

Examine the Management Cluster Deployment

When the management cluster is deployed, either from the installer interface or from a configuration file using Tanzu CLI, Tanzu Kubernetes Grid uses a Kubernetes in Docker kind cluster on the jumpbox to create a temporary management cluster. kind is a tool for running Kubernetes clusters locally using Docker containers as Kubernetes nodes.

Tanzu Kubernetes Grid uses the temporary management cluster to provision the final management cluster on AWS. For information about how to examine and verify your Tanzu Kubernetes Grid management cluster deployment, see Examine the Management Cluster Deployment.

Deploy Workload Clusters

After deploying the management cluster, you can create the workload clusters. The context of the management cluster is updated automatically, so you can begin interacting with the management cluster.

Run the following command to create a basic workload cluster:

```
tanzu cluster create <cluster_name> --plan=prod
```

Workload clusters can be highly customized through YAML manifests and applied to the management cluster for deployment and lifecycle management. To generate a YAML template to update and modify to your own needs, use the `--dry-run` switch. Edit the manifests to meet your requirements and apply them to the cluster.

Example:

```
tanzu cluster create <workload_cluster> --plan=prod --worker-machine-count 3 --dry-run
```

After the workload cluster is created, the current context changes to the new workload cluster.

For more information on cluster lifecycle and management, see Manage Clusters.

Troubleshooting Tips for Tanzu Kubernetes Grid

For tips to help you to troubleshoot common problems that you might encounter when installing Tanzu Kubernetes Grid and deploying Tanzu Kubernetes clusters, see Troubleshooting Tips for Tanzu Kubernetes Grid.

Install and Configure Packages into Workload Clusters

A package in Tanzu Kubernetes Grid is a collection of related software that supports or extends the core functionality of the Kubernetes cluster in which the package is installed. Tanzu Kubernetes Grid includes two types of packages, auto-managed packages and CLI-managed packages. For more information about packages in Tanzu Kubernetes Grid, see Install and Configure Packages.

Auto-Managed Packages

Tanzu Kubernetes Grid automatically installs the auto-managed packages during cluster creation. For more information about auto-managed packages, see Auto-Managed Packages.

CLI-Managed Packages

A CLI-managed packages package is an optional component of a Kubernetes cluster that you can install and manage with the Tanzu CLI. These packages are installed after cluster creation. CLI-managed packages are grouped into package repositories in the Tanzu CLI. If a package repository that contains CLI-managed packages is available in the target cluster, you can use the Tanzu CLI to install and manage any of the packages from that repository.

Using the Tanzu CLI, you can install cli-managed packages from the built-in `tanzu-standard` package repository or from package repositories that you add to your target cluster. From the `tanzu-standard` package repository, you can install the Cert Manager, Contour, Fluent Bit, Grafana, Harbor, and Prometheus packages. See CLI-Managed Packages for more information.

Recommended packages:

- **Cert Manager** for automating the management and issuance of TLS certificates. See Installing Cert Manager.
- **Contour** for ingress control. See Implementing Ingress Control with Contour. For use a private load balancer, set `service.beta.kubernetes.io/aws-load-balancer-internal: "true"` in the annotations for the service. This setting also applies to the Contour ingress and controls.
- **Fluent Bit** for log processing and forwarding. See Implementing Log Forwarding with Fluent Bit
- **Prometheus** and **Grafana** for monitoring. See Implementing Monitoring with Prometheus and Grafana
- **Multus** for multi networking. Implementing Multiple Pod Network Interfaces with Multus

Logs and Troubleshooting

For information about how to find the Tanzu Kubernetes Grid logs, how to troubleshoot frequently encountered Tanzu Kubernetes Grid issues, and how to use the Crash Recovery and Diagnostics tool, see Logs and Troubleshooting.

Delete Clusters

The procedures in this section are optional. They are provided in case you want to clean up your production or lab environment.

Delete a Workload Cluster

To delete a provisioned workload cluster, first set your context back to the management cluster.

```
kubectl config use-context [mgmt_cluster_name]-admin@[mgmt_cluster_name]
```

From the management cluster context, run:

```
tanzu cluster delete <cluster_name>
```

Delete a Management Cluster

Use this procedure to delete the management cluster as well as all of the AWS objects that Tanzu Kubernetes Grid created, such as VPC, subnets, and NAT Gateways.

> Be sure to wait until all the workload clusters have been reconciled before deleting the management cluster, or you will need to manually clean up the infrastructure.

Run the following command to delete the management cluster and related objects:

```
tanzu cluster delete <management-cluster-name>
```

Tanzu Kubernetes Grid Upgrade

For information about how to upgrade to Tanzu Kubernetes Grid 2.1, see Tanzu Kubernetes Grid Upgrade.

VMware Tanzu Kubernetes Grid on vSphere Networking in an Air-Gapped Environment Reference Design

VMware Tanzu for Kubernetes Operations (informally known as TKO) simplifies operating Kubernetes for multi-cloud deployment by centralizing management and governance for clusters and teams across on-premises, public clouds, and edge. Tanzu for Kubernetes Operations delivers an open-source aligned Kubernetes distribution with consistent operations and management to support infrastructure and application modernization.

An air-gapped environment is a network security measure employed to ensure a computer or computer network is secure by physically isolating it from unsecured networks, such as the public Internet or an unsecured local area network. This means a computer or network is disconnected from all other systems.

This document lays out a reference design for deploying Tanzu Kubernetes Grid on vSphere Networking in an air-gapped environment and offers a high-level overview of the different components required for setting up a Tanzu Kubernetes Grid environment.

Supported Component Matrix

The following table provides the component versions and interoperability matrix supported with the reference design:

Software Components	Version
Tanzu Kubernetes Grid	2.3.0
VMware vSphere ESXi	8.0 U1
VMware vCenter (VCSA)	8.0 U1
VMware vSAN	8.0 U1
NSX Advanced LB	22.1.3

For the latest information, see VMware Product Interoperability Matrix.

Infrastructure Components

The following components are used in the reference architecture:

- **Tanzu Kubernetes Grid Instance (TKG)** - A Tanzu Kubernetes Grid instance is a full deployment of Tanzu Kubernetes Grid, including the management cluster, the deployed workload clusters, and the shared and in-cluster services that you configure..

- **NSX Advanced Load Balancer** - Tanzu Kubernetes Grid leverages NSX Advanced Load Balancer to provide L4 load balancing for the Tanzu Kubernetes Clusters Control-Plane HA, and L4/L7 ingress to the applications deployed in the Tanzu Kubernetes Clusters. Users access the applications by connecting to the Virtual IP address (VIP) of the Virtual Service provisioned by NSX ALB
- **User-Managed Tanzu Packages:**
  - **Cert Manager** - Provides automated certificate management. It runs by default in management clusters.
  - **Contour** - Provides layer 7 ingress control to deployed HTTP(S) applications. Tanzu Kubernetes Grid includes signed binaries for Contour. Deploying Contour is a prerequisite for deploying the Prometheus, Grafana, and Harbor extensions.
  - **Fluent Bit** - Collects data and logs from different sources, unifies them, and sends them to multiple destinations. Tanzu Kubernetes Grid includes signed binaries for Fluent Bit.
  - **Prometheus** - Provides out-of-the-box health monitoring of Kubernetes clusters. The Tanzu Kubernetes Grid implementation of Prometheus includes Alert Manager. You can configure Alert Manager to notify you when certain events occur.
  - **Grafana** - Provides monitoring dashboards for displaying key health metrics of Kubernetes clusters. Tanzu Kubernetes Grid includes an implementation of Grafana.
  - **Harbor Image Registry** - Provides a centralized location to push, pull, store, and scan container images used in Kubernetes workloads. It supports storing artifacts and includes enterprise-grade features such as role-based access control (RBAC), retention policies, automated garbage cleanup, and Docker hub proxying.
  - **Multus CNI** - Enables attaching multiple network interfaces to pods. Multus CNI is a container network interface (CNI) plugin for Kubernetes that lets you attach multiple network interfaces to a single pod and associate each with a different address range.
- **Bastion Host -** Bastion host is the physical/virtual machine where you download the required installation images/binaries (for TKG installation) from the internet. This machine needs to be outside the air-gapped environment. The downloaded items then need to be shipped to the bootstrap machine which is inside the air-gapped environment.
- **Local Image Registry -** An image registry provides a location for pushing, pulling, storing, and scanning container images used in the Tanzu Kubernetes Grid environment. The image registry is also used for day-2 operations of the Tanzu Kubernetes clusters, such as storing application images, upgrading Tanzu Kubernetes clusters, and so forth.

In an air-gapped environment, there are two ways to deploy an image registry:

- **Existing Image Registry -** An image registry pre-existing in the environment with a project created for storing TKG binaries. The bootstrap machine has access to this registry. After unzipping the tarball present at the bootstrap machine, the operator uses a script included in the tarball to push the TKG binaries to the TKG project. This registry can be a Harbor registry or any other container registry solution.
- **New Image Registry -** If there is no pre-existing image registry in the environment, a new registry instance can be deployed. The easiest way to create a new image registry instance is VM-based deployment using OVA and then push the TKG binaries to the appropriate project. VM-based

deployments are only supported by VMware Global Support Services to host the system images for air-gapped or Internet-restricted deployments. Do not use this method for hosting application images.

Tanzu Kubernetes Grid Components

VMware Tanzu Kubernetes Grid (TKG) provides organizations with a consistent, upstream-compatible, regional Kubernetes substrate that is ready for end-user workloads and ecosystem integrations. You can deploy Tanzu Kubernetes Grid across software-defined datacenters (SDDC) and public cloud environments, including vSphere, Microsoft Azure, and Amazon EC2.

Tanzu Kubernetes Grid comprises the following components:

Management Cluster - A management cluster is the first element that you deploy when you create a Tanzu Kubernetes Grid instance. The management cluster is a Kubernetes cluster that performs the role of the primary management and operational center for the Tanzu Kubernetes Grid instance. The management cluster is purpose-built for operating the platform and managing the lifecycle of Tanzu Kubernetes clusters.

ClusterClass API - Tanzu Kubernetes Grid 2 functions through the creation of a management Kubernetes cluster which houses ClusterClass API. The ClusterClass API then interacts with the infrastructure provider to service workload Kubernetes cluster lifecycle requests.The earlier primitives of Tanzu Kubernetes Clusters will still exist for Tanzu Kubernetes Grid 1.X . A new feature has been introduced as a part of Cluster API called ClusterClass which reduces the need for redundant templating and enables powerful customization of clusters. The whole process for creating a cluster using ClusterClass is the same as before but with slightly different parameters.

Workload Clusters - Workload Clusters are the Kubernetes clusters in which your application workloads run. These clusters are also referred to as Tanzu Kubernetes clusters. Workload Clusters can run different versions of Kubernetes, depending on the needs of the applications they run.

Shared Service Cluster - Each Tanzu Kubernetes Grid instance can only have one shared services cluster. You will deploy this cluster only if you intend to deploy shared services such as Contour and Harbor.

Tanzu Kubernetes Cluster Plans - A cluster plan is a blueprint that describes the configuration with which to deploy a Tanzu Kubernetes cluster. It provides a set of configurable values that describe settings like the number of control plane machines, worker machines, VM types, and so on. This release of Tanzu Kubernetes Grid provides two default templates, dev and prod.

Tanzu Kubernetes Grid Instance - A Tanzu Kubernetes Grid instance is the full deployment of Tanzu Kubernetes Grid, including the management cluster, the workload clusters, and the shared services cluster that you configure.

Tanzu CLI - A command-line utility that provides the necessary commands to build and operate Tanzu management and Tanzu Kubernetes clusters. Starting with TKG 2.3.0, Tanzu Core CLI is now distributed separately from Tanzu Kubernetes Grid. For instructions on how to install the Tanzu CLI for use with Tanzu Kubernetes Grid, refer Install the Tanzu CLI

Carvel Tools - Carvel is an open-source suite of reliable, single-purpose, composable tools that aid in building, configuring, and deploying applications to Kubernetes. Tanzu Kubernetes Grid uses the following Carvel tools:

- **ytt -** A command-line tool for templating and patching YAML files. You can also use `ytt` to collect fragments and piles of YAML into modular chunks for reuse.

- **kapp -** The application deployment CLI for Kubernetes. It allows you to install, upgrade, and delete multiple Kubernetes resources as one application.
- **kbld -** An image-building and resolution tool.
- **imgpkg -** A tool that enables Kubernetes to store configurations and the associated container images as OCI images, and to transfer these images.
- **yq -** a lightweight and portable command-line YAML, JSON, and XML processor. `yq` uses `jq`-like syntax but works with YAML files as well as JSON and XML.

Bootstrap Machine - The bootstrap machine is the laptop, host, or server on which you download and run the Tanzu CLI. This is where the initial bootstrapping of a management cluster occurs before it is pushed to the platform where it will run.

Tanzu Kubernetes Grid Installer - The Tanzu Kubernetes Grid installer is a graphical wizard that you launch by running the `tanzu management-cluster create --ui` command. The installer wizard runs locally on the bootstrap machine and provides a user interface to guide you through the process of deploying a management cluster.**Carvel Tools -** Carvel is an open-source suite of reliable, single-purpose, composable tools that aid in building, configuring, and deploying applications to Kubernetes.

Tanzu Kubernetes Grid Storage

Many storage options are available and Kubernetes is agnostic about which option you choose.

For Kubernetes stateful workloads, Tanzu Kubernetes Grid installs the vSphere Container Storage interface (vSphere CSI) to provision Kubernetes persistent volumes for pods automatically. While the default vSAN storage policy can be used, site reliability engineers (SREs) and administrators should evaluate the needs of their applications and craft a specific vSphere Storage Policy. vSAN storage policies describe classes of storage such as SSD and NVME, as well as cluster quotas.

In vSphere 7u1+ environments with vSAN, the vSphere CSI driver for Kubernetes also supports creating NFS File Volumes, which support ReadWriteMany access modes. This allows for provisioning volumes which can be read and written from multiple pods simultaneously. To support this, the vSAN File Service must be enabled.

You can also use other types of vSphere datastores. There are Tanzu Kubernetes Grid Cluster Plans that operators can define to use a certain vSphere datastore when creating new workload clusters. All developers would then have the ability to provision container-backed persistent volumes from that underlying datastore.

Tanzu Kubernetes Clusters Networking

A Tanzu Kubernetes cluster provisioned by the Tanzu Kubernetes Grid supports two Container Network Interface (CNI) options:

- Antrea
- Calico

Both are open-source software that provide networking for cluster pods, services, and ingress. When you deploy a Tanzu Kubernetes cluster using Tanzu CLI with the default configuration, Antrea CNI is automatically enabled in the cluster.

Tanzu Kubernetes Grid also supports Multus CNI that can be installed through Tanzu user-managed packages. Multus CNI lets you attach multiple network interfaces to a single pod and associate each interface with a different address range.

To provision a Tanzu Kubernetes cluster using a non-default CNI, see the following instructions:

- Deploy Tanzu Kubernetes clusters with calico
- Implement Multiple Pod Network Interfaces with Multus

Each CNI is suitable for a different use case. The following table lists common use cases for the three CNIs that Tanzu Kubernetes Grid supports. This table will help you with information on selecting the right CNI in your Tanzu Kubernetes Grid implementation.

CNI	Use Case	Pros and Cons
Antrea	Enable Kubernetes pod networking with IP overlay networks using VXLAN or Geneve for encapsulation. Optionally encrypt node-to-node communication using IPSec packet encryption. Antrea supports advanced network use cases like kernel bypass and network service mesh.	**Pros:** - Provide an option to Configure Egress IP Pool or Static Egress IP for the Kubernetes Workloads.
Calico	Calico is used in environments where factors like network performance, flexibility, and power are essential. For routing packets between nodes, Calico leverages the BGP routing protocol instead of an overlay network. This eliminates the need to wrap packets with an encapsulation layer resulting in increased network performance for Kubernetes workloads.	**Pros:** - Support for Network Policies. - High network performance. - SCTP Support. **Cons:** - No multicast support.
Multus	Multus CNI provides multiple interfaces per each Kubernetes pod. Using Multus CRDs, you can specify which pods get which interfaces and allow different interfaces depending on the use case.	Pros - Separation of data/control planes. - Separate security policies can be used for separate interfaces. - Supports SR-IOV, DPDK, OVS-DPDK, and VPP workloads in Kubernetes with both cloud native and NFV based applications in Kubernetes.

Tanzu Kubernetes Grid Infrastructure Networking

Tanzu Kubernetes Grid on vSphere can be deployed on various networking stacks including

- VMware NSX-T Data Center Networking.
- vSphere Networking (VDS).

The scope of this document is limited to vSphere Networking.

TKG on vSphere Networking with NSX Advanced Load Balancer

Tanzu Kubernetes Grid when deployed on the vSphere networking uses the distributed port groups to provide connectivity to Kubernetes control plane VMs, worker nodes, services, and applications. All hosts from the cluster where Tanzu Kubernetes clusters are deployed are connected to the distributed switch that provides connectivity to the Kubernetes environment.

You can configure NSX Advanced Load Balancer in Tanzu Kubernetes Grid as:

- L4 load balancer for an application hosted on the TKG cluster.
- The L7 ingress service provider for the application hosted on the TKG cluster.
- L4 load balancer for the Kubernetes cluster control plane API server.

Each workload cluster integrates with NSX Advanced Load Balancer by running an Avi Kubernetes Operator (AKO) on one of its nodes. The cluster's AKO calls the Kubernetes API to manage the lifecycle of load balancing and ingress resources for its workloads.

NSX Advanced Load Balancer Licensing

NSX ALB requires a license to enable and utilize the available load balancing features. VMware Tanzu for Kubernetes Operations supports the following license editions:

- VMware NSX Advance Load Balancer Enterprise Edition.
- VMware NSX Advanced Load Balancer essentials for Tanzu.

The Enterprise Edition is the default licensing tier for an Avi Controller. A new Avi Controller is set up in the Enterprise Edition licensing tier, and the Controller can be switched from one edition to another. For more information about NSX ALB Feature comparison, see NSX Advanced Load Balancer Editions.

VMware NSX ALB Enterprise Edition

The VMware NSX ALB Enterprise Edition is a full-featured Avi Vantage license that includes load balancing, GSLB, WAF, and so on.

For more information about VMware NSX ALB Enterprise edition, see VMware NSX ALB Enterprise Edition.

VMware NSX Advanced Load Balancer essentials for Tanzu

Starting with Avi Vantage release 20.1.2, VMware NSX ALB essentials for Tanzu edition is supported on Avi Vantage. NSX ALB essentials for Tanzu has been introduced to provide basic Layer 4 load balancing services for Tanzu Basic and Standard edition customers.

For more information on VMware NSX ALB essentials for Tanzu edition, see VMware NSX ALB essentials for Tanzu.

NSX Advanced Load Balancer Components

NSX Advanced Load Balancer is deployed in Write Access Mode in the vSphere environment. This mode grants NSX Advanced Load Balancer Controller full write access to the vCenter which helps in automatically creating, modifying, and removing service engines (SEs) and other resources as needed to adapt to changing traffic needs. The core components of NSX Advanced Load Balancer are as follows:

- **NSX Advanced Load Balancer Controller** - NSX Advanced Load Balancer Controller manages Virtual Service objects and interacts with the vCenter Server infrastructure to manage the lifecycle of the service engines (SEs). It is the central repository for the configurations and policies related to services and management, and it provides the portal for viewing the health of VirtualServices and SEs and the associated analytics that NSX Advanced Load Balancer provides.
- **NSX Advanced Load Balancer Service Engine** - The service engines (SEs) are lightweight VMs that handle all data plane operations by receiving and executing instructions from the controller. The SEs perform load balancing and all client- and server-facing network interactions.
- **Cloud -** Clouds are containers for the environment that NSX Advanced Load Balancer is installed or operating within. During initial setup of NSX Advanced Load Balancer, a default cloud, named Default-Cloud, is created. This is where the first controller is deployed into Default-Cloud. Additional clouds may be added, containing SEs and virtual services.
- **Avi Kubernetes Operator (AKO)** - It is a Kubernetes operator that runs as a pod in the Supervisor Cluster and Tanzu Kubernetes clusters, and it provides ingress and load balancing functionality. AKO translates the required Kubernetes objects to NSX Advanced Load Balancer objects and automates the implementation of ingresses, routes, and services on the service engines (SE) through the NSX Advanced Load Balancer Controller.
- **AKO Operator (AKOO)** - This is an operator which is used to deploy, manage, and remove the AKO pod in Kubernetes clusters. This operator when deployed creates an instance of the AKO controller and installs all the relevant objects like:
  - AKO `StatefulSet`
  - `ClusterRole` and `ClusterRoleBinding`
  - `ConfigMap` required for the AKO controller and other artifacts.

Tanzu Kubernetes Grid management clusters have an AKO operator installed out of the box during cluster deployment. By default, a Tanzu Kubernetes Grid management cluster has a couple of AkoDeploymentConfig created which dictates when and how AKO pods are created in the workload clusters. For more information, see AKO Operator documentation.

Optionally, you can enter one or more cluster labels to identify clusters on which to selectively enable NSX ALB or to customize NSX ALB settings for different groups of clusters. This is useful in the following scenarios: - You want to configure different sets of workload clusters to different Service Engine Groups to implement isolation or to support more Service type Load Balancers than one Service Engine Group’s capacity. - You want to configure different sets of workload clusters to different Clouds because they are deployed in different sites.

To enable NSX ALB selectively rather than globally, add labels in the format **key: value** pair in the management cluster config file. This will create a default AKO Deployment Config (ADC) on management cluster with the NSX ALB settings provided. Labels that you define here will be used to create a label selector. Only workload cluster objects that have the matching labels will have the load balancer enabled.

To customize the NSX ALB settings for different groups of clusters, create an AKO Deployment Config (ADC) on management cluster by customizing the NSX ALB settings, and providing a unique label selector for the ADC. Only the workload cluster objects that have the matching labels will have these custom settings applied.

You can label the cluster during the workload cluster deployment or label it manually post cluster creation. If you define multiple key-values, you need to apply all of them. - Provide an AVI_LABEL in the below format in the workload cluster deployment config file, and it will automatically label the cluster and select the matching ADC based on the label selector during the cluster deployment. `AVI_LABELS: | 'type': 'tkg-workloadset01'` - Optionally, you can manually label the cluster object of the corresponding workload cluster with the labels defined in ADC. `kubectl label cluster <cluster-name> type=tkg-workloadset01`

Each environment configured in NSX Advanced Load Balancer is referred to as a cloud. Each cloud in NSX Advanced Load Balancer maintains networking and NSX Advanced Load Balancer Service Engine settings. The cloud is configured with one or more VIP networks to provide IP addresses to load balancing (L4 or L7) virtual services created under that cloud.

The virtual services can span across multiple service engines if the associated Service Engine Group is configured in the Active/Active HA mode. A service engine can belong to only one Service Engine group at a time.

IP address allocation for virtual services can be over DHCP or using NSX Advanced Load Balancer in-built IPAM functionality. The VIP networks created or configured in NSX Advanced Load Balancer are associated with the IPAM profile.

Tanzu Kubernetes Grid Clusters Recommendations

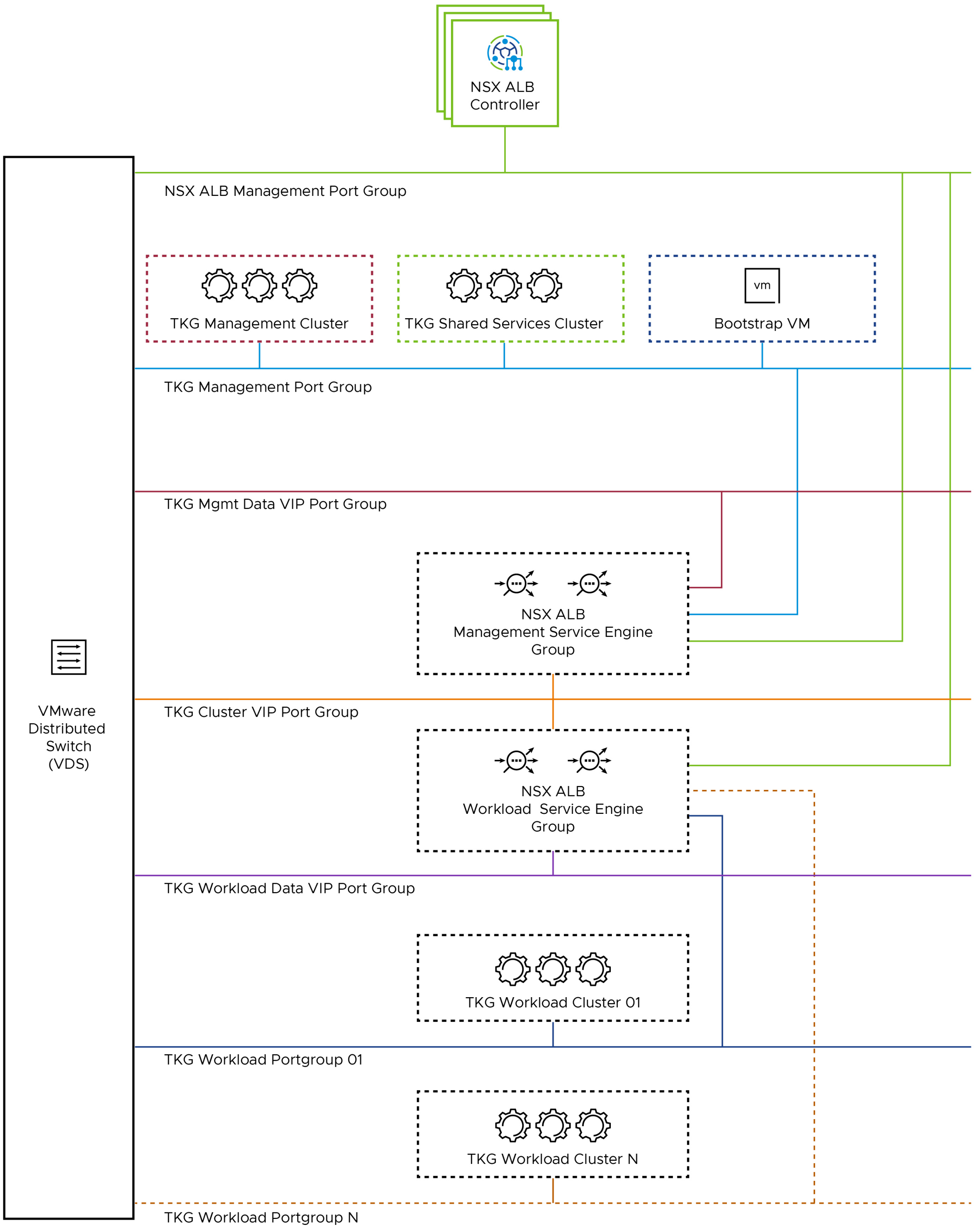
Decision ID	Design Decision	Design Justification	Design Implications
TKO-TKG-001	Use NSX Advanced Load Balancer as your control plane endpoint provider and for application load balancing.	AVI is tightly coupled with TKG and vSphere. Since AVI is a VMware product customers will have single point of contact for Support.	Adds NSX Advanced Load Balancer License Cost to the solution
TKO-TKG-002	Deploy Tanzu Kubernetes Management clusters in large form factor	Large form factor should suffice for pinniped and velero deployment. This must be capable of accommodating 100+ Tanzu Workload Clusters	Consume more Resources from Infrastructure.
TKO-TKG-003	Deploy Tanzu Kubernetes clusters with prod plan.	This deploys multiple control plane nodes and provides high availability for the control plane.	Consume more Resources from Infrastructure.
TKO-TKG-004	Enable identity management for Tanzu Kubernetes Grid clusters.	Role-based access control to Tanzu Kubernetes Grid clusters.	Required External Identity Management
TKO-TKG-005	Enable Machine Health Checks for TKG clusters.	MachineHealthCheck controller helps to provide health monitoring and auto-repair for management and workload clusters Machines.	NA

Generic Network Architecture

For the deployment of Tanzu Kubernetes Grid in the vSphere environment, it is required to build separate networks for the Tanzu Kubernetes Grid management cluster and workload clusters, NSX Advanced Load

Balancer management, cluster-VIP network for control plane HA, Tanzu Kubernetes Grid management VIP or data network, and Tanzu Kubernetes Grid workload data or VIP network.

The network reference design can be mapped into this general framework:


NSX ALB Controller
NSX ALB Management Port Group
TKG Management Cluster
TKG Shared Services Cluster
vm
Bootstrap VM
TKG Management Port Group
TKG Mgmt Data VIP Port Group
NSX ALB Management Service Engine Group
VMware Distributed Switch (VDS)
TKG Cluster VIP Port Group
NSX ALB Workload Service Engine Group
TKG Workload Data VIP Port Group
TKG Workload Cluster 01
TKG Workload Portgroup 01
TKG Workload Cluster N
TKG Workload Portgroup N

This topology enables the following benefits:

- Isolate and separate SDDC management components (vCenter, ESX) from the Tanzu Kubernetes Grid components. This reference design allows only the minimum connectivity between the Tanzu

Kubernetes Grid clusters and NSX Advanced Load Balancer to the vCenter Server.

- Isolate and separate NSX Advanced Load Balancer management network from the Tanzu Kubernetes Grid management segment and the Tanzu Kubernetes Grid workload segments.
- Depending on the workload cluster type and use case, multiple workload clusters may leverage the same workload network or new networks can be used for each workload cluster. To isolate and separate Tanzu Kubernetes Grid workload cluster networking from each other it's recommended to make use of separate networks for each workload cluster and configure the required firewall between these networks. For more information, see Firewall Requirements.
- Separate provider and tenant access to the Tanzu Kubernetes Grid environment.
    - Only provider administrators need access to the Tanzu Kubernetes Grid management cluster. This prevents tenants from attempting to connect to the TKG management cluster.
- Only allow tenants to access their Tanzu Kubernetes Grid workload clusters and restrict access to this cluster from other tenants. network-architecture

Network Requirements

As per the defined architecture, the list of required networks is as follows:

Network Type	DHCP Service	Description & Recommendations
NSX ALB Management Network	Optional	NSX ALB controllers and SEs will be attached to this network. DHCP is not a mandatory requirement on this network as NSX ALB can take care of IPAM.
TKG Management Network	Yes	Control plane and worker nodes of the TKG Management cluster and Shared Service cluster will be attached to this network. Creating a shared service cluster on a separate network is also supported.
TKG Workload Network	Yes	Control plane and worker nodes of TKG Workload Clusters will be attached to this network.
TKG Cluster VIP/Data Network	No	Virtual services for Control plane HA of all TKG clusters (Management, Shared service, and Workload). Reserve sufficient IPs depending on the number of TKG clusters planned to be deployed in the environment, NSX ALB takes care of IPAM on this network.
TKG Management VIP/Data Network	No	Virtual services for all user-managed packages (such as Contour and Harbor) hosted on the Shared service cluster.
TKG Workload VIP/Data Network	No	Virtual services for all applications hosted on the Workload clusters. Reserve sufficient IPs depending on the number of applications that are planned to be hosted on the Workload clusters along with scalability considerations.

Network Recommendations

The key network recommendations for a production-grade Tanzu Kubernetes Grid deployment with VDS Networking are as follows:

Decision ID	Design Decision	Design Justification	Design Implications
TKO-NET-001	Use Dedicated networks for the Management Cluster Nodes and Workload Cluster Nodes.	To have a flexible firewall and security policies.	Additional VLAN Required (OPEX overhead)
TKO-NET-002	Use Dedicated VIP network for the Application Hosted in Management and Workload Cluster.	To have a flexible firewall and security policies.	Additional VLAN Required (OPEX overhead)
TKO-NET-003	Shared Service Cluster uses Management network and Application VIP network of Management Cluster.	Host Shared Services like Harbor.	VLAN Based Firewall Policies are not possible

With Tanzu Kubernetes Grid 2.3 and above, you can use Node IPAM, which simplifies the allocation and management of IP addresses for cluster nodes within the cluster. This eliminates the need for external DHCP configuration.

Node IPAM can be configured for standalone management clusters on vSphere, and the associated class-based workload clusters that they manage. In the Tanzu Kubernetes Grid Management configuration file, a dedicated Node IPAM pool is defined for the management cluster only.

The following types of Node IPAM pools are available for workload clusters:

- **InClusterIPPool -** Configures IP pools that are only available to workload clusters in the same management cluster namespace. For example, default.
- **GlobalInClusterIPPool -** Configures IP pools with addresses that can be allocated to workload clusters across multiple namespaces.

Node IPAM in TKG provides flexibility in managing IP addresses for both management and workload clusters that allows efficient IP allocation and management within the cluster environment.

Subnet and CIDR Examples

For the purpose of demonstration, this document makes use of the following Subnet CIDR for TKO deployment:

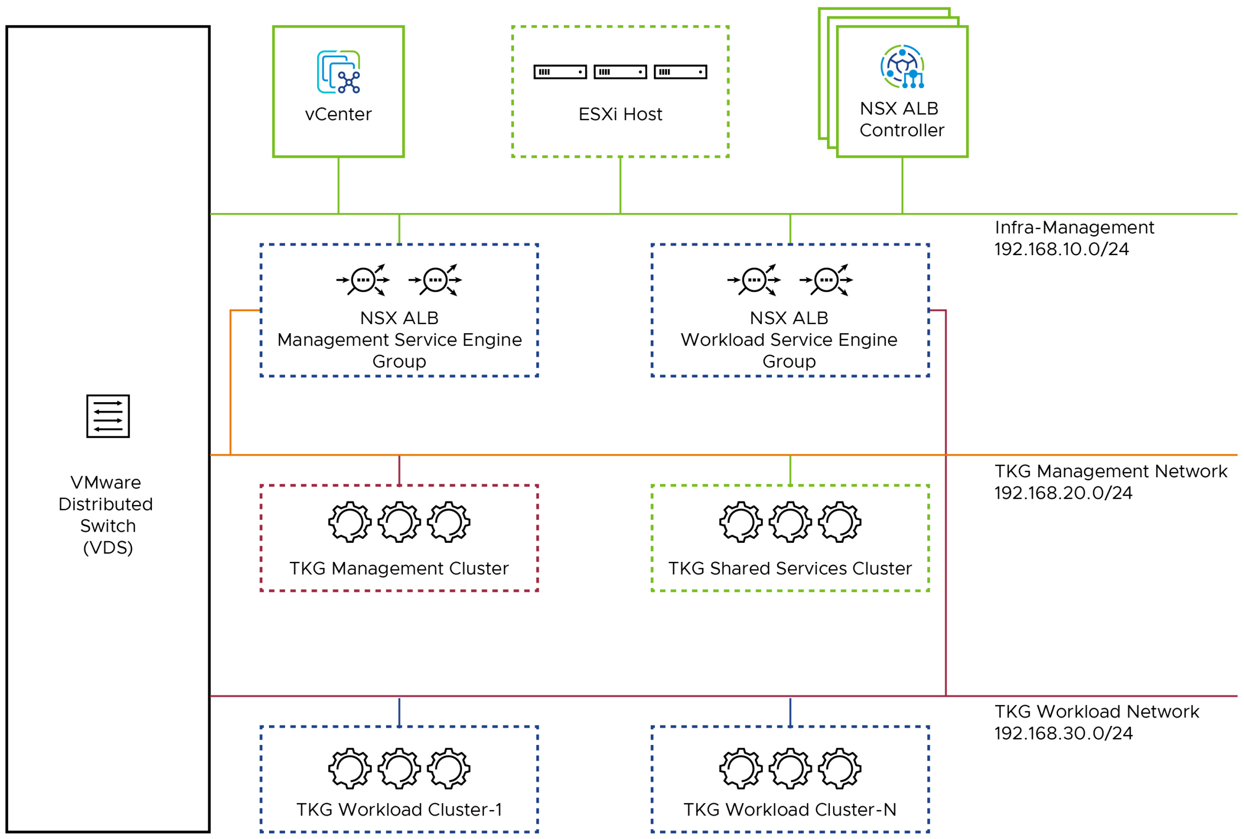
Network Type	Port Group Name	Gateway CIDR	DHCP Pool	NSX ALB IP Pool
NSX ALB Management Network	`sfo01-w01-vds01-albmanagement`	172.16.10.1/24	N/A	172.16.10.100-172.16.10.200
TKG Management Network	`sfo01-w01-vds01-tkgmanagement`	172.16.40.1/24	172.16.40.100-172.16.40.200	N/A
TKG Management VIP Network	`sfo01-w01-vds01-tkgmanagementvip`	172.16.50.1/24	N/A	172.16.50.100-172.16.50.200
TKG Cluster VIP Network	`sfo01-w01-vds01-tkgclustervip`	172.16.80.1/24	N/A	172.16.80.100-172.16.80.200
TKG Workload VIP Network	`sfo01-w01-vds01-tkgworkloadvip`	172.16.70.1/24	N/A	172.16.70.100 - 172.16.70.200

Network Type	Port Group Name	Gateway CIDR	DHCP Pool	NSX ALB IP Pool
TKG Workload Segment	`sfo01-w01-vds01-tkgshared`	172.16.60.1/24	172.16.60.100-172.16.60.200	N/A

3-Network Architecture

For POC environments and minimal networks requirement, you can proceed with 3 network architecture. In this design, we deploy the Tanzu Kubernetes Grid into 3 networks as Infrastructure Management Network, TKG Management Network and TKG Workload Network. This design allows us to use only 3 networks and ensures the isolation between Infra VMs, TKG Management and TKG Workload components.

This network reference design can be mapped into this general framework:


vCenter
ESXi Host
NSX ALB Controller
Infra-Management 192.168.10.0/24
NSX ALB Management Service Engine Group
NSX ALB Workload Service Engine Group
VMware Distributed Switch (VDS)
TKG Management Network 192.168.20.0/24
TKG Management Cluster
TKG Shared Services Cluster
TKG Workload Network 192.168.30.0/24
TKG Workload Cluster-1
TKG Workload Cluster-N

This topology enables the following benefits: - Deploy the NSX ALB components on the existing infrastructure management network which reduces an additional network usage. - Isolate and separate the NSX ALB, SDDC management components (vCenter and ESX) from the VMware Tanzu Kubernetes Grid components. - Club TKG Mgmt Cluster VIP, TKG Mgmt Data VIP, TKG Mgmt into a single network `TKG-Mgmt-Network`, that ensures that the TKG Management components are deployed in a common network, and removes additional network overhead and firewall rules. - Club TKG Workload Cluster VIP, TKG Workload Data VIP, TKG Workload into a single network `TKG-Workload-Network`, that ensures that the TKG Workload components are deployed in a common network. - Separate the Management control plane/Data VIP and the Workload control plane/Data VIP into different networks to enhance the isolation and security.

Network Requirements

Network Type	DHCP Service	Description
Infrastructure Management Network	Optional	NSX ALB controllers and Service Engines (SE) are attached to this network. DHCP is not a mandatory requirement on this network as NSX ALB manages the SE networking with IPAM. This network also hosts core infrastructure components such as, vCenter, ESXi hosts, DNS, NTP, and so on.
TKG Management Network	Yes	Control plane and worker nodes of the TKG Management cluster and the shared services clusters are attached to this network. The IP Assignment is managed through DHCP. TKG Management cluster VIP and TKG Management Data VIP assignment is also managed from the same network using NSX ALB Static IP pool. Ensure that DHCP range does not interfere with the NSX ALB IP Block reservation.
TKG Workload Network	Yes	Control plane and worker nodes of the TKG Workload cluster and the shared services clusters are attached to this network. IP Assignment is managed done through DHCP. TKG Workload cluster VIP and TKG Workload Data VIP assignment is also managed from the same network using NSX ALB Static IP pool. Ensure that DHCP range does not interfere with the NSX ALB IP Block reservation.

Subnet and CIDR Examples:

Network Type	Gateway CIDR	DHCP Pool	NSX ALB IP Pool
Infrastructure Management	192.168.10.1/24	N/A	192.168.10.101-192.168.10.200
TKG Management Network	192.168.20.1/24	192.168.20.2 - 192.168.20.100	192.168.20.101 - 192.168.20.200
TKG Workload Network	192.168.30.1/24	192.168.30.2 - 192.168.30.100	192.168.30.101 - 192.168.30.200

Firewall Requirements

To prepare the firewall, you need to gather the following information:

1. NSX ALB Controller nodes and Cluster IP address.
2. NSX ALB Management Network CIDR
3. TKG Management Network CIDR.
4. TKG Workload Network CIDR.
5. TKG Cluster VIP Range.
6. TKG Management VIP Range.
7. TKG Workload VIP Range.
8. Bastion host IP address.
9. Bootstrap machine IP address.
10. VMware Harbor registry IP

11. vCenter Server IP
12. DNS server IP(s)
13. NTP Server IP(s)
14. DHCP Server IP(s)

The following table provides a list of firewall rules based on the assumption that there is no firewall within a subnet/VLAN.

Source	Destination	Protocol:Port	Description
Bastion Host	Internet	TCP:80/443	To download installation binaries required for TKG installation.
Bootstrap VM	vCenter Server	TCP:443	To create resource pools, VM folders and so on in vCenter.
Bootstrap VM	NSX ALB Controller nodes and Cluster IP Address.	TCP:443	To access the NSX ALB portal for configuration.
TKG Management Network CIDR TKG Workload Network CIDR.	DNS Server NTP Server	UDP:53 UDP:123	DNS Service Time Synchronization
TKG Management Network CIDR TKG Workload Network CIDR.	DHCP Server	UDP: 67, 68	Allows TKG nodes to get DHCP addresses.
TKG Management Network CIDR TKG Workload Network CIDR.	vCenter IP	TCP:443	Allows components to access vCenter to create VMs and Storage Volumes.
TKG Management Network CIDR TKG Workload Network CIDR.	Harbor Registry	TCP:443	Allows components to retrieve container images. This registry needs to be a private registry.
TKG Management Network CIDR TKG Workload Network CIDR.	TKG Cluster VIP Range. p> **Note:** In a 3 Network design, destination network is “TKG Mgmt Network”	TCP:6443	For the management cluster to configure shared services and workload clusters. Allow Workload cluster to register with management cluster.
TKG Management Network CIDR TKG Workload Network CIDR.	NSX ALB Controllers and Cluster IP Address.	TCP:443	Allow Avi Kubernetes Operator (AKO) and AKO Operator (AKOO) access to Avi Controller.

Source	Destination	Protocol:Port	Description
NSX Advanced Load Balancer Management Network	vCenter and ESXi Hosts	TCP:443	Allow NSX Advanced Load Balancer to discover vCenter objects and deploy SEs as required.
NSX Advanced Load Balancer Controller Nodes	DNS server NTP Server	TCP/UDP:53 UDP:123	DNS Service Time Synchronization
Admin network	Bootstrap VM	SSH:22	To deploy, manage, and configure TKG clusters.
deny-all	any	any	deny

NSX Advanced Load Balancer Recommendations

The following table provides the recommendations for configuring NSX Advanced Load Balancer in a vSphere with Tanzu environment.

Decision ID	Design Decision	Design Justification	Design Implications
TKO-ALB-001	Deploy NSX ALB controller cluster nodes on a network dedicated to NSX-ALB.	Isolate NSX ALB traffic from infrastructure management traffic and Kubernetes workloads.	Additional Network (VLAN) is required
TKO-ALB-002	Deploy 3 NSX ALB controllers nodes.	To achieve high availability for the NSX ALB platform. In clustered mode, NSX ALB availability is not impacted by an individual controller node failure. The failed node can be removed from the cluster and redeployed if recovery is not possible. Provides the highest level of uptime for a site	Additional resource requirements
TKO-ALB-003	Under Compute policies Create 'VM-VM anti-affinity' rule that prevents collocation of the NSX ALB Controllers VMs on the same host.	vSphere will take care of placing the NSX Advanced Load Balancer Controller VMs in a way that always ensures maximum HA.	Affinity Rules needs to be configured manually.
TKO-ALB-004	Use static IP addresses for the NSX ALB controllers.	NSX ALB Controller cluster uses management IP addresses to form and maintain quorum for the control plane cluster. Any changes to management IP addresses will be disruptive.	None
TKO-ALB-005	Use NSX ALB IPAM for service engine data network and virtual services.	Simplify the IP address management for Virtual Service and Service engine from NSX ALB	None
TKO-ALB-006	Reserve an IP address in the NSX ALB management subnet to be used as the cluster IP address for the controller cluster.	NSX ALB portal is always accessible over cluster IP address regardless of a specific individual controller node failure.	Additional IP is required.
TKO-ALB-007	Create a dedicated resource pool with appropriate reservations for NSX ALB controllers.	Guarantees the CPU and Memory allocation for NSX ALB Controllers and avoids performance degradation in case of resource contention.	None

Decision ID	Design Decision	Design Justification	Design Implications
TKO-ALB-008	Replace default NSX ALB certificates with Custom CA or Public CA-signed certificates that contains SAN entries of all Controller nodes	To establish a trusted connection with other infra components, and the default certificate doesn't include SAN entries which is not acceptable by Tanzu.	None, SAN entries are not applicable if using wild card certificate
TKO-ALB-009	Configure NSX ALB backup with a remote server as backup location	Periodic backup of NSX ALB configuration database is recommended. The database defines all clouds, all virtual services, all users, and others. As best practice store backups in an external location to provide backup capabilities in case of entire cluster failure	Additional Operational Overhead. Additional infrastructure Resource.
TKO-ALB-010	Configure Remote logging for NSX ALB Controller to send events on Syslog.	For operations teams to be able to centrally monitor NSX ALB and escalate alerts events must be sent from the NSX ALB Controller	Additional Operational Overhead. Additional infrastructure Resource.
TKO-ALB-011	Use LDAP/SAML based Authentication for NSX ALB	Helps to Maintain Role based Access Control	Additional Configuration is required

NSX Advanced Load Balancer Service Engine Recommendations

Decision ID	Design Decision	Design Justification	Design Implications
TKO-ALB-SE-001	NSX ALB Service Engine High Availability set to Active/Active	Provides higher resiliency, optimum performance, and utilization compared to N+M and/or Active/Standby.	Requires NSX ALB Enterprise Licensing. Only the Active/Standby mode is supported with NSX ALB essentials for Tanzu license. Certain applications might not work in the Active/Active mode. For example, applications that preserve the client IP use the Legacy Active/Standby HA mode.
TKO-ALB-SE-002	Dedicated Service Engine Group for the TKG Management	SE resources are guaranteed for TKG Management Stack and provides data path segregation for Management and Tenant Application	Dedicated service engine Groups increase licensing cost.
TKO-ALB-SE-003	Dedicated Service Engine Group for the TKG Workload Clusters Depending on the nature and type of workloads (dev/prod/test)	SE resources are guaranteed for single or set of workload clusters and provides data path segregation for Tenant Application hosted on workload clusters	Dedicated service engine Groups increase licensing cost.
TKO-ALB-SE-004	Enable ALB Service Engine Self Elections	Enable SEs to elect a primary amongst themselves in the absence of connectivity to the NSX ALB controller	Requires NSX ALB Enterprise Licensing. This feature is not supported with NSX ALB essentials for Tanzu license.

Decision ID	Design Decision	Design Justification	Design Implications
TKO-ALB-SE-005	Enable 'Dedicated dispatcher CPU' on Service Engine Groups that contain the Service Engine VMs of 4 or more vCPUs. **Note:** This setting should be enabled on SE Groups that are servicing applications and has high network requirements.	This will enable a dedicated core for packet processing enabling high packet pipeline on the Service Engine VMs. **Note:** By default, the packet processing core also processes load-balancing flows.	Consume more Resources from Infrastructure.
TKO-ALB-SE-006	Set 'Placement across the Service Engines' setting to 'Compact'.	This allows maximum utilization of capacity (Service Engine).	None
TKO-ALB-SE-007	Set the SE size to a minimum 2vCPU and 4GB of Memory	This configuration should meet the most generic use case	For services that require higher throughput, these configuration needs to be investigated and modified accordingly.
TKO-ALB-SE-008	Under Compute policies Create a 'VM-VM anti-affinity rule for SE engines part of the same SE group that prevents collocation of the Service Engine VMs on the same host.	vSphere will take care of placing the Service Engine VMs in a way that always ensures maximum HA for the Service Engines part of a Service Engine group	Affinity Rules needs to be configured manually.
TKO-ALB-SE-009	Reserve Memory and CPU for Service Engines.	The Service Engines are a critical infrastructure component providing load-balancing services to mission-critical applications. Guarantees the CPU and Memory allocation for SE VM and avoids performance degradation in case of resource contention	You must perform additional configuration to set up the reservations.

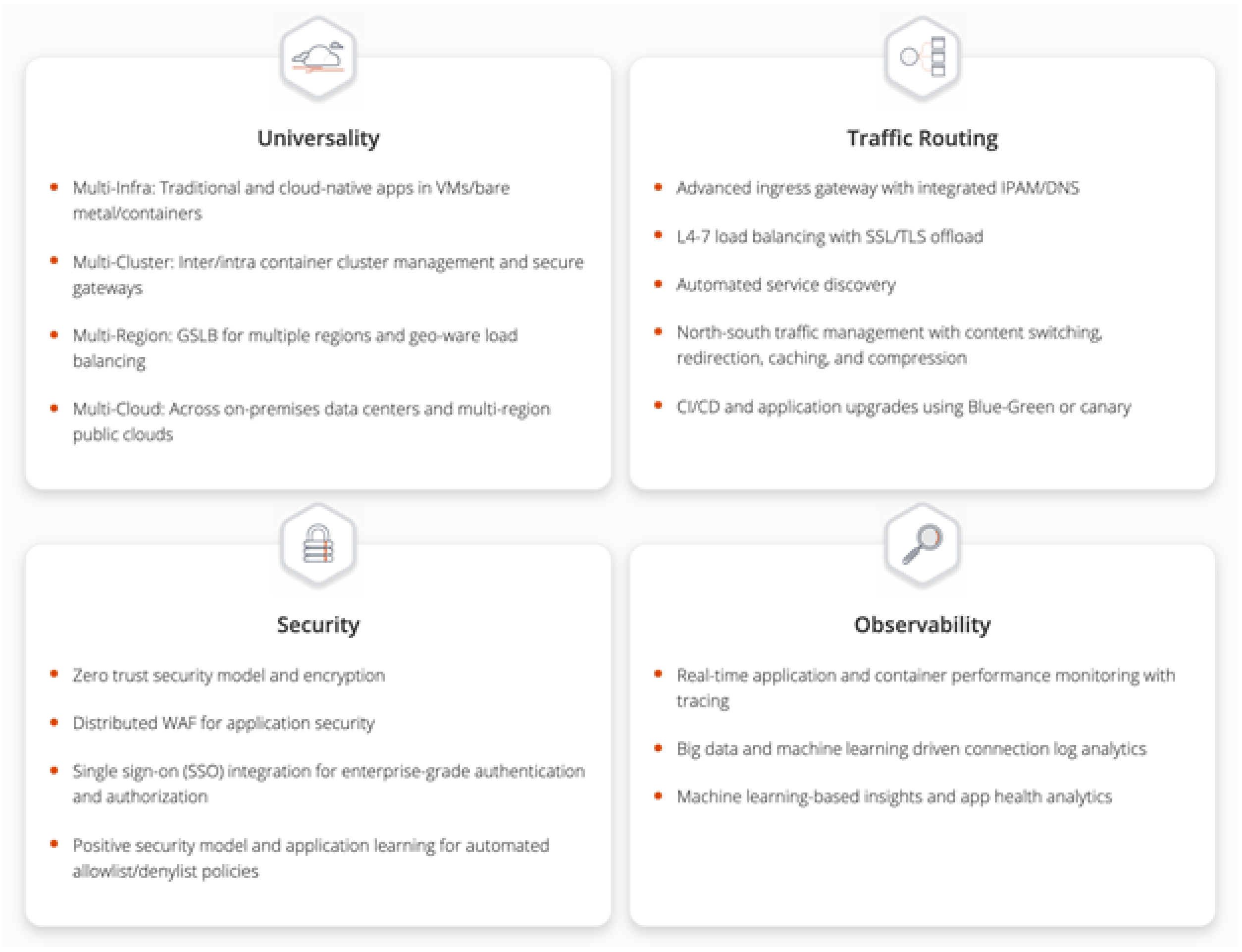
Kubernetes Ingress Routing

The default installation of Tanzu Kubernetes Grid does not have any ingress controller installed. Users can use Contour (available for installation through Tanzu Packages) or any third-party ingress controller of their choice.

Contour is an open-source controller for Kubernetes ingress routing. Contour can be installed in the shared services cluster on any Tanzu Kubernetes Cluster. Deploying Contour is a prerequisite if you want to deploy the Prometheus, Grafana, and Harbor Packages on a workload cluster.

For more information about Contour, see the Contour site and Implementing Ingress Control with Contour.

Another option is to use the NSX Advanced Load Balancer Kubernetes ingress controller (available only with the NSX ALB Enterprise license) which offers an advanced L7 ingress for containerized applications that are deployed in the Tanzu Kubernetes workload cluster.



Universality

- Multi-Infra: Traditional and cloud-native apps in VMs/bare metal/containers
- Multi-Cluster: Inter/intra container cluster management and secure gateways
- Multi-Region: GSLB for multiple regions and geo-ware load balancing
- Multi-Cloud: Across on-premises data centers and multi-region public clouds

Traffic Routing

- Advanced ingress gateway with integrated IPAM/DNS
- L4-7 load balancing with SSL/TLS offload
- Automated service discovery
- North-south traffic management with content switching, redirection, caching, and compression
- CI/CD and application upgrades using Blue-Green or canary

Security

- Zero trust security model and encryption
- Distributed WAF for application security
- Single sign-on (SSO) integration for enterprise-grade authentication and authorization
- Positive security model and application learning for automated allowlist/denylist policies

Observability

- Real-time application and container performance monitoring with tracing
- Big data and machine learning driven connection log analytics
- Machine learning-based insights and app health analytics

For more information about the NSX Advanced Load Balancer ingress controller, see Configuring L7 Ingress with NSX Advanced Load Balancer.

Tanzu Service Mesh, which is a SaaS offering for modern applications running across multi-cluster, multi-clouds, also offers an ingress controller based on Istio.

The following table provides general recommendations on when you should use a specific ingress controller for your Kubernetes environment.

Ingress Controller	Use Cases
Contour	Use contour when only north-south traffic is needed in a Kubernetes cluster. You can apply security policies for north-south traffic by defining the policies in the applications manifest file. Contour is a reliable solution for simple Kubernetes workloads.
NSX ALB Ingress controller	Use the NSX ALB ingress controller when a containerized application requires features like local and global server load balancing (GSLB), web application firewall, performance monitoring, direct routing from LB to pod and so on.
Istio	Use Istio ingress controller when you intend to provide security, traffic direction, and insights within the cluster (east-west traffic) and between the cluster and the outside world (north-south traffic).

NSX ALB as in L4+L7 Ingress Service Provider

As a load balancer, NSX Advanced Load Balancer provides an L4+L7 load balancing solution for vSphere. It includes a Kubernetes operator that integrates with the Kubernetes API to manage the lifecycle of load balancing and ingress resources for workloads.

Legacy ingress services for Kubernetes include multiple disparate solutions. The services and products contain independent components that are difficult to manage and troubleshoot. The ingress services have reduced observability capabilities with little analytics, and they lack comprehensive visibility into the applications that run on the system. Cloud-native automation is difficult in the legacy ingress services.

In comparison to the legacy Kubernetes ingress services, NSX Advanced Load Balancer has comprehensive load balancing and ingress services features. As a single solution with a central control, NSX Advanced Load Balancer is easy to manage and troubleshoot. NSX Advanced Load Balancer supports real-time telemetry with an insight into the applications that run on the system. The elastic auto-scaling and the decision automation features highlight the cloud-native automation capabilities of NSX Advanced Load Balancer.

NSX ALB with Enterprise Licensing also lets you configure L7 ingress for your workload clusters by using one of the following options:

- L7 ingress in ClusterIP mode
- L7 ingress in NodePortLocal mode
- L7 ingress in NodePort mode
- NSX ALB L4 ingress with Contour L7 ingress

L7 Ingress in ClusterIP Mode

This option enables NSX Advanced Load Balancer L7 ingress capabilities, including sending traffic directly from the service engines (SEs) to the pods, preventing multiple hops that other ingress solutions need when sending packets from the load balancer to the right node where the pod runs. The ALB controller creates a virtual service with a backend pool with the pod IP addresses which helps to send the traffic directly to the pods.

However, each workload cluster needs a dedicated SE group for Avi Kubernetes Operator (AKO) to work, which could increase the number of SEs you need for your environment. This mode is used when you have a small number of workload clusters.

L7 Ingress in NodePort Mode

The NodePort mode is the default mode when AKO is installed on Tanzu Kubernetes Grid. This option allows your workload clusters to share SE groups and is fully supported by VMware. With this option, the services of your workloads must be set to NodePort instead of ClusterIP even when accompanied by an ingress object. This ensures that NodePorts are created on the worker nodes and traffic can flow through the SEs to the pods via the NodePorts. Kube-Proxy, which runs on each node as DaemonSet, creates network rules to expose the application endpoints to each of the nodes in the format “NodeIP:NodePort”. The NodePort value is the same for a service on all the nodes. It exposes the port on all the nodes of the Kubernetes Cluster, even if the pods are not running on it.

L7 Ingress in NodePortLocal Mode

This feature is supported only with Antrea CNI. The primary difference between this mode and the NodePort mode is that the traffic is sent directly to the pods in your workload cluster through node ports without interfering Kube-proxy. With this option, the workload clusters can share SE groups. Similar to the ClusterIP mode, this option avoids the potential extra hop when sending traffic from the NSX Advanced Load Balancer SEs to the pod by targeting the right nodes where the pods run.

Antrea agent configures NodePortLocal port mapping rules at the node in the format “NodeIP:Unique Port” to expose each pod on the node on which the pod of the service is running. The default range of the port number is 61000-62000. Even if the pods of the service are running on the same Kubernetes node, Antrea agent publishes unique ports to expose the pods at the node level to integrate with the load balancer.

NSX ALB L4 Ingress with Contour L7 Ingress

This option does not have all the NSX Advanced Load Balancer L7 ingress capabilities but uses it for L4 load balancing only and leverages Contour for L7 Ingress. This also allows sharing SE groups across workload clusters. This option is supported by VMware and it requires minimal setup.

NSX Advanced Load Balancer L7 Ingress Recommendations

Decision ID	Design Decision	Design Justification	Design Implications
TKO-ALB-L7-001	Deploy NSX ALB L7 ingress in NodePortLocal mode.	1. Gives good Network hop efficiency. 2. Helps to reduce the east-west traffic and encapsulation overhead. 3.Service Engine groups are shared across clusters and the load-balancing persistence is also supported.	Supported only with Antrea CNI with IPV4 addressing. To configure L7 Ingress, you need NSX ALB Enterprise Licensing.

Container Registry

VMware Tanzu for Kubernetes Operations using Tanzu Kubernetes Grid includes Harbor as a container registry. Harbor provides a location for pushing, pulling, storing, and scanning container images used in your Kubernetes clusters.

Harbor registry is used for day-2 operations of the Tanzu Kubernetes workload clusters. Typical day-2 operations include tasks such as pulling images from Harbor for application deployment, pushing custom images to Harbor and so on.

You may use one of the following methods to install Harbor:

- **Tanzu Kubernetes Grid Package deployment** - VMware recommends this installation method for general use cases. The Tanzu packages, including Harbor, must either be pulled directly from VMware or be hosted in an internal registry.
- **VM-based deployment using OVA** - VMware recommends this installation method in cases where Tanzu Kubernetes Grid is being installed in an air-gapped or Internet-restricted environment, and no pre-existing image registry exists to host the Tanzu Kubernetes Grid system images. VM-based deployments are only supported by VMware Global Support Services to host the system images for air-gapped or Internet-restricted deployments. Do not use this method for hosting application images.

If you are deploying Harbor without a publicly signed certificate, you must include the Harbor root CA in your Tanzu Kubernetes Grid clusters. To do so, follow the procedure in Trust Custom CA Certificates on Cluster Nodes.

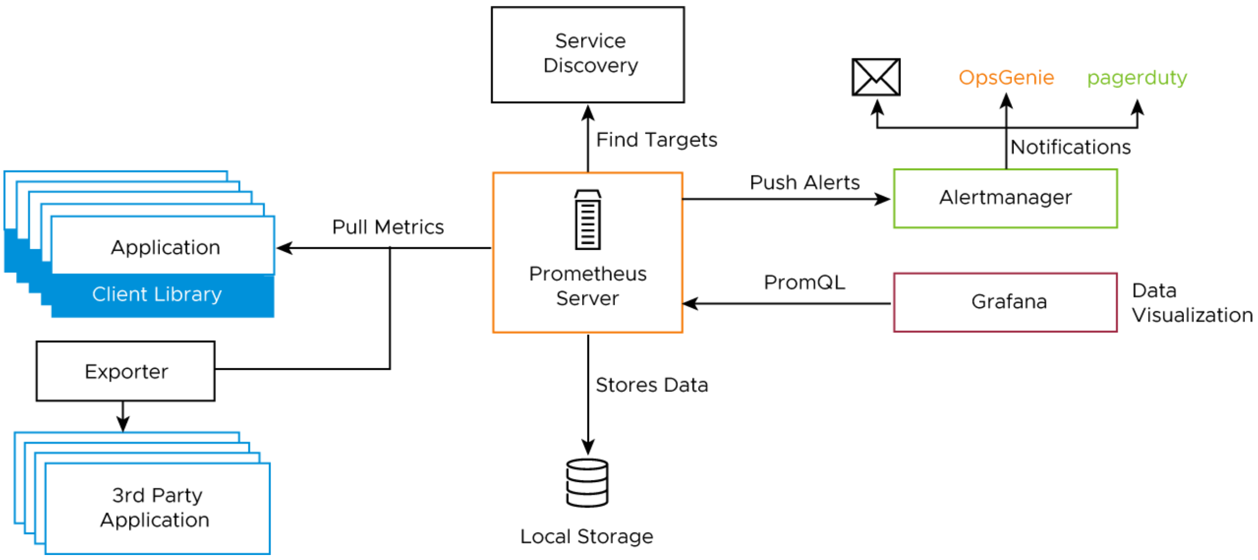
Tanzu Kubernetes Grid Monitoring

In an air-gapped environment, monitoring for the Tanzu Kubernetes clusters is provided through Prometheus and Grafana. Tanzu Kubernetes Grid includes signed binaries for Prometheus and Grafana that you can deploy on Tanzu Kubernetes clusters to monitor cluster health and services

- Prometheus is an open-source system monitoring and alerting toolkit. It can collect metrics from target clusters at specified intervals, evaluate rule expressions, display the results, and trigger alerts if certain conditions arise. The Tanzu Kubernetes Grid implementation of Prometheus includes **Alert Manager**, which you can configure to notify you when certain events occur.
- Grafana is open-source visualization and analytics software. It allows you to query, visualize, alert on, and explore your metrics no matter where they are stored.

Both Prometheus and Grafana are installed via user-managed Tanzu packages by creating the deployment manifests and invoking the kubectl command to deploy the packages in the Tanzu Kubernetes clusters.

The following diagram shows how the monitoring components on a cluster interact.



You can use out-of-the-box Kubernetes dashboards or can create new dashboards to monitor compute, network, or storage utilization of Kubernetes objects such as Clusters, Namespaces, Pods and so on.

Tanzu Kubernetes Grid Logging

Metrics and logs are critical for any system or application as they provide insights into the system's or application's activity. It is important to have a central place to observe a multitude of metrics and log sources from multiple endpoints.

Log processing and forwarding in Tanzu Kubernetes Grid is provided via Fluent Bit. Fluent bit binaries are available as part of extensions and can be installed on management cluster or in workload cluster. Fluent Bit is a light-weight log processor and forwarder that allows you to collect data and logs from different sources, unify them, and send them to multiple destinations. VMware Tanzu Kubernetes Grid includes signed binaries for Fluent Bit that you can deploy on management clusters and on Tanzu Kubernetes clusters to provide a log-forwarding service.

Fluent Bit makes use of the Input Plug-ins, the filters, and the Output Plug-ins. The Input Plug-ins define the source from where it can collect data, and the Output plug-ins define the destination where it should send the information. The Kubernetes filter will enrich the logs with Kubernetes metadata, specifically labels and annotations. Once you configure Input and Output plug-ins on the Tanzu Kubernetes Grid cluster. Fluent Bit is installed as a user-managed package.

Fluent Bit integrates with logging platforms such as VMware Aria Operations for Logs, Elasticsearch, Kafka, Splunk, or an HTTP endpoint. For more details about configuring Fluent Bit to your logging provider, see Implement Log Forwarding with Fluent Bit.

Bring Your Own Images for Tanzu Kubernetes Grid Deployment

You can build custom machine images for Tanzu Kubernetes Grid to use as a VM template for the management and Tanzu Kubernetes (workload) cluster nodes that it creates. Each custom machine image packages a base operating system (OS) version and a Kubernetes version, along with any additional customizations, into an image that runs on vSphere, Microsoft Azure infrastructure, and AWS (EC2) environments.

A custom image must be based on the operating system (OS) versions that are supported by Tanzu Kubernetes Grid. The table below provides a list of the operating systems that are supported for building custom images for TKG.

vSphere	AWS	Azure
- Ubuntu 20.04 - Ubuntu 18.04 - RHEL 8 - Photon OS 3 - Windows 2019	- Ubuntu 20.04 - Ubuntu 18.04 - Amazon Linux 2	- Ubuntu 20.04 - Ubuntu 18.04

Refer Tanzu Kubernetes Releases and Custom Node Images

For additional information on building custom images for Tanzu Kubernetes Grid, see the Build Machine Images

- Linux Custom Machine Images
- Windows Custom Machine Images

Compliance and Security

VMware published Tanzu Kubernetes releases (TKrs), along with compatible versions of Kubernetes and supporting components. Use the latest stable and generally-available update of the OS version that it packages, containing all current CVE and USN fixes, as of the day that the image is built. The image files are signed by VMware and have file names that contain a unique hash identifier.

VMware provides FIPS-capable Kubernetes OVA which can be used to deploy FIPS compliant TKG management and workload clusters. Tanzu Kubernetes Grid core components such as Kubelet, Kube-apiserver, Kube-controller manager, Kube-proxy, Kube-scheduler, Kubectl, Etcd, Coredns, Containerd, and Cri-tool are made FIPS compliant by compiling them with the BoringCrypto FIPS modules, an open-source cryptographic library that provides FIPS 140-2 approved algorithms.

Summary

Tanzu Kubernetes Grid on vSphere on hyper-converged hardware offers high-performance potential, convenience, and addresses the challenges of creating, testing, and updating on-premises Kubernetes platforms in a consolidated production environment. This validated approach will result in a near-production quality installation with all the application services needed to serve combined or uniquely separated workload types through a combined infrastructure solution.

This plan meets many Day-0 needs for quickly aligning product capabilities to full stack infrastructure, including networking, firewalling, load balancing, workload compute alignment, and other capabilities.

Appendix A - Configure Node Sizes

The Tanzu CLI creates the individual nodes of management clusters and Tanzu Kubernetes clusters according to the settings that you provide in the configuration file.

On vSphere, you can configure all node VMs to have the same predefined configurations, set different predefined configurations for control plane and worker nodes, or customize the configurations of the nodes. By using these settings, you can create clusters that have nodes with different configurations to the management cluster nodes. You can also create clusters in which the control plane nodes and worker nodes have different configurations.

Use Predefined Node Configurations

The Tanzu CLI provides the following predefined configurations for cluster nodes:

Size	CPU	Memory (in GB)	Disk (in GB)
Small	2	4	20
Medium	2	8	40
Large	4	16	40
Extra-large	8	32	80

To create a cluster in which all of the control plane and worker node VMs are the same size, specify the `SIZE` variable. If you set the `SIZE` variable, all nodes will be created with the configuration that you set.

- `SIZE: "large"`

To create a cluster in which the control plane and worker node VMs are different sizes, specify the `CONTROLPLANE_SIZE` and `WORKER_SIZE` options.

- `CONTROLPLANE_SIZE: "medium"`
- `WORKER_SIZE: "large"`

You can combine the `CONTROLPLANE_SIZE` and `WORKER_SIZE` options with the `SIZE` option. For example, if you specify `SIZE: "large"` with `WORKER_SIZE: "extra-large"`, the control plane nodes will be set to large and worker nodes will be set to extra-large.

- `SIZE: "large"`
- `WORKER_SIZE: "extra-large"`

Define Custom Node Configurations

You can customize the configuration of the nodes rather than using the predefined configurations.

To use the same custom configuration for all nodes, specify the `VSPHERE_NUM_CPUS`, `VSPHERE_DISK_GIB`, and `VSPHERE_MEM_MIB` options.

- `VSPHERE_NUM_CPUS: 2`
- `VSPHERE_DISK_GIB: 40`
- `VSPHERE_MEM_MIB: 4096`

To define different custom configurations for control plane nodes and worker nodes, specify the `VSPHERE_CONTROL_PLANE_*` and `VSPHERE_WORKER_*`

- `VSPHERE_CONTROL_PLANE_NUM_CPUS: 2`
- `VSPHERE_CONTROL_PLANE_DISK_GIB: 20`
- `VSPHERE_CONTROL_PLANE_MEM_MIB: 8192`
- `VSPHERE_WORKER_NUM_CPUS: 4`
- `VSPHERE_WORKER_DISK_GIB: 40`
- `VSPHERE_WORKER_MEM_MIB: 4096`

Appendix B - NSX Advanced Load Balancer Sizing Guidelines

NSX Advanced Load Balancer Controller Sizing Guidelines

Controllers are classified into the following categories:

Classification	vCPUs	Memory (GB)	Virtual Services	Avi SE Scale
Essentials	4	24	0-50	0-10
Small	6	24	0-200	0-100
Medium	10	32	200-1000	100-200
Large	16	48	1000-5000	200-400

The number of virtual services that can be deployed per controller cluster is directly proportional to the controller cluster size. For more information, see the NSX Advanced Load Balancer Configuration Maximums Guide.

Service Engine Sizing Guidelines

The service engines can be configured with a minimum of 1 vCPU core and 2 GB RAM up to a maximum of 64 vCPU cores and 256 GB RAM. The following table provides guidance for sizing a service engine VM with regards to performance:

Performance metric	Per core performance	Maximum performance on a single Service Engine VM
HTTP Throughput	5 Gbps	7 Gbps
HTTP requests per second	50k	175k
SSL Throughput	1 Gbps	7 Gbps
SSL TPS (RSA2K)	750	40K
SSL TPS (ECC)	2000	40K

Multiple performance vectors or features may have an impact on performance. For instance, to achieve 1 Gb/s of SSL throughput and 2000 TPS of SSL with EC certificates, NSX Advanced Load Balancer recommends two cores.

Deploy Tanzu Kubernetes Grid on vSphere Networking in an Air-Gapped Environment

VMware Tanzu Kubernetes Grid (informally known as TKG) provides a consistent, upstream-compatible, regional Kubernetes substrate that is ready for end-user workloads and ecosystem integrations.

An air-gap installation method is used when the Tanzu Kubernetes Grid bootstrapper and cluster nodes components are unable to connect to the Internet to download the installation binaries from the public VMware Registry during Tanzu Kubernetes Grid installation or upgrades.

The scope of this document is limited to providing deployment steps based on the reference design in Tanzu Kubernetes Grid on vSphere Networking. This document does not provide any deployment procedures for the underlying SDDC components.

Supported Component Matrix

The following table provides the component versions and interoperability matrix supported with the reference design:

Software Components	Version
Tanzu Kubernetes Grid	2.3.0
VMware vSphere ESXi	8.0U1 and later
VMware vCenter (VCSA)	8.0U1 and later
VMware vSAN	8.0U1 and later
NSX Advanced LB	22.1.3

For the latest information, see VMware Product Interoperability Matrix.

Prepare your Environment for Deploying Tanzu Kubernetes Grid

Before deploying Tanzu Kubernetes Grid in vSphere environment, ensure that your environment is set up as described in the following sections:

- General Requirements
- Network Requirements
- Firewall Requirements

General Requirements

- vSphere 8.0 U1 or greater instance with an Enterprise Plus license
- Your Software-defined data center (SDDC) environment has the following objects in place:
  - A vSphere cluster with at least 3 hosts, on which vSphere DRS is enabled. If you are using vSAN for shared storage, it is recommended that you use 4 ESXi hosts.
  - A distributed switch with port groups for Tanzu Kubernetes Grid components. For more information about port groups, see Network Requirements.
  - Dedicated resource pool where you can deploy the Tanzu Kubernetes Grid instance.
  - VM folders where you can collect the Tanzu Kubernetes Grid VMs.
  - A shared datastore with sufficient capacity for the control plane and worker node VMs.
  - Network Time Protocol (NTP) service is running on all ESXi hosts and vCenter, and time is synchronized from the centralized NTP servers.
  - A host/server/VM based on Linux that acts as your **bastion host** and is located **outside the Internet-restricted environment (i.e. connected to the Internet)**. The installation binaries for Tanzu Kubernetes Grid and NSX Advanced Load Balancer will be downloaded on this machine. You will need to transfer files from this bastion host to your Internet-restricted environment (proxy connection, shared drive, USB drive, sneakernet, and so on).
  - A host/server/VM **inside your Internet-restricted environment** based on Linux/Windows, which acts as your bootstrap machine and has Tanzu CLI, Kubectl and docker installed. This document uses a virtual machine based on CentOS. An internal Harbor registry will be installed on the same machine.
- vSphere account with permissions as described in Required Permissions for the vSphere Account.

You can download and import supported older versions of Kubernetes in order to deploy workload clusters on the intended Kubernetes versions. In Tanzu Kubernetes Grid nodes, it is recommended to not use hostnames with “.local” domain suffix. For more information, see KB article

Resource Pools and VM Folders

The sample entries of the resource pools and folders that need to be created are as follows.

Resource Type	Sample Resource Pool Name	Sample Folder Name
NSX ALB Components	`tkg-alb-components`	`tkg-alb-components`
TKG Management components	`tkg-management-components`	`tkg-management-components`
TKG Shared Service Components	`tkg-sharedsvc-components`	`tkg-sharedsvc-components`
TKG Workload components	`tkg-workload01-components`	`tkg-workload01-components`

Network Requirements

Create Port groups on vSphere Distributed Switch for deploying Tanzu Kubernetes Grid components as defined in Network Requirements in the reference architecture.

Firewall Requirements

Ensure that the firewall is set up as described in Firewall Requirements.

Subnet and CIDR Examples

For the purpose of this demonstration, this document uses the following CIDR for TKO deployment. Please change the values to reflect your environment.

Network Type	Port Group Name	Gateway CIDR	DHCP Pool	NSX ALB IP Pool
NSX ALB Management Network	sfo01-w01-vds01-albmanagement	172.16.10.1/24	N/A	172.16.10.100-172.16.10.200
TKG Management Network	sfo01-w01-vds01-tkgmanagement	172.16.40.1/24	172.16.40.100-172.16.40.200	N/A
TKG Management VIP Network	sfo01-w01-vds01-tkgmanagementvip	172.16.50.1/24	N/A	172.16.50.100-172.16.50.200
TKG Cluster VIP Network	sfo01-w01-vds01-tkgclustervip	172.16.80.1/24	N/A	172.16.80.100-172.16.80.200
TKG Workload VIP Network	sfo01-w01-vds01-tkgworkloadvip	172.16.70.1/24	N/A	172.16.70.100 - 172.16.70.200
TKG Workload Segment	sfo01-w01-vds01-tkgworkload	172.16.60.1/24	172.16.60.100-172.16.60.200	N/A

These are sample network ranges. Modify them according to your environment and security requirements.

Deployment Overview

Here are the high-level steps for deploying Tanzu Kubernetes Grid on vSphere VDS networking in an air-gapped environment:

- Deploy and Configure Bastion Host

- Install Harbor Image Registry
- Deploy and Configure Bootstrap VM
- Deploy and Configure NSX Advanced Load Balancer
- Deploy Tanzu Kubernetes Grid Management Cluster
- Deploy Tanzu Kubernetes Grid Shared Services Cluster
- Deploy Tanzu Kubernetes Grid Workload Cluster
- Deploy User-Managed Packages

Deploy and Configure Bastion Host

The bastion host is the physical or virtual machine where you download the images and binaries required for Tanzu Kubernetes Grid installation from the Internet. You will then transfer the downloaded items to the bootstrap machine, which is located inside the air-gapped environment.

You must ensure that the following options are available: - A browser is available on the bastion host to download the binaries from the Internet. - The bastion host has the following hardware configuration: - CPU: 1 - Memory: 4 GB - Storage (HDD): 200 GB or greater.

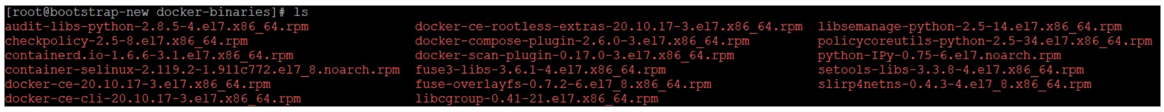
> The following instructions are for CentOS 7. If you are using any other operating system for your bastion host, change the commands accordingly.

Prerequisites

1. Download the binaries for Docker Engine and associated dependencies.

   ```
   ### Create a directory for collecting docker installation binaries

   mkdir docker-binaries && cd docker-binaries

   ### Add docker repository to the yum command

   yum install yum-utils -y

   yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-c
   e.repo

   ### Download docker and associated dependencies

   yumdownloader --resolve docker-ce docker-ce-cli containerd.io docker-compose-pl
   ugin
   ```

   The `yumdownloader` command downloads the following binaries:

   ```
   [root@bootstrap-new docker-binaries]# ls
   audit-libs-python-2.8.5-4.el7.x86_64.rpm             docker-ce-rootless-extras-20.10.17-3.el7.x86_64.rpm  libsemanage-python-2.5-14.el7.x86_64.rpm
   checkpolicy-2.5-8.el7.x86_64.rpm                     docker-compose-plugin-2.6.0-3.el7.x86_64.rpm         policycoreutils-python-2.5-34.el7.x86_64.rpm
   containerd.io-1.6.6-3.1.el7.x86_64.rpm               docker-scan-plugin-0.17.0-3.el7.x86_64.rpm           python-IPy-0.75-6.el7.noarch.rpm
   container-selinux-2.119.2-1.911c772.el7_8.noarch.rpm fuse3-libs-3.6.1-4.el7.x86_64.rpm                    setools-libs-3.3.8-4.el7.x86_64.rpm
   docker-ce-20.10.17-3.el7.x86_64.rpm                  fuse-overlayfs-0.7.2-6.el7_8.x86_64.rpm              slirp4netns-0.4.3-4.el7_8.x86_64.rpm
   docker-ce-cli-20.10.17-3.el7.x86_64.rpm              libcgroup-0.41-21.el7.x86_64.rpm
   ```

2. Download Harbor OVA from Broadcom Support.
3. Download the NSX Advanced Load Balancer OVA from Broadcom Support.

4. Download Tanzu CLI, Kubectl, and the Kubernetes OVA images from the Tanzu Kubernetes Grid product download page. Tanzu CLI and its plug-ins need to be installed on the bastion host and the bootstrap machine. Starting with TKG 2.3.0, Tanzu Core CLI is now distributed separately from Tanzu Kubernetes Grid. For more infromation about installing the Tanzu CLI for using with Tanzu Kubernetes Grid, see Install the Tanzu CLI.

5. Download the yq installation binary from the mikefarah / yq GitHub repository.

Configure Bastion Host

1. Install Tanzu CLI.

   ```
   tar -xvf tanzu-cli-linux-amd64.tar
   cd ./v0.90.1/
   install tanzu-cli-linux_amd64 /usr/local/bin/tanzu
   chmod +x /usr/local/bin/tanzu
   ```

   Run the `tanzu version` command to check that the correct version of Tanzu is installed and executable.

   ```
   # tanzu version

   version: v0.90.1
   buildDate: 2023-06-29
   sha: 8945351c
   ```

2. Install the Tanzu CLI plug-ins.

   ```
   tanzu plugin group search
   [i] Reading plugin inventory for "projects.registry.vmware.com/tanzu_cli/plugin
   s/plugin-inventory:latest", this will take a few seconds.
   GROUP                DESCRIPTION      LATEST
   vmware-tkg/default   Plugins for TKG  v2.3.0

   tanzu plugin install --group vmware-tkg/default
   [i] Installing plugin 'isolated-cluster:v0.30.1' with target 'global'
   [i] Installing plugin 'management-cluster:v0.30.1' with target 'kubernetes'
   [i] Installing plugin 'package:v0.30.1' with target 'kubernetes'
   [i] Installing plugin 'pinniped-auth:v0.30.1' with target 'global'
   [i] Installing plugin 'secret:v0.30.1' with target 'kubernetes'
   [i] Installing plugin 'telemetry:v0.30.1' with target 'kubernetes'
   [ok] successfully installed all plugins from group 'vmware-tkg/default:v2.3.0'

   #Accept EULA
   tanzu config eula accept
   [ok] Marking agreement as accepted.
   ```

3. Download the Images.

   Before downloading the images, ensure that the disk partition where you download the images has 65 GB of available space.

   ```
   tanzu isolated-cluster download-bundle --source-repo <SOURCE-REGISTRY> --tkg-ve
   rsion <TKG-VERSION> --ca-certificate <SECURITY-CERTIFICATE>
   ```

  - SOURCE-REGISTRY is the IP address or the hostname of the registry where the images are stored.
  - TKG-VERSION is the version of Tanzu Kubernetes Grid that you want to deploy in the proxied or air-gapped environment.
  - SECURITY-CERTIFICATE is the security certificate of the registry where the images are stored. To bypass the security certificate validation, use –insecure, instead of –ca-certificate. Both the strings are optional. If you do not specify any value, the system validates the default server security certificate.

```
tanzu isolated-cluster download-bundle --source-repo projects.registry.vmware.c
om/tkg --tkg-version v2.3.0
```

The image bundle in the form of TAR files, along with the `publish-images-fromtar.yaml` file, is downloaded . The YAML file defines the mapping between the images and the TAR files.

4. Download the Tanzu CLI plug-ins.

   Download the plugin-inventory image along with all selected plug-in images as a tar.gz file on the local disk of a machine which has internet access using the Tanzu plug-in download-bundle command.

```
tanzu plugin download-bundle --group vmware-tkg/default:v2.3.0 --to-tar plugin_
bundle_tkg_latest.tar.gz
```

5. Copy the files to the bootstrap machine after bootstrap machine deployment.

   Copy the following files to the offline machine, which is the bootstrap machine in the proxied or air-gapped environment, through a USB thumb drive or other medium: * Image TAR files * YAML files * Tanzu CLI plugins * Tanzu CLI, Kubectl & Carvel Tools - kbld, kapp, ytt and imgpkg

Install Harbor Image Registry

Install Harbor only if you don’t have any existing image repository in your environment.

For more information about deploying and configuring Harbor, see Deploy an Offline Harbor Registry on vSphere.

This VM-based harbor deployment is only supported for hosting the TKG system images in an internet-restricted or air-gapped environment. To deploy a scalable and highly-available Harbor that can manage large numbers of images for hosted apps in a production environment, deploy the Harbor package to TKG clusters as described in Install Harbor for Service Registry in Creating and Managing TKG 2.3 Workload Clusters with the Tanzu CLI.

Deploy and Configure Bootstrap VM

The deployment of the Tanzu Kubernetes Grid management and workload clusters is facilitated by setting up a bootstrap machine where you install the Tanzu CLI and Kubectl utilities which are used to create and manage the Tanzu Kubernetes Grid instance. This machine also keeps the Tanzu Kubernetes Grid and

Kubernetes configuration files for your deployments. The bootstrap machine can be a laptop, host, or server running on Linux, macOS, or Windows from where you deploy the management and the workload clusters.

The bootstrap machine runs a local `kind` cluster when Tanzu Kubernetes Grid management cluster deployment is started. Once the `kind` cluster is fully initialized, the configuration is used to deploy the actual management cluster on the backend infrastructure. After the management cluster is fully configured, the local `kind` cluster is deleted and future configurations are performed with the Tanzu CLI.

For this deployment, a Photon-based virtual machine is used as the bootstrap machine. For more information about how configuring a macOS or a Windows machine, see Install the Tanzu CLI and Other Tools.

The bootstrap machine must meet the following prerequisites:

- A minimum of 6 GB of RAM, 2-core CPU, 160 GB storage.
- System time is synchronized with a Network Time Protocol (NTP) server.
- Docker and containerd binaries are installed. For instructions on how to install Docker, see Docker documentation.
- Ensure that the bootstrap VM is connected to the Tanzu Kubernetes Grid management network, `sfo01-w01-vds01-tkgmanagement`.

To install Tanzu CLI, Tanzu Plug-ins, and Kubectl utility on the bootstrap machine, perform the following instructions: 1. Copy files to the bootstrap machine.

Copy the following files downloaded in Bastion host through a USB thumb drive or other medium: * Image TAR files * YAML files * Tanzu CLI Plugins

1. Copy the following Linux CLI packages from Bastion host:
   - VMware Tanzu CLI v0.90.1 for Linux
   - kubectl cluster CLI v1.26.5 for Linux
   - Carvel tools - kbld, kapp, ytt & imgpkg
   - yq binaries
2. Execute the following commands to install Tanzu Kubernetes Grid CLI, kubectl CLIs, and Carvel tools:

```
## Install required packages
install tar, zip, unzip, wget

## Install Tanzu Kubernetes Grid CLI
tar -xvf tanzu-cli-linux-amd64.tar
cd ./v0.90.1/
install tanzu-cli-linux_amd64 /usr/local/bin/tanzu
chmod +x /usr/local/bin/tanzu

## Verify Tanzu CLI version

 # tanzu version

version: v0.90.1
buildDate: 2023-06-29
sha: 8945351c
```

3. Log in to the private registry on the offline machine:

   ```
   docker login <URL>

   docker login harbor.sfo01.rainpole.vmw
   Username: admin
   Password:
   WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
   Configure a credential helper to remove this warning. See
   https://docs.docker.com/engine/reference/commandline/login/#credentials-store

   Login Succeeded
   ```

   > If your private registry uses a self-signed certificate, save the CA certificate of the registry in `/etc/docker/certs.d/registry.example.com/ca.crt` and add certificate verification process by following the instructions in Adding Certificate Configuration for the Custom Registry.

4. Upload the images to the private registry:

   ```
   tanzu isolated-cluster upload-bundle --source-directory <SOURCE-DIRECTORY> --de
   stination-repo <DESTINATION-REGISTRY> --ca-certificate <SECURITY-CERTIFICATE>
   ```

   - SOURCE-DIRECTORY is the path to the location where the image TAR files are stored.
   - DESTINATION-REGISTRY is the path to the private registry where the images will be hosted in the air-gapped environment.
   - SECURITY-CERTIFICATE is the security certificate of the private registry where the images will be hosted in the proxied or air-gapped environment.

   ```
    Example: tanzu isolated-cluster upload-bundle --source-directory /home/test/tk
   g-images/ --destination-repo harbor.sfo01.rainpole.vmw/tkgm-images --ca-certifi
   cate /etc/docker/certs.d/harbor.sfo01.rainpole.vmw/harbor.sfo01.rainpole.vmw-c
   a.crt
   ```

5. Upload the CLI plug-ins bundle to harbor repository:

   ```
   tanzu plugin upload-bundle --tar ./plugin_bundle_tkg_latest.tar.gz --to-repo ha
   rbor.sfo01.rainpole.vmw/tkgm-images/
   ```

6. Run tanzu plugin source command to set default discovery source to the images uploaded in internal harbor registry:

   ```
   tanzu plugin source update default --uri harbor.sfo01.rainpole.vmw/tkgm-images/
   plugin-inventory:latest
   ```

   > we can skip step 4, 5 and 6 if the Bastion host accesses the private registry directly. You can directly upload the files from the Bastion host to the private

registry.

7. Install the kubectl utility.

```
gunzip kubectl-linux-v1.26.5+vmware.2.gz
mv gunzip kubectl-linux-v1.26.5+vmware.2 /usr/local/bin/kubectl && chmod +x /us
r/local/bin/kubectl
```

   Run `kubectl version --short=true` to check that the correct version of kubectl is installed and executable.

8. Configure the environment variables.

   By default the Tanzu global config file, `config.yaml`, which gets created when you first run `tanzu init` command, points to the repository URL https://projects.registry.vmware.com to fetch the Tanzu plug-ins for installation. Because there is no Internet in the environment, the commands fail after some time.

   To ensure that Tanzu Kubernetes Grid always pulls images from the local private registry, run the Tanzu `export` command to add `TKG_CUSTOM_IMAGE_REPOSITORY` to the global Tanzu CLI configuration file, `~/.config/tanzu/config.yaml`.

   If your image registry is configured with a public signed CA certificate, set the following environment variables.

```
export TKG_CUSTOM_IMAGE_REPOSITORY=custom-image-repository.io/yourproject

export TKG_CUSTOM_IMAGE_REPOSITORY_SKIP_TLS_VERIFY=false
```

   If your registry solution uses self-signed certificates, also add TKG_CUSTOM_IMAGE_REPOSITORY_CA_CERTIFICATE in base64-encoded format to the global Tanzu CLI configuration file. For self-signed certificates, set the following environment variables:

```
export TKG_CUSTOM_IMAGE_REPOSITORY=custom-image-repository.io/yourproject

export TKG_CUSTOM_IMAGE_REPOSITORY_SKIP_TLS_VERIFY=false

export TKG_CUSTOM_IMAGE_REPOSITORY_CA_CERTIFICATE LS0t[...]tLS0tLQ==

# Example
root@photon-a17e54311cf [ ~ ]# export TKG_CUSTOM_IMAGE_REPOSITORY=harbor.sfo01.
rainpole.vmw/tkgm-images
root@photon-a17e54311cf [ ~ ]# export TKG_CUSTOM_IMAGE_REPOSITORY_SKIP_TLS_VERI
FY=false
root@photon-a17e54311cf [ ~ ]# export TKG_CUSTOM_IMAGE_REPOSITORY_CA_CERTIFICAT
E=LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUZnekNDQTJ1Z0F3SUJBZ0lVUXhvVVV5VnpwU
VlkdUlXbStwL3dxZ0JDSFVrd0RRWUpLb1pJaHZjTkFRRUwKQlFBd1VURUxNQWtHQTFVRUJoTUNRMDR4
RERBS0JnTlZCQWdNQTFCRlN6RVFNQTRHQTFVRUJ3d0hRbVZwU21sdQpaekVQTUEwR0ExVUVDZ3dHVms
xM1lYSmxNUkV3RHdZRFZRUUREQWhJWVhKaWIzSkRRVEFlRncweU16QTRNRGN3Ck5qVTBOVGRhRncwek
16QTRNRFF3TmpVME5UZGFNRkV4Q3pBSkJnTlZCQVlUQWtOT01Rd3dDZ1lEVlFRSURBTlEKUlVzeEVEQ
U9CZ05WQkFjTUIwSmxhVXBwYm1jeER6QU5CZ05WQkFvTUJsWk5kMkZ5WlRFUk1BOEdBMVVFQXd3SQpT
R0Z5WW05eVEwRXdnZ0lpTUEwR0NTcUdTSWIzRFFFQkFRVUFBNElDRHdBd2dnSUtBb0lDQVFDckFtajB
VR1lUCjVobm5OeTNIMVdkcUhIejBHbFphWVpGQlhvQVhlWitZa2ljeXJhOFpFeVJCTFBKZVNOcldzMW
45dFp1RUg2WDgKM3JJbzUwdzhrZTYwYU9OOU80OVlVXZExZKzYxYUxrN1pkNGkzcU9IQTVKSFU1cE5De
```

```
nU0bThaQ1F1bUp2SzFVSQpwQ1lQNnFtNGxSQUFvWFVzWGZ0S24vRkJ4bGdReTNhUjJ1Y0IzdXA5UndD
RllDLzA5TVd5ZjErUmhja3ZvWExxRCmppUWx4aHZ0NFpxeG12b09KMi9lbUorTHBqbENZaXBRVkNwN3N
peVM1ZGIvRmw5U1VWSFVlRDhpdzgxKzZTaEEKenRpZXY1U20zbGE0UGw1cUU4Vm9EVEJTUXRpMmQzUX
B2M05IR0Z4UUNWMjVTM1BVcWxPQ1Z0dDFtSHRjZEhSUQpTOEN2SE9DVlczRHd5ZXB2Y3M4STJFYlU3M
0VQN3JKOEJtL21GQUFwd2F1NXpjMmtxRGJRclMrTjhEeWFpcGY4Ckgxa3FFVlU2VmEvbCtjZHFSbkc1
ZWVmM21vT3dKMDBPa2ZqSWtCUExSOC9iZVdDR0R4RVE1RkZlRXhxSTdhNUIKcm1FK25vTTBjMFRQSFp
5Z3c4UnZVU3hQbGJ3MFZkVzA5MHdQOHB3WGFGYVBPMmhtT2lvVGJiVkpkMVhqaGRFZAoxWTdmQktYKz
l6UUxjb3dlcmF5cDRaSUlSTTNTZnRaZkc5bjErN3pmcVBpamVoTGdiUkUrcHVaaUQwbi9xUXIvClNzO
WU5MXZQR1U2NW82VjYvL2Y5MVZmVHJRUmVJV0JWUWNubCtobGpPOVpUeHRRMlVwRnU3TDRwT2RJZ1JH
RTgKcDVjZE91TDF1cWltTi8rTW5hckV3K2JFalc0ajF3cnFQUUlEQVFBQm8xTXdVVEFkQmdOVkhRNEV
GZ1FVdEozbQpvR3RrcVJiVGl2ZFkzQWxnMTZUOWpCZ3dId1lEVlIwakJCZ3dGb0FVdEozbW9HdGtxUm
JUaXZkWTNBbGcxNlQ5CmpCZ3dEd1lEVlIwVEFFRSC9CQVV3QXdFQi96QU5CZ2txaGtpRzl3MEJBUXNGQ
UFPQ0FnRUFUMFBGcUduUjVRdEoKZm1PMTdwSmVDK0IwM3NOVGozS0VzZjUybmJXUnprWm9xNjMrRDZn
VURtU3FMMm1RSExRUy9WNWhadmIzZTFCTApDSXpEZmF5cG14K2k3MnlIbVRRNGRBUW5NT2hCUm8rMEN
VOGpZa0t4TjllN1NhOTlERk0yVE5Bb1pzcEs4ZEVUCmlUa0p2bHlHS2tyellkNEFvczN3dU9uVE85VW
xCK1FqQmI4TUVDc3lMR2U4VGxJMk4vOFdwMFMySG1QUHVNeU8KemJ0RUUrSWJXaW1qT1lLRU44cHlUY
3plelVnZHpGcXJ3bjVKdWZSejdockV6MnE5ZG9sYm0vTGRNK3pnUHVzZwo0a3lnVjlxWU5KcldvVExxX
bFJKNzRyQmFOZFpYM3BOV1VGRGtjQ3JkSkloWFFESWdWUDNWa2xJZ21zUUhTU3IrCktJYSt5R1p3MU5
hODZML2R0djNrR1ovN2VRMHNHVzVpS3R3VmY5UnBqYTdXL3ZhTTA1OTdFWGNSSGZ2cHRxeFQKcjNOcn
FTQmZkTlJtNStXOVh3c0RxNDl3dFdERE52OHNDS2JrenI2Q0JHYUxXSHFGRWtCOHpiTGlJQVBYd2Vqa
ApSOTh5TnY1ZjBzb0ltZlg1R3REY2RMZjd3dGg0UGlvRGloZklrRXVzd0twVGN1WWo3clR0SnFYTFV2
b25jZkV0Ck93cytHa2c5L0ZHd0p6ZkJlYUNrNHVXbGw5bC9JanZ6azdydkc0Z1VXa2tMVi9SdmVzbFl
EZVlSUXBnYzRybzgKdFFpMThvb3V4RGZuMTlSS2JPVjNtNm5uTlYwdzlHdjZiUGxqbjlRaDB0MWJOaU
xwZThJeWhGb0VPOFpVYTBnSApCZ29PbXlGZHQ0VTlQclIvZTdNcWZzM2tQVjZkbmtzPQotLS0tLUVOR
CBDRVJUSUZJQ0FURS0tLS0t
```

If you reboot the VM, the above configuration will be set to default.

9. Install the Tanzu CLI plug-ins.

```
root@photon-a17e54311cf [ ~ ]# tanzu plugin group search
GROUP                DESCRIPTION      LATEST
vmware-tkg/default   Plugins for TKG  v2.3.0

root@photon-a17e54311cf [ ~ ]# tanzu plugin install --group vmware-tkg/default
[i] Installing plugin 'isolated-cluster:v0.30.1' with target 'global'
[i] Plugin binary for 'isolated-cluster:v0.30.1' found in cache
[i] Installing plugin 'management-cluster:v0.30.1' with target 'kubernetes'
[i] Plugin binary for 'management-cluster:v0.30.1' found in cache
[i] Installing plugin 'package:v0.30.1' with target 'kubernetes'
[i] Plugin binary for 'package:v0.30.1' found in cache
[i] Installing plugin 'pinniped-auth:v0.30.1' with target 'global'
[i] Plugin binary for 'pinniped-auth:v0.30.1' found in cache
[i] Installing plugin 'secret:v0.30.1' with target 'kubernetes'
[i] Plugin binary for 'secret:v0.30.1' found in cache
[i] Installing plugin 'telemetry:v0.30.1' with target 'kubernetes'
[i] Plugin binary for 'telemetry:v0.30.1' found in cache
[ok] successfully installed all plugins from group 'vmware-tkg/default:v2.3.0'
```

After installing the Tanzu plug-ins, run the Tanzu plug-in list command to check the versions of the plug-ins and their installation status.

10. Install Carvel tools.

Tanzu Kubernetes Grid uses the following tools from the Carvel open-source project:

- ytt - a command-line tool for templating and patching YAML files. You can also use ytt to collect fragments and piles of YAML into modular chunks for easy re-use.
- kapp - the application deployment CLI for Kubernetes. It allows you to install, upgrade, and delete multiple Kubernetes resources as one application.
- kbld - an image-building and resolution tool.
- imgpkg - a tool that enables Kubernetes to store configurations and the associated container images as OCI images, and to transfer these images.
- Install ytt.

  ```
  gunzip ytt-linux-amd64-v0.45.0+vmware.2.gz

  chmod ugo+x ytt-linux-amd64-v0.45.0+vmware.2 &&  mv ./ytt-linux-amd64-v0.
  45.0+vmware.2 /usr/local/bin/ytt
  ```

  Run `ytt --version` to check that the correct version of ytt is installed and executable.
- Install kapp.

  ```
  gunzip kapp-linux-amd64-v0.55.0+vmware.2.gz

  chmod ugo+x kapp-linux-amd64-v0.55.0+vmware.2 && mv ./kapp-linux-amd64-v
  0.55.0+vmware.2 /usr/local/bin/kapp
  ```

  Run `kapp --version` to check that the correct version of kapp is installed and executable.
- Install kbld.

  ```
  gunzip kbld-linux-amd64-v0.37.0+vmware.2.gz

  chmod ugo+x kbld-linux-amd64-v0.37.0+vmware.2 && mv ./kbld-linux-amd64-v
  0.37.0+vmware.2 /usr/local/bin/kbld
  ```

  Run `kbld --version` to check that the correct version of kbld is installed and executable.
- Install imgpkg.

  ```
  gunzip imgpkg-linux-amd64-v0.36.0+vmware.2.gz
  chmod ugo+x imgpkg-linux-amd64-v0.36.0+vmware.2 && mv ./imgpkg-linux-amd6
  4-v0.36.0+vmware.2 /usr/local/bin/imgpkg
  ```

  Run `imgpkg --version` to check that the correct version of imgpkg is installed and executable.

11. Install yq.

    yq is a lightweight and portable command-line YAML processor. You can download yq by clicking here.

    ```
    tar -zxvf yq_linux_amd64.tar.gz

    mv yq_linux_amd64 /usr/local/bin/yq
    ```

Run the `yq -V` command to check that the correct version of yq is installed and executable.

12. Run the following commands to start the Docker service and enable it to start at boot. The Photon OS has Docker installed by default.

```
## Check Docker service status
systemctl status docker

## Start Docker Service
systemctl start docker

## To start Docker Service at boot
systemctl enable docker
```

13. Execute the following commands to ensure that the bootstrap machine uses cgroup v1.

```
docker info | grep -i cgroup

## You should see the following
Cgroup Driver: cgroupfs
```

14. Create an SSH key-pair.

    This is required for Tanzu CLI to connect to vSphere from the bootstrap machine. The public key part of the generated key will be passed during the Tanzu Kubernetes Grid management cluster deployment.

```
### Generate public/Private key pair.

ssh-keygen -t rsa -b 4096 -C "email@example.com"

### Add the private key to the SSH agent running on your machine and enter the
password you created in the previous step

ssh-add ~/.ssh/id_rsa

### If the above command fails, execute "eval $(ssh-agent)" and then rerun the
command.
```

    Make a note of the public key from the file **$home/.ssh/id_rsa.pub**. You need this while creating a config file for deploying the Tanzu Kubernetes Grid management cluster.

15. If your bootstrap machine runs Linux or Windows Subsystem for Linux, and it has a Linux kernel built after the May 2021 Linux security patch, for example Linux 5.11 and 5.12 with Fedora, run the following command:

```
sudo sysctl net/netfilter/nf_conntrack_max=131072
```

Import the Base Image Template in vCenter Server

Before you proceed with the management cluster creation, ensure that the base image template is imported into vSphere and is available as a template. To import a base image template into vSphere:

1. Go to the Tanzu Kubernetes Grid downloads page and download a Tanzu Kubernetes Grid OVA for the cluster nodes.

For the management cluster, download either a Photon or Ubuntu based Kubernetes v1.26.5 OVA.

> Custom OVA with a custom Tanzu Kubernetes release (TKr) is also supported, as described in Build Machine Images.

For workload clusters, OVA can have any supported combination of OS and Kubernetes version, as packaged in a Tanzu Kubernetes release.

> Make sure you download the most recent OVA base image templates in the event of security patch releases. You can find updated base image templates that include security patches on the Tanzu Kubernetes Grid product download page.

2. In the vSphere client, right-click an object in the vCenter Server inventory and select **Deploy OVF template**.
3. Select **Local file**, click the button to upload files, and select the downloaded OVA file on your local machine.
4. Follow the installer prompts to deploy a VM from the OVA.
5. Click **Finish** to deploy the VM. When the OVA deployment finishes, right-click the VM and select **Template** > **Convert to Template**.

> Do not power on the VM before you convert it to a template.

6. **If using non administrator SSO account**: In the VMs and Templates view, right-click the new template, select **Add Permission**, and assign the **tkg-user** to the template with the **TKG role**.

For information about how to create the user and role for Tanzu Kubernetes Grid, see Required Permissions for the vSphere Account.

Import NSX Advanced Load Balancer in Content Library

Create a content library following the instructions provided in Create a Library in VMware vSphere documentation. You will store the NSX Advanced Load Balancer OVA in the library.

To import the OVA into the content library, see Import Items to a Content Library.

Deploy and Configure NSX Advanced Load Balancer

NSX Advanced Load Balancer (ALB) is an enterprise-grade integrated load balancer that provides L4 - L7 load balancer support. It is recommended for vSphere deployments without NSX-T, or when there are unique scaling requirements.

NSX Advanced Load Balancer is deployed in Write Access Mode in the vSphere Environment. This mode grants NSX Advanced Load Balancer controllers full write access to vCenter that helps in automatically creating, modifying, and removing service engines (SEs) and other resources as needed to adapt to changing traffic needs.

The following table provides a sample IP address and FQDN set for the NSX Advanced Load Balancer controllers:

Controller Node	IP Address	FQDN
Node 1 Primary	172.16.10.11	`sfo01albctlr01a.sfo01.rainpole.local`
Node 2 Secondary	172.16.10.12	`sfo01albctlr01b.sfo01.rainpole.local`
Node 3 Secondary	172.16.10.13	`sfo01albctlr01c.sfo01.rainpole.local`
HAAddress	172.16.10.10	`sfo01albctlr01.sfo01.rainpole.local`

Follow these steps to deploy and configure NSX Advanced Load Balancer:

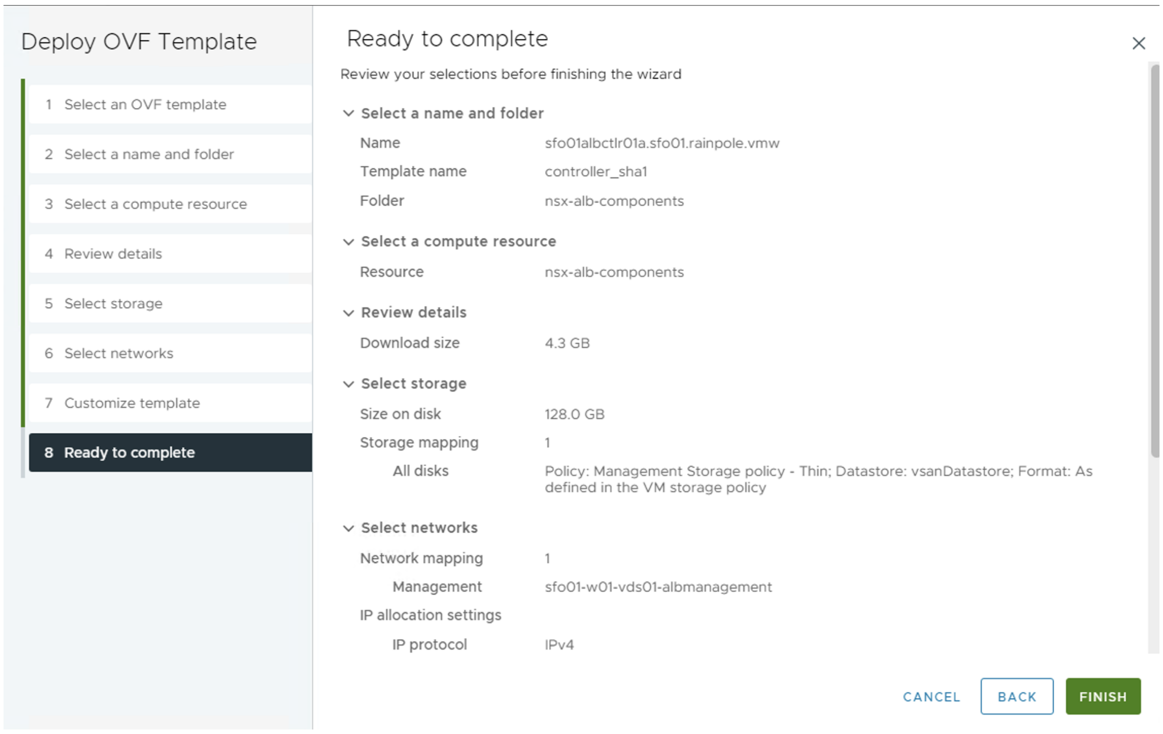
1. Deploy NSX Advanced Load Balancer
2. NSX Advanced Load Balancer: Initial setup
3. NSX Advanced Load Balancer: Licensing
4. NSX Advanced Load Balancer: Controller High Availability
5. NSX Advanced Load Balancer: Certificate Management
6. NSX Advanced Load Balancer: Create vCenter Cloud and SE Groups
7. NSX Advanced Load Balancer: Configure Network and IPAM & DNS Profiles

Deploy NSX Advanced Load Balancer

As part of the prerequisites, you must have the NSX Advanced Load Balancer 21.1.4 OVA downloaded and imported to the content library. Deploy the NSX Advanced Load Balancer under the resource pool **“nsx-alb-components”** and place it under the folder **“nsx-alb-components”**.

To deploy NSX Advanced Load Balancer, complete the following steps.

1. Log in to **vCenter** and go to **Home** > **Content Libraries**.
2. Select the content library under which the NSX Advanced Load Balancer OVA is placed.
3. Click on **OVA & OVF Templates**.
4. Right-click the NSX Advanced Load Balancer image and select **New VM from this Template**.
5. On the Select name and folder page, enter a name and select a folder for the NSX Advanced Load Balancer VM as **nsx-alb-components**.
6. On the Select a compute resource page, select the resource pool **nsx-alb-components**.
7. On the Review details page, verify the template details and click **Next**.
8. On the Select storage page, select a storage policy from the VM Storage Policy drop-down menu and choose the datastore location where you want to store the virtual machine files.
9. On the Select networks page, select the network **sfo01-w01-vds01-albmanagement** and click **Next**.
10. On the Customize template page, provide the NSX Advanced Load Balancer management network details such as IP address, subnet mask, and gateway, and click **Next**.
11. On the Ready to complete page, review the page and click **Finish**.



Ready to complete

Review your selections before finishing the wizard

Select a name and folder

Name	sfo01albctlr01a.sfo01.rainpole.vmw
Template name	controller_sha1
Folder	nsx-alb-components

Select a compute resource

Resource	nsx-alb-components

Review details

Download size	4.3 GB

Select storage

Size on disk	128.0 GB
Storage mapping	1
All disks	Policy: Management Storage policy - Thin; Datastore: vsanDatastore; Format: As defined in the VM storage policy

Select networks

Network mapping	1
Management	sfo01-w01-vds01-albmanagement
IP allocation settings	
IP protocol	IPv4

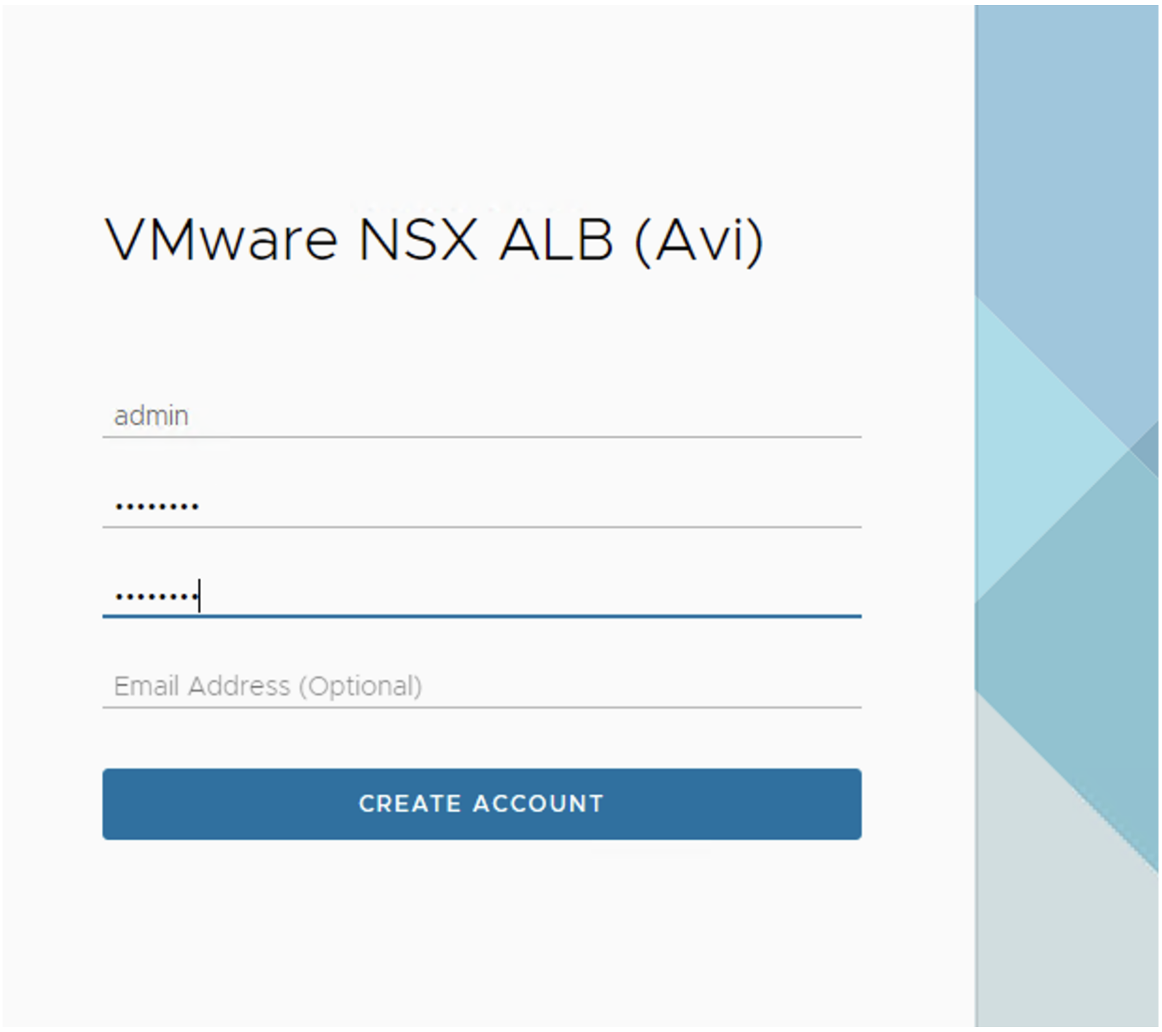
A new task for creating the virtual machine appears in the **Recent Tasks** pane. After the task is complete, the NSX Advanced Load Balancer virtual machine is created on the selected resource. Power on the virtual machine and give it a few minutes for the system to boot. Upon successful boot up, go to NSX Advanced Load Balancer on your browser.

While the system is booting up, a blank web page or a 503 status code may appear.

NSX Advanced Load Balancer: Initial Setup

After NSX Advanced Load Balancer is successfully deployed and running, go to NSX Advanced Load Balancer on your browser using the URL https://*<IP/FQDN>* and configure the basic system settings:

1. Set admin password and click **Create Account**.


VMware NSX ALB (Avi)

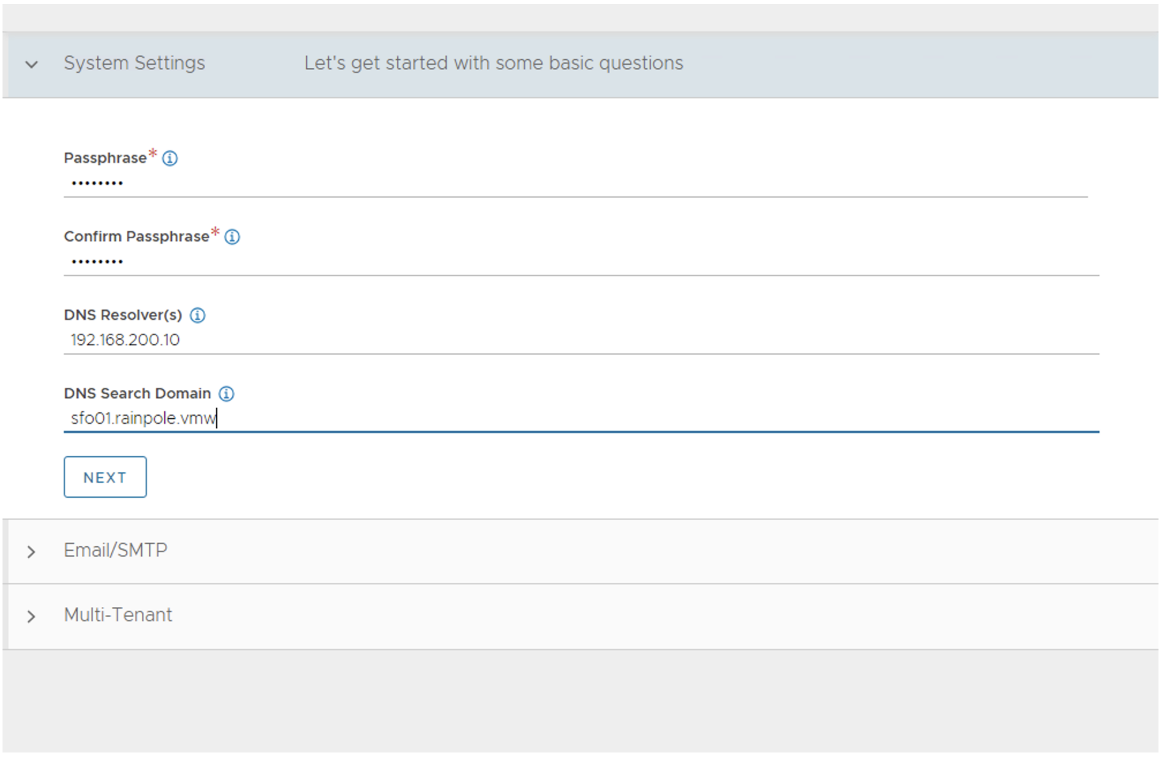
admin

••••••••

••••••••

Email Address (Optional)

2. On the Welcome page, under **System Settings**, set backup passphrase and provide DNS information, and click **Next**.


System Settings — Let's get started with some basic questions

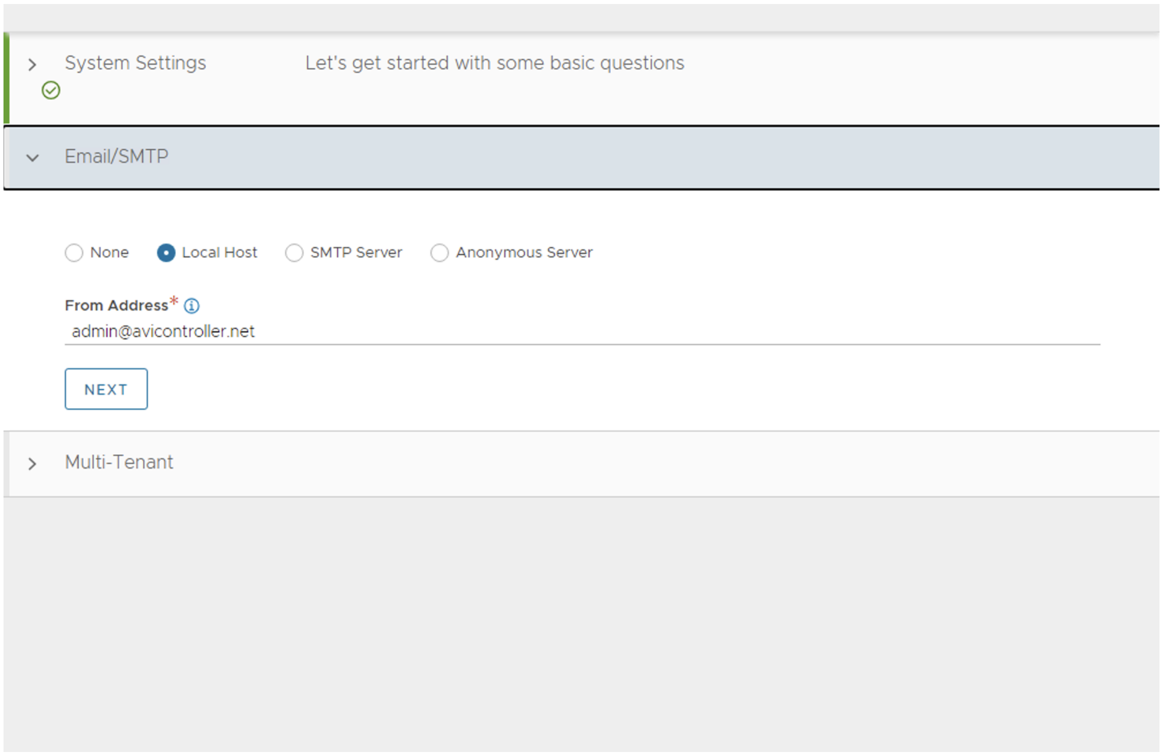
Passphrase*: ••••••••

Confirm Passphrase*: ••••••••

DNS Resolver(s): 192.168.200.10

DNS Search Domain: sfo01.rainpole.vmw

3. Under **Email/SMTP**, provide email and SMTP information, and click **Next**.



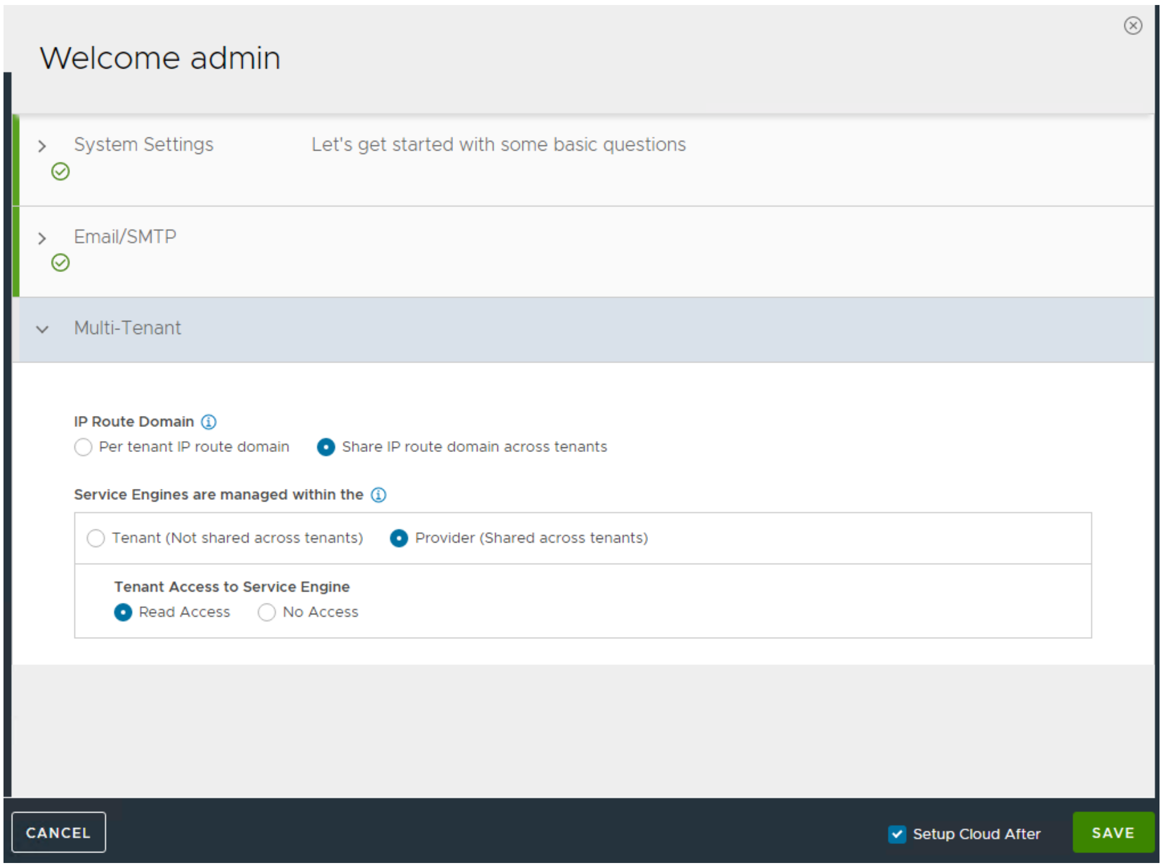
System Settings — Let's get started with some basic questions

Email/SMTP

( ) None [✓ Local Host] ( ) SMTP Server ( ) Anonymous Server

From Address*: admin@avicontroller.net

4. Under **Multi-Tenant**, configure settings as follows and click **Save**.
   - IP Route Domain: Share IP route domain across tenants
   - Service Engines are managed within the: Provider (Shared across tenants)
   - Tenant Access to Service Engine: Read Access



Welcome admin

System Settings — Let's get started with some basic questions

Email/SMTP

Multi-Tenant

IP Route Domain: ( ) Per tenant IP route domain [✓ Share IP route domain across tenants]

Service Engines are managed within the: ( ) Tenant (Not shared across tenants) [✓ Provider (Shared across tenants)]

Tenant Access to Service Engine: [✓ Read Access] ( ) No Access

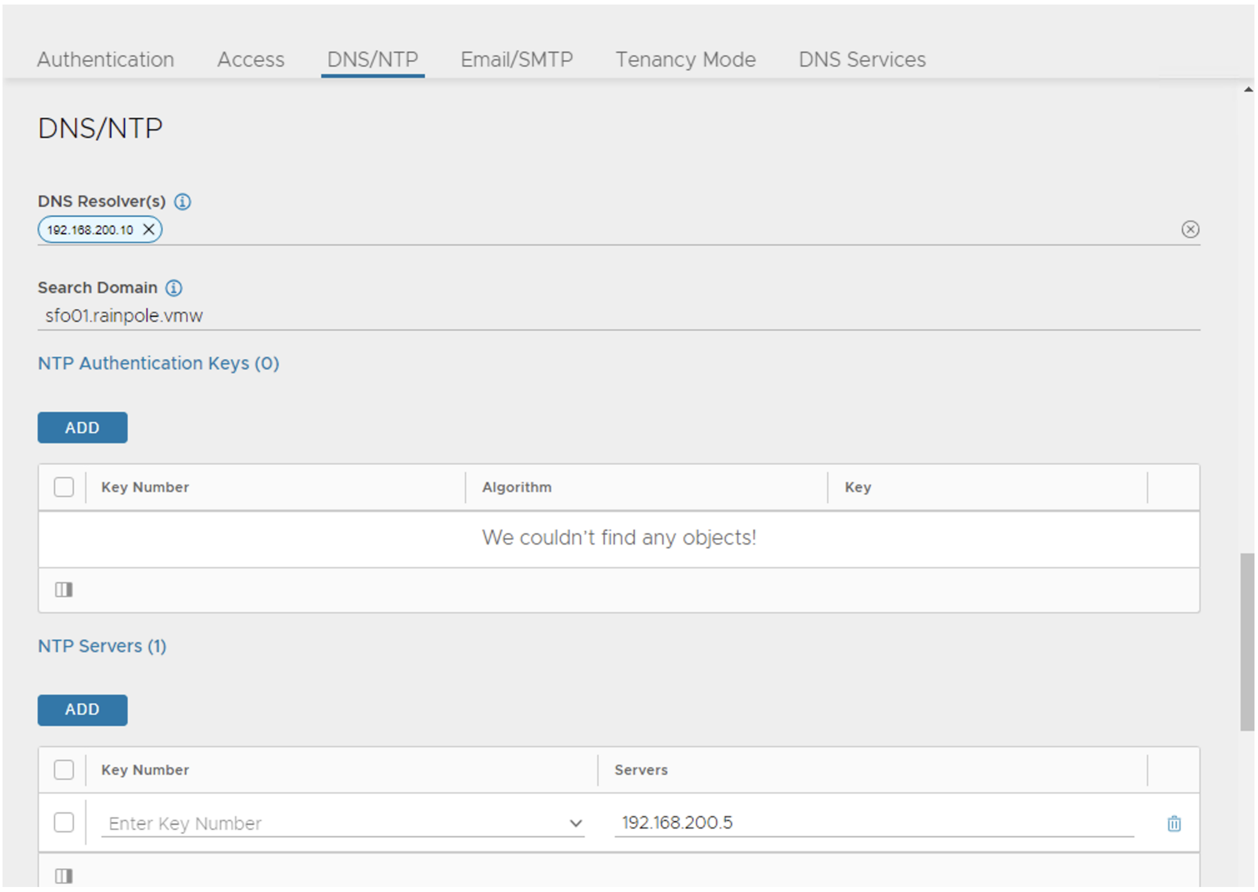
[✓ Setup Cloud After]

If you did not select the **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.

NSX Advanced Load Balancer: NTP Configuration

To configure NTP, go to **Administration** > **Settings** > **DNS/NTP > Edit** and add your NTP server details under **DNS/NTP** and click **Save**.

You may also delete the default NTP servers.



DNS/NTP

DNS Resolver(s): 192.168.200.10

Search Domain: sfo01.rainpole.vmw

NTP Authentication Keys (0)

Key Number	Algorithm	Key
We couldn't find any objects!		

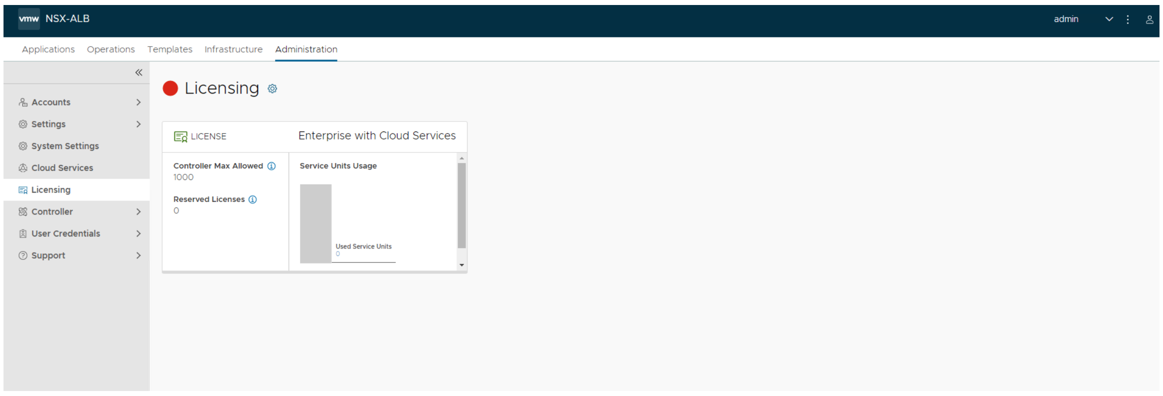
NTP Servers (1)

Key Number	Servers
Enter Key Number	192.168.200.5

NSX Advanced Load Balancer: Licensing

You can configure the license tier as NSX ALB Enterprise or NSX ALB Essentials for VMware Tanzu as per the feature requirement. This section focuses on enabling NSX Advanced Load Balancer using **Enterprise Tier (VMware NSX ALB Enterprise)** license model.

1. To configure licensing, go to **Administration** > **System Settings** > **Licensing**, and click on the gear icon to change the license type to Enterprise.



Licensing

LICENSE Enterprise with Cloud Services

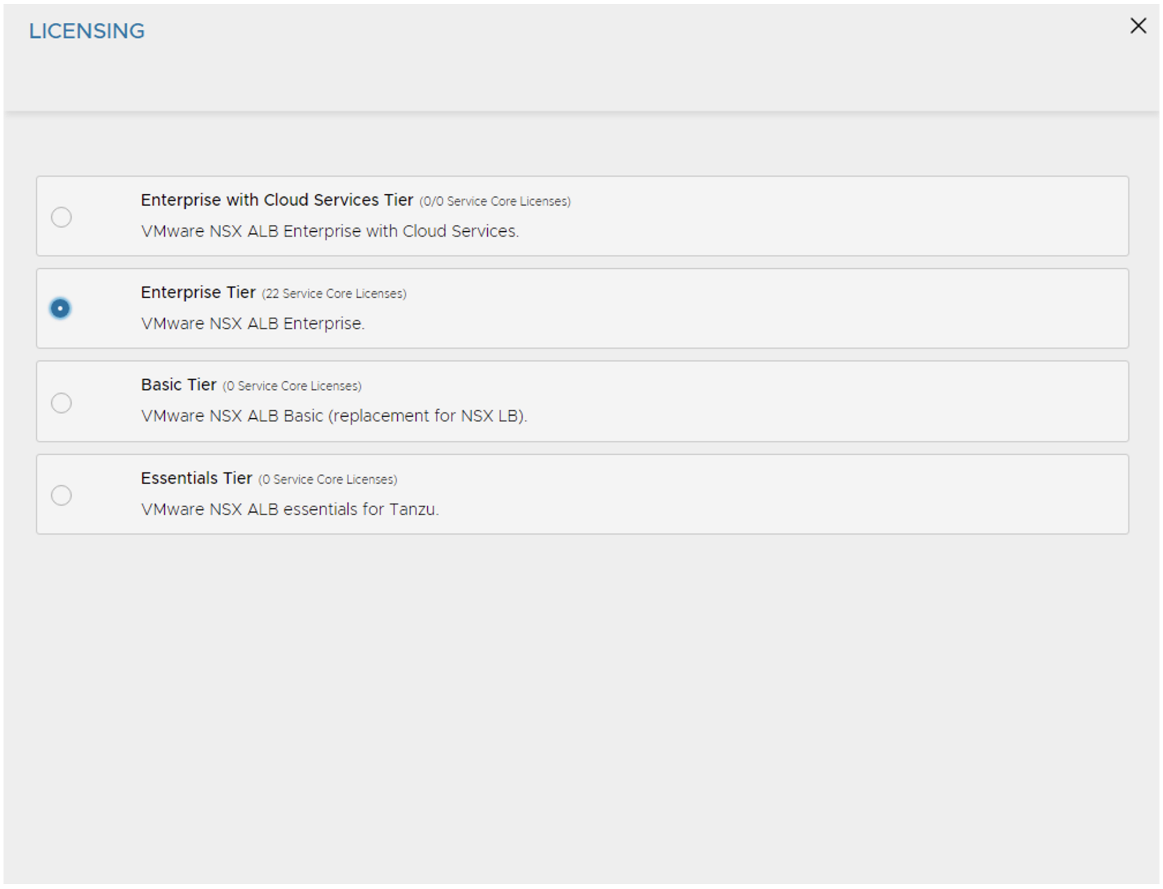
Controller Max Allowed 1000

Reserved Licenses 0

Service Units Usage

Used Service Units 0

2. Select Enterprise Tier as the license type and click **Save**.



LICENSING

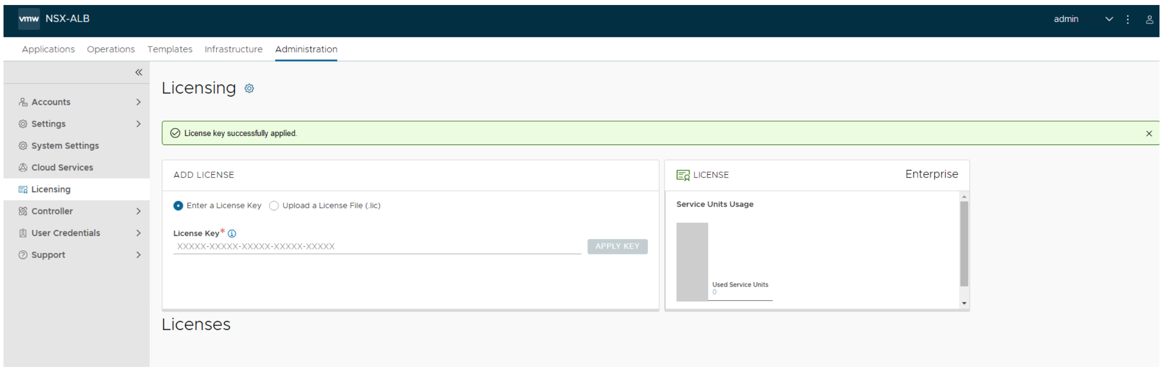
( ) Enterprise with Cloud Services Tier (0/0 Service Core Licenses)
VMware NSX ALB Enterprise with Cloud Services.

[✓ Enterprise Tier] (22 Service Core Licenses)
VMware NSX ALB Enterprise.

( ) Basic Tier (0 Service Core Licenses)
VMware NSX ALB Basic (replacement for NSX LB).

( ) Essentials Tier (0 Service Core Licenses)
VMware NSX ALB essentials for Tanzu.

3. Once the license tier is changed, apply the NSX Advanced Load Balancer Enterprise license key. If you have a license file instead of a license key, apply the license by clicking on the **Upload a License File(.lic)** option.



Licensing

License key successfully applied.

ADD LICENSE

[✓ Enter a License Key] ( ) Upload a License File (.lic)

License Key* XXXXX-XXXXX-XXXXX-XXXXX-XXXXX APPLY KEY

LICENSE Enterprise

Service Units Usage

Used Service Units 0

Licenses

NSX Advanced Load Balancer: Controller High Availability

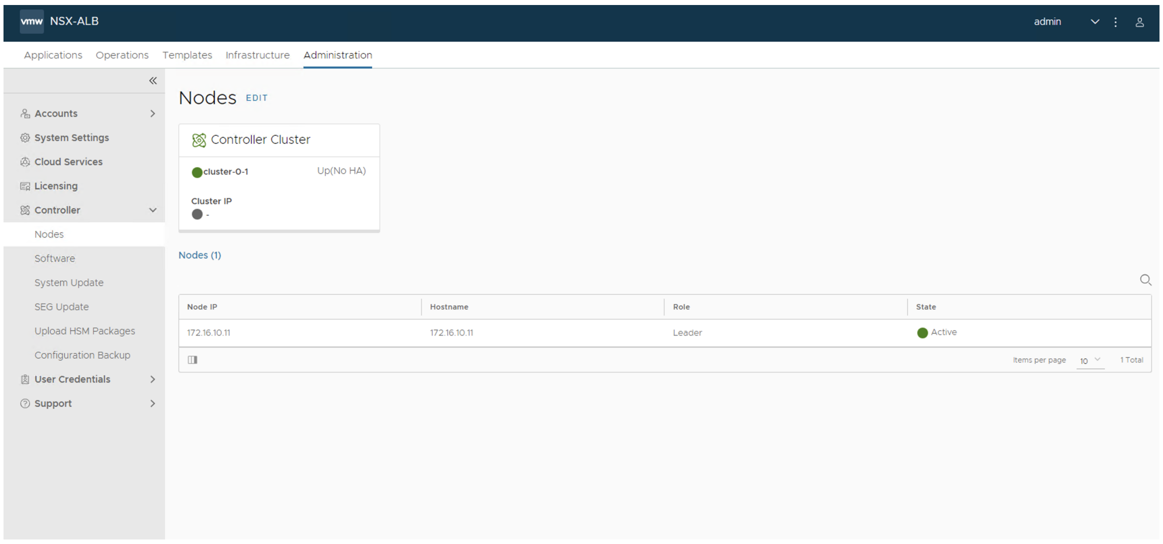
In a production environment, it is recommended to deploy additional controller nodes and configure the controller cluster for high availability and disaster recovery. Adding 2 additional nodes to create a 3-node cluster provides node-level redundancy for the controller and also maximizes performance for CPU-intensive analytics functions.

To run a 3-node controller cluster, you deploy the first node and perform the initial configuration, and set the cluster IP address. After that, you deploy and power on two more controller VMs, but you must not run the initial configuration wizard or change the admin password for these controllers VMs. The configuration of the first controller VM is assigned to the two new controller VMs.

The first controller of the cluster receives the Leader role. The second and third controllers work as Follower.

Perform the following steps to configure NSX Advanced Load Balancer cluster:

1. Log in to the primary NSX Advanced Load Balancer controller and go to **Administrator** > **Controller** > **Nodes**, and click **Edit**.



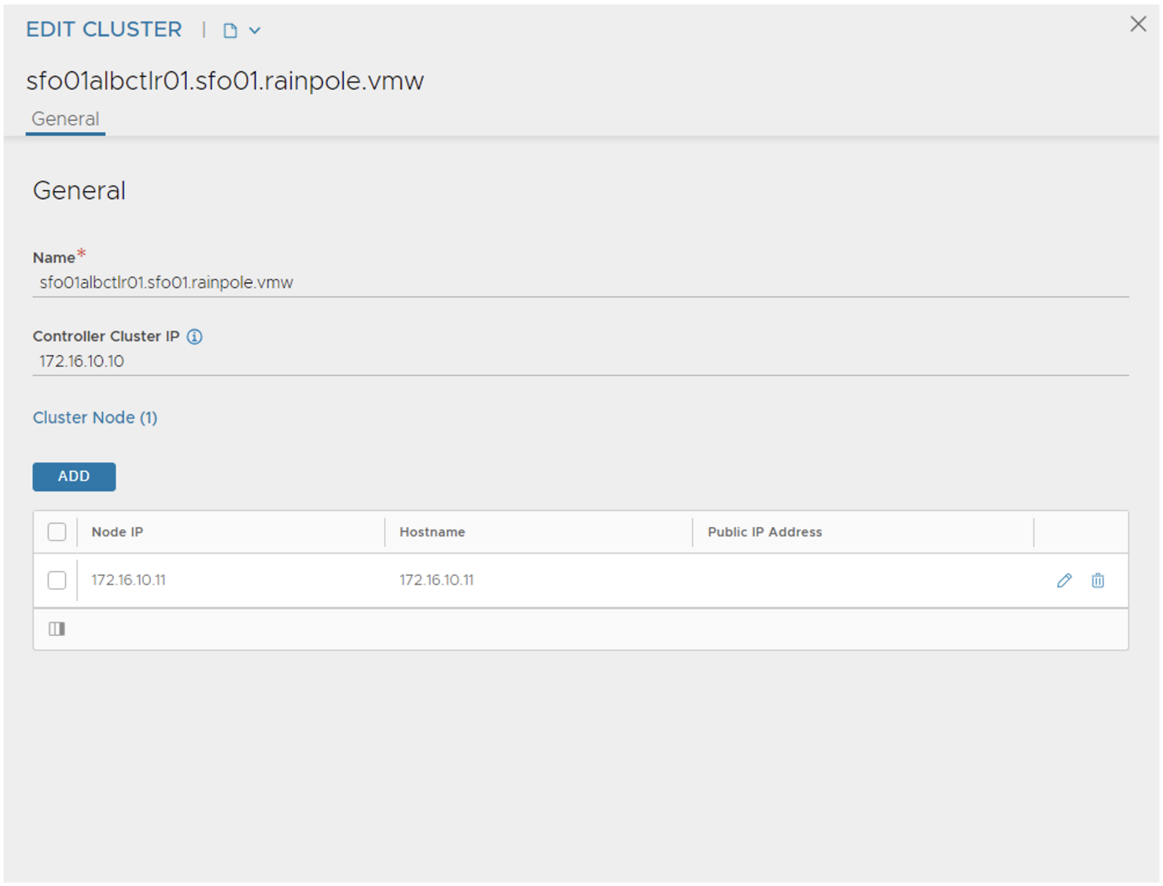
cluster-0-1 Up(No HA)

Cluster IP -

Node IP	Hostname	Role	State
172.16.10.11	172.16.10.11	Leader	● Active

Items per page 10 1 Total

2. Specify **Name** and **Controller Cluster IP**, and click **Save**. This IP address must be from the NSX Advanced Load Balancer management network.



sfo01albctlr01.sfo01.rainpole.vmw

General

Name*
sfo01albctlr01.sfo01.rainpole.vmw

Controller Cluster IP
172.16.10.10

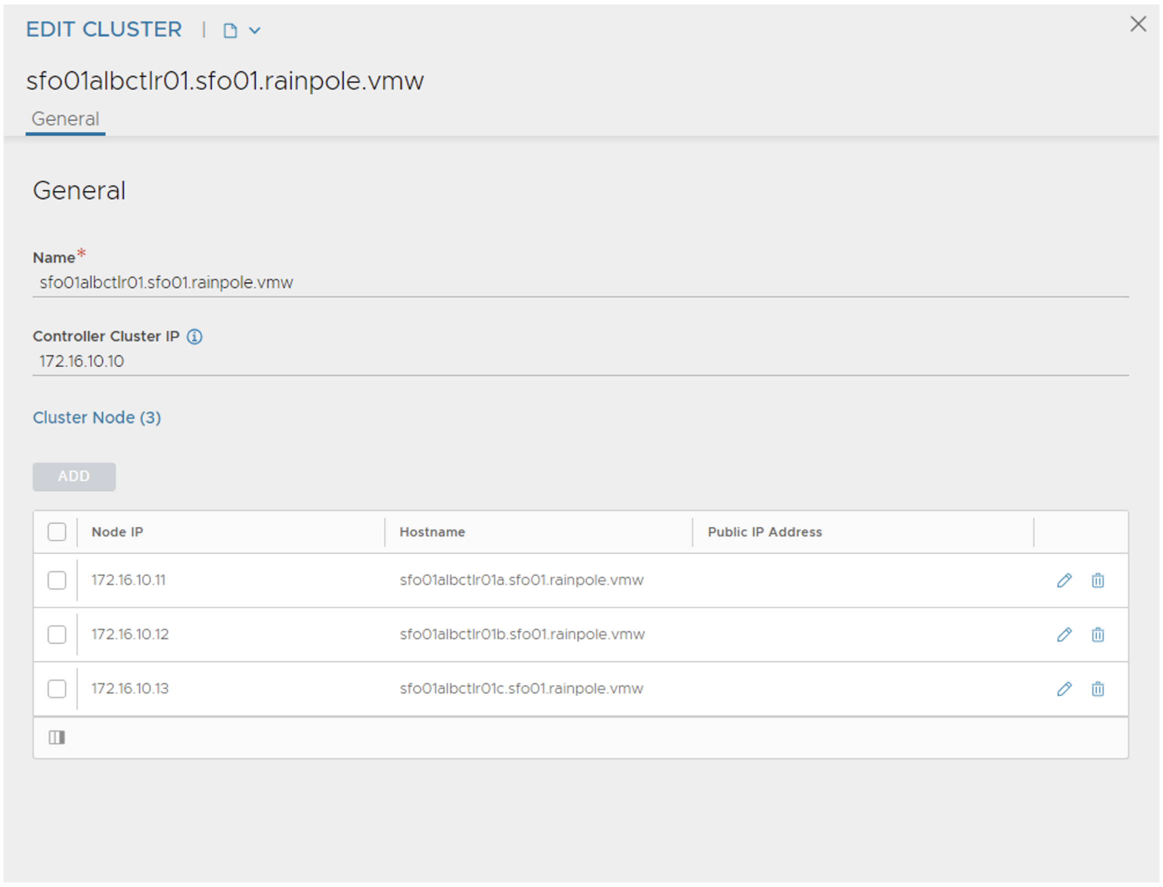
Cluster Node (1)

Node IP	Hostname	Public IP Address
172.16.10.11	172.16.10.11	

3. Deploy the 2nd and 3rd NSX Advanced Load Balancer controller nodes by using steps in Deploy NSX Advanced Load Balancer.

4. Log into the primary NSX Advanced Load Balancer controller using the Controller Cluster IP/FQDN and go to **Administrator** > **Controller** > **Nodes**, and click **Edit**. The Edit Controller Configuration popup appears.

5. In the **Cluster Nodes** field, enter the IP address for the 2nd and 3rd controller, and click **Save**.



EDIT CLUSTER

sfo01albctlr01.sfo01.rainpole.vmw

General

General

Name*
sfo01albctlr01.sfo01.rainpole.vmw

Controller Cluster IP
172.16.10.10

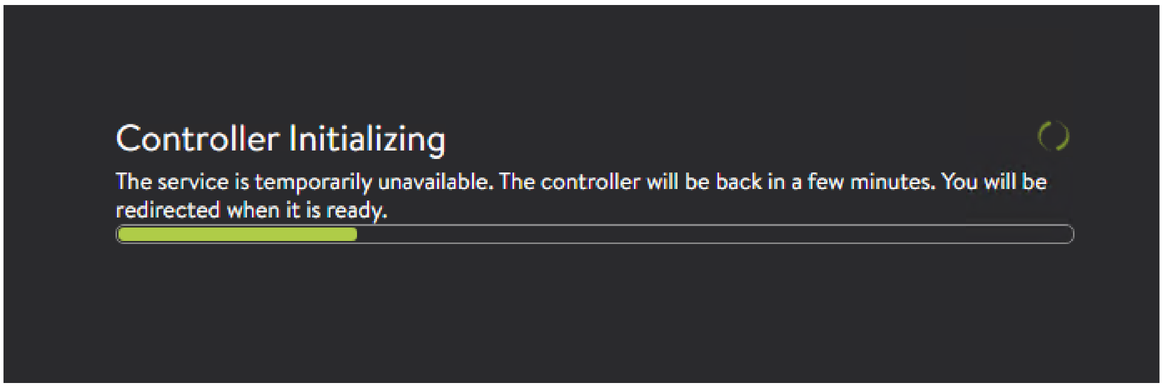
Cluster Node (3)

ADD

Node IP	Hostname	Public IP Address
172.16.10.11	sfo01albctlr01a.sfo01.rainpole.vmw	
172.16.10.12	sfo01albctlr01b.sfo01.rainpole.vmw	
172.16.10.13	sfo01albctlr01c.sfo01.rainpole.vmw	

After you complete these steps, the primary NSX Advanced Load Balancer controller becomes the leader for the cluster and invites the other controllers to the cluster as members.

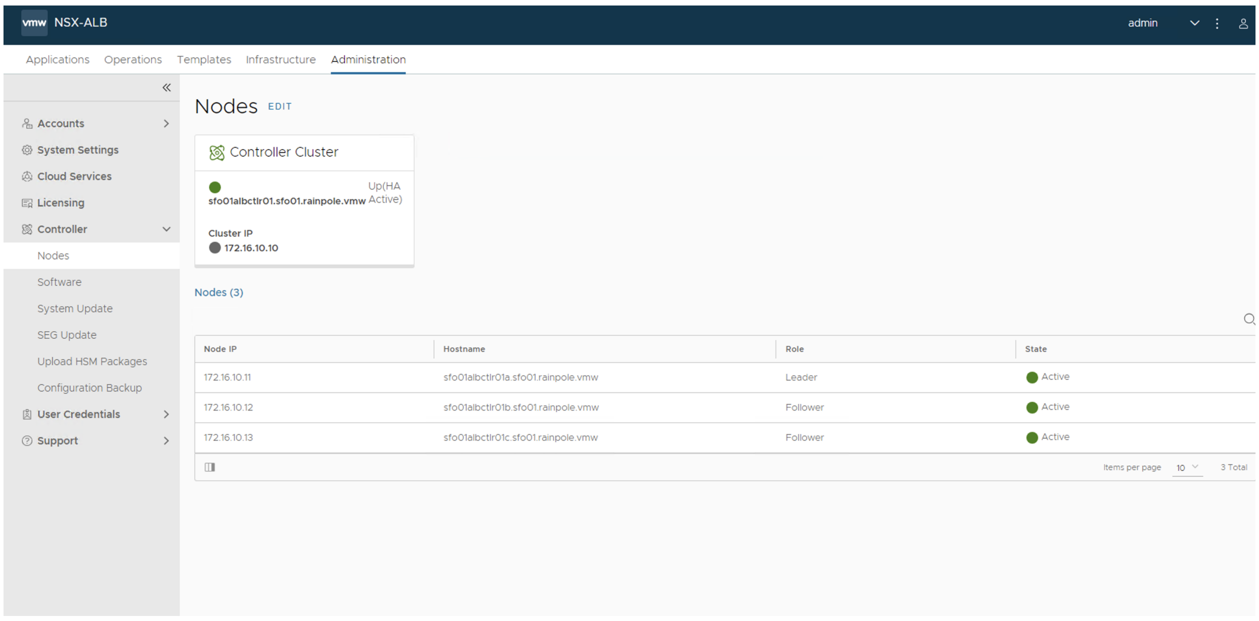
NSX Advanced Load Balancer then performs a warm reboot of the cluster. This process can take approximately 10-15 minutes. You are automatically logged out of the controller node where you are currently logged in. Enter the cluster IP address in the browser, to see details about the cluster formation task.



Controller Initializing

The service is temporarily unavailable. The controller will be back in a few minutes. You will be redirected when it is ready.

The configuration of the primary (leader) controller is synchronized to the new member nodes when the cluster comes online following the reboot. After the cluster is successfully formed, you can see the following status:



Nodes

Controller Cluster

sfo01albctlr01.sfo01.rainpole.vmw Up(HA Active)

Cluster IP
172.16.10.10

Nodes (3)

Node IP	Hostname	Role	State
172.16.10.11	sfo01albctlr01a.sfo01.rainpole.vmw	Leader	● Active
172.16.10.12	sfo01albctlr01b.sfo01.rainpole.vmw	Follower	● Active
172.16.10.13	sfo01albctlr01c.sfo01.rainpole.vmw	Follower	● Active

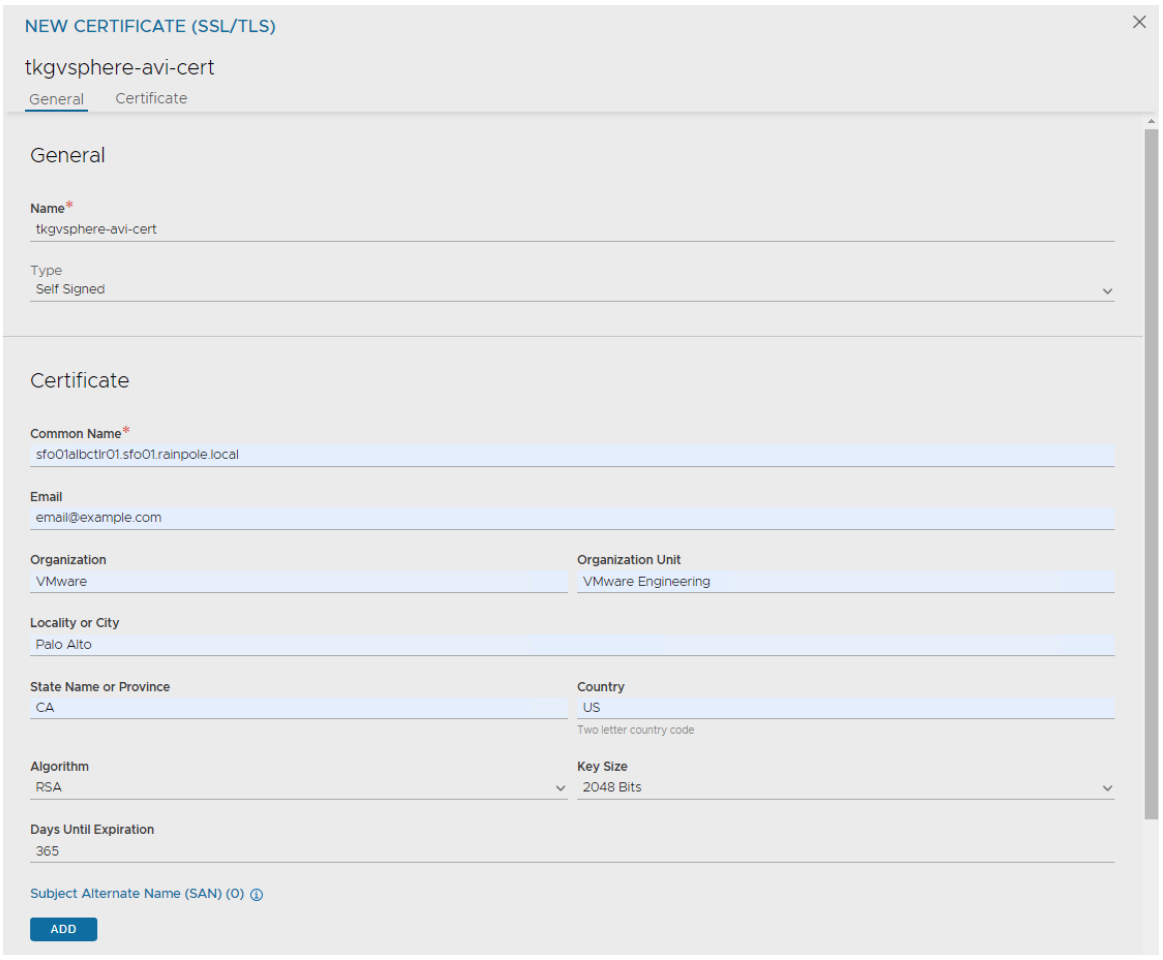
Items per page 10 3 Total

In the following tasks, all NSX Advanced Load Balancer configurations are done by connecting to the NSX ALB Controller Cluster IP/FQDN.

NSX Advanced Load Balancer: Certificate Management

The default system-generated controller certificate generated for SSL/TSL connections will not have the required subject alternate name (SAN) entries. Perform the following steps to create a controller certificate:

1. Log in to the NSX Advanced Load Balancer controller and go to **Templates** > **Security** > **SSL/TLS Certificates**.
2. Click **Create** and select **Controller Certificate**. You can either generate a self-signed certificate, generate CSR, or import a certificate. For the purpose of this document, a self-signed certificate is generated.
3. Provide all required details as per your infrastructure requirements and in the **Subject Alternate Name (SAN)** field, provide IP address and FQDN of all NSX Advanced Load Balancer controllers including NSX Advanced Load Balancer cluster IP and FQDN, and click **Save**.


NEW CERTIFICATE (SSL/TLS)

tkgvsphere-avi-cert

General   Certificate

General

Name*
tkgvsphere-avi-cert

Type
Self Signed

Certificate

Common Name*
sfo01albctlr01.sfo01.rainpole.local

Email
email@example.com

Organization
VMware

Organization Unit
VMware Engineering

Locality or City
Palo Alto

State Name or Province
CA

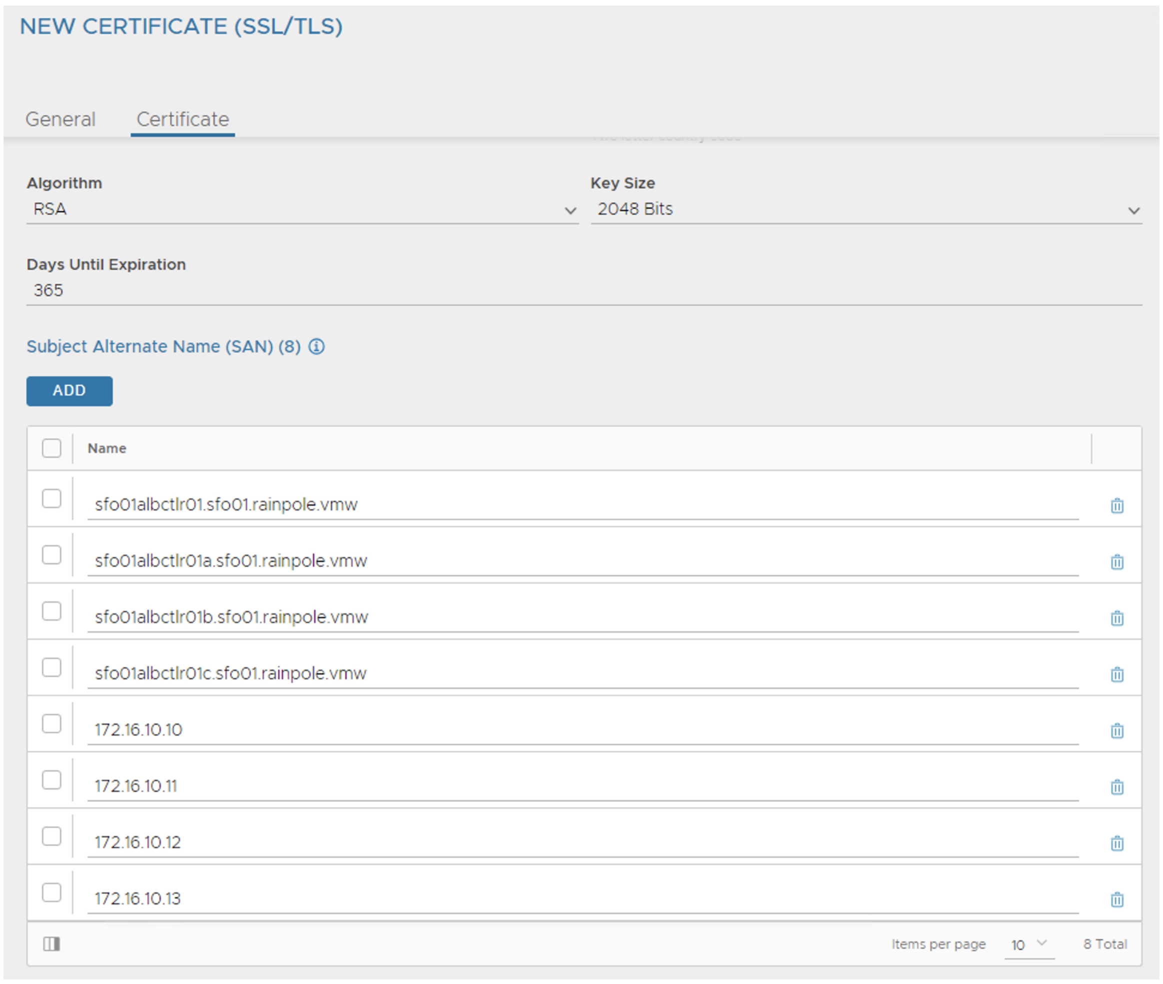
Country
US
Two letter country code

Algorithm
RSA

Key Size
2048 Bits

Days Until Expiration
365

Subject Alternate Name (SAN) (0)



Algorithm: RSA

Key Size: 2048 Bits

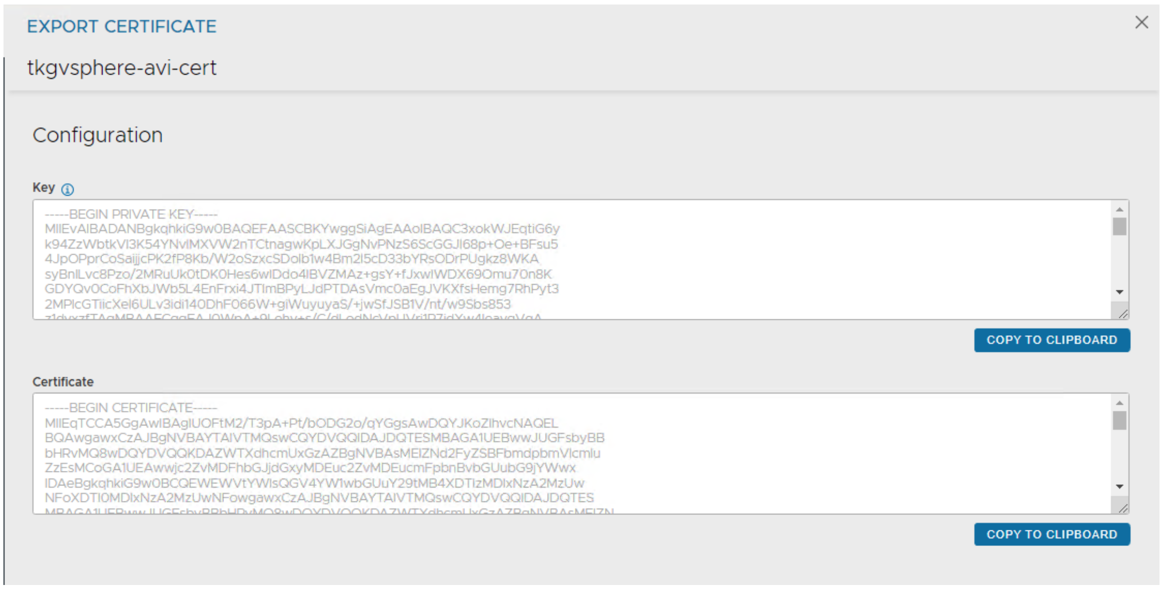
Days Until Expiration: 365

Subject Alternate Name (SAN) (8)

Name
sfo01albctlr01.sfo01.rainpole.vmw
sfo01albctlr01a.sfo01.rainpole.vmw
sfo01albctlr01b.sfo01.rainpole.vmw
sfo01albctlr01c.sfo01.rainpole.vmw
172.16.10.10
172.16.10.11
172.16.10.12
172.16.10.13

Items per page 10 8 Total

4. After the certificate is created, capture the certificate contents as this is required while deploying the Tanzu Kubernetes Grid management cluster. To capture the certificate content, click the Download icon next to certificate, and click **Copy to clipboard** under **Certificate**.



tkgvsphere-avi-cert

Configuration

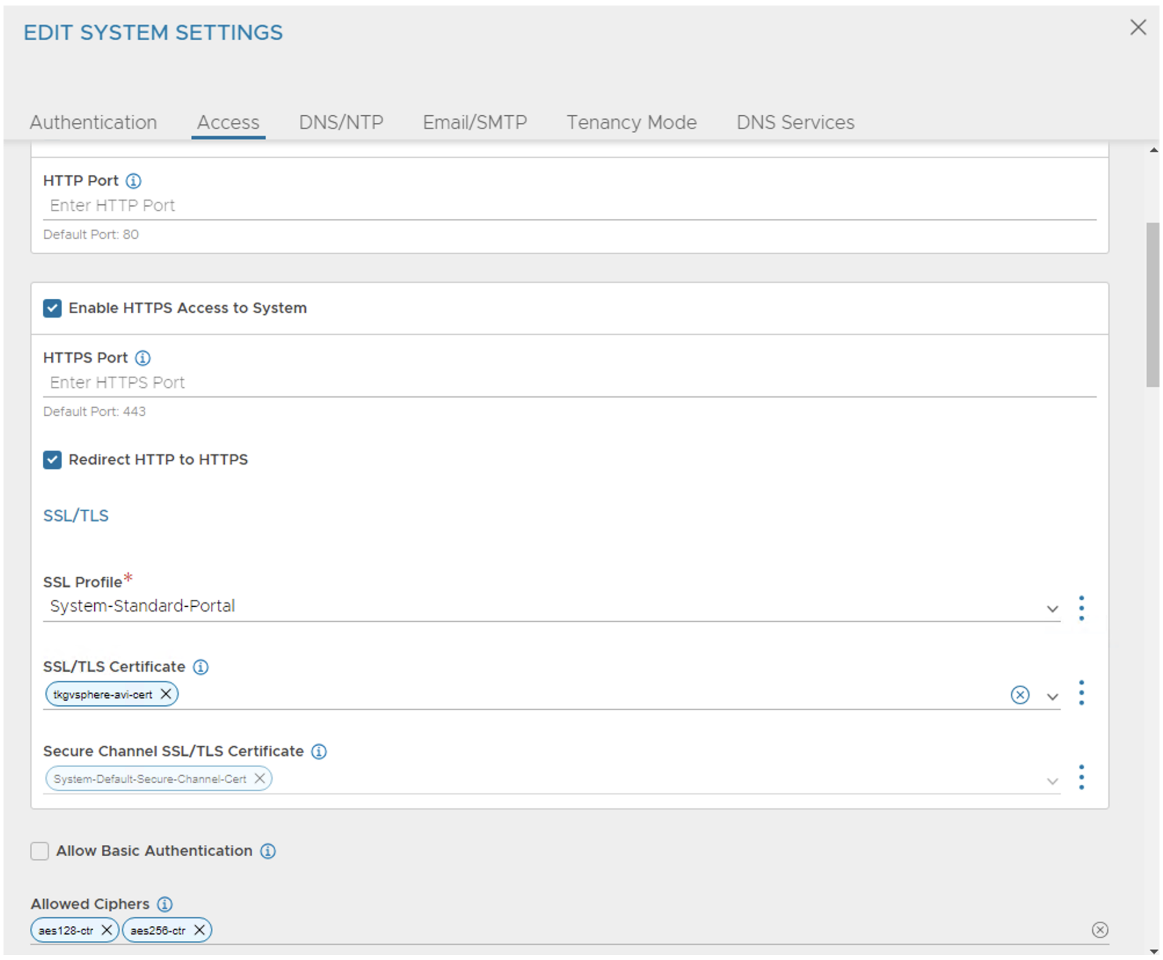
Key

```
-----BEGIN PRIVATE KEY-----
MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQC3xokWJEqtiG6y
k94ZzWbtkVl3K54YNvlMXVW2nTCtnagwKpLXJGgNvPNzS6ScGGJl68p+Oe+BFsu5
4JpOPprCoSaijjcPK2fP8Kb/W2oSzxcSDolb1w4Bm2l5cD33bYRsODrPUgkz8WKA
syBnlLvc8Pzo/2MRuUk0tDK0Hes6wIDdo4IBVZMAz+gsY+fJxwlWDX69Omu70n8K
GDYQv0CoFhXbJWb5L4EnFrxi4JTlmBPyLJdPTDAsVmc0aEgJVKXfsHemg7RhPyt3
2MPlcGTiicXel6ULv3idi140DhF066W+giWuyuyaS/+jwSfJSB1V/nt/w9Sbs853
```

Certificate

```
-----BEGIN CERTIFICATE-----
MIIEqTCCA5GgAwIBAgIUOFtM2/T3pA+Pt/bODG2o/qYGgsAwDQYJKoZIhvcNAQEL
BQAwgawxCzAJBgNVBAYTAlVTMQswCQYDVQQIDAJDQTESMBAGA1UEBwwJUGFsbyBB
bHRvMQ8wDQYDVQQKDAZWTXdhcmUxGzAZBgNVBAsMElZNd2FyZSBFbmdpbmVlcmlu
ZzEsMCoGA1UEAwwjc2ZvMDFhbGJjdGxyMDEuc2ZvMDEucmFpbnBvbGUubG9jYWwx
IDAeBgkqhkiG9w0BCQEWEWVtYWlsQGV4YW1wbGUuY29tMB4XDTIzMDIxNzA2MzUw
NFoXDTI0MDIxNzA2MzUwNFowgawxCzAJBgNVBAYTAlVTMQswCQYDVQQIDAJDQTES
```

5. To replace the certificate, go to **Administration** > **System Settings**, and edit it under **Access**. You can replace the SSL/TSL certificate to previously created certificate and click **Save**.



EDIT SYSTEM SETTINGS

Authentication | Access | DNS/NTP | Email/SMTP | Tenancy Mode | DNS Services

HTTP Port: Enter HTTP Port
Default Port: 80

[✓] Enable HTTPS Access to System

HTTPS Port: Enter HTTPS Port
Default Port: 443

[✓] Redirect HTTP to HTTPS

SSL/TLS

SSL Profile*: System-Standard-Portal

SSL/TLS Certificate: tkgvsphere-avi-cert

Secure Channel SSL/TLS Certificate: System-Default-Secure-Channel-Cert

[ ] Allow Basic Authentication

Allowed Ciphers: aes128-ctr, aes256-ctr

6. Log out and log in to NSX Advanced Load Balancer.

NSX Advanced Load Balancer: Create vCenter Cloud and SE Groups

NSX Advanced Load Balancer can be deployed in multiple environments for the same system. Each environment is called a cloud. The following procedure provides steps on how to create a VMware vCenter cloud, and as shown in the architecture two service engine (SE) groups are created.

Service Engine Group 1: Service engines part of this service engine group hosts:

- Virtual services that load balances control plane nodes of Management Cluster and Shared services cluster.
- Virtual services for all load balancer functionalities requested by Tanzu Kubernetes Grid management cluster and Shared services cluster.

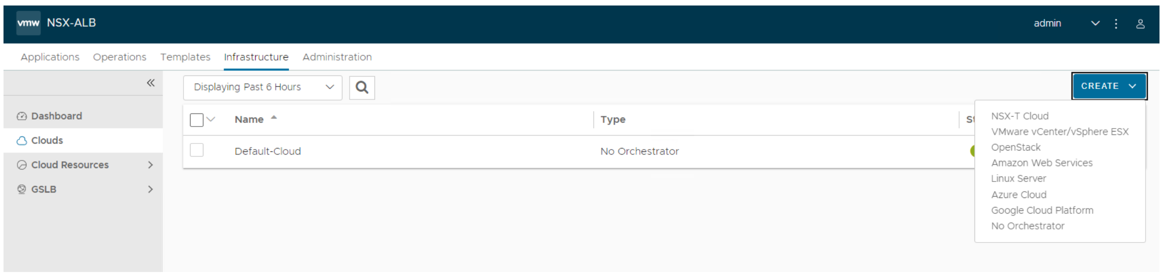
Service Engine Group 2: Service engines part of this service engine group hosts virtual services that load balances control plane nodes and virtual services for all load balancer functionalities requested by the workload clusters mapped to this SE group.

- Based on your requirements, you can create additional SE groups for the workload clusters. - Multiple workload clusters can be mapped to a single SE group. - A Tanzu Kubernetes Grid cluster can be mapped to only one SE group for application load balancer services. - Control plane VIP for the workload clusters will be placed on the respective

> Service Engine group assigned through AKO Deployment Config (ADC) during cluster creation.

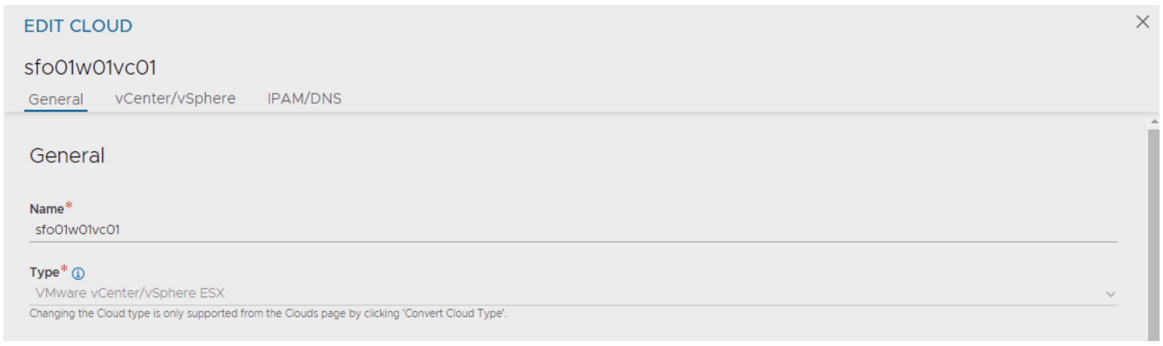
For information about mapping a specific service engine group to Tanzu Kubernetes Grid workload cluster, see Configure NSX Advanced Load Balancer in Tanzu Kubernetes Grid Workload Cluster.

1. Log in to NSX Advanced Load Balancer and go to **Infrastructure** > **Clouds** > **Create** > **VMware vCenter/vSphere ESX**.



Name	Type
Default-Cloud	No Orchestrator

2. Under **General** pane, in the **Name** field, enter a Cloud name.

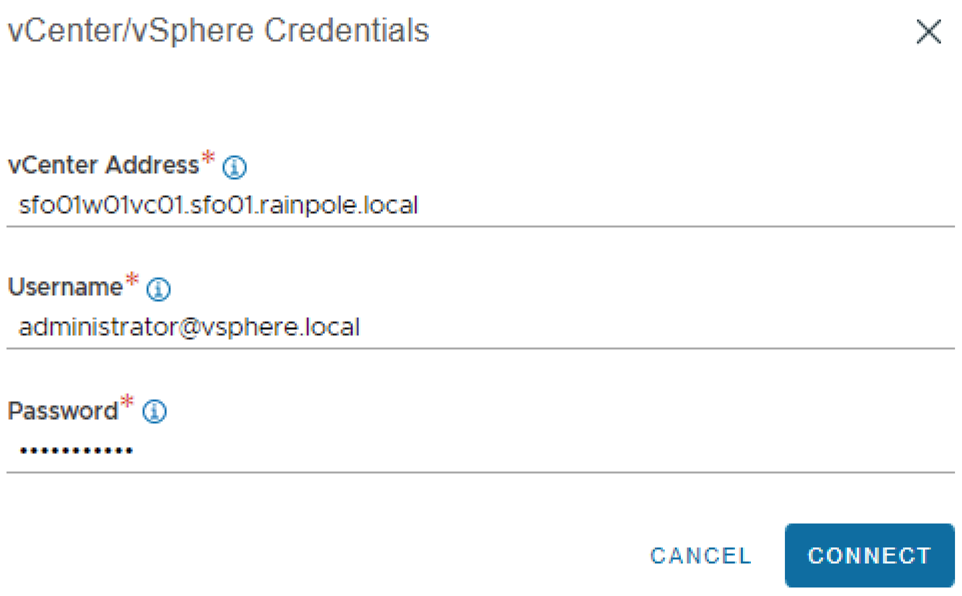


Name* sfo01w01vc01

Type* VMware vCenter/vSphere ESX

Changing the Cloud type is only supported from the Clouds page by clicking 'Convert Cloud Type'.

3. Under the **vCenter/vSphere** pane, specify the vCenter address, Username, and Password, and click **CONNECT**.

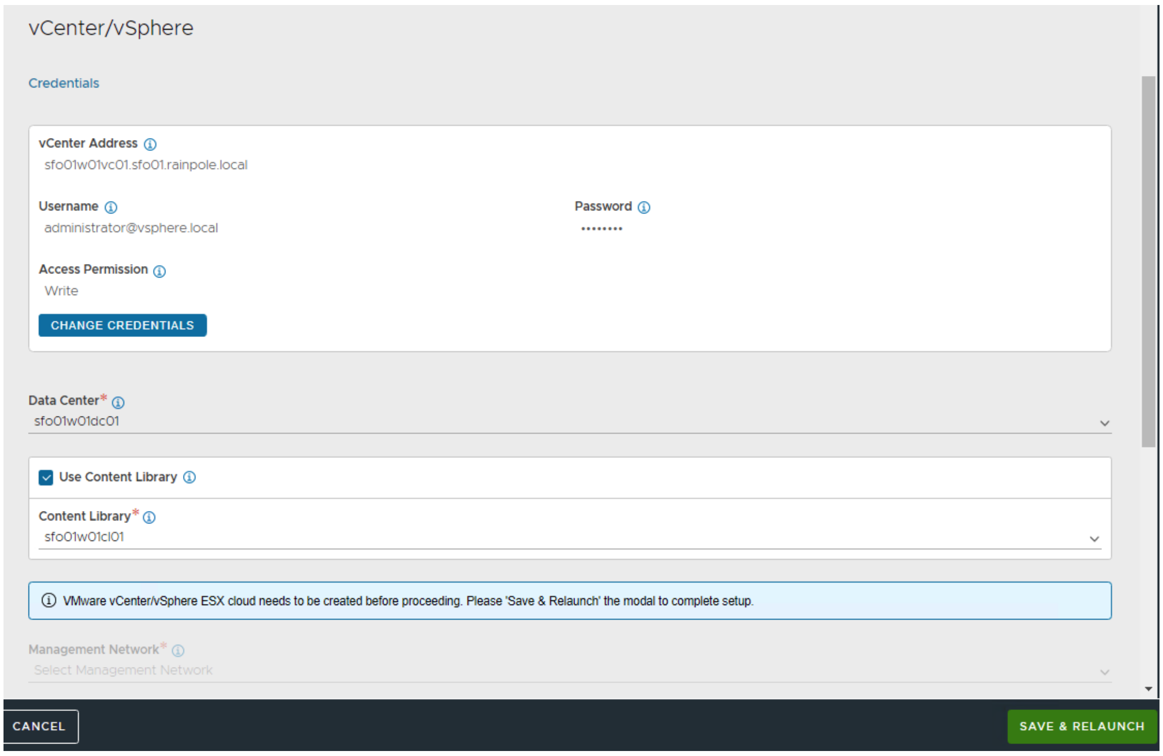


vCenter Address* sfo01w01vc01.sfo01.rainpole.local

Username* administrator@vsphere.local

Password* ••••••••••••

4. Under the **Data Center** pane, choose the data center from the Data Center drop-down menu.Select **Content Library** for SE template and click **SAVE & LAUNCH**.



vCenter/vSphere

Credentials

vCenter Address
sfo01w01vc01.sfo01.rainpole.local

Username
administrator@vsphere.local

Password
••••••••

Access Permission
Write

Data Center*
sfo01w01dc01

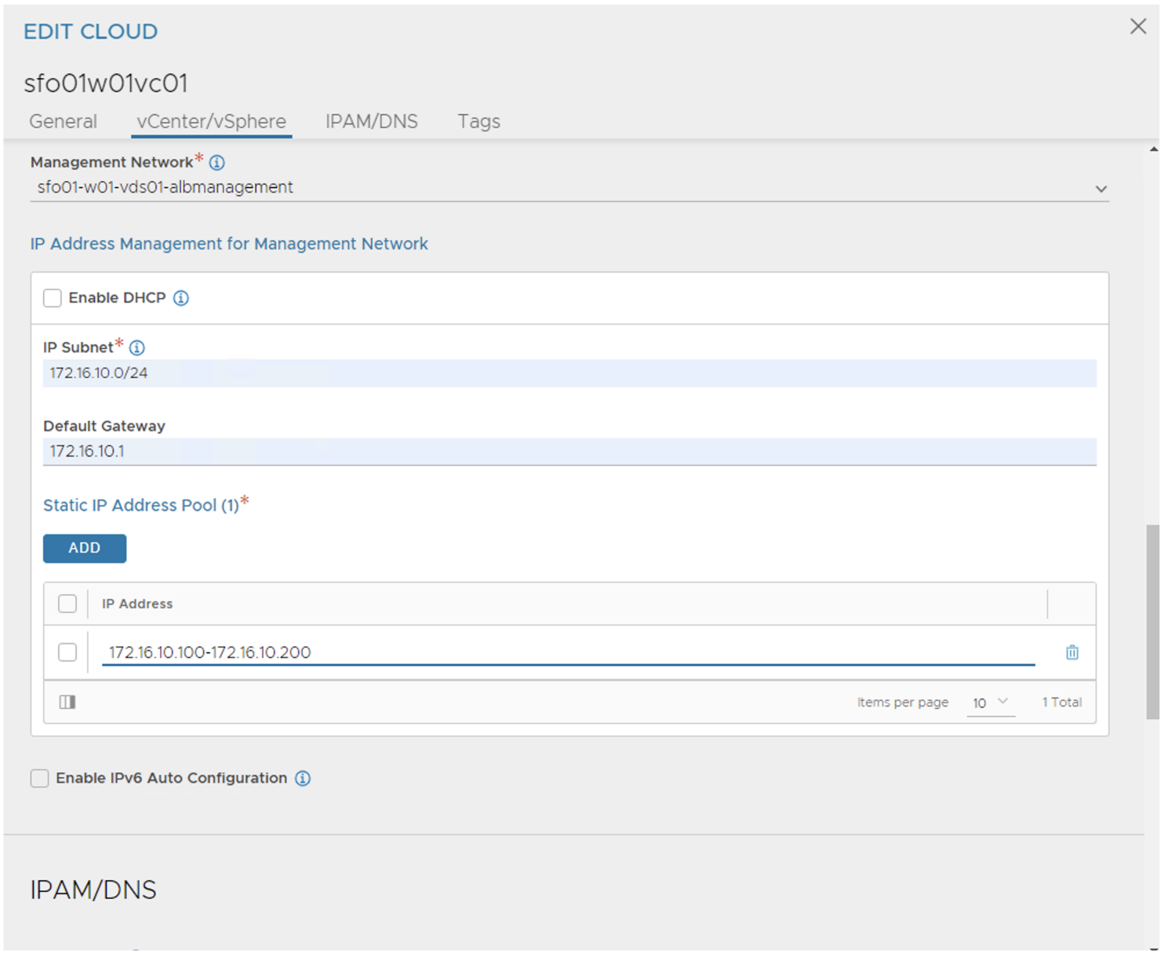
Use Content Library ✓

Content Library*
sfo01w01cl01

VMware vCenter/vSphere ESX cloud needs to be created before proceeding. Please 'Save & Relaunch' the modal to complete setup.

Management Network*
Select Management Network

5. Select the Management Network from the **Management Network** drop-down menu to choose the NSX Advanced Load Balancer management network for service engines. Enter a static IP address pool for SEs and VIP, and click **Complete**.



EDIT CLOUD

sfo01w01vc01

Management Network*
sfo01-w01-vds01-albmanagement

IP Address Management for Management Network

Enable DHCP ☐

IP Subnet*
172.16.10.0/24

Default Gateway
172.16.10.1

Static IP Address Pool (1)*

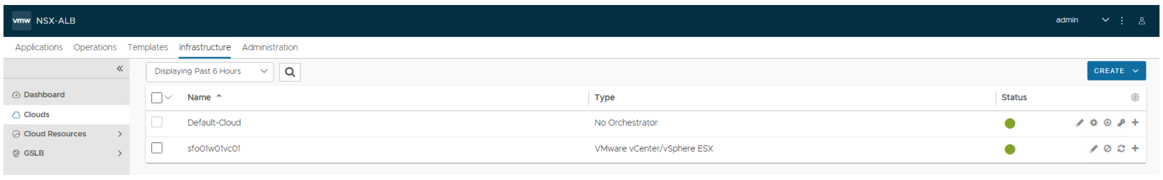
IP Address
172.16.10.100-172.16.10.200

Items per page 10 1 Total

Enable IPv6 Auto Configuration ☐

IPAM/DNS

6. Wait for the cloud to get configured and the status to turn green.



Name	Type	Status
Default-Cloud	No Orchestrator	●
sfo01w01vc01	VMware vCenter/vSphere ESX	●

7. To create a service engine group for Tanzu Kubernetes Grid management clusters, under the **Infrastructure** tab, go to **Cloud Resources** > **Service Engine Group**. From the **Select Cloud** drop-down menu, select the cloud created in the previous step and click **Create**.

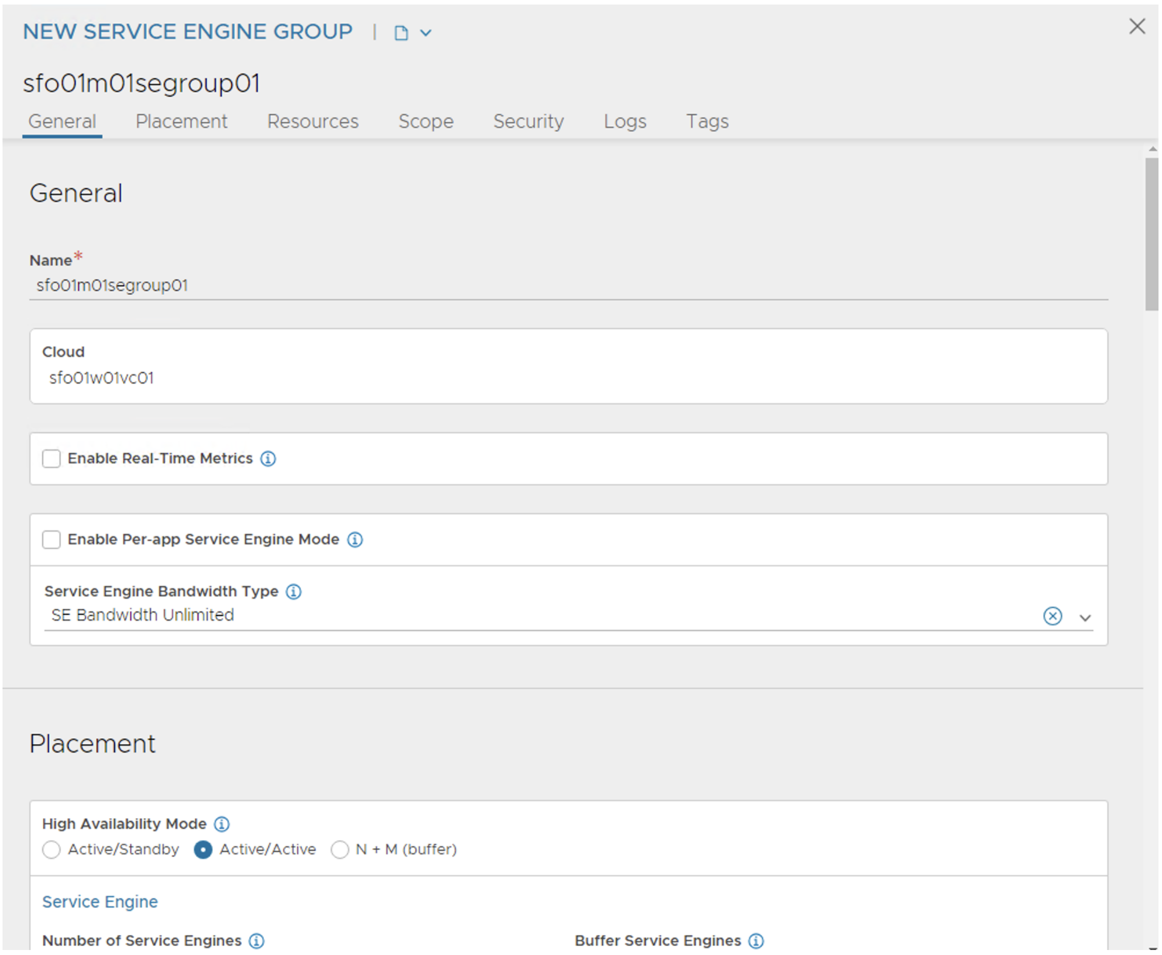
   The following components are created in NSX Advanced Load Balancer.

Object	Sample Name
vCenter Cloud	sfo01w01vc01
Service Engine Group 1	sfo01m01segroup01
Service Engine Group 2	sfo01w01segroup01

8. Enter a name for the Tanzu Kubernetes Grid management service engine group and set the following parameters:

Parameter	Value
High availability mode	Active/Active - NSX ALB Enterprise edition. Active/Standby - NSX ALB Essentials for Tanzu edition.
Enable Service Engine Self Election	Supported with NSX ALB Enterprise edition.
Memory for caching	Supported with NSX ALB Enterprise edition. You must set the value to 0 for essentials.
Memory per Service Engine	4
vCPU per Service Engine	2

   Use the default values for the rest of the parameters.



sfo01m01segroup01

General

Name*: sfo01m01segroup01

Cloud: sfo01w01vc01

☐ Enable Real-Time Metrics

☐ Enable Per-app Service Engine Mode

Service Engine Bandwidth Type: SE Bandwidth Unlimited

Placement

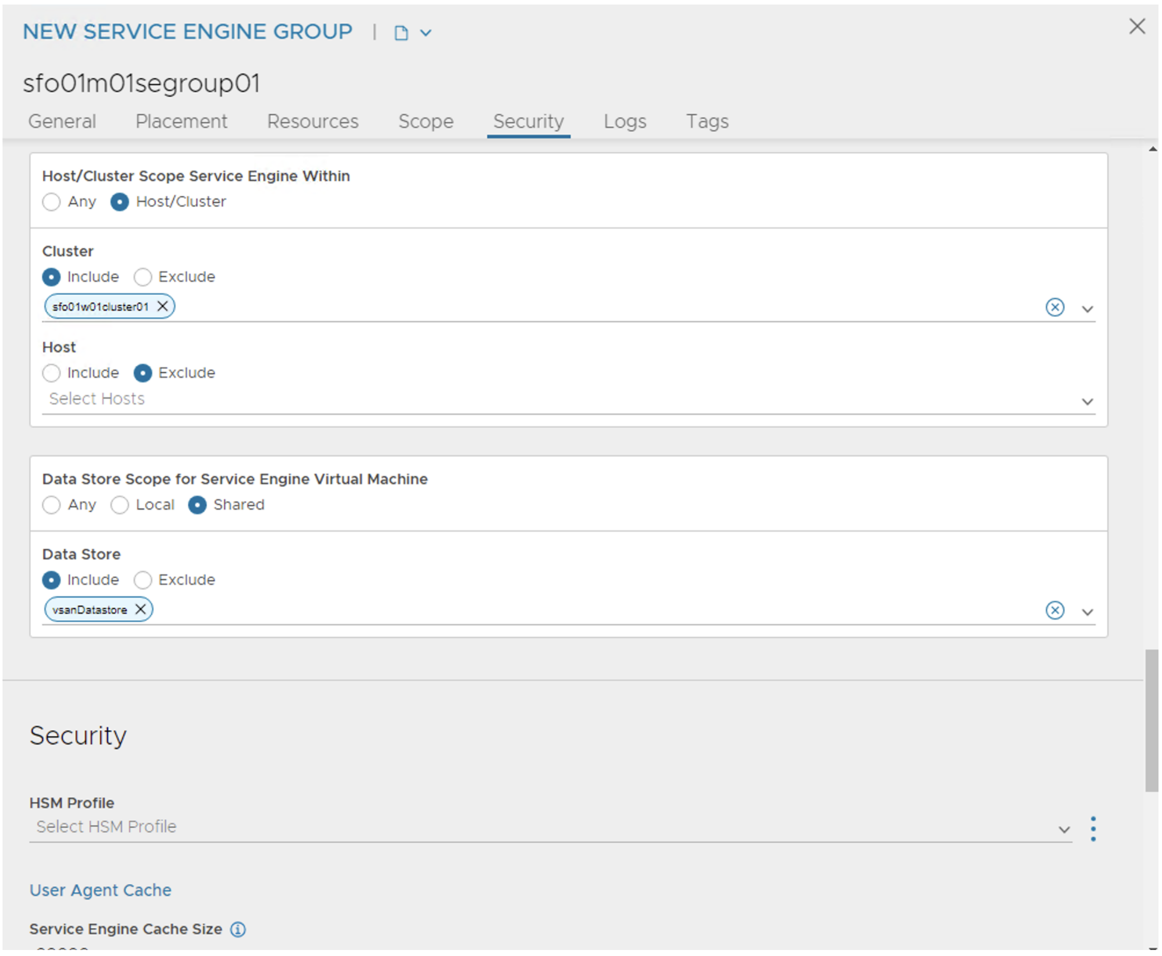
High Availability Mode: Active/Standby [✓ Active/Active] N + M (buffer)

Service Engine

Number of Service Engines

Buffer Service Engines

For advanced configuration, click on the **Advanced tab**, specify a specific cluster and datastore for service engine placement. Then, change the NSX_ALB SE folder name and service engine name prefix, and click **Save**.



Host/Cluster Scope Service Engine Within
Any [✓ Host/Cluster]

Cluster
[✓ Include] Exclude
sfo01w01cluster01

Host
Include [✓ Exclude]
Select Hosts

Data Store Scope for Service Engine Virtual Machine
Any Local [✓ Shared]

Data Store
[✓ Include] Exclude
vsanDatastore

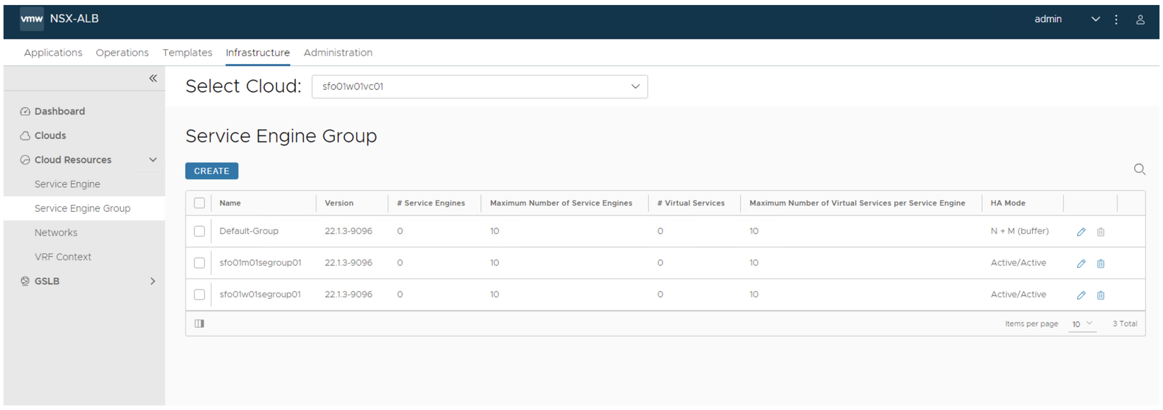
Security

HSM Profile
Select HSM Profile

User Agent Cache

Service Engine Cache Size

9. Repeat steps 7 and 8 to create another service engine group for Tanzu Kubernetes Grid workload clusters. After completing this step, you will have created two service engine groups.



Select Cloud: sfo01w01vc01

Service Engine Group

Name	Version	# Service Engines	Maximum Number of Service Engines	# Virtual Services	Maximum Number of Virtual Services per Service Engine	HA Mode
Default-Group	22.1.3-9096	0	10	0	10	N + M (buffer)
sfo01m01segroup01	22.1.3-9096	0	10	0	10	Active/Active
sfo01w01segroup01	22.1.3-9096	0	10	0	10	Active/Active

Items per page 10 3 Total

NSX Advanced Load Balancer: Configure Network and IPAM Profile

Configure Tanzu Kubernetes Grid Networks in NSX Advanced Load Balancer

As part of the cloud creation in NSX Advanced Load Balancer, only management network has been configured in NSX Advanced Load Balancer. Perform the following steps to configure these networks:

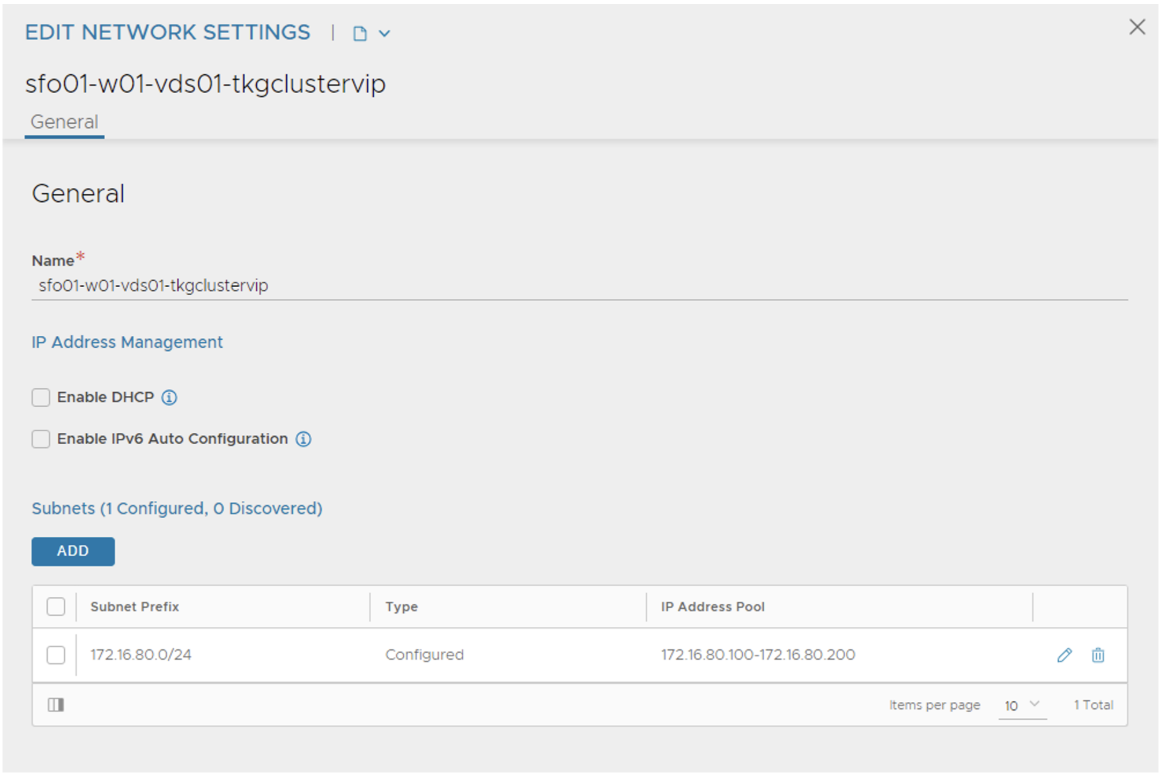
- TKG Management Network
- TKG Workload Network
- TKG Cluster VIP/Data Network

- TKG Management VIP/Data Network
- TKG Workload VIP/Data Network
- Log in to NSX Advanced Load Balancer and go to **Infrastructure** > **Cloud Resources** > **Networks**.
- Select the desired cloud. All the networks available in vCenter are listed.
- Click on the edit icon next for the network and configure as follows. Change the provided details as per your SDDC configuration.

> Not all networks are auto-discovered. For those networks, manually add the subnet.

Network Name	DHCP	Subnet	Static IP Pool
sfo01-w01-vds01-tkgmanagement	Yes	172.16.40.0/24	NA
sfo01-w01-vds01-tkgworkload	Yes	172.16.60.0/24	NA
sfo01-w01-vds01-tkgclustervip	No	172.16.80.0/24	172.16.80.100 - 172.16.80.200
sfo01-w01-vds01-tkgmanagementvip	No	172.16.50.0/24	172.16.50.100 - 172.16.50.200
sfo01-w01-vds01-tkgworkloadvip	No	172.16.70.0/24	172.16.70.100 - 172.16.70.200

The following image shows a sample network configuration for network `sfo01-w01-vds01-tkgclustervip`. You should apply the same configuration in `sfo01-w01-vds01-tkgmanagementvip` and `sfo01-w01-vds01-tkgworkloadvip`.



sfo01-w01-vds01-tkgclustervip

General

Name*
sfo01-w01-vds01-tkgclustervip

IP Address Management

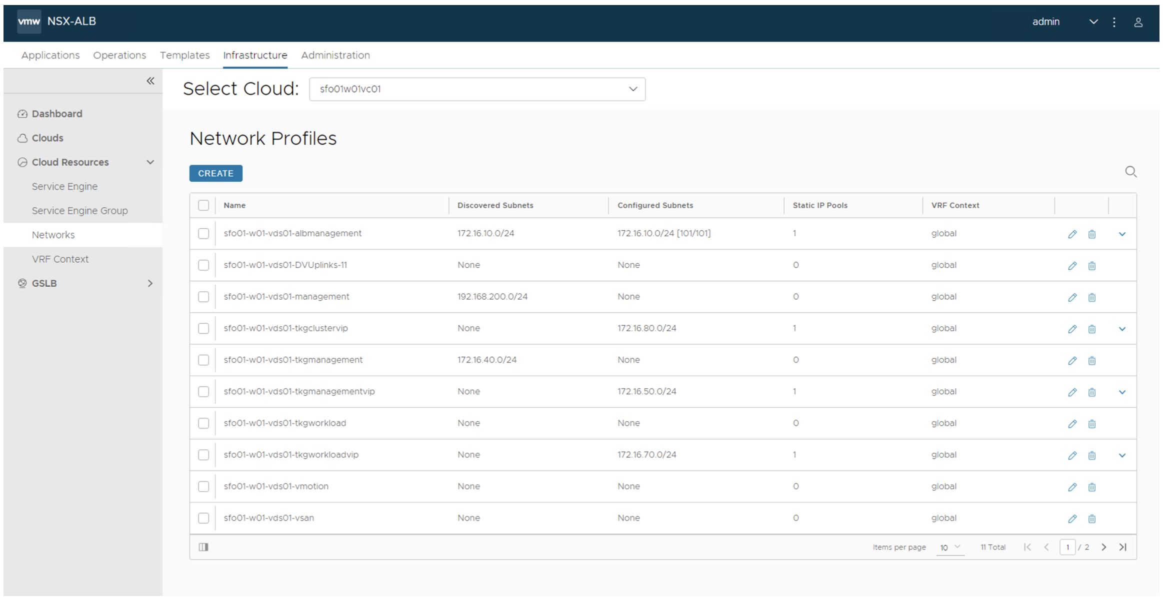
☐ Enable DHCP
☐ Enable IPv6 Auto Configuration

Subnets (1 Configured, 0 Discovered)

Subnet Prefix	Type	IP Address Pool
172.16.80.0/24	Configured	172.16.80.100-172.16.80.200

The `sfo01-w01-vds01-tkgmanagement` and `sfo01-w01-vds01-tkgworkload` network should be enabled with DHCP.

After the networks are configured, the configuration must look like the following image.



Select Cloud: sfo01w01vc01

Network Profiles

Name	Discovered Subnets	Configured Subnets	Static IP Pools	VRF Context
sfo01-w01-vds01-albmanagement	172.16.10.0/24	172.16.10.0/24 [101/101]	1	global
sfo01-w01-vds01-DVUplinks-11	None	None	0	global
sfo01-w01-vds01-management	192.168.200.0/24	None	0	global
sfo01-w01-vds01-tkgclustervip	None	172.16.80.0/24	1	global
sfo01-w01-vds01-tkgmanagement	172.16.40.0/24	None	0	global
sfo01-w01-vds01-tkgmanagementvip	None	172.16.50.0/24	1	global
sfo01-w01-vds01-tkgworkload	None	None	0	global
sfo01-w01-vds01-tkgworkloadvip	None	172.16.70.0/24	1	global
sfo01-w01-vds01-vmotion	None	None	0	global
sfo01-w01-vds01-vsan	None	None	0	global

Items per page 10 11 Total 1 / 2

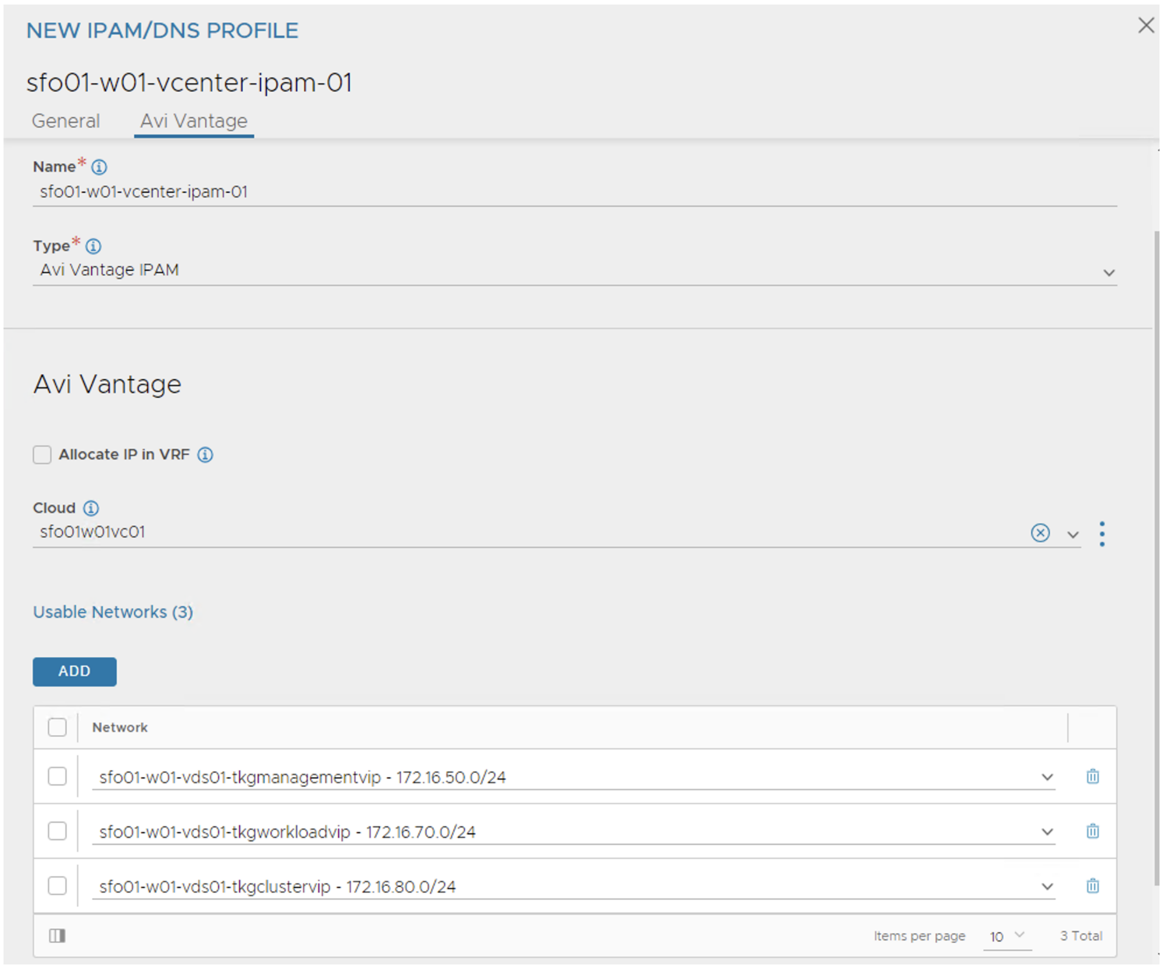
Create IPAM and DNS Profile in NSX Advanced Load Balancer and Attach to Cloud

At this point, all the required networks related to Tanzu functionality are configured in NSX Advanced Load Balancer, except for Tanzu Kubernetes Grid management and workload network which uses DHCP. NSX Advanced Load Balancer provides IPAM service for Tanzu Kubernetes Grid cluster VIP network, management VIP network, and workload VIP network.

Perform the following steps to create an IPAM profile and attach it to the vCenter cloud created earlier:

1. Log in to NSX Advanced Load Balancer and go to **Templates** > **Profiles** > **IPAM/DNS Profiles** > **Create** > **IPAM Profile**, provide the following details, and click **Save**.

Parameter	Value
Name	sfo01-w01-vcenter-ipam-01
Type	AVI Vintage IPAM
Cloud for Usable Networks	Tanzu-vcenter-01 (created earlier in this deployment)
Usable Networks	sfo01-w01-vds01-tkgclustervip sfo01-w01-vds01-tkgmanagementvip sfo01-w01-vds01-tkgworkloadvip


NEW IPAM/DNS PROFILE

sfo01-w01-vcenter-ipam-01

General | Avi Vantage

Name*: sfo01-w01-vcenter-ipam-01

Type*: Avi Vantage IPAM

Avi Vantage

☐ Allocate IP in VRF

Cloud: sfo01w01vc01

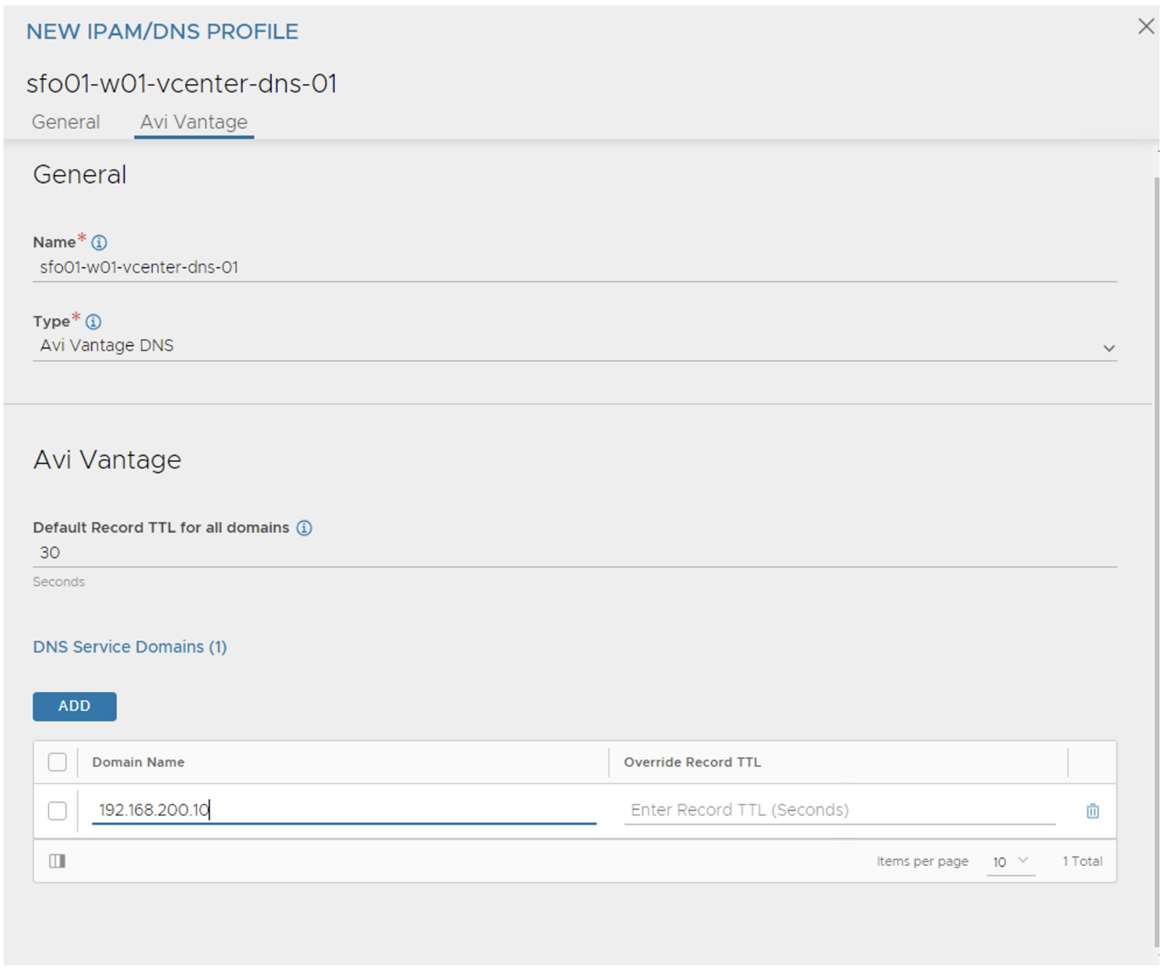
Usable Networks (3)

ADD

	Network
☐	sfo01-w01-vds01-tkgmanagementvip - 172.16.50.0/24
☐	sfo01-w01-vds01-tkgworkloadvip - 172.16.70.0/24
☐	sfo01-w01-vds01-tkgclustervip - 172.16.80.0/24

Items per page 10 · 3 Total

2. Click **Create** > **DNS Profile** and provide the domain name.


NEW IPAM/DNS PROFILE

sfo01-w01-vcenter-dns-01

General

Name* sfo01-w01-vcenter-dns-01

Type* Avi Vantage DNS

Avi Vantage

Default Record TTL for all domains: 30 Seconds

DNS Service Domains (1)

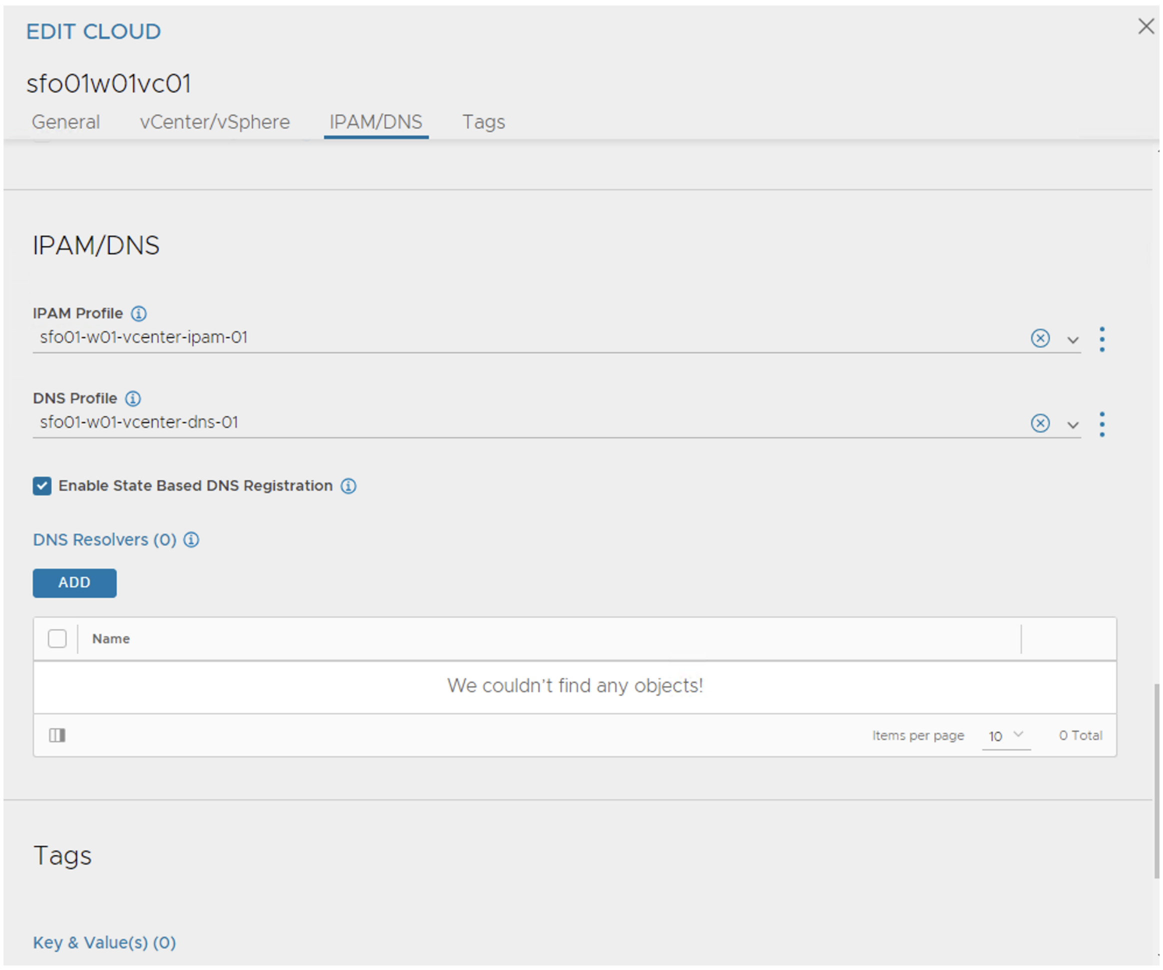
Domain Name	Override Record TTL
192.168.200.10	Enter Record TTL (Seconds)

3. Attach the IPAM and DNS profiles to the `sfo01w01vc01` cloud.

   1. Navigate to **Infrastructure** > **Clouds**.

   2. Edit the sfo01w01vc01 cloud.

   3. Under IPAM/DNS section, choose the IPAM and DNS profiles created earlier and save the updated configuration.


EDIT CLOUD

sfo01w01vc01

General | vCenter/vSphere | IPAM/DNS | Tags

IPAM/DNS

IPAM Profile: sfo01-w01-vcenter-ipam-01

DNS Profile: sfo01-w01-vcenter-dns-01

☑ Enable State Based DNS Registration

DNS Resolvers (0)

ADD

	Name
We couldn't find any objects!	

Items per page 10 | 0 Total

Tags

Key & Value(s) (0)

The above steps complete the NSX Advanced Load Balancer configuration. The next step is to deploy and configure a bootstrap machine. The bootstrap machine is used to deploy and manage Tanzu Kubernetes clusters.

Deploy Tanzu Kubernetes Grid Management Cluster

The management cluster is a Kubernetes cluster that runs Cluster API operations on a specific cloud provider to create and manage workload clusters on that provider.

The management cluster is where you configure the shared and in-cluster services that the workload clusters use.

You can deploy management clusters in the following ways:

- Run the Tanzu Kubernetes Grid installer, a wizard interface that guides you through the process of deploying a management cluster.
- Create and edit YAML configuration files, and use them with Tanzu CLI commands to deploy a management cluster.

Before creating a management cluster using the Tanzu CLI, you must define its configuration in a YAML configuration file that provides the base configuration for the cluster. When you deploy the management cluster from the CLI, you specify the YAML file by using the `--file` option of the `tanzu mc create` command.

In an air-gapped environment, we recommend deploying a management cluster using a YAML configuration file. You can use the templates provided in the following section to deploy management clusters on vSphere.

Management Cluster Configuration Template

The templates include all of the options that are relevant to deploying management clusters on vSphere. You can copy this template and use it to deploy management clusters to vSphere.

The environment variables that you have set, override values from a cluster configuration file. To use all settings from a cluster configuration file, remove any conflicting environment variables before you deploy the management cluster from the CLI.

Image repository configuration is very important details which will not be part of default config file when we are creating from TKG UI.

```
#! ---------------------------------------------------------------------
#! Basic cluster creation configuration
#! ---------------------------------------------------------------------

CLUSTER_NAME:
CLUSTER_PLAN: <dev/prod>
INFRASTRUCTURE_PROVIDER: vsphere
ENABLE_CEIP_PARTICIPATION: <true/false>
ENABLE_AUDIT_LOGGING: <true/false>
CLUSTER_CIDR: 100.96.0.0/11
SERVICE_CIDR: 100.64.0.0/13
# CAPBK_BOOTSTRAP_TOKEN_TTL: 30m

#! ---------------------------------------------------------------------
#! vSphere configuration
#! ---------------------------------------------------------------------

VSPHERE_SERVER:
VSPHERE_USERNAME:
VSPHERE_PASSWORD:
VSPHERE_DATACENTER:
VSPHERE_RESOURCE_POOL:
VSPHERE_DATASTORE:
VSPHERE_FOLDER:
VSPHERE_NETWORK: <tkg-management-network>
VSPHERE_CONTROL_PLANE_ENDPOINT: #Leave blank as VIP network is configured in NSX ALB a
nd IPAM is configured with VIP network

# VSPHERE_TEMPLATE:

VSPHERE_SSH_AUTHORIZED_KEY:
VSPHERE_TLS_THUMBPRINT:
VSPHERE_INSECURE: <true/false>
DEPLOY_TKG_ON_VSPHERE7: true

#! ---------------------------------------------------------------------
#! Node configuration
#! ---------------------------------------------------------------------
```

```
# SIZE:
# CONTROLPLANE_SIZE:
# WORKER_SIZE:
# OS_NAME: ""
# OS_VERSION: ""
# OS_ARCH: ""
# VSPHERE_NUM_CPUS: 2
# VSPHERE_DISK_GIB: 40
# VSPHERE_MEM_MIB: 4096
# VSPHERE_CONTROL_PLANE_NUM_CPUS: 2
# VSPHERE_CONTROL_PLANE_DISK_GIB: 40
# VSPHERE_CONTROL_PLANE_MEM_MIB: 8192
# VSPHERE_WORKER_NUM_CPUS: 2
# VSPHERE_WORKER_DISK_GIB: 40
# VSPHERE_WORKER_MEM_MIB: 4096

#! ---------------------------------------------------------------------
#! NSX Advanced Load Balancer configuration
#! ---------------------------------------------------------------------

AVI_CA_DATA_B64:
AVI_CLOUD_NAME:
AVI_CONTROL_PLANE_HA_PROVIDER: <true/false>
AVI_CONTROL_PLANE_NETWORK:
AVI_CONTROL_PLANE_NETWORK_CIDR:
AVI_CONTROLLER:
AVI_DATA_NETWORK:
AVI_DATA_NETWORK_CIDR:
AVI_ENABLE: <true/false>
AVI_LABELS:
AVI_MANAGEMENT_CLUSTER_CONTROL_PLANE_VIP_NETWORK_CIDR:
AVI_MANAGEMENT_CLUSTER_CONTROL_PLANE_VIP_NETWORK_NAME:
AVI_MANAGEMENT_CLUSTER_SERVICE_ENGINE_GROUP:
AVI_MANAGEMENT_CLUSTER_VIP_NETWORK_CIDR:
AVI_MANAGEMENT_CLUSTER_VIP_NETWORK_NAME:
AVI_PASSWORD: <base 64 encoded AVI password>
AVI_SERVICE_ENGINE_GROUP:
AVI_USERNAME:


#! ---------------------------------------------------------------------
#! Image repository configuration
#! ---------------------------------------------------------------------

TKG_CUSTOM_IMAGE_REPOSITORY: ""
TKG_CUSTOM_IMAGE_REPOSITORY_SKIP_TLS_VERIFY: false
TKG_CUSTOM_IMAGE_REPOSITORY_CA_CERTIFICATE: ""

#! ---------------------------------------------------------------------
#! Machine Health Check configuration
#! ---------------------------------------------------------------------

ENABLE_MHC:
# ENABLE_MHC_CONTROL_PLANE: <true/false>
# ENABLE_MHC_WORKER_NODE: <true/flase>

#! ---------------------------------------------------------------------
#! Identity management configuration
#! ---------------------------------------------------------------------

IDENTITY_MANAGEMENT_TYPE: "none"

#! Settings for IDENTITY_MANAGEMENT_TYPE: "oidc"
# CERT_DURATION: 2160h
# CERT_RENEW_BEFORE: 360h
# OIDC_IDENTITY_PROVIDER_CLIENT_ID:
# OIDC_IDENTITY_PROVIDER_CLIENT_SECRET:
# OIDC_IDENTITY_PROVIDER_GROUPS_CLAIM: groups
# OIDC_IDENTITY_PROVIDER_ISSUER_URL:
# OIDC_IDENTITY_PROVIDER_SCOPES: "email,profile,groups"
# OIDC_IDENTITY_PROVIDER_USERNAME_CLAIM: email

#! Settings for IDENTITY_MANAGEMENT_TYPE: "ldap"
# LDAP_BIND_DN:
# LDAP_BIND_PASSWORD:
# LDAP_HOST:
# LDAP_USER_SEARCH_BASE_DN:
# LDAP_USER_SEARCH_FILTER:
# LDAP_USER_SEARCH_USERNAME: userPrincipalName
# LDAP_USER_SEARCH_ID_ATTRIBUTE: DN
# LDAP_USER_SEARCH_EMAIL_ATTRIBUTE: DN
# LDAP_USER_SEARCH_NAME_ATTRIBUTE:
# LDAP_GROUP_SEARCH_BASE_DN:
# LDAP_GROUP_SEARCH_FILTER:
# LDAP_GROUP_SEARCH_USER_ATTRIBUTE: DN
# LDAP_GROUP_SEARCH_GROUP_ATTRIBUTE:
# LDAP_GROUP_SEARCH_NAME_ATTRIBUTE: cn
# LDAP_ROOT_CA_DATA_B64:
```

For a full list of configurable values and to learn more about the fields present in the template file, see Tanzu Configuration File Variable Reference.

Create a file using the values provided in the template and save the file with a `.yaml` extension. For more information about a sample YAML file to use for deploying a management cluster, see Appendix Section.

After you have created or updated the cluster configuration file, you can deploy a management cluster by running the `tanzu mc create --file CONFIG-FILE` command, where CONFIG-FILE is the name of the configuration file. Below is the sample config file for deploying the TKG Management cluster in an air-gapped environment.

```
#! ---------------------------------------------------------------------
#! Basic cluster creation configuration
#! ---------------------------------------------------------------------

CLUSTER_NAME: sfo01w01vc01
CLUSTER_PLAN: prod
INFRASTRUCTURE_PROVIDER: vsphere
ENABLE_CEIP_PARTICIPATION: "true"
ENABLE_AUDIT_LOGGING: "true"
CLUSTER_CIDR: 100.96.0.0/11
SERVICE_CIDR: 100.64.0.0/13
# CAPBK_BOOTSTRAP_TOKEN_TTL: 30m

#! ---------------------------------------------------------------------
#! vSphere configuration
#! ---------------------------------------------------------------------
```

```
VSPHERE_SERVER: sfo01w01vc01.sfo01.rainpole.local
VSPHERE_USERNAME: administrator@vsphere.local
VSPHERE_PASSWORD: <encoded:Vk13YXJlMSE=>
VSPHERE_DATACENTER: /sfo01w01dc01
VSPHERE_RESOURCE_POOL: /sfo01w01dc01/host/tkg-management-components/Resources/tkg-mana
gement-components
VSPHERE_DATASTORE: /sfo01w01dc01datastore/vsanDatastore
VSPHERE_FOLDER: /sfo01w01dc01vm/tkg-management-components
VSPHERE_NETWORK: /sfo01w01dc01/network/sfo01-w01-vds01-tkgmanagement
VSPHERE_CONTROL_PLANE_ENDPOINT: #Leave blank as VIP network is configured in NSX ALB a
nd IPAM is configured with VIP network

# VSPHERE_TEMPLATE:

VSPHERE_SSH_AUTHORIZED_KEY: ssh-rsa AAAA[...]== email@example.com
VSPHERE_TLS_THUMBPRINT: DC:FA:81:1D:CA:08:21:AB:4E:15:BD:2B:AE:12:2C:6B:CA:65:49:B8
VSPHERE_INSECURE: "false"
DEPLOY_TKG_ON_VSPHERE7: true

#! ---------------------------------------------------------------------
#! Node configuration
#! ---------------------------------------------------------------------

OS_NAME: photon
OS_VERSION: "3"
OS_ARCH: amd64
VSPHERE_CONTROL_PLANE_NUM_CPUS: 2
VSPHERE_CONTROL_PLANE_DISK_GIB: 40
VSPHERE_CONTROL_PLANE_MEM_MIB: 8192
VSPHERE_WORKER_NUM_CPUS: 2
VSPHERE_WORKER_DISK_GIB: 40
VSPHERE_WORKER_MEM_MIB: 8192

#! ---------------------------------------------------------------------
#! NSX Advanced Load Balancer configuration
#! ---------------------------------------------------------------------

AVI_CA_DATA_B64: LS0t[...]tLS0tLQ==
AVI_CLOUD_NAME: sfo01w01vc01
AVI_CONTROL_PLANE_HA_PROVIDER: "true"
AVI_CONTROL_PLANE_NETWORK: sfo01-w01-vds01-tkgclustervip
AVI_CONTROL_PLANE_NETWORK_CIDR: 172.16.80.0/24
AVI_CONTROLLER: sfo01albctlr01.sfo01.rainpole.local
AVI_DATA_NETWORK: sfo01-w01-vds01-tkgworkloadvip
AVI_DATA_NETWORK_CIDR: 172.16.70.0/24
AVI_ENABLE: "true"
AVI_LABELS:
AVI_MANAGEMENT_CLUSTER_CONTROL_PLANE_VIP_NETWORK_CIDR: 172.16.80.0/24
AVI_MANAGEMENT_CLUSTER_CONTROL_PLANE_VIP_NETWORK_NAME: sfo01-w01-vds01-tkgclustervip
AVI_MANAGEMENT_CLUSTER_SERVICE_ENGINE_GROUP: sfo01m01segroup01
AVI_MANAGEMENT_CLUSTER_VIP_NETWORK_CIDR: 172.16.50.0/24
AVI_MANAGEMENT_CLUSTER_VIP_NETWORK_NAME: sfo01-w01-vds01-tkgmanagementvip
AVI_PASSWORD: <encoded:Vk13YXJlMSE=>
AVI_SERVICE_ENGINE_GROUP: sfo01w01segroup01
AVI_USERNAME: admin


#! ---------------------------------------------------------------------
#! Image repository configuration
#! ---------------------------------------------------------------------

TKG_CUSTOM_IMAGE_REPOSITORY: "harbor-sa.lab.vmw/tkg-160"
TKG_CUSTOM_IMAGE_REPOSITORY_SKIP_TLS_VERIFY: false
TKG_CUSTOM_IMAGE_REPOSITORY_CA_CERTIFICATE: LS0t[...]tLS0tLQ==

#! ---------------------------------------------------------------------
#! Machine Health Check configuration
#! ---------------------------------------------------------------------

ENABLE_MHC: true

#! ---------------------------------------------------------------------
#! Identity management configuration
#! ---------------------------------------------------------------------

IDENTITY_MANAGEMENT_TYPE: "none"

#! ---------------------------------------------------------------------
```

To create Management cluster, execute the following command:

```
tanzu management-cluster create --file config.yaml
```

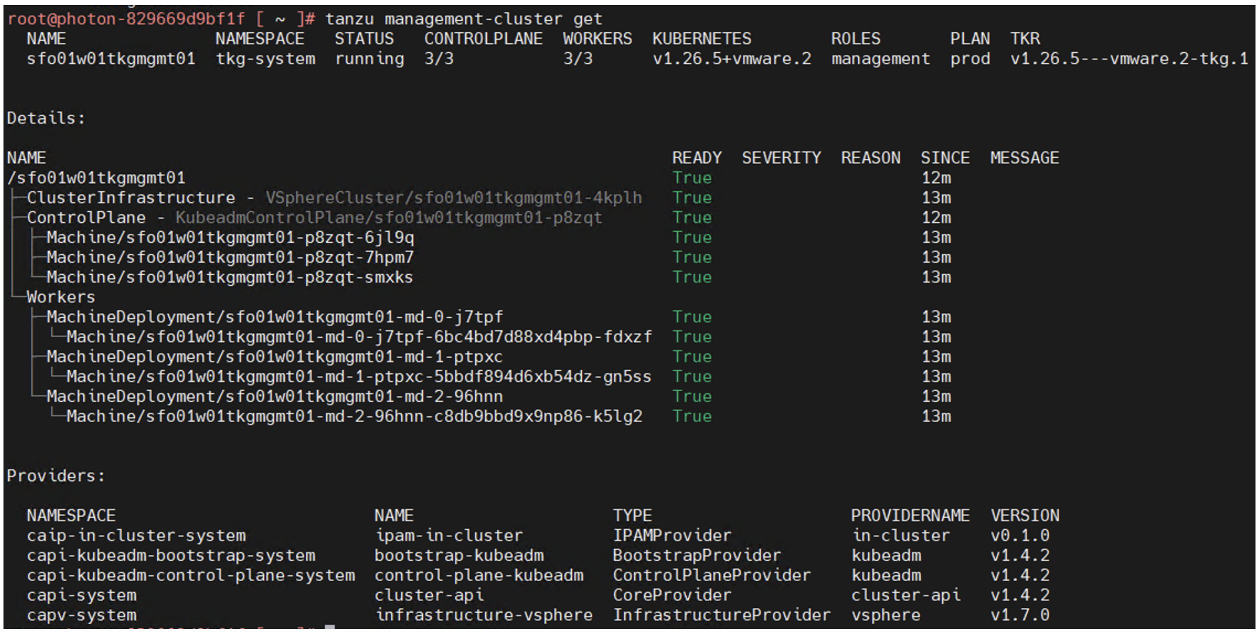
The cluster deployment logs are streamed in the terminal when you run the `tanzu mc create` command. The first run of `tanzu mc create` takes longer than subsequent runs because it has to pull the required Docker images into the image store on your bootstrap machine. Subsequent runs do not require this step, and thus the process is faster.

While the cluster is being deployed, you will find that a virtual service is created in NSX Advanced Load Balancer and new service engines are deployed in vCenter by NSX Advanced Load Balancer. The service engines are mapped to the SE Group `sfo01m01segroup01`.

Now you can access the Tanzu Kubernetes Grid management cluster from the bootstrap machine and perform additional tasks such as verifying the management cluster health and deploying the workload clusters.

To get the status of the Tanzu Kubernetes Grid management cluster execute the following command:

```
tanzu management-cluster get
```



```
root@photon-829669d9bf1f [ ~ ]# tanzu management-cluster get
  NAME                NAMESPACE   STATUS   CONTROLPLANE  WORKERS  KUBERNETES        ROLES       PLAN  TKR
  sfo01w01tkgmgmt01  tkg-system  running  3/3           3/3      v1.26.5+vmware.2  management  prod  v1.26.5---vmware.2-tkg.1


Details:

NAME                                                                  READY  SEVERITY  REASON  SINCE  MESSAGE
/sfo01w01tkgmgmt01                                                    True                     12m
├─ClusterInfrastructure - VSphereCluster/sfo01w01tkgmgmt01-4kplh      True                     13m
├─ControlPlane - KubeadmControlPlane/sfo01w01tkgmgmt01-p8zqt          True                     12m
│ ├─Machine/sfo01w01tkgmgmt01-p8zqt-6jl9q                             True                     13m
│ ├─Machine/sfo01w01tkgmgmt01-p8zqt-7hpm7                             True                     13m
│ └─Machine/sfo01w01tkgmgmt01-p8zqt-smxks                             True                     13m
└─Workers
  ├─MachineDeployment/sfo01w01tkgmgmt01-md-0-j7tpf                    True                     13m
  │ └─Machine/sfo01w01tkgmgmt01-md-0-j7tpf-6bc4bd7d88xd4pbp-fdxzf     True                     13m
  ├─MachineDeployment/sfo01w01tkgmgmt01-md-1-ptpxc                    True                     13m
  │ └─Machine/sfo01w01tkgmgmt01-md-1-ptpxc-5bbdf894d6xb54dz-gn5ss     True                     13m
  └─MachineDeployment/sfo01w01tkgmgmt01-md-2-96hnn                    True                     13m
    └─Machine/sfo01w01tkgmgmt01-md-2-96hnn-c8db9bbd9x9np86-k5lg2      True                     13m


Providers:

  NAMESPACE                          NAME                    TYPE                    PROVIDERNAME  VERSION
  caip-in-cluster-system             ipam-in-cluster         IPAMProvider            in-cluster    v0.1.0
  capi-kubeadm-bootstrap-system      bootstrap-kubeadm       BootstrapProvider       kubeadm       v1.4.2
  capi-kubeadm-control-plane-system  control-plane-kubeadm   ControlPlaneProvider    kubeadm       v1.4.2
  capi-system                        cluster-api             CoreProvider            cluster-api   v1.4.2
  capv-system                        infrastructure-vsphere  InfrastructureProvider  vsphere       v1.7.0
```

To interact with the management cluster using the kubectl command, retrieve the management cluster `kubeconfig` and switch to the cluster context to run kubectl commands.

```
# kubectl config get-contexts
Get Context of  Management cluster

]# kubectl config use-context sfo01w01tkgmgmt01-admin@sfo01w01tkgmgmt01
Switched to context "sfo01w01tkgmgmt01-admin@sfo01w01tkgmgmt01".

]# kubectl get nodes
NAME                                                 STATUS   ROLES           AGE
VERSION
sfo01w01tkgmgmt01-md-0-fsf4s-85b59574dx8pgbx-rkndw   Ready    <none>          3d13h
v1.26.5+vmware.2
sfo01w01tkgmgmt01-md-1-x42wp-7c689d7f44xbpj9z-djmhw  Ready    <none>          3d13h
v1.26.5+vmware.2
sfo01w01tkgmgmt01-md-2-5gzjf-74dd88f65bxkcczp-twqnv  Ready    <none>          3d13h
v1.26.5+vmware.2
sfo01w01tkgmgmt01-rbnp4-lmlsf                        Ready    control-plane   3d13h
v1.26.5+vmware.2
sfo01w01tkgmgmt01-rbnp4-pj29x                        Ready    control-plane   3d13h
v1.26.5+vmware.2
sfo01w01tkgmgmt01-rbnp4-zfwwd                        Ready    control-plane   3d13h
v1.26.5+vmware.2



# kubectl get apps -A
tkg-system   ako-operator                                      Reconcile succeeded
4m55s          3d13h
tkg-system   sfo01w01tkgmgmt01-antrea                          Reconcile succeeded
13m            3d13h
tkg-system   sfo01w01tkgmgmt01-capabilities                    Reconcile succeeded
79s            3d13h
tkg-system   sfo01w01tkgmgmt01-load-balancer-and-ingress-service   Reconcile succeeded
9m11s          3d13h
tkg-system   sfo01w01tkgmgmt01-metrics-server                  Reconcile succeeded
61s            3d13h
```

```
tkg-system   sfo01w01tkgmgmt01-pinniped                          Reconcile succeeded
7m6s           3d13h
tkg-system   sfo01w01tkgmgmt01-secretgen-controller              Reconcile succeeded
33s            3d13h
tkg-system   sfo01w01tkgmgmt01-tkg-storageclass                  Reconcile succeeded
2m             3d13h
tkg-system   sfo01w01tkgmgmt01-vsphere-cpi                       Reconcile succeeded
3m32s          3d13h
tkg-system   sfo01w01tkgmgmt01-vsphere-csi                       Reconcile succeeded
10m            3d13h
tkg-system   tanzu-addons-manager                                Reconcile succeeded
103s           3d13h
tkg-system   tanzu-auth                                          Reconcile succeeded
39s            3d13h
tkg-system   tanzu-cliplugins                                    Reconcile succeeded
7m53s          3d13h
tkg-system   tanzu-core-management-plugins                       Reconcile succeeded
8m10s          3d13h
tkg-system   tanzu-featuregates                                  Reconcile succeeded
3m6s           3d13h
tkg-system   tanzu-framework                                     Reconcile succeeded
32s            3d13h
tkg-system   tkg-clusterclass                                    Reconcile succeeded
5m22s          3d13h
tkg-system   tkg-pkg                                             Reconcile succeeded
67s            3d13h
tkg-system   tkr-service                                         Reconcile succeeded
9m37s          3d13h
tkg-system   tkr-source-controller                               Reconcile succeeded
2m1s           3d13h
tkg-system   tkr-vsphere-resolver                                Reconcile succeeded
111s           3d13h
```

The Tanzu Kubernetes Grid management cluster is successfully deployed. You can now proceed with configuring custom ADCs, and creating shared services and workload clusters.

Configure AKO Deployment Config (ADC) for Workload Clusters

Tanzu Kubernetes Grid v2.3.0 management clusters with NSX Advanced Load Balancer are deployed with the following 2 AKODeploymentConfigs:

- `install-ako-for-management-cluster`: default configuration for management cluster
- `install-ako-for-all`: default configuration for all workload clusters. By default, all the workload clusters reference this file for their virtual IP networks and service engine (SE) groups. This ADC configuration does not enable NSX L7 Ingress by default.

As per this Tanzu deployment, create 2 more ADCs:

- `tanzu-ako-for-shared`: Used by shared services cluster to deploy the virtual services in `TKG Mgmt SE Group` and the loadbalancer applications in `TKG Management VIP Network`.
- `tanzu-ako-for-workload-L7-ingress`: Use this ADC only if you would like to enable NSX Advanced Load Balancer L7 ingress on workload cluster. Otherwise, leave the cluster labels empty to apply the network configuration from default ADC `install-ako-for-all`.

Configure AKODeploymentConfig (ADC) for Shared Services Cluster

As per the defined architecture, shared services cluster uses the same control plane and data plane network as the management cluster. Shared services cluster control plane endpoint uses `TKG Cluster VIP Network`, application loadbalancing uses `TKG Management Data VIP network` and the virtual services are deployed in `sfo01m01segroup01` SE group. This configuration is enforced by creating a custom AKO Deployment Config (ADC) and applying the respective `NSXALB_LABELS` while deploying the shared services cluster.

The format of the AKODeploymentConfig YAML file is as follows:

```
apiVersion: networking.tkg.tanzu.vmware.com/v1alpha1
kind: AKODeploymentConfig
metadata:
  finalizers:
     - ako-operator.networking.tkg.tanzu.vmware.com
  generation: 2
  name: <Unique name of AKODeploymentConfig>
spec:
  adminCredentialRef:
    name: nsx-alb-controller-credentials
    namespace: tkg-system-networking
  certificateAuthorityRef:
    name: nsx-alb-controller-ca
    namespace: tkg-system-networking
  cloudName: <NAME OF THE CLOUD in ALB>
  clusterSelector:
    matchLabels:
      <KEY>: <VALUE>
  controlPlaneNetwork:
    cidr: <TKG-Cluster-VIP-CIDR>
    Name: <TKG-Cluster-VIP-Network>
  controller: <NSX ALB CONTROLLER IP/FQDN>
  dataNetwork:
    cidr: <TKG-Mgmt-Data-VIP-CIDR>
    name: <TKG-Mgmt-Data-VIP-Name>
  extraConfigs:
   cniPlugin: antrea
   disableStaticRouteSync: true
   ingress:
      defaultIngressController: false
      disableIngressClass: true
      nodeNetworkList:
      - networkName: <TKG-Mgmt-Network>
  serviceEngineGroup: <Mgmt-Cluster-SEG>
```

The sample AKODeploymentConfig with sample values in place is as follows. You should add the respective NSX_ALB label `type=shared-services` while deploying shared services cluster to enforce this network configuration.

- cloud: `sfo01w01vc01`
- service engine group: `sfo01m01segroup01`
- Control Plane network: `sfo01-w01-vds01-tkgclustervip`

- VIP/data network: `sfo01-w01-vds01-tkgmanagementvip`
- Node Network: `sfo01-w01-vds01-tkgmanagement`

```
apiVersion: networking.tkg.tanzu.vmware.com/v1alpha1
kind: AKODeploymentConfig
metadata:
  name: tanzu-ako-for-shared
spec:
  adminCredentialRef:
    name: avi-controller-credentials
    namespace: tkg-system-networking
  certificateAuthorityRef:
    name: avi-controller-ca
    namespace: tkg-system-networking
  cloudName: sfo01w01vc01
  clusterSelector:
    matchLabels:
      type: shared-services
  controlPlaneNetwork:
    cidr: 172.16.80.0/24
    name: sfo01-w01-vds01-tkgclustervip
  controller: 172.16.10.10
  dataNetwork:
    cidr: 172.16.80.0/24
    name: sfo01-w01-vds01-tkgclustervip
  extraConfigs:
    cniPlugin: antrea
    disableStaticRouteSync: true
    ingress:
      defaultIngressController: false
      disableIngressClass: true
      nodeNetworkList:
      - networkName: sfo01-w01-vds01-tkgshared
  serviceEngineGroup: sfo01m01segroup01
```

After you have the AKO configuration file ready, use the `kubectl` command to set the context to Tanzu Kubernetes Grid management cluster and create the ADC:

```
# kubectl config use-context sfo01w01tkgmgmt01-admin@sfo01w01tkgmgmt01
Switched to context "sfo01w01tkgmgmt01-admin@sfo01w01tkgmgmt01".


# kubectl apply -f ako-shared-services.yaml
akodeploymentconfig.networking.tkg.tanzu.vmware.com/tanzu-ako-for-shared created
```

Use the following command to list all AKODeploymentConfig created under the management cluster:

```
# kubectl get adc
NAME                                 AGE
install-ako-for-all                  21h
install-ako-for-management-cluster   21h
tanzu-ako-for-shared                 113s
```

Configure AKO Deployment Config (ADC) for Workload Cluster to Enable NSX ALB L7 Ingress with NodePortLocal Mode

VMware recommends using NSX Advanced Load Balancer L7 ingress with NodePortLocal mode for the L7 application load balancing. This is enabled by creating a custom ADC with ingress settings enabled, and then applying the `AVI_LABELS` while deploying the workload cluster.

As per the defined architecture, workload cluster cluster control plane endpoint uses `TKG Cluster VIP Network`, application loadbalancing uses `TKG Workload Data VIP network`, and the virtual services are deployed in `sfo01w01segroup01` SE group.

Below are the changes in ADC Ingress section when compare to the default ADC.

- **disableIngressClass**: set to `false` to enable NSX ALB L7 Ingress.
- **nodeNetworkList**: Provide the values for TKG workload network name and CIDR.
- **serviceType**: L7 Ingress type. We recommend to use `NodePortLocal`.
- **shardVSSize**: Virtual service size.

NSX ALB L7 Ingress feature requires Enterprise edition license. If you do not wish to enable L7 feature/applied with ALB essentials for Tanzu license, disable the L7 feature by setting the value `disableIngressClass` to `true`.

The format of the AKODeploymentConfig YAML file for enabling NSX ALB L7 Ingress is as follows:

```
apiVersion: networking.tkg.tanzu.vmware.com/v1alpha1
kind: AKODeploymentConfig
metadata:
  name: <unique-name-for-adc>
spec:
  adminCredentialRef:
    name: NSX_ALB-controller-credentials
    namespace: tkg-system-networking
  certificateAuthorityRef:
    name: NSX_ALB-controller-ca
    namespace: tkg-system-networking
  cloudName: <cloud name configured in nsx alb>
  clusterSelector:
    matchLabels:
      <KEY>: <value>
  controller: <ALB-Controller-IP/FQDN>
  controlPlaneNetwork:
    cidr: <TKG-Cluster-VIP-Network-CIDR>
    name: <TKG-Cluster-VIP-Network-CIDR>
  dataNetwork:
    cidr: <TKG-Workload-VIP-network-CIDR>
    name: <TKG-Workload-VIP-network-CIDR>
  serviceEngineGroup: <Workload-Cluster-SEG>
  extraConfigs:
    cniPlugin: antrea
    disableStaticRouteSync: false                               # required
    ingress:
      disableIngressClass: false                                # required
      nodeNetworkList:                                          # required
        - networkName: <TKG-Workload-Network>
          cidrs:
            - <TKG-Workload-Network-CIDR>
      serviceType: NodePortLocal                                # required
      shardVSSize: MEDIUM                                       # required
```

The AKODeploymentConfig with sample values in place is as follows. You must add the respective NSX ALB label `workload-l7-enabled=true` while deploying shared services cluster to enforce this network configuration.

- cloud: `sfo01w01vc01`
- service engine group: `sfo01w01segroup01`
- Control Plane network: `sfo01-w01-vds01-tkgclustervip`
- VIP/data network: `sfo01-w01-vds01-tkgworkloadvip`
- Node Network: `sfo01-w01-vds01-tkgworkload`

```
apiVersion: networking.tkg.tanzu.vmware.com/v1alpha1
kind: AKODeploymentConfig
metadata:
  name: tanzu-ako-for-workload-l7-ingress
spec:
  adminCredentialRef:
    name: NSX_ALB-controller-credentials
    namespace: tkg-system-networking
  certificateAuthorityRef:
    name: NSX_ALB-controller-ca
    namespace: tkg-system-networking
  cloudName: sfo01w01vc01
  clusterSelector:
    matchLabels:
      workload-l7-enabled: "true"
  controller: 172.16.10.10
  controlPlaneNetwork:
    cidr: 172.16.80.0/24
    name: sfo01-w01-vds01-tkgclustervip
  dataNetwork:
    cidr: 172.16.70.0/24
    name: sfo01-w01-vds01-tkgworkloadvip
  serviceEngineGroup: sfo01w01segroup01
  extraConfigs:
    cniPlugin: antrea
    disableStaticRouteSync: false                               # required
    ingress:
      disableIngressClass: false                                # required
      nodeNetworkList:                                          # required
        - networkName: sfo01-w01-vds01-tkgworkload
          cidrs:
            - 172.16.60.0/24
      serviceType: NodePortLocal                                # required
      shardVSSize: MEDIUM                                       # required
```

Use the `kubectl` command to set the context to Tanzu Kubernetes Grid management cluster, and create the ADC:

```
# kubectl config use-context sfo01w01tkgmgmt01-admin@sfo01w01tkgmgmt01
Switched to context "sfo01w01tkgmgmt01-admin@sfo01w01tkgmgmt01".

# kubectl apply -f workload-adc-l7.yaml
akodeploymentconfig.networking.tkg.tanzu.vmware.com/tanzu-ako-for-workload-l7-ingress
createdmentconfig.networking.tkg.tanzu.vmware.com/tanzu-ako-for-workload-l7-ingress cr
eated
```

Use the following command to list all AKODeploymentConfig created under the management cluster:

```
# kubectl get adc
NAME                                 AGE
install-ako-for-all                  22h
install-ako-for-management-cluster   22h
tanzu-ako-for-shared                 82m
tanzu-ako-for-workload-l7-ingress    25s
```

Now that you have successfully created the AKO deployment config, you need to apply the cluster labels while deploying the workload clusters to enable NSX Advanced Load Balancer L7 Ingress with NodePortLocal mode.

Deploy Tanzu Kubernetes Grid Shared Services Cluster

Each Tanzu Kubernetes Grid instance can have only one shared services cluster. Create a shared services cluster if you intend to deploy Harbor.

The procedures for deploying a shared services cluster and workload cluster are almost the same. A key difference is that for the shared service cluster you add the `tanzu-services` label to the shared services cluster, as its cluster role. This label identifies the shared services cluster to the management cluster and workload clusters.

Shared services cluster uses the custom ADC `tanzu-ako-for-shared` created earlier to apply the network settings similar to the management cluster. This is enforced by applying the AVI_LABEL `type:shared-services` while deploying the shared services cluster.

Deployment of the shared services cluster is done by creating a YAML file and invoking the `tanzu cluster create -f <file-name>` command. The YAML file used for shared services deployment is usually a bit smaller than the YAML used for the management cluster deployment because you don’t need to define the AVI fields except `AVI_CONTROL_PLANE_HA_PROVIDER` & `AVI_LABELS` in the YAML.

The following is a sample YAML for deploying a shared services cluster:

```
CLUSTER_NAME: sfo01w01shared01
CLUSTER_PLAN: prod
INFRASTRUCTURE_PROVIDER: vsphere
ENABLE_CEIP_PARTICIPATION: "true"
ENABLE_AUDIT_LOGGING: "true"
CLUSTER_CIDR: 100.96.0.0/11
SERVICE_CIDR: 100.64.0.0/13
VSPHERE_SERVER: sfo01w01vc01.sfo01.rainpole.local
VSPHERE_USERNAME: administrator@vsphere.local
VSPHERE_PASSWORD: <encoded:Vk13YXJlMSE=>
VSPHERE_DATACENTER: /sfo01w01dc01
```

```
VSPHERE_RESOURCE_POOL: /sfo01w01dc01/host/sfo01w01vc01/Resources/tkg-sharedsvc-compone
nts
VSPHERE_DATASTORE: /sfo01w01dc01/datastore/vsanDatastore
VSPHERE_FOLDER: /sfo01w01dc01/vm/tkg-sharedsvc-components
VSPHERE_NETWORK: /sfo01w01dc01/network/sfo01-w01-vds01-tkgmanagement
VSPHERE_CONTROL_PLANE_ENDPOINT: #Leave blank as VIP network is configured in NSX ALB a
nd IPAM is configured with VIP network
VSPHERE_SSH_AUTHORIZED_KEY: ssh-rsa AAAA[...]== email@example.com
VSPHERE_TLS_THUMBPRINT: DC:FA:81:1D:CA:08:21:AB:4E:15:BD:2B:AE:12:2C:6B:CA:65:49:B8
VSPHERE_INSECURE: "false"
OS_NAME: photon
OS_VERSION: "3"
OS_ARCH: amd64
VSPHERE_CONTROL_PLANE_NUM_CPUS: 2
VSPHERE_CONTROL_PLANE_DISK_GIB: 40
VSPHERE_CONTROL_PLANE_MEM_MIB: 8192
VSPHERE_WORKER_NUM_CPUS: 2
VSPHERE_WORKER_DISK_GIB: 40
VSPHERE_WORKER_MEM_MIB: 8192
AVI_CONTROL_PLANE_HA_PROVIDER: "true"
AVI_LABELS: |
    'type': 'shared-services'
TKG_CUSTOM_IMAGE_REPOSITORY: "harbor-sa.lab.vmw/tkg-160"
TKG_CUSTOM_IMAGE_REPOSITORY_SKIP_TLS_VERIFY: false
TKG_CUSTOM_IMAGE_REPOSITORY_CA_CERTIFICATE: LS0t[...]tLS0tLQ==
ENABLE_MHC: true
IDENTITY_MANAGEMENT_TYPE: "none"
```

To create Workload Cluster, execute the following command:

```
tanzu cluster create --file config.yaml
```

Cluster creation takes approximately 15-20 minutes to complete. Verify the health of the cluster and validate that the cluster labels are applied.

1. Connect to the Tanzu Management Cluster context and verify the cluster labels for the workload cluster.

    ```
    ## verify the workload  service cluster creation

    tanzu cluster list
    NAME                      NAMESPACE  STATUS    CONTROLPLANE  WORKERS  KUBERNETES
    ROLES   PLAN  TKR
    sfo01w01tkgshared01    default    running   3/3           3/3      v1.26.5+vmwa
    re.2  <none>  prod  v1.26.5---vmware.2-tkg.1

    ## Connect to tkg management cluster

    kubectl config use-context sfo01w01tkgmgmt01-admin@sfo01w01tkgmgmt01

    ## Add the tanzu-services label to the shared services cluster as its cluster r
    ole. In the following command "sfo01w01tkgshared01" is the name of the shared s
    ervice cluster

    kubectl label cluster.cluster.x-k8s.io/sfo01w0tkgshared01 cluster-role.tkg.tanz
    u.vmware.com/tanzu-services="" --overwrite=true
    cluster.cluster.x-k8s.io/sfo01w0tkgshared01 labeled
    ```

```
## Validate AVI_LABEL applied to shared serice cluster

kubectl get cluster sfo01w0tkgshared01 --show-labels
NAME                    PHASE         AGE    VERSION             LABELS

sfo01w0tkgshared01      Provisioned   105m   v1.26.5+vmware.2    cluster-role.tk
g.tanzu.vmware.com/tanzu-services=,networking.tkg.tanzu.vmware.com/avi=tanzu-ak
o-for-shared,tanzuKubernetesRelease=v1.26.5---vmware.2-tkg.1,tkg.tanzu.vmware.c
om/cluster-name=sfo01w0tkgshared01,type=shared-services
```

2. Connect to admin context of the workload cluster by using the following commands and validate the ako pod status:

```
## Use the following command to get the admin context of workload Cluster.

tanzu cluster kubeconfig get sfo01w0tkgshared01 --admin

Credentials of cluster 'sfo01w0tkgshared01' have been saved
You can now access the cluster by running 'kubectl config use-context sfo01w0tk
gshared01-admin@sfo01w0tkgshared01'


## Use the following command to use the context of workload Cluster

kubectl config use-context sfo01w0tkgshared01-admin@sfo01w0tkgshared01

Switched to context "sfo01w0tkgshared01-admin@sfo01w0tkgshared01".

# Verify that ako pod gets deployed in avi-system namespace

kubectl get pods -n avi-system
NAME    READY   STATUS    RESTARTS   AGE
ako-0   1/1     Running   0          73m

# verify the nodes and pods status by running the command:
kubectl get nodes -o wide

kubectl get pods -A
```

Now the shared services cluster is successfully created.

Deploy Tanzu Kubernetes Grid Workload Cluster

Deployment of the workload cluster is done using a YAML similar to the shared services cluster YAML but customized for the workload cluster placement objects.

The following is a sample YAML for deploying the workload cluster:

```
CLUSTER_NAME: sfo01w01workload01
CLUSTER_PLAN: prod
INFRASTRUCTURE_PROVIDER: vsphere
ENABLE_CEIP_PARTICIPATION: "true"
ENABLE_AUDIT_LOGGING: "true"
CLUSTER_CIDR: 100.96.0.0/11
SERVICE_CIDR: 100.64.0.0/13
VSPHERE_SERVER: sfo01w01vc01.sfo01.rainpole.local
```

```
VSPHERE_USERNAME: administrator@vsphere.local
VSPHERE_PASSWORD: <encoded:Vk13YXJlMSE=>
VSPHERE_DATACENTER: /tkgm-internet-dc1
VSPHERE_RESOURCE_POOL: /sfo01w01dc01/host/sfo01w01vc01/Resources/tkg-workload01-compon
ents
VSPHERE_DATASTORE: /sfo01w01dc01/datastore/vsanDatastore
VSPHERE_FOLDER: /sfo01w01dc01/vm/tkg-workload01-components
VSPHERE_NETWORK: /sfo01w01dc01/network/sfo01-w01-vds01-tkgworkload
VSPHERE_CONTROL_PLANE_ENDPOINT: #Leave blank as VIP network is configured in NSX ALB a
nd IPAM is configured with VIP network
VSPHERE_SSH_AUTHORIZED_KEY: ssh-rsa AAAA[...]== email@example.com
VSPHERE_TLS_THUMBPRINT: DC:FA:81:1D:CA:08:21:AB:4E:15:BD:2B:AE:12:2C:6B:CA:65:49:B8
VSPHERE_INSECURE: "false"
OS_NAME: photon
OS_VERSION: "3"
OS_ARCH: amd64
VSPHERE_CONTROL_PLANE_NUM_CPUS: 2
VSPHERE_CONTROL_PLANE_DISK_GIB: 40
VSPHERE_CONTROL_PLANE_MEM_MIB: 8192
VSPHERE_WORKER_NUM_CPUS: 2
VSPHERE_WORKER_DISK_GIB: 40
VSPHERE_WORKER_MEM_MIB: 8192
AVI_CONTROL_PLANE_HA_PROVIDER: "true"
AVI_LABELS: |
    'workload-l7-enabled': 'true'
TKG_CUSTOM_IMAGE_REPOSITORY: "harbor-sa.lab.vmw/tkg-160"
TKG_CUSTOM_IMAGE_REPOSITORY_SKIP_TLS_VERIFY: false
TKG_CUSTOM_IMAGE_REPOSITORY_CA_CERTIFICATE: LS0t[...]tLS0tLQ==
ENABLE_MHC: true
IDENTITY_MANAGEMENT_TYPE: "none"
```

To create Workload Cluster, execute the following command:

```
tanzu cluster create --file config.yaml
```

Cluster creation roughly takes 15-20 minutes to complete. Verify the health of the cluster and apply the labels.

Once the Workload cluster is created, verify the cluster labels and ako pod status.

1. Connect to the Tanzu Management Cluster context and verify the cluster labels for the workload cluster.

```
## verify the workload  service cluster creation

tanzu cluster list
NAME                   NAMESPACE  STATUS   CONTROLPLANE  WORKERS  KUBERNETES
ROLES   PLAN  TKR

sfo01w01shared01       default    running   3/3           3/3      v1.26.5+vmwa
re.2  <none>  prod   v1.26.5---vmware.2-tkg.1

sfo01w01workload01   default       running   3/3           3/3      v1.26.5+vmwa
re.2  <none>  prod   v1.26.5---vmware.2-tkg.1
```

```
kubectl config use-context sfo01w01vc01-admin@sfo01w01vc01

## Validate that TMC has applied the AVI_LABEL while deploying the cluster

kubectl get cluster sfo01w01workload01 --show-labels
NAME                    PHASE          AGE    VERSION   LABELS

sfo01w01workload01   Provisioned   105m             networking.tkg.tanzu.vmwar
e.com/avi=tanzu-ako-for-workload-l7-ingress,tanzuKubernetesRelease=v1.249---vmw
are.1-tkg.1,tkg.tanzu.vmware.com/cluster-name=sfo01w01workload01,workload-l7-en
abled=true
```

2. Connect to admin context of the workload cluster by using the following commands and validate the ako pod status.

```
## Use the following command to get the admin context of workload Cluster.

tanzu cluster kubeconfig get sfo01w01workload01 --admin

Credentials of cluster 'sfo01w01workload01' have been saved
You can now access the cluster by running 'kubectl config use-context sfo01w01w
orkload01-admin@sfo01w01workload01'


## Use the following command to use the context of workload Cluster

kubectl config use-context sfo01w01workload01-admin@sfo01w01workload01

Switched to context "sfo01w01workload01-admin@sfo01w01workload01".

# Verify that ako pod gets deployed in avi-system namespace

kubectl get pods -n avi-system
NAME    READY   STATUS    RESTARTS   AGE
ako-0   1/1     Running   0          73m

# verify the nodes and pods status by running the command:
kubectl get nodes -o wide

kubectl get pods -A
```

You can see that the workload cluster is successfully deployed and the AKO pod is deployed on the cluster. You can now deploy user-managed packages on this cluster.

Deploy User-Managed Packages

User-managed packages are installed after workload cluster creation. These packages extend the core functionality of Kubernetes clusters created by Tanzu Kubernetes Grid.

Tanzu Kubernetes Grid includes the following user-managed packages. These packages provide in-cluster and shared services to the Kubernetes clusters that are running in your Tanzu Kubernetes Grid environment. For more information, see Installing and Managing Packages with the Tanzu CLI .

With TKG v2.3, the Tanzu Standard package repository is versioned and distributed separately from TKG, and its versioning is based on a date stamp. For TKG v2.3, the latest compatible Tanzu Standard repository version is v2023.7.13

Function	Package	Location
Certificate Management	Cert Manager	Workload and shared services cluster
Container networking	Multus	Workload cluster
Container registry	Harbor	Shared services cluster
Ingress control	Contour	Workload and shared services cluster
Log forwarding	Fluent Bit	Workload cluster
Monitoring	Grafana Prometheus	Workload cluster

User-managed packages can be installed via CLI by invoking the `tanzu package install` command. Before installing the user-managed packages, ensure that you have switched to the context of the cluster where you want to install the packages.

Also, ensure that the tanzu-standard repository is configured on the cluster where you want to install the packages.

You can run the command `tanzu package repository list -A` to verify this. Also, ensure that the repository status is `Reconcile succeeded`.

```
# Add Private Registry to the workload Cluster

tanzu package repository add tanzu-standard --url harbor.sfo01.rainpole.vmw/tkgm-image
s/packages/standard/repo -n tkg-system

# tanzu package repository list -A

NAMESPACE   NAME              SOURCE
STATUS
tkg-system  tanzu-standard  (imgpkg) harbor.sfo01.rainpole.vmw/tkgm-images/packages/st
andard/repo  Reconcile succeeded

#tanzu package available list -A

NAMESPACE   NAME                                             DISPLAY-NAME
tkg-system  cert-manager.tanzu.vmware.com                    cert-manager
tkg-system  contour.tanzu.vmware.com                         contour
tkg-system  external-csi-snapshot-webhook.tanzu.vmware.com  external-csi-snapshot-webh
ook
tkg-system  external-dns.tanzu.vmware.com                    external-dns
tkg-system  fluent-bit.tanzu.vmware.com                      fluent-bit
tkg-system  fluxcd-helm-controller.tanzu.vmware.com          Flux Helm Controller
tkg-system  fluxcd-kustomize-controller.tanzu.vmware.com     Flux Kustomize Controller
tkg-system  fluxcd-source-controller.tanzu.vmware.com        Flux Source Controller
tkg-system  grafana.tanzu.vmware.com                         grafana
tkg-system  harbor.tanzu.vmware.com                          harbor
tkg-system  multus-cni.tanzu.vmware.com                      multus-cni
tkg-system  prometheus.tanzu.vmware.com                      prometheus
tkg-system  whereabouts.tanzu.vmware.com                     whereabouts
```

Install Cert Manager

The first package that you should install on your cluster is the **cert-manager** package which adds certificates and certificate issuers as resource types in Kubernetes clusters and simplifies the process of obtaining, renewing and using those certificates.

1. Capture the available Cert Manager package versions.

```
# tanzu package available list cert-manager.tanzu.vmware.com -A

NAMESPACE   NAME                           VERSION                RELEASED-AT
tkg-system  cert-manager.tanzu.vmware.com  1.1.0+vmware.1-tkg.2   2020-11-24 1
8:00:00 +0000 UTC
tkg-system  cert-manager.tanzu.vmware.com  1.1.0+vmware.2-tkg.1   2020-11-24 1
8:00:00 +0000 UTC
tkg-system  cert-manager.tanzu.vmware.com  1.11.1+vmware.1-tkg.1  2023-01-11 1
2:00:00 +0000 UTC
tkg-system  cert-manager.tanzu.vmware.com  1.5.3+vmware.2-tkg.1   2021-08-23 1
7:22:51 +0000 UTC
tkg-system  cert-manager.tanzu.vmware.com  1.5.3+vmware.4-tkg.1   2021-08-23 1
7:22:51 +0000 UTC
tkg-system  cert-manager.tanzu.vmware.com  1.5.3+vmware.7-tkg.1   2021-08-23 1
7:22:51 +0000 UTC
tkg-system  cert-manager.tanzu.vmware.com  1.5.3+vmware.7-tkg.3   2021-08-23 1
7:22:51 +0000 UTC
tkg-system  cert-manager.tanzu.vmware.com  1.7.2+vmware.1-tkg.1   2021-10-29 1
2:00:00 +0000 UTC
tkg-system  cert-manager.tanzu.vmware.com  1.7.2+vmware.3-tkg.1   2021-10-29 1
2:00:00 +0000 UTC
tkg-system  cert-manager.tanzu.vmware.com  1.7.2+vmware.3-tkg.3   2021-10-29 1
2:00:00 +0000 UTC
```

2. Install the `cert-manager` package.

   Capture the latest version from the previous command, if there are multiple versions available, check the **RELEASED-AT** to collect the version of the latest one. This document uses the version 1.7.2+vmware.3-tkg.3 for installation.

   The following command installs the `cert-manager` package:

```
tanzu package install cert-manager --package cert-manager.tanzu.vmware.com --na
mespace cert-manager-package --version <AVAILABLE-PACKAGE-VERSION>

# tanzu package install cert-manager --package cert-manager.tanzu.vmware.com --
namespace cert-manager-package --version 1.7.2+vmware.3-tkg.3
8:05:31AM: Creating service account 'cert-manager-cert-manager-package-sa'
8:05:31AM: Creating cluster admin role 'cert-manager-cert-manager-package-clust
er-role'
8:05:31AM: Creating cluster role binding 'cert-manager-cert-manager-package-clu
ster-rolebinding'
8:05:31AM: Creating overlay secrets
8:05:31AM: Creating package install resource
8:05:31AM: Waiting for PackageInstall reconciliation for 'cert-manager'
8:05:31AM: Fetch started (1s ago)
8:05:32AM: Fetching
        | apiVersion: vendir.k14s.io/v1alpha1
        | directories:
        | - contents:
        |   - imgpkgBundle:
```

```
        |       image: harbor.sfo01.rainpole.vmw/tkgm-images/packages/standard/
repo@sha256:cac4e2d8a3e98be121a86e687b57d8058dba5f0ba240f3db5008bc85e5ac04cf
        |     path: .
        |   path: "0"
        | kind: LockConfig
        |
8:05:32AM: Fetch succeeded
8:05:33AM: Template succeeded (1s ago)
```

3. Confirm that the `cert-manager` package has been installed successfully and the status is `Reconcile succeeded`.

```
]# tanzu package installed get cert-manager -n cert-manager-package
NAME:                    cert-manager
PACKAGE-NAME:            cert-manager.tanzu.vmware.com
PACKAGE-VERSION:         1.7.2+vmware.3-tkg.3
STATUS:                  Reconcile succeeded
CONDITIONS:              [{ReconcileSucceeded True  }]
USEFUL-ERROR-MESSAGE:
```

Install Contour

Contour is an open-source Kubernetes ingress controller providing the control plane for the Envoy edge and service proxy. Tanzu Kubernetes Grid includes signed binaries for Contour and Envoy, which you can deploy into workload clusters to provide ingress control services in those clusters.

After you have set up the cluster, you must first create the configuration file that is used when you install the Contour package and then install the package.

Package installation can be customized by entering the user-configurable values in YAML format. Following is an example YAML for customizing Contour installation.

```
---
infrastructure_provider: vsphere
namespace: tanzu-system-ingress
contour:
 configFileContents: {}
 useProxyProtocol: false
 replicas: 2
 pspNames: "vmware-system-restricted"
 logLevel: info
envoy:
 service:
   type: LoadBalancer
   annotations: {}
   nodePorts:
     http: null
     https: null
   externalTrafficPolicy: Cluster
   disableWait: false
 hostPorts:
   enable: true
   http: 80
   https: 443
 hostNetwork: false
 terminationGracePeriodSeconds: 300
 logLevel: info
 pspNames: null
certificates:
 duration: 8760h
 renewBefore: 360h
```

For a full list of user-configurable values, see Configure the Contour Extension.

1. Capture the available Contour package versions.

   ```
   # tanzu package available list contour.tanzu.vmware.com -A

   NAME                       VERSION                 RELEASED-AT
   contour.tanzu.vmware.com   1.24.4+vmware.1-tkg.1   2023-04-28 00:00:00 +0000 UTC
   ```

   Capture the latest version from the previous command. If there are multiple versions available, check the **RELEASED-AT** to collect the version of the latest one. This document make use of version 1.24.4+vmware.1-tkg.1 for installation.

2. Install the Contour package.

   ```
   tanzu package install contour --package contour.tanzu.vmware.com --version <AVA
   ILABLE-PACKAGE-VERSION> --values-file <path to contour-data-values.yaml> --name
   space tanzu-contour-ingress

   # kubectl create namespace tanzu-system-ingress
   # kubectl create namespace tanzu-contour-ingress
   # tanzu package install contour --package contour.tanzu.vmware.com --version 1.
   24.4+vmware.1-tkg.1 --values-file contour-data-values.yaml --namespace tanzu-co
   ntour-ingress


   8:12:04AM: Creating service account 'contour-tanzu-contour-ingress-sa'
   8:12:04AM: Creating cluster admin role 'contour-tanzu-contour-ingress-cluster-r
   ole'
   8:12:04AM: Creating cluster role binding 'contour-tanzu-contour-ingress-cluster
   -rolebinding'
   8:12:04AM: Creating secret 'contour-tanzu-contour-ingress-values'
   8:12:04AM: Creating overlay secrets
   8:12:04AM: Creating package install resource
   8:12:04AM: Waiting for PackageInstall reconciliation for 'contour'
   8:12:04AM: Fetch started (1s ago)
   8:12:05AM: Fetching
           | apiVersion: vendir.k14s.io/v1alpha1
           | directories:
           | - contents:
           |   - imgpkgBundle:
           |       image: harbor.sfo01.rainpole.vmw/tkgm-images/packages/standard/
   repo@sha256:20db584c146086a789ab29e3efd24a8b406054a945607322abd134f38c603013
           |     path: .
           |   path: "0"
           | kind: LockConfig
           |
   8:12:05AM: Fetch succeeded
   8:12:06AM: Template succeeded
   ```

3. Confirm that the Contour package has been installed and the status is `Reconcile succeeded`.

```
# tanzu package installed get contour --namespace tanzu-contour-ingress

NAME:                         contour
PACKAGE-NAME:                 contour.tanzu.vmware.com
PACKAGE-VERSION:              1.22.3+vmware.1-tkg.1
STATUS:                       Reconcile succeeded
CONDITIONS:                   [{ReconcileSucceeded True  }]
USEFUL-ERROR-MESSAGE:
```

Install Harbor

Harbor is an open-source container registry. Harbor Registry may be used as a private registry for container images that you want to deploy to Tanzu Kubernetes clusters.

Tanzu Kubernetes Grid includes signed binaries for Harbor, which you can deploy into:

- A workload cluster to provide container registry services for that clusters
- A shared services cluster to provide container registry services for other Tanzu Kubernetes (workload) clusters.

When deployed as a shared service, Harbor is available to all of the workload clusters in a given Tanzu Kubernetes Grid instance.

Perform the following procedure to deploy Harbor into a workload cluster or a shared services cluster:

1. Confirm that the Harbor package is available in the cluster and retrieve the version of the available package.

   ```
   # tanzu package available list harbor.tanzu.vmware.com -A

   - Retrieving package versions for harbor.tanzu.vmware.com...

    NNAME                        VERSION                RELEASED-AT
    harbor.tanzu.vmware.com  2.8.2+vmware.2-tkg.1  2023-06-08 10:18:00 +0000 UTC
   ```

2. Create a configuration file named `harbor-data-values.yaml` by executing the following commands:

   ```
   image_url=$(kubectl -n tkg-system get packages harbor.tanzu.vmware.com.2.8.2+vm
   ware.2-tkg.1 -o jsonpath='{.spec.template.spec.fetch[0].imgpkgBundle.image}')

   imgpkg pull -b $image_url -o /tmp/harbor-package --registry-ca-cert-path /etc/d
   ocker/certs.d/harbor.tanzu.lab/ca.crt

   cp /tmp/harbor-package/config/values.yaml harbor-data-values.yaml
   ```

3. Set the mandatory passwords and secrets in the `harbor-data-values.yaml` file.

   ```
   bash /tmp/harbor-package/config/scripts/generate-passwords.sh harbor-data-value
   s.yaml
   ```

4. Edit the `harbor-data-values.yaml` file and configure the values for the following mandatory parameters.

   - namespace

   - port
   - harborAdminPassword
   - secretKey

   You can also change the values for other parameters to meet the requirements for your deployment. For the full list of the user-configurable values, see Deploy Harbor into a Cluster.

5. Remove the comments in the `harbor-data-values.yaml` file.

```
yq -i eval '... comments=""' harbor-data-values.yaml
```

6. Install the Harbor package by executing the following command:

```
# kubectl create namespace tanzu-system-registry
# kubectl create namespace tanzu-harbor-registry
# tanzu package install harbor --package-name harbor.tanzu.vmware.com --version 
2.8.2+vmware.2-tkg.1 --values-file harbor-data-values.yaml --namespace tanzu-ha
rbor-registry

 8:01:14AM: Creating service account 'harbor-tanzu-system-registry-sa'
 8:01:14AM: Creating cluster admin role 'harbor-tanzu-system-registry-cluster-r
ole'
 8:01:15AM: Creating cluster role binding 'harbor-tanzu-system-registry-cluster
-rolebinding'
 8:01:15AM: Creating secret 'harbor-tanzu-system-registry-values'
 8:01:15AM: Creating overlay secrets
 8:01:15AM: Creating package install resource
 8:01:15AM: Waiting for PackageInstall reconciliation for 'harbor'
 8:01:15AM: Fetch started (6s ago)
        | 8:04:50AM:  L ongoing: waiting on pod/harbor-registry-78c99df744-v8ps
j (v1) namespace: tanzu-system-registry
        | 8:04:50AM:     ^ Condition Ready is not True (False)
        | 8:04:52AM: ok: reconcile deployment/harbor-registry (apps/v1) namespa
ce: tanzu-system-registry
        | 8:04:52AM: ---- applying complete [50/50 done] ----
        | 8:04:52AM: ---- waiting complete [50/50 done] ----
        | Succeeded
8:04:52AM: Deploy succeeded
```

7. Confirm that the Harbor package has been installed and the status is `Reconcile succeeded`.

```
# tanzu package installed get harbor --namespace tanzu-system-registry


NAME:                    harbor
PACKAGE-NAME:            harbor.tanzu.vmware.com
PACKAGE-VERSION:         2.8.2+vmware.2-tkg.1
STATUS:                  Reconcile succeeded
CONDITIONS:              [{ReconcileSucceeded True  }]
USEFUL-ERROR-MESSAGE:
```

Install Prometheus

Prometheus is a system and service monitoring system. It collects metrics from configured targets at given intervals, evaluates rule expressions, displays the results, and can trigger alerts if some condition is observed to be true. Alertmanager handles alerts generated by Prometheus and routes them to their receiving endpoints.

Do the following to deploy Prometheus into a workload cluster:

1. Capture the available Prometheus version.

```
# tanzu package available list prometheus.tanzu.vmware.com -A

NAME                          VERSION                  RELEASED-AT
prometheus.tanzu.vmware.com  2.27.0+vmware.1-tkg.1  2021-05-12 18:00:00 +0000 U
TC
prometheus.tanzu.vmware.com  2.27.0+vmware.2-tkg.1  2021-05-12 18:00:00 +0000 U
TC
prometheus.tanzu.vmware.com  2.36.2+vmware.1-tkg.1  2022-06-23 18:00:00 +0000 U
TC
prometheus.tanzu.vmware.com  2.37.0+vmware.1-tkg.1  2022-10-25 18:00:00 +0000 U
TC
prometheus.tanzu.vmware.com  2.37.0+vmware.2-tkg.1  2022-10-25 18:00:00 +0000 U
TC
prometheus.tanzu.vmware.com  2.37.0+vmware.3-tkg.1  2022-10-25 18:00:00 +0000 U
TC
prometheus.tanzu.vmware.com  2.43.0+vmware.2-tkg.1  2023-03-21 18:00:00 +0000 U
TC
```

   Capture the latest version from the previous command. If there are multiple versions available, check the **RELEASED-AT** to collect the version of the latest one. This document uses the version 2.43.0+vmware.2-tkg.1 for installation.

2. Retrieve the template of the Prometheus package's default configuration:

```
image_url=$(kubectl -n tkg-system get packages prometheus.tanzu.vmware.com.2.4
3.0+vmware.2-tkg.1 -o jsonpath='{.spec.template.spec.fetch[0].imgpkgBundle.imag
e}')

imgpkg pull -b $image_url -o /tmp/prometheus-package-2.43.0+vmware.2-tkg.1--reg
istry-ca-cert-path /etc/docker/certs.d/harbor.tanzu.lab/ca.crt

cp /tmp/prometheus-package-2.43.0+vmware.2-tkg.1/config/values.yaml prometheus-
data-values.yaml
```

   This creates a configuration file named `prometheus-data-values.yaml` that you can modify.

3. To customize the Prometheus installation, modify the following values:

Key	Default Value	Modified value
Ingress.tlsCertificate.tls.crt	Null	Note: This is optional.
ingress.tlsCertificate.tls.key	Null	Cert Key provided in Input file. Note: This is optional.
ingress.enabled	false	true
ingress.virtual_host_fqdn	prometheus.system.tanzu	prometheus.your-domain

To see a full list of user configurable configuration parameters, see Prometheus Package Configuration Parameters.

4. After you make the necessary changes to your `prometheus-data-values.yaml` file, remove all comments in the file:

```
yq -i eval '... comments=""' prometheus-data-values.yaml
```

5. Install the Prometheus package.

```
# kubectl create namespace tanzu-system-monitoring
# kubectl create namespace tanzu-prometheus-monitoring
# tanzu package install prometheus --package-name prometheus.tanzu.vmware.com -
-version 2.43.0+vmware.2-tkg.1 --values-file prometheus-data-values.yaml --name
space tanzu-prometheus-monitoring

8:20:09AM: Creating service account 'prometheus-tanzu-system-monitoring-sa'
8:20:09AM: Creating cluster admin role 'prometheus-tanzu-system-monitoring-clus
ter-role'
8:20:09AM: Creating cluster role binding 'prometheus-tanzu-system-monitoring-cl
uster-rolebinding'
8:20:09AM: Creating secret 'prometheus-tanzu-system-monitoring-values'
8:20:09AM: Creating overlay secrets
8:20:09AM: Creating package install resource
8:20:09AM: Waiting for PackageInstall reconciliation for 'prometheus'

        | 8:22:02AM:  L ok: waiting on replicaset/alertmanager-56f6ccfc64 (app
s/v1) namespace: tanzu-system-monitoring
        | 8:22:02AM:  L ok: waiting on pod/alertmanager-56f6ccfc64-h5tl9 (v1) n
amespace: tanzu-system-monitoring
        | 8:22:03AM: ok: reconcile deployment/alertmanager (apps/v1) namespace:
tanzu-system-monitoring
        | 8:22:03AM: ---- waiting on 1 changes [35/36 done] ----
        | 8:22:23AM: ok: reconcile deployment/prometheus-server (apps/v1) names
pace: tanzu-system-monitoring
        | 8:22:23AM: ---- applying complete [36/36 done] ----
        | 8:22:23AM: ---- waiting complete [36/36 done] ----
        | Succeeded
8:22:23AM: Deploy succeeded (1s ago)
```

6. Confirm that the Prometheus package has been installed successfully and the status is `Reconcile succeeded`.

```
# tanzu package installed get prometheus -n tanzu-prometheus-monitoring

NAME:                    prometheus
PACKAGE-NAME:            prometheus.tanzu.vmware.com
PACKAGE-VERSION:         2.43.0+vmware.2-tkg.1
STATUS:                  Reconcile succeeded
CONDITIONS:              [{ReconcileSucceeded True  }]
USEFUL-ERROR-MESSAGE:
```

Install Grafana

Grafana allows you to query, visualize, alert on, and explore metrics irrespective of their storage location. Grafana provides tools to form graphs and visualizations from application data.

Grafana is configured with Prometheus as a default data source. If you have customized the Prometheus deployment namespace and it is not deployed in the default namespace, `tanzu-system-monitoring`, you need to change the Grafana data source configuration in the following code.

1. Retrieve the version of the available package.

```
# tanzu package available list grafana.tanzu.vmware.com -A

NAME                        VERSION                  RELEASED-AT
grafana.tanzu.vmware.com  7.5.16+vmware.1-tkg.1  2022-05-19 18:00:00 +0000 UTC
grafana.tanzu.vmware.com  7.5.17+vmware.1-tkg.2  2022-05-19 18:00:00 +0000 UTC
grafana.tanzu.vmware.com  7.5.7+vmware.1-tkg.1   2021-05-19 18:00:00 +0000 UTC
grafana.tanzu.vmware.com  7.5.7+vmware.2-tkg.1   2021-05-19 18:00:00 +0000 UTC
grafana.tanzu.vmware.com  9.5.1+vmware.2-tkg.1   2022-05-19 18:00:00 +0000 UTC
```

   Capture the latest version from the previous command. If there are multiple versions available, check the **RELEASED-AT** to collect the version of the latest one. This document uses the version 9.5.1+vmware.2-tkg.1 for installation.

2. Retrieve the template of the Grafana package's default configuration.

```
image_url=$(kubectl -n tkg-system get packages grafana.tanzu.vmware.com.9.5.1+v
mware.2-tkg.1  -o jsonpath='{.spec.template.spec.fetch[0].imgpkgBundle.image}')

imgpkg pull -b $image_url -o /tmp/grafana-package-9.5.1+vmware.2-tkg.1 --regist
ry-ca-cert-path /etc/docker/certs.d/harbor.tanzu.lab/ca.crt

cp /tmp/grafana-package-9.5.1+vmware.2-tkg.1/config/values.yaml grafana-data-va
lues.yaml
```

   This creates a configuration file named `grafana-data-values.yaml` that you can modify. For a full list of user-configurable values, see Grafana Package Configuration Parameters.

3. Edit grafana-data-values.yaml and replace the following with your custom values.

Key	**Default Value**	**Modified value**
secret.admin_password	Null	Your password in Base64 encoded format.
grafana.service.type	LoadBalancer	NodePort
ingress.virtual_host_fqdn	grafana.system.tanzu	User-Provided FQDN from Input file.
ingress.tlsCertificate.tls.crt	Null	Full chain cert provided in Input file.
ingress.tlsCertificate.tls.key	Null	Full chain cert provided in Input file.

4. (Optional) Modify the Grafana data source configuration.

   Grafana is configured with Prometheus as a default data source. If you have customized the Prometheus deployment namespace and it is not deployed in the default namespace, `tanzu-system-monitoring`, you need to change the Grafana data source configuration in `grafana-data-values.yaml`.

```
datasources:
        - name: Prometheus
          type: prometheus
          url: prometheus-server.<change-to-prometheus-namespace>.svc.cluster.l
ocal
```

5. Remove all comments from `grafana-data-values.yaml` file.

```
yq -i eval '... comments=""' grafana-data-values.yaml
```

6. Install Grafana.

```
# kubectl create namespace tanzu-system-dashboards
# kubectl create namespace tanzu-grafana-dashboards
#  tanzu package install grafana --package-name grafana.tanzu.vmware.com --vers
ion 9.5.1+vmware.2-tkg.1 --values-file grafana-data-values.yaml --namespace tan
zu-grafana-dashboards

8:12:41AM: Creating service account 'grafana-tanzu-system-dashboards-sa'
8:12:42AM: Creating cluster admin role 'grafana-tanzu-system-dashboards-cluster
-role'
8:12:42AM: Creating cluster role binding 'grafana-tanzu-system-dashboards-clust
er-rolebinding'
8:12:42AM: Creating secret 'grafana-tanzu-system-dashboards-values'
8:12:42AM: Creating overlay secrets
8:12:42AM: Creating package install resource
8:12:42AM: Waiting for PackageInstall reconciliation for 'grafana'
        | 8:14:19AM: ongoing: reconcile deployment/grafana (apps/v1) namespace:
tanzu-system-dashboards
        | 8:14:19AM:  ^ Waiting for 1 unavailable replicas
        | 8:14:19AM:  L ok: waiting on replicaset/grafana-58656c5f9b (apps/v1)
namespace: tanzu-system-dashboards
        | 8:14:19AM:  L ongoing: waiting on pod/grafana-58656c5f9b-mjphv (v1) n
amespace: tanzu-system-dashboards
        | 8:14:19AM:     ^ Condition Ready is not True (False)
        | 8:14:31AM: ok: reconcile deployment/grafana (apps/v1) namespace: tanz
u-system-dashboards
        | 8:14:31AM: ---- applying complete [18/18 done] ----
        | 8:14:31AM: ---- waiting complete [18/18 done] ----
        | Succeeded
8:14:31AM: Deploy succeeded
```

7. Confirm that the Grafana package has been installed and the status is `Reconcile succeeded`.

```
# tanzu package installed get grafana -n tanzu-grafana-dashboards

NAME:                    grafana
PACKAGE-NAME:            grafana.tanzu.vmware.com
PACKAGE-VERSION:         9.5.1+vmware.2-tkg.1
STATUS:                  Reconcile succeeded
CONDITIONS:              [{ReconcileSucceeded True  }]
USEFUL-ERROR-MESSAGE:
```

Install Fluent Bit

Fluent Bit is a lightweight log processor and forwarder that allows you to collect data and logs from different sources, unify them, and send them to multiple destinations.

The current release of Fluent Bit allows you to gather logs from management clusters or Tanzu Kubernetes clusters running in vSphere, Amazon EC2, and Azure. You can then forward them to a log storage provider such as Elastic Search, Kafka, Splunk, or an HTTP endpoint.

The example shown in this document uses HTTP endpoint `vRealize Log Insight` for forwarding logs from Tanzu Kubernetes clusters.

1. Retrieve the version of the available package.

```
# tanzu package available list fluent-bit.tanzu.vmware.com -A

NAME                         VERSION                 RELEASED-AT
fluent-bit.tanzu.vmware.com  1.7.5+vmware.1-tkg.1    2021-05-13 18:00:00 +0000 U
TC
fluent-bit.tanzu.vmware.com  1.7.5+vmware.2-tkg.1    2021-05-13 18:00:00 +0000 U
TC
fluent-bit.tanzu.vmware.com  1.8.15+vmware.1-tkg.1   2022-05-24 18:00:00 +0000 U
TC
fluent-bit.tanzu.vmware.com  1.9.5+vmware.1-tkg.2    2022-06-23 18:00:00 +0000 U
TC
fluent-bit.tanzu.vmware.com  2.1.2+vmware.1-tkg.1    2022-06-23 18:00:00 +0000 U
TC
```

   Capture the latest version from the previous command. If there are multiple versions available, check the **RELEASED-AT** to collect the version of the latest one. This document uses the version 2.1.2+vmware.1-tkg.1 for installation.

2. Retrieve the template of the Fluent Bit package's default configuration.

```
image_url=$(kubectl -n tkg-system get packages fluent-bit.tanzu.vmware.com.2.1.
2+vmware.1-tkg.1  -o jsonpath='{.spec.template.spec.fetch[0].imgpkgBundle.imag
e}')

imgpkg pull -b $image_url -o /tmp/fluent-bit-2.1.2+vmware.1-tkg.1 --registry-ca
-cert-path /etc/docker/certs.d/harbor.tanzu.lab/ca.crt

cp /tmp/fluent-bit-2.1.2+vmware.1-tkg.1/config/values.yaml fluentbit-data-value
s.yaml
```

3. Modify the resulting `fluentbit-data-values.yaml` file and configure the endpoint as per your requirement. A sample endpoint configuration for sending logs to vRealize Log Insight Cloud over HTTP is shown in the following example.

```
[OUTPUT]
        Name                  syslog
        Match                 *
        Host                  vrli.lab.vmw
        Port                  514
        Mode                  udp
        Syslog_Format         rfc5424
        Syslog_Hostname_key   tkg_cluster
        Syslog_Appname_key    pod_name
        Syslog_Procid_key     container_name
        Syslog_Message_key    message
        Syslog_SD_key        k8s
        Syslog_SD_key        labels
        Syslog_SD_key        annotations
        Syslog_SD_key        tkg
```

4. Deploy Fluent Bit.

```
# kubectl create namespace tanzu-system-logging
# kubectl create namespace tanzu-fluent-bit-logging

 tanzu package install fluent-bit --package-name fluent-bit.tanzu.vmware.com --
version 2.1.2+vmware.1-tkg.1 --namespace tanzu-fluent-bit-logging --values-file
fluent-bit-data-values.yaml

ℹ   Installing package 'fluent-bit.tanzu.vmware.com'
ℹ   Getting package metadata for 'fluent-bit.tanzu.vmware.com'
ℹ   Creating service account 'fluent-bit-tanzu-fluent-bit-logging-sa'
ℹ   Creating cluster admin role 'fluent-bit-tanzu-fluent-bit-logging-cluster-rol
e'
ℹ   Creating cluster role binding 'fluent-bit-tanzu-fluent-bit-logging-cluster-r
olebinding'
ℹ   Creating package resource
ℹ   Waiting for 'PackageInstall' reconciliation for 'fluent-bit'
ℹ   'PackageInstall' resource install status: Reconciling
ℹ   'PackageInstall' resource install status: ReconcileSucceeded
ℹ
Added installed package 'fluent-bit'
```

5. Confirm that the Fluent Bit package has been installed and the status is `Reconcile succeeded`.

```
# tanzu package installed get fluent-bit --namespace tanzu-fluent-bit-logging

NAME:                    fluent-bit
PACKAGE-NAME:            fluent-bit.tanzu.vmware.com
PACKAGE-VERSION:         2.1.2+vmware.1-tkg.1
STATUS:                  Reconcile succeeded
CONDITIONS:              [{ReconcileSucceeded True  }]
USEFUL-ERROR-MESSAGE:
```

VMware Tanzu Kubernetes Grid on vSphere with NSX-T Networking in Air-Gapped Environment Reference Design

VMware Tanzu Kubernetes Grid (informally known as TKG) (multi-cloud) provides organizations with a consistent, upstream-compatible, regional Kubernetes substrate that is ready for end-user workloads and ecosystem integrations.

An air-gapped environment is a network security measure employed to ensure a computer or computer network is secure by physically isolating it from unsecured networks, such as the public Internet or an unsecured local area network. This means a computer or network is disconnected from all other systems.

This document lays out a reference design for deploying Tanzu Kubernetes Grid (informally known as TKG) on NSX-T Data Center Networking in an air-gapped environment and offers a high-level overview of the different components required for setting up a Tanzu Kubernetes Grid environment.

Supported Component Matrix

The following table provides the component versions and interoperability matrix supported with the reference design:

Software Components	Version
Tanzu Kubernetes Grid	2.3.0
VMware vSphere ESXi	8.0 U1 or later
VMware vCenter Server	8.0 U1 or later
VMware NSX	4.1.0.2

For the latest interoperability information about other VMware products and versions, see the VMware Interoperability Matrix.

Components

The following components are used in the reference architecture:

- **Tanzu Kubernetes Grid (TKG)** - Enables creation and lifecycle management of Kubernetes clusters.
- **NSX Advanced Load Balancer Enterprise Edition** - Provides layer 4 service type load balancer and layer 7 ingress support.

- **Tanzu User-Managed Packages:** User-managed packages are distributed through package repositories. The `tanzu-standard` package repository includes the following user-managed packages:
    - **Cert Manager** - Provides automated certificate management. It runs by default in management clusters.
    - **Contour** - Provides layer 7 ingress control to deployed HTTP(S) applications. Tanzu Kubernetes Grid includes signed binaries for Contour. Deploying Contour is a prerequisite for deploying Prometheus, Grafana, and Harbor extensions.
    - **Fluent Bit** - Collects data and logs from different sources, unifies them, and sends them to multiple destinations. Tanzu Kubernetes Grid includes signed binaries for Fluent Bit.
    - **Prometheus** - Provides out-of-the-box health monitoring of Kubernetes clusters. The Tanzu Kubernetes Grid implementation of Prometheus includes an Alert Manager. You can configure Alert Manager to notify you when certain events occur.
    - **Grafana** - Provides monitoring dashboards for displaying key health metrics of Kubernetes clusters. Tanzu Kubernetes Grid includes an implementation of Grafana.
    - **Harbor Image Registry** - Provides a centralized location to push, pull, store, and scan container images used in Kubernetes workloads. It supports storing artifacts and includes enterprise-grade features such as RBAC, retention policies, automated garbage clean up, and Docker hub proxying.
    - **Multus CNI** - Enables attaching multiple network interfaces to pods. Multus CNI is a container network interface (CNI) plugin for Kubernetes that lets you attach multiple network interfaces to a single pod and associate each interface with a different address range.
- **Bastion Host** - Bastion host is the physical/virtual machine where you download the required installation images/binaries (for Tanzu Kubernetes Grid installation) from the Internet. This machine needs to be outside the air-gapped environment. The downloaded items then need to be shipped to the bootstrap machine which is inside the air-gapped environment.
- **Jumpbox/Bootstrap Machine** - The bootstrap machine is where you run the Tanzu CLI and other utilities such as Kubectl, Kind. Here is the initial bootstrapping of a management cluster occurs before it is pushed to the platform where it runs.

The binaries for Tanzu Kubernetes Grid installation are made available in ISO or tarball format on this machine. This machine should have access to the infrastructure components such as the vCenter server and the components that are deployed during the installation of Tanzu Kubernetes Grid. This machine should have a browser installed to access the UI of the components described above.

With TKG 2.1.0, instead of custom script, a new Tanzu CLI plugin isolated-cluster has been provided which will pre-populate air gapped internal registry.

This new Tanzu CLI plug-in contains two separate commands :

- **Download-bundle** - Downloads the images and bundles as tar files. Along with the downloads, a YAML file gets created that contains the mapping from the image name to the tar location.
- **Upload-bundle** - Upload images to private repository. Bootstrap VM should have access to this private repository for Tanzu installation.

- **Local Image Registry** - An image registry provides a location for pushing, pulling, storing, and scanning container images used in the Tanzu Kubernetes Grid environment. The image registry is also used for day-2 operations of the Tanzu Kubernetes clusters. Typical day-2 operations include tasks such as storing application images, upgrading Tanzu Kubernetes clusters, etc.

In an air-gapped environment, there are a couple of possible solutions for using an image registry:

- **Existing Image Registry** - An image registry pre-existing in the environment with a project created for storing Tanzu Kubernetes Grid binaries and the bootstrap machine has access to this registry. The operator unzip the TAR file present in the bootstrap machine and pushes the Tanzu Kubernetes Grid binaries to the Tanzu Kubernetes Grid project using the script present in the TAR file. This registry can be a Harbor registry or any other container registry solution.
- **New Image Registry** - If there is no pre-existing image registry in the environment, a new registry instance can be deployed. The easiest way to create a new image registry instance is VM-based deployment using OVA, and then push the TKG binaries to the appropriate project. VM-based deployments are only supported by VMware Global Support Services to host the system images for air-gapped or Internet-restricted deployments. Do not use this method for hosting application images.

Tanzu Kubernetes Grid Components

VMware Tanzu Kubernetes Grid provides organizations with a consistent, upstream-compatible, regional Kubernetes substrate that is ready for end-user workloads and ecosystem integrations. You can deploy Tanzu Kubernetes Grid across software-defined datacenters (SDDC) and public cloud environments, including vSphere, Microsoft Azure, and Amazon EC2.

Tanzu Kubernetes Grid comprises the following components:

- **Management Cluster** - A management cluster is the first element that you deploy when you create a Tanzu Kubernetes Grid instance. The management cluster is a Kubernetes cluster that performs the role of the primary management and operational center for the Tanzu Kubernetes Grid instance. The management cluster is purpose-built for operating the platform and managing the lifecycle of Tanzu Kubernetes clusters.
- **ClusterClass API** - Tanzu Kubernetes Grid 2 functions through the creation of a management Kubernetes cluster which holds ClusterClass API. The ClusterClass API then interacts with the infrastructure provider to service workload Kubernetes cluster lifecycle requests. The earlier primitives of Tanzu Kubernetes clusters will still exist for Tanzu Kubernetes Grid 1.X . The Cluster API also contains ClusterClass which reduces the need for redundant templating, and enables powerful customization of clusters. The process for creating a cluster using ClusterClass is same as before with a set of different parameters.
- **Tanzu Kubernetes Cluster** - Tanzu Kubernetes clusters are the Kubernetes clusters in which your application workloads run. These clusters are also referred to as workload clusters. Tanzu Kubernetes clusters can run different versions of Kubernetes, depending on the needs of the applications they run.
- **Shared Services Cluster** - Each Tanzu Kubernetes Grid instance can only have one shared services cluster. You deploy this cluster only if you intend to deploy shared services such as Contour and Harbor.

- **Tanzu Kubernetes Cluster Plans** - A cluster plan is a blueprint that describes the configuration with which to deploy a Tanzu Kubernetes cluster. It provides a set of configurable values that describe settings like the number of control plane machines, worker machines, VM types, and so on.

  This release of Tanzu Kubernetes Grid provides two default templates; dev, and prod. You can create and use custom plans to meet your requirements.

- **Tanzu Kubernetes Grid Instance** - A Tanzu Kubernetes Grid instance is the full deployment of Tanzu Kubernetes Grid, including the management cluster, the workload clusters, and the shared services cluster that you configure.

- **Tanzu CLI** - A command-line utility that provides the necessary commands to build and operate Tanzu management and Tanzu Kubernetes clusters. Starting with TKG 2.3.0, Tanzu Core CLI is now distributed separately from Tanzu Kubernetes Grid. For more information about installing the Tanzu CLI for use with Tanzu Kubernetes Grid, see Install the Tanzu CLI.

- **Bootstrap Machine** - The bootstrap machine is the laptop, host, or server on which you download and run the Tanzu CLI. This is where the initial bootstrapping of a management cluster occurs before it is pushed to the platform where it runs. This machine also houses a Harbor instance where all the required Tanzu Kubernetes Grid installation binaries are pushed.

- **Carvel Tools** - An open-source suite of tools. Carvel provides a set of reliable, single-purpose, composable tools that aid in your application building, configuration, and deployment to Kubernetes. Tanzu Kubernetes Grid uses the following tools from the Carvel open-source project:

  - **ytt** - A command-line tool for templating and patching YAML files. You can also use `ytt` to collect fragments and piles of YAML into modular chunks for reuse.

  - **kapp** - The application deployment CLI for Kubernetes. It allows you to install, upgrade, and delete multiple Kubernetes resources as one application.

  - **kbld** - An image-building and resolution tool.

  - **imgpkg** - A tool that enables Kubernetes to store configurations and the associated container images as OCI images, and to transfer these images.

  - **yq** - A lightweight and portable command-line YAML, JSON, and XML processor. `yq` uses `jq`-like syntax but works with YAML files as well as JSON and XML.

- **Tanzu Kubernetes Grid Installer** - The Tanzu Kubernetes Grid installer is a CLI/graphical wizard that provides an option to deploy a management cluster. You launch this installer locally on the bootstrap machine by running the `tanzu management-cluster create` command.

Tanzu Kubernetes Grid Storage

Tanzu Kubernetes Grid integrates with shared datastores available in the vSphere infrastructure. The following types of shared datastores are supported:

- vSAN
- VMFS
- NFS
- vVols

Tanzu Kubernetes Grid uses storage policies to integrate with shared datastores. The policies represent datastores and manage the storage placement of such objects as control plane VMs, container images, and persistent storage volumes.

Tanzu Kubernetes Grid Cluster Plans can be defined by operators to use a certain vSphere Datastore when creating new workload clusters. All developers would then have the ability to provision container-backed persistent volumes from that underlying datastore.

Tanzu Kubernetes Grid is agnostic about which option you choose. For Kubernetes stateful workloads, Tanzu Kubernetes Grid installs the vSphere Container Storage interface (vSphere CSI) to automatically provision Kubernetes persistent volumes for pods.

Tanzu Kubernetes Clusters Networking

A Tanzu Kubernetes cluster provisioned by the Tanzu Kubernetes Grid supports two Container Network Interface (CNI) options:

- Antrea
- Calico

Both are open-source softwares that provide networking for cluster pods, services, and ingress.

When you deploy a Tanzu Kubernetes cluster using Tanzu CLI, Antrea CNI is automatically enabled in the cluster.

To provision a Tanzu Kubernetes cluster using a non-default CNI, see Deploy Tanzu Kubernetes clusters with Calico.

Each CNI is suitable for a different use case. The following table lists common use cases for the three CNIs that Tanzu Kubernetes Grid supports. This table helps you with information on selecting the right CNI in your Tanzu Kubernetes Grid implementation.

CNI	**Use Case**	**Pros and Cons**
Antrea	Enable Kubernetes pod networking with IP overlay networks using VXLAN or Geneve for encapsulation. Optionally, encrypt node-to-node communication using IPSec packet encryption. Antrea supports advanced network use cases like kernel bypass and network service mesh.	Pros: - Provide an option to configure egress IP address pool or static egress IP address for the Kubernetes workloads.
Calico	Calico is used in environments where factors like network performance, flexibility, and power are essential. For routing packets between nodes, Calico leverages the BGP routing protocol instead of an overlay network. This eliminates the need to wrap packets with an encapsulation layer resulting in increased network performance for Kubernetes workloads.	Pros: - Support for network policies - High network performance - SCTP support Cons: - No multicast support

Tanzu Kubernetes Grid Infrastructure Networking

Tanzu Kubernetes Grid on vSphere can be deployed on various networking stacks including

- VMware NSX-T Data Center Networking
- vSphere Networking (VDS)

> The scope of this document is limited to NSX-T Data Center Networking with NSX Advanced load balancer Enterprise Edition.

Tanzu Kubernetes Grid on NSX-T Networking with NSX Advanced Load Balancer

When deployed on VMware NSX-T Networking, Tanzu Kubernetes Grid uses the NSX-T logical segments and gateways to provide connectivity to Kubernetes control plane VMs, worker nodes, services, and applications. All hosts from the cluster where Tanzu Kubernetes clusters are deployed are configured as NSX-T transport nodes, which provide network connectivity to the Kubernetes environment.

You can configure NSX Advanced Load Balancer in Tanzu Kubernetes Grid as:

- L4 load balancer for application hosted on the TKG cluster.
- The L7 ingress service provider for the applications in the clusters that are deployed on vSphere.
- L4 load balancer for the control plane API server.

Each workload cluster integrates with NSX Advanced Load Balancer by running an Avi Kubernetes Operator (AKO) on one of its nodes. The cluster's AKO calls the Kubernetes API to manage the lifecycle of load balancing and ingress resources for its workloads.

NSX Advanced Load Balancer Components

NSX Advanced Load Balancer is deployed in Write Access Mode in VMware NSX Environment. This mode grants NSX Advanced Load Balancer controllers full write access to vCenter which helps in automatically creating, modifying, and removing service engines (SEs) and other resources as needed to adapt to changing traffic needs. The core components of NSX Advanced Load Balancer are as follows:

- **NSX Advanced Load Balancer Controller** - NSX Advanced Load Balancer controller manages virtual service objects and interacts with the vCenter Server infrastructure to manage the lifecycle of the service engines (SEs). It is the central repository for the configurations and policies related to services and management, and it provides the portal for viewing the health of VirtualServices and SEs and the associated analytics that NSX Advanced Load Balancer provides.
- **NSX Advanced Load Balancer Service Engine** - The service engines (SEs) are lightweight VMs that handle all data plane operations by receiving and executing instructions from the controller. The SEs perform load balancing and all client- and server-facing network interactions.
- **Service Engine Group -** Service engines are created within a group, which contains the definition of how the SEs should be sized, placed, and made highly available. Each cloud has at least one SE group.
- **Cloud -** Clouds are containers for the environment that NSX Advanced Load Balancer is installed or operating within. During the initial setup of NSX Advanced Load Balancer, a default cloud, named

`Default-Cloud`, is created. This is where the first controller is deployed into Default-Cloud. Additional clouds may be added containing SEs and virtual services.

- **Avi Kubernetes Operator (AKO)** - It is a Kubernetes operator that runs as a pod in the Supervisor Cluster and Tanzu Kubernetes clusters, and it provides ingress and load balancing functionality. AKO translates the required Kubernetes objects to NSX Advanced Load Balancer objects and automates the implementation of ingresses, routes, and services on the service engines (SE) through the NSX Advanced Load Balancer Controller.
- **AKO Operator (AKOO)** - This is an operator which is used to deploy, manage, and remove the AKO pod in Kubernetes clusters. This operator when deployed creates an instance of the AKO controller and installs all the relevant objects like:
    - AKO `Statefulset`
    - `Clusterrole` and `Clusterrolebinding`
    - `Configmap` (required for the AKO controller and other artifacts).

Tanzu Kubernetes Grid management clusters have an AKO operator installed out-of-the-box during cluster deployment. By default, a Tanzu Kubernetes Grid management cluster has a couple of `AkoDeploymentConfig` created which dictates when and how AKO pods are created in the workload clusters. For more information, see AKO Operator documentation.

Optionally, you can enter one or more cluster labels to identify clusters on which to selectively enable NSX ALB or to customize NSX ALB settings for different groups of clusters. This is useful in the following scenarios: - You want to configure different sets of workload clusters to different Service Engine Groups to implement isolation or to support more Service type Load Balancers than one Service Engine Group's capacity. - You want to configure different sets of workload clusters to different Clouds because they are deployed in different sites.

To enable NSX ALB selectively rather than globally, add labels in the format **key: value** pair in the management cluster config file. This will create a default AKO Deployment Config (ADC) on management cluster with the NSX ALB settings provided. Labels that you define here will be used to create a label selector. Only workload cluster objects that have the matching labels will have the load balancer enabled.

To customize the NSX ALB settings for different groups of clusters, create an AKO Deployment Config (ADC) on management cluster by customizing the NSX ALB settings, and providing a unique label selector for the ADC. Only the workload cluster objects that have the matching labels will have these custom settings applied.

You can label the cluster during the workload cluster deployment or label it manually post cluster creation. If you define multiple key-values, you need to apply all of them. - Provide an AVI_LABEL in the below format in the workload cluster deployment config file, and it will automatically label the cluster and select the matching ADC based on the label selector during the cluster deployment. `AVI_LABELS: | 'type': 'tkg-workloadset01'` - Optionally, you can manually label the cluster object of the corresponding workload cluster with the labels defined in ADC. `kubectl label cluster <cluster-name> type=tkg-workloadset01`

Each environment configured in NSX Advanced Load Balancer is referred to as a cloud. Each cloud in NSX Advanced Load Balancer maintains networking and service engine settings. The cloud is configured with one or more VIP networks to provide IP addresses to load balancing (L4/L7) virtual services created under that cloud.

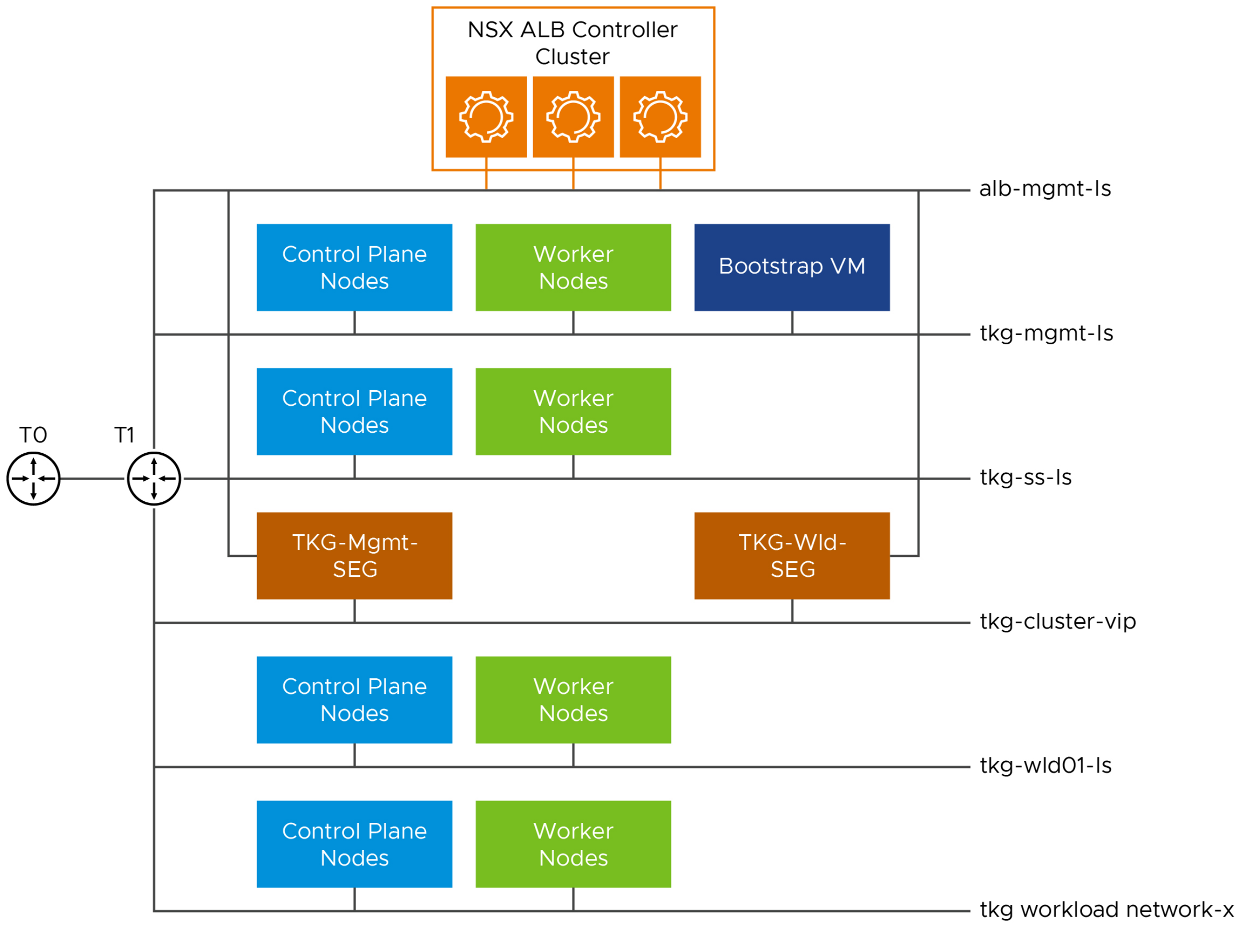
The virtual services can be spanned across multiple service engines if the associated SE group is configured in Active/Active HA mode. A service engine can belong to only one SE group at a time.

IP address allocation for virtual services can be over DHCP or through the in-built IPAM functionality of NSX Advanced Load Balancer. The VIP networks created or configured in NSX Advanced Load Balancer are associated with the IPAM profile.

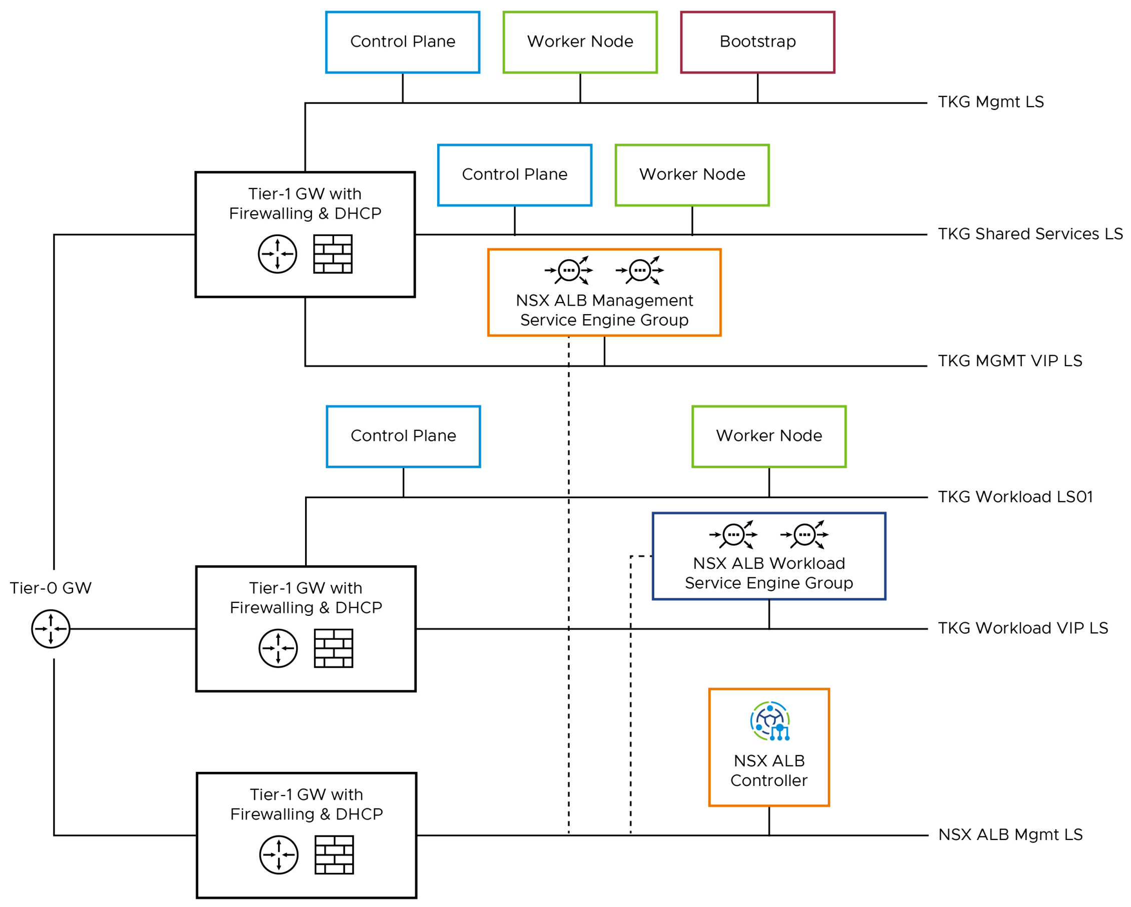
Network Architecture

For the deployment of Tanzu Kubernetes Grid in the VMware NSX-T environment, it is required to build separate networks for the Tanzu Kubernetes Grid management cluster and workload clusters, NSX Advanced Load Balancer management, and cluster-VIP network for control plane HA.

The network reference design can be mapped into this general framework. This design uses a single VIP network for control plane L4 load balancing and application L4/L7. This design is mostly suited for dev/test environment.



Another reference design that can be implemented in production environment is shown below, and it uses separate VIP network for the applications deployed in management/shared services and the workload cluster.



This topology enables the following benefits:

- Isolate and separate SDDC management components (vCenter, ESX) from the Tanzu Kubernetes Grid components. This reference design allows only the minimum connectivity between the Tanzu Kubernetes Grid clusters and NSX Advanced Load Balancer to the vCenter server.
- Isolate and separate the NSX Advanced Load Balancer management network from the Tanzu Kubernetes Grid management segment and workload segments.
- Depending on the workload cluster type and use case, multiple workload clusters may leverage the same workload network or new networks can be used for each workload cluster. To isolate and separate Tanzu Kubernetes Grid workload cluster networking from each other, it is recommended to make use of separate networks for each workload cluster and configure the required firewall between these networks. For more information, see Firewall Recommendations.
- Separate provider and tenant access to the Tanzu Kubernetes Grid environment.
  - Only provider administrators need access to the Tanzu Kubernetes Grid management cluster. This prevents tenants from attempting to connect to the Tanzu Kubernetes Grid management cluster.
- Only allow tenants to access their Tanzu Kubernetes Grid workload clusters and restrict access to this cluster from other tenants.

Network Requirements

As per the defined architecture, the list of required networks follows:

Network Type	DHCP Service	Description & Recommendations
NSX ALB Management Logical Segment	Optional	NSX ALB controllers and SEs are attached to this network. DHCP is not a mandatory requirement on this network as NSX ALB can handle IPAM services for the management network.
TKG Management Logical Segment	Yes	Control plane and worker nodes of TKG management cluster are attached to this network.
TKG Shared Service Logical Segment	Yes	Control plane and worker nodes of TKG shared services cluster are attached to this network.
TKG Workload Logical Segment	Yes	Control plane and worker nodes of TKG workload clusters are attached to this network.
TKG Management VIP Logical Segment	No	Virtual services for control plane HA of all TKG clusters (management, shared services, and workload). Reserve sufficient IP addresses depending on the number of TKG clusters planned to be deployed in the environment. NSX Advanced Load Balancer takes care of IPAM on this network.
TKG Workload VIP Logical Segment	No	Virtual services for applications deployed in the workload cluster. The applications can be of type Load balancer or Ingress. Reserve sufficient IP addresses depending on the number of applications planned to be deployed in the environment. NSX Advanced Load Balancer takes care of IPAM on this network.

You can also select `TKG Workload VIP` network for control plane HA of the workload cluster if you wish so.

Subnet and CIDR Examples

This document uses the following CIDRs for Tanzu Kubernetes Grid deployment:

Network Type	Segment Name	Gateway CIDR	DHCP Pool in NSX-T	NSX ALB IP Pool
NSX ALB Management Network	sfo01-w01-vds01-albmanagement	172.16.10.1/24	N/A	172.16.10.100 - 172.16.10.200
TKG Management VIP Network	sfo01-w01-vds01-tkgclustervip	172.16.80.1/24	N/A	172.16.80.100 - 172.16.80.200
TKG Management Network	sfo01-w01-vds01-tkgmanagement	172.16.40.1/24	172.16.40.100 - 172.16.40.200	N/A
TKG Shared Service Network	sfo01-w01-vds01-tkgshared	172.16.50.1/24	172.16.50.100-172.16.50.200	N/A
TKG Workload Network	sfo01-w01-vds01-tkgworkload	172.16.60.1/24	172.16.60.100-172.16.60.200	N/A
TKG Workload VIP Network	sfo01-w01-vds01-workloadvip	172.16.70.1/24	172.16.70.100-172.16.70.200	N/A

Firewall Requirements

To prepare the firewall, you must collect the following information:

1. NSX ALB Controller nodes and Cluster IP address.
2. NSX ALB Management Network CIDR.
3. TKG Management Network CIDR
4. TKG Shared Services Network CIDR
5. TKG Workload Network CIDR
6. TKG Cluster VIP Address Range
7. Client Machine IP Address
8. Bootstrap machine IP Address
9. Harbor registry IP address
10. vCenter Server IP.
11. DNS server IP(s).
12. NTP Server(s).
13. NSX-T nodes and VIP address.

The following table provides a list of firewall rules based on the assumption that there is no firewall within a subnet/VLAN:

Source	Destination	Protocol:Port	Description	Configured On
NSX Advanced Load Balancer controllers and Cluster IP address	vCenter and ESXi hosts	TCP:443	Allows NSX ALB to discover vCenter objects and deploy SEs as required.	NSX ALB Tier-1 Gateway
NSX Advanced Load Balancer controllers and Cluster IP address	NSX nodes and VIP address.	TCP:443	Allows NSX ALB to discover NSX Objects (logical routers and logical segments, and so on).	NSX ALB Tier-1 Gateway
NSX Advanced Load Balancer management network CIDR	DNS Server. NTP Server	UDP:53 UDP:123	DNS Service Time synchronization	NSX ALB Tier-1 Gateway
Client Machine	NSX Advanced Load Balancer controllers and Cluster IP address	TCP:443	To access NSX Advanced Load Balancer portal.	NSX ALB Tier-1 Gateway
Client Machine	Bootstrap VM IP address	SSH:22	To deploy,configure and manage TKG clusters.	TKG Mgmt Tier-1 Gateway
TKG management network CIDR TKG shared services network CIDR	DNS Server NTP Server	UDP:53 UDP:123	DNS Service Time Synchronization	TKG Mgmt Tier-1 Gateway

Source	Destination	Protocol:Port	Description	Configured On
TKG management network CIDR TKG shared services network CIDR	vCenter Server	TCP:443	Allows components to access vCenter to create VMs and storage volumes	TKG Mgmt Tier-1 Gateway
TKG management network CIDR TKG shared services network CIDR	Harbor Registry	TCP:443	Allows components to retrieve container images. This registry can be a local or a public image registry.	TKG Mgmt Tier-1 Gateway
TKG management network CIDR TKG shared services network CIDR	TKG Management VIP Network	TCP:6443	For management cluster to configure workload cluster. Allows shared cluster to register with management cluster.	TKG Mgmt Tier-1 Gateway
TKG management network CIDR TKG shared services network CIDR	NSX Advanced Load Balancer management network CIDR	TCP:443	Allow Avi Kubernetes Operator (AKO) and AKO Operator (AKOO) access to NSX ALB controller.	TKG Mgmt Tier-1 Gateway
TKG workload network CIDR	DNS Server NTP Server	UDP:53 UDP:123	DNS Service Time Synchronization	TKG Workload Tier-1 Gateway
TKG workload network CIDR	vCenter Server	TCP:443	Allows components to access vCenter to create VMs and storage volumes.	TKG Workload Tier-1 Gateway
TKG workload network CIDR	Harbor Registry	TCP:443	Allows components to retrieve container images. This registry can be a local or a public image registry.	TKG Workload Tier-1 Gateway
TKG workload network CIDR	TKG Management VIP Network	TCP:6443	Allow TKG workload clusters to register with TKG management cluster.	TKG Workload Tier-1 Gateway
TKG workload network CIDR	NSX Advanced Load Balancer management network CIDR	TCP:443	Allow Avi Kubernetes Operator (AKO) and AKO Operator (AKOO) access to NSX ALB controller.	TKG Workload Tier-1 Gateway
deny-all	any	any	deny	All Tier-1 gateways

Design Recommendations

NSX Advanced Load Balancer Recommendations

The following table provides the recommendations for configuring NSX Advanced Load Balancer in a Tanzu Kubernetes Grid environment:

Decision ID	Design Decision	Design Justification	Design Implications
TKO-ALB-001	Deploy NSX ALB controller cluster nodes on a network dedicated to NSX ALB.	Isolate NSX ALB traffic from infrastructure management traffic and Kubernetes workloads.	Allows ease of management for the controllers. Additional Network (VLAN) is required.
TKO-ALB-002	Deploy 3 NSX ALB controller nodes.	To achieve high availability for the NSX ALB platform.In clustered mode, NSX ALB availability is not impacted by an individual controller node failure. The failed node can be removed from the cluster and redeployed if recovery is not possible.	Clustered mode requires more compute and storage resources.
TKO-ALB-003	Initial setup should be done only on one NSX ALB controller VM out of the three deployed, to create an NSX ALB controller cluster.	NSX ALB controller cluster is created from an initialized NSX ALB controller which becomes the cluster leader. Follower NSX ALB controller nodes need to be uninitialized to join the cluster.	NSX ALB controller cluster creation fails if more than one NSX ALB controller is initialized.
TKO-ALB-004	Use static IP addresses for the NSX ALB controllers.	NSX ALB controller cluster uses management IP addresses to form and maintain quorum for the control plane cluster. Any changes to management IP addresses are disruptive.	NSX ALB Controller control plane might go down if the management IP addresses of the controller node changes.
TKO-ALB-005	Use NSX ALB IPAM for service engine data network and virtual services.	Guarantees IP address assignment for service engine data NICs and virtual services.	Removes the corner case scenario when the DHCP server runs out of the lease or is down.
TKO-ALB-006	Reserve an IP address in the NSX ALB management subnet to be used as the cluster IP address for the controller cluster.	NSX ALB portal is always accessible over cluster IP address regardless of a specific individual controller node failure.	NSX ALB administration is not affected by an individual controller node failure.
TKO-ALB-007	Shared service engines for the same type of workload (dev/test/prod) clusters.	Minimize the licensing cost	Each service engine contributes to the CPU core capacity associated with a license. Sharing service engines can help reduce the licensing cost.
TKO-ALB-008	Configure anti-affinity rules for the NSX ALB controller cluster.	This is to ensure that no two controllers end up in same ESXi host and thus avoid single point of failure.	Anti-Affinity rules need to be created manually.
TKO-ALB-009	Configure backup for the NSX ALB Controller cluster.	Backups are required if the NSX ALB Controller becomes inoperable or if the environment needs to be restored from a previous state.	To store backups, a SCP capable backup location is needed. SCP is the only supported protocol currently.
TKO-ALB-010	Create an NSX-T Cloud connector on NSX ALB controller for each NSX transport zone requiring load balancing.	An NSX-T Cloud connector configured on the NSX ALB controller provides load balancing for workloads belonging to a transport zone on NSX-T.	None

Decision ID	Design Decision	Design Justification	Design Implications
TKO-ALB-011	Configure Remote logging for NSX ALB Controller to send events on Syslog.	For operations teams to be able to centrally monitor NSX ALB and escalate alerts events must be sent from the NSX ALB Controller	Additional Operational Overhead. Additional infrastructure Resource.
TKO-ALB-012	Use LDAP/SAML based Authentication for NSX ALB	Helps to maintain Role based Access Control	Additional Configuration is required.

NSX Advanced Load Balancer Service Engine Recommendations

Decision ID	Design Decision	Design Justification	Design Implications
TKO-ALB-SE-001	Configure SE Group for Active/Active HA mode.	Provides optimum resiliency, performance, and utilization.	Certain applications might not work in Active/Active mode. For instance, applications that require preserving client IP address. In such cases, use the legacy Active/Standby HA mode.
TKO-ALB-SE-002	Configure anti-affinity rule for the SE VMs.	This is ensure that no two SEs in the same SE group end up on same ESXi Host and thus avoid single point of failure.	Anti-Affinity rules need to be created manually.
TKO-ALB-SE-003	Configure CPU and memory reservation for the SE VMs.	This is to ensure that service engines don’t compete with other VMs during resource contention.	CPU and memory reservation is configured at SE group level.
TKO-ALB-SE-004	Enable ‘Dedicated dispatcher CPU’ on SE groups that contain the SE VMs of 4 or more vCPUs. Note: This setting must be enabled on SE groups that are servicing applications that have high network requirement.	This enables a dedicated core for packet processing enabling high packet pipeline on the SE VMs.	None.
TKO-ALB-SE-005	Dedicated Service Engine Group for the TKG Management	SE resources are guaranteed for TKG Management Stack and provides data path segregation for Management and Tenant Application	Dedicated service engine Groups increase licensing cost.
TKO-ALB-SE-006	Dedicated Service Engine Group for the TKG Workload Clusters Depending on the nature and type of workloads (dev/prod/test)	SE resources are guaranteed for single or set of workload clusters and provides data path segregation for Tenant Application hosted on workload clusters	Dedicated service engine Groups increase licensing cost.
TKO-ALB-SE-007	Set ‘Placement across the Service Engines’ setting to ‘distributed’.	This allows for maximum fault tolerance and even utilization of capacity.	None
TKO-ALB-SE-008	Set the SE size to a minimum 2vCPU and 4GB of Memory	This configuration should meet the most generic use case	For services that require higher throughput, these configuration needs to be investigated and modified accordingly.

Decision ID	Design Decision	Design Justification	Design Implications
TKO-ALB-SE-009	Enable ALB Service Engine Self Elections	Enable SEs to elect a primary amongst themselves in the absence of connectivity to the NSX ALB controller	None

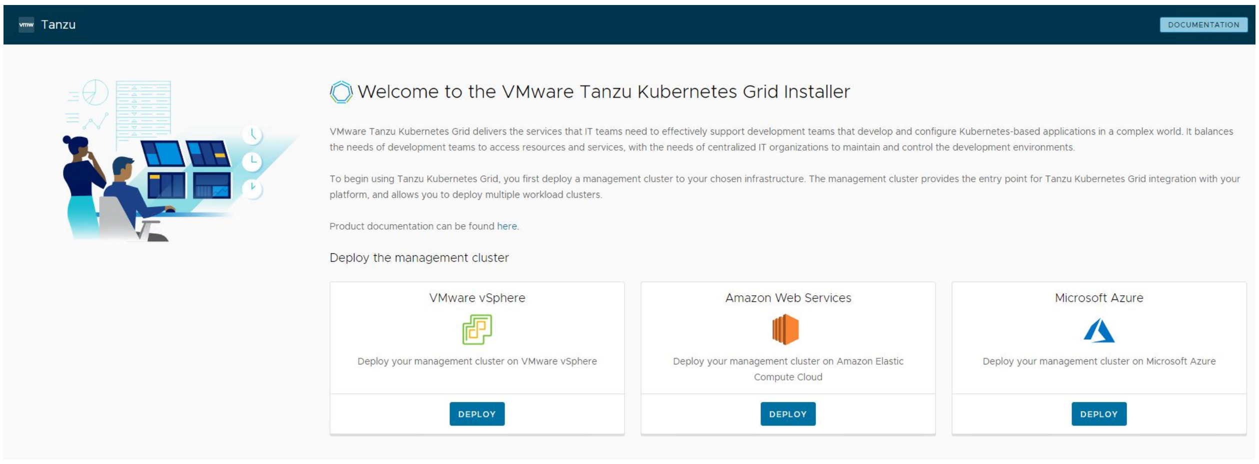
Installation Experience

Tanzu Kubernetes Grid management cluster is the first component that you deploy to get started with Tanzu Kubernetes Grid.

You can deploy the management cluster in one of the following ways:

- Run the Tanzu Kubernetes Grid installer, a wizard interface that guides you through the process of deploying a management cluster.
- Create and edit YAML configuration files, and use them to deploy a management cluster with the CLI commands. This is the recommended method if you are installing a Tanzu Kubernetes Grid management cluster in an air-gapped environment.

By using the current version of the The Tanzu Kubernetes Grid Installation user interface, you can install Tanzu Kubernetes Grid on VMware vSphere, AWS, and Microsoft Azure. The UI provides a guided experience tailored to the IaaS, in this case on VMware vSphere backed by NSX-T Data Center networking.



Welcome to the VMware Tanzu Kubernetes Grid Installer

VMware Tanzu Kubernetes Grid delivers the services that IT teams need to effectively support development teams that develop and configure Kubernetes-based applications in a complex world. It balances the needs of development teams to access resources and services, with the needs of centralized IT organizations to maintain and control the development environments.

To begin using Tanzu Kubernetes Grid, you first deploy a management cluster to your chosen infrastructure. The management cluster provides the entry point for Tanzu Kubernetes Grid integration with your platform, and allows you to deploy multiple workload clusters.

Product documentation can be found here.

Deploy the management cluster

VMware vSphere
Deploy your management cluster on VMware vSphere

Amazon Web Services
Deploy your management cluster on Amazon Elastic Compute Cloud

Microsoft Azure
Deploy your management cluster on Microsoft Azure

The installation process takes you through the setup of a `management cluster` on your vSphere with NSX-T environment. Once the management cluster is deployed, you can make use of Tanzu CLI to deploy Tanzu Kubernetes shared services and workload clusters.

To deploy the Tanzu Kubernetes Grid management cluster directly from CLI, see the supplemental information Cluster Deployment Parameters for a sample yaml file used for deployment.

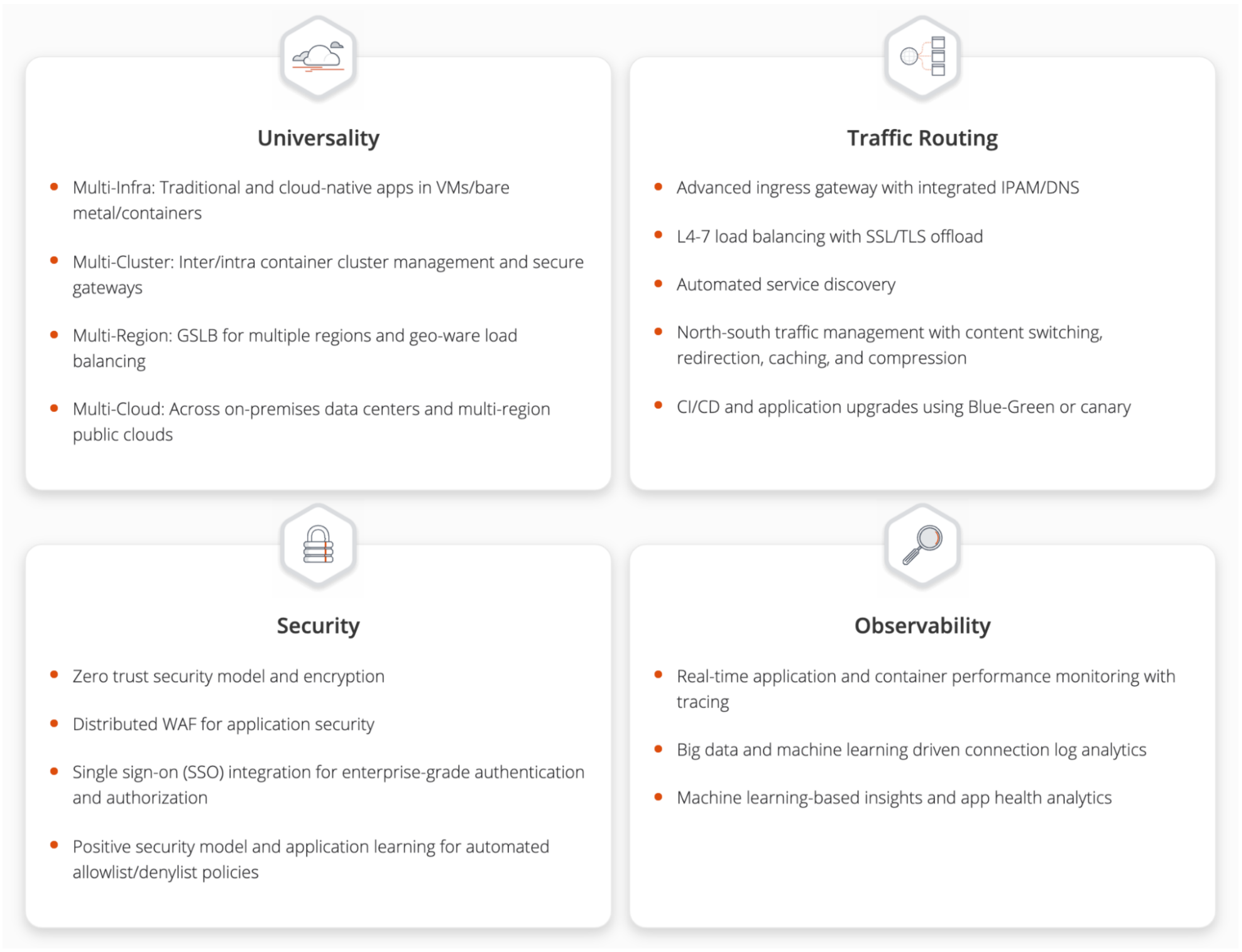
Kubernetes Ingress Routing

The default installation of Tanzu Kubernetes Grid does not have any default ingress controller deployed. Users can use Contour (available for installation through Tanzu Packages), or any third-party ingress controller of their choice.

Contour is an open-source controller for Kubernetes ingress routing. Contour can be installed in the shared services cluster on any Tanzu Kubernetes cluster. Deploying Contour is a prerequisite if you want to deploy the Prometheus, Grafana, and Harbor packages on a workload cluster.

For more information about Contour, see the Contour website and Implementing Ingress Control with Contour.

Another option is to use the NSX Advanced Load Balancer Kubernetes ingress controller (available only with the NSX ALB Enterprise license) which offers an advanced L7 ingress for containerized applications that are deployed in the Tanzu Kubernetes workload cluster.



Universality

- Multi-Infra: Traditional and cloud-native apps in VMs/bare metal/containers
- Multi-Cluster: Inter/intra container cluster management and secure gateways
- Multi-Region: GSLB for multiple regions and geo-ware load balancing
- Multi-Cloud: Across on-premises data centers and multi-region public clouds

Traffic Routing

- Advanced ingress gateway with integrated IPAM/DNS
- L4-7 load balancing with SSL/TLS offload
- Automated service discovery
- North-south traffic management with content switching, redirection, caching, and compression
- CI/CD and application upgrades using Blue-Green or canary

Security

- Zero trust security model and encryption
- Distributed WAF for application security
- Single sign-on (SSO) integration for enterprise-grade authentication and authorization
- Positive security model and application learning for automated allowlist/denylist policies

Observability

- Real-time application and container performance monitoring with tracing
- Big data and machine learning driven connection log analytics
- Machine learning-based insights and app health analytics

For more information about the NSX ALB ingress controller, see Configuring L7 Ingress with NSX Advanced Load Balancer.

The following table provides general recommendations on when you should use a specific ingress controller for your Kubernetes environment.

Ingress Controller	Use Cases
Contour	Use Contour when only north-south traffic is needed in a Kubernetes cluster. You can apply security policies for north-south traffic by defining the policies in the applications manifest file. It's a reliable solution for simple Kubernetes workloads.
NSX Advanced Load Balancer ingress controller	Use NSX Advanced Load Balancer ingress controller when a containerized application requires features like local and global server load balancing (GSLB), web application firewall (WAF), performance monitoring, direct routing from LB to pod, etc.

NSX Advanced Load Balancer as an L4+L7 Ingress Service Provider

NSX Advanced Load Balancer provides an L4+L7 load balancing solution for vSphere. It includes a Kubernetes operator that integrates with the Kubernetes API to manage the lifecycle of load balancing and ingress resources for workloads.

Legacy ingress services for Kubernetes include multiple disparate solutions. The services and products contain independent components that are difficult to manage and troubleshoot. The ingress services have reduced observability capabilities with little analytics, and they lack comprehensive visibility into the applications that run on the system. Cloud-native automation is difficult in the legacy ingress services.

In comparison to the legacy Kubernetes ingress services, NSX Advanced Load Balancer has comprehensive load balancing and ingress services features. As a single solution with a central control, NSX Advanced Load Balancer is easy to manage and troubleshoot. NSX Advanced Load Balancer supports real-time telemetry with an insight into the applications that run on the system. The elastic auto-scaling and the decision automation features highlight the cloud-native automation capabilities of NSX Advanced Load Balancer.

NSX Advanced Load Balancer also lets you configure L7 ingress for your workload clusters by using one of the following options:

- L7 ingress in ClusterIP mode
- L7 ingress in NodePortLocal mode
- L7 ingress in NodePort mode
- NSX Advanced Load Balancer L4 ingress with Contour L7 ingress

L7 Ingress in ClusterIP Mode

This option enables NSX Advanced Load Balancer L7 ingress capabilities, including sending traffic directly from the service engines (SEs) to the pods, preventing multiple hops that other ingress solutions need when sending packets from the load balancer to the right node where the pod runs. The NSX Advanced Load Balancer controller creates a virtual service with a backend pool with the pod IP addresses which helps send the traffic directly to the pods.

However, each workload cluster needs a dedicated SE group for Avi Kubernetes Operator (AKO) to work, which could increase the number of SEs you need for your environment. This mode is used when you have a small number of workload clusters.

L7 Ingress in NodePort Mode

The NodePort mode is the default mode when AKO is installed on Tanzu Kubernetes Grid. This option allows your workload clusters to share SE groups and is fully supported by VMware. With this option, the services of your workloads must be set to NodePort instead of ClusterIP even when accompanied by an ingress object. This ensures that NodePorts are created on the worker nodes and traffic can flow through the SEs to the pods via the NodePorts. Kube-Proxy, which runs on each node as DaemonSet, creates network rules to expose the application endpoints to each of the nodes in the format “NodeIP:NodePort”. The NodePort value is the same for a service on all the nodes. It exposes the port on all the nodes of the Kubernetes Cluster, even if the pods are not running on it.

L7 Ingress in NodePortLocal Mode

This feature is supported only with Antrea CNI. You must enable this feature on a workload cluster before its creation. The primary difference between this mode and the NodePort mode is that the traffic is sent directly to the pods in your workload cluster through node ports without interfering Kube-proxy. With this option, the workload clusters can share SE groups. Similar to the ClusterIP Mode, this option avoids the potential extra hop when sending traffic from the NSX Advanced Load Balancer SEs to the pod by targeting the right nodes where the pods run.

Antrea agent configures NodePortLocal port mapping rules at the node in the format “NodeIP:Unique Port” to expose each pod on the node on which the pod of the service is running. The default range of the port number is 61000-62000. Even if the pods of the service are running on the same Kubernetes node, Antrea agent publishes unique ports to expose the pods at the node level to integrate with the load balancer.

NSX ALB L4 Ingress with Contour L7 Ingress

This option does not have all the NSX Advanced Load Balancer L7 ingress capabilities but uses it for L4 load balancing only and leverages Contour for L7 ingress. This also allows sharing SE groups across workload clusters. This option is supported by VMware and it requires minimal setup.

NSX Advanced Load Balancer L7 Ingress Recommendations

Decision ID	Design Decision	Design Justification	Design Implications
TKO-ALB-L7-001	Deploy NSX ALB L7 ingress in NodePortLocal mode.	1. Network hop efficiency is gained by bypassing the kube-proxy to receive external traffic to applications. 2. TKG clusters can share SE groups, optimizing or maximizing capacity and license consumption. 3. Pod’s node port only exist on nodes where the Pod is running, and it helps to reduce the east-west traffic and encapsulation overhead. 4. Better session persistence.	1. This is supported only with Antrea CNI. 2. `NodePortLocal` mode is currently only supported for nodes running Linux or Windows with IPv4 addresses. Only TCP and UDP service ports are supported (not SCTP). For more information, see Antrea NodePortLocal Documentation.

VMware recommends using NSX Advanced Load Balancer L7 ingress with the NodePortLocal mode as it gives you a distinct advantage over other modes as mentioned below:

- Although there is a constraint of one SE group per Tanzu Kubernetes Grid cluster, which results in increased license capacity, ClusterIP provides direct communication to the Kubernetes pods, enabling persistence and direct monitoring of individual pods.
- NodePort resolves the issue for needing a SE group per workload cluster, but a kube-proxy is created on each and every workload node even if the pod doesn’t exist in it, and there’s no direct connectivity. Persistence is then broken.
- NodePortLocal is the best of both use cases. Traffic is sent directly to the pods in your workload cluster through node ports without interfering with kube-proxy. SE groups can be shared and load balancing persistence is supported.

Network Recommendations

The key network recommendations for a production-grade Tanzu Kubernetes Grid deployment with NSX-T Data Center Networking are as follows:

Decision ID	Design Decision	Design Justification	Design Implications
TKO-NET-001	Use separate logical segments for management cluster, shared services cluster, workload clusters, and VIP network.	To have a flexible firewall and security policies.	Sharing the same network for multiple clusters can complicate firewall rules creation.
TKO-NET-002	Configure DHCP for each TKG cluster network.	Tanzu Kubernetes Grid does not support static IP address assignments for Kubernetes VM components.	IP address pool can be used for the TKG clusters in absence of the DHCP.
TKO-NET-003	Use NSX for configuring DHCP	This avoids setting up dedicated DHCP server for TKG.	For a simpler configuration, make use of the DHCP local server to provide DHCP services for required segments.
TKO-NET-004	Create a overlay-backed NSX segment connected to a Tier-1 gateway for the SE management for the NSX-T Cloud of overlay type.	This network is used for the controller to the SE connectivity.	None
TKO-NET-005	Create a overlay-backed NSX segment as data network for the NSX-T Cloud of overlay type.	The SEs are placed on overlay segments created on Tier-1 gateway.	None

With Tanzu Kubernetes Grid 2.3 and above, you can use Node IPAM, which simplifies the allocation and management of IP addresses for cluster nodes within the cluster. This eliminates the need for external DHCP configuration.

The Node IPAM can be configured for standalone management clusters on vSphere, and the associated class-based workload clusters that they manage. In the Tanzu Kubernetes Grid Management configuration file, a dedicated Node IPAM pool is defined for the management cluster only.

The following types of Node IPAM pools are available for workload clusters:

- **InClusterIPPool -** Configures IP pools that are only available to workload clusters in the same management cluster namespace. For example, default.
- **GlobalInClusterIPPool -** Configures IP pools with addresses that can be allocated to workload clusters across multiple namespaces.

Node IPAM in TKG provides flexibility in managing IP addresses for both management and workload clusters that allows efficient IP allocation and management within the cluster environment.

Tanzu Kubernetes Grid Clusters Recommendations

Decision ID	Design Decision	Design Justification	Design Implications
TKO-TKG-001	Use NSX ALB as your control plane endpoint provider and for application load balancing.	Eliminates the requirement for an external load balancer and additional configuration changes on your Tanzu Kubernetes Grid clusters.	Add NSX ALB License cost to the solution.

Decision ID	Design Decision	Design Justification	Design Implications
TKO-TKG-002	Use NSX Advanced Load Balancer as your control plane endpoint provider and for application load balancing.	Eliminates the requirement for an external load balancer and additional configuration changes on your Tanzu Kubernetes Grid clusters.	Adds NSX Advanced Load Balancer License cost to the solution.
TKO-TKG-003	Deploy Tanzu Kubernetes Management cluster in large form factor.	Large form factor should suffice to integrate TKG Management cluster with TMC, pinniped and Velero. This must be capable of accommodating 100+ Tanzu Workload Clusters.	Consume more resources from infrastructure.
TKO-TKG-004	Deploy the Tanzu Kubernetes Cluster with prod plan(Management and Workload Clusters).	Deploying three control plane nodes ensures the state of your Tanzu Kubernetes Cluster control plane stays healthy in the event of a node failure.	Consume more resources from infrastructure.
TKO-TKG-005	Enable identity management for Tanzu Kubernetes Grid clusters.	To avoid usage of administrator credentials and ensure that required users with right roles have access to Tanzu Kubernetes Grid clusters.	Required external Identity Management.
TKO-TKG-006	Enable MachineHealthCheck for TKG clusters.	vSphere HA and MachineHealthCheck interoperability work together to enhance workload resiliency.	NA

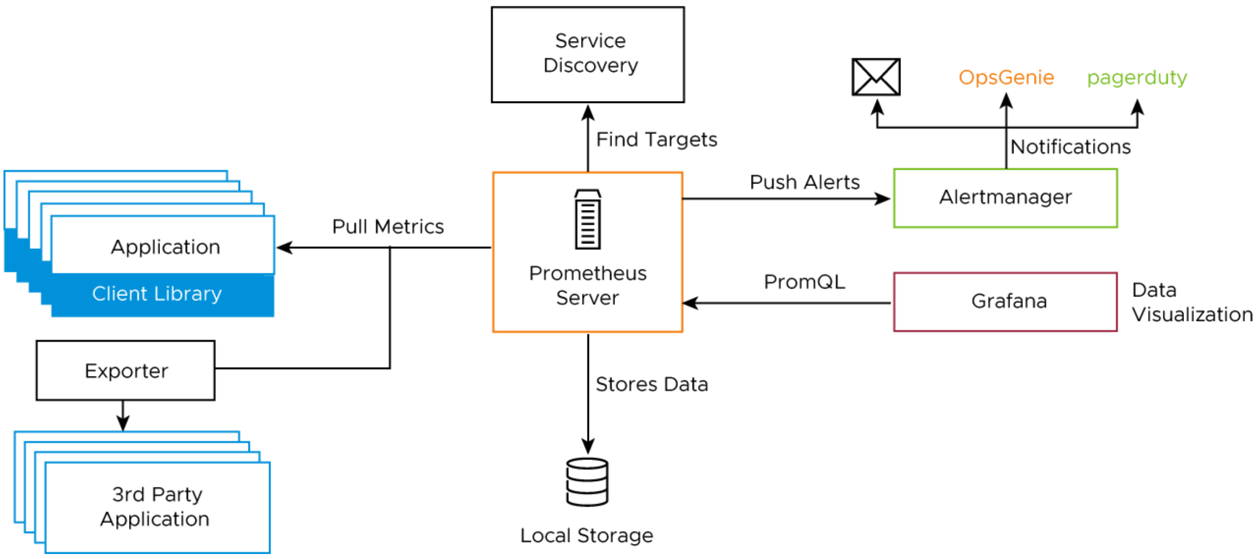
Tanzu Kubernetes Grid Monitoring

In an air-gapped environment, monitoring for the Tanzu Kubernetes clusters is provided through Prometheus and Grafana.

- Prometheus is an open-source system monitoring and alerting toolkit. It can collect metrics from target clusters at specified intervals, evaluate rule expressions, display the results, and trigger alerts if certain conditions arise. The Tanzu Kubernetes Grid implementation of Prometheus includes **Alert Manager**, which you can configure to notify you when certain events occur.
- Grafana is open-source visualization and analytics software. It allows you to query, visualize, alert on, and explore your metrics no matter where they are stored.

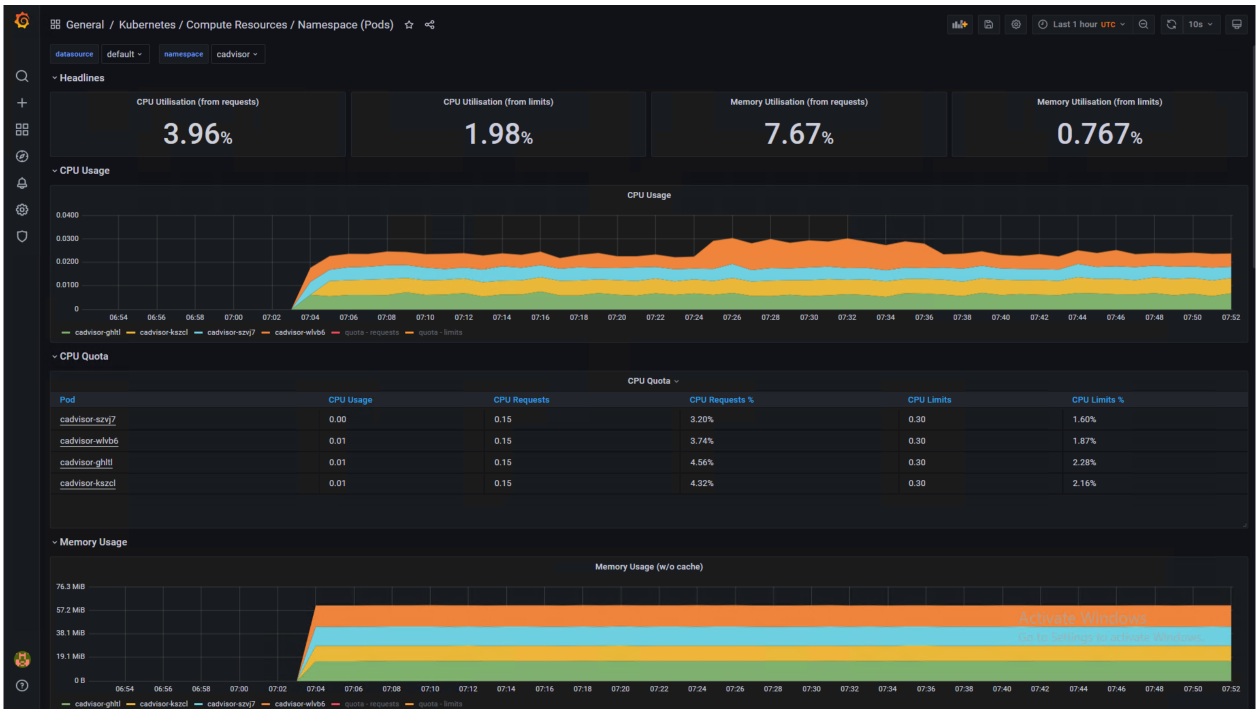
Both Prometheus and Grafana are installed through user-managed Tanzu packages by creating the deployment manifests and invoking the `tanzu package install` command to deploy the packages in the Tanzu Kubernetes clusters.

The following diagram shows how the monitoring components on a cluster interact.



You can use out-of-the-box Kubernetes dashboards or you can create new dashboards to monitor compute, network, and storage utilization of Kubernetes objects such as Clusters, Namespaces, Pods, etc. See the sample dashboards shown below:

Namespace (Pods) Compute Resources Utilization Dashboard

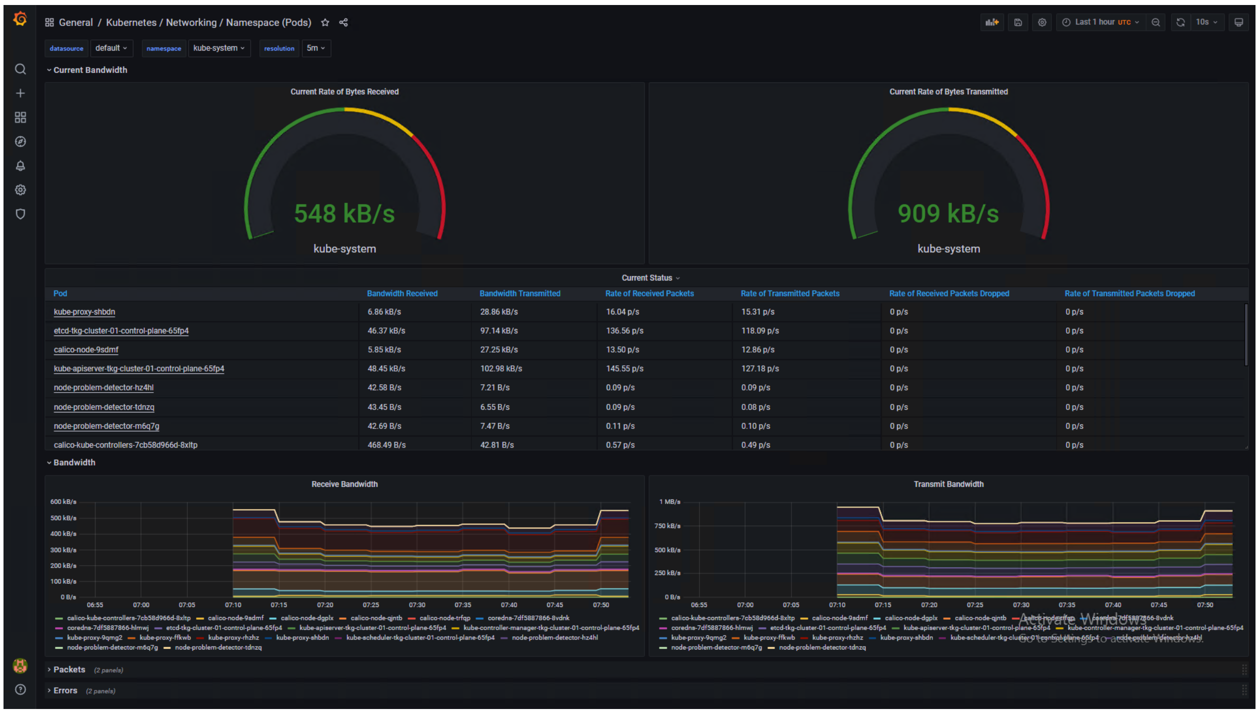


CPU Utilisation (from requests)	CPU Utilisation (from limits)	Memory Utilisation (from requests)	Memory Utilisation (from limits)
3.96%	1.98%	7.67%	0.767%

CPU Quota

Pod	CPU Usage	CPU Requests	CPU Requests %	CPU Limits	CPU Limits %
cadvisor-szvj7	0.00	0.15	3.20%	0.30	1.60%
cadvisor-wlvb6	0.01	0.15	3.74%	0.30	1.87%
cadvisor-ghltl	0.01	0.15	4.56%	0.30	2.28%
cadvisor-kszcl	0.01	0.15	4.32%	0.30	2.16%

Namespace (Pods) Networking Utilization Dashboard



General / Kubernetes / Networking / Namespace (Pods)

datasource: default | namespace: kube-system | resolution: 5m

Current Bandwidth

Current Rate of Bytes Received: 548 kB/s kube-system

Current Rate of Bytes Transmitted: 909 kB/s kube-system

Current Status

Pod	Bandwidth Received	Bandwidth Transmitted	Rate of Received Packets	Rate of Transmitted Packets	Rate of Received Packets Dropped	Rate of Transmitted Packets Dropped
kube-proxy-shbdn	6.86 kB/s	28.86 kB/s	16.04 p/s	15.31 p/s	0 p/s	0 p/s
etcd-tkg-cluster-01-control-plane-65fp4	46.37 kB/s	97.14 kB/s	136.56 p/s	118.09 p/s	0 p/s	0 p/s
calico-node-9sdmf	5.85 kB/s	27.25 kB/s	13.50 p/s	12.86 p/s	0 p/s	0 p/s
kube-apiserver-tkg-cluster-01-control-plane-65fp4	48.45 kB/s	102.98 kB/s	145.55 p/s	127.18 p/s	0 p/s	0 p/s
node-problem-detector-hz4hl	42.58 B/s	7.21 B/s	0.09 p/s	0.09 p/s	0 p/s	0 p/s
node-problem-detector-tdnzq	43.45 B/s	6.55 B/s	0.09 p/s	0.08 p/s	0 p/s	0 p/s
node-problem-detector-m6q7g	42.69 B/s	7.47 B/s	0.11 p/s	0.10 p/s	0 p/s	0 p/s
calico-kube-controllers-7cb58d966d-8xltp	468.49 B/s	42.81 B/s	0.57 p/s	0.49 p/s	0 p/s	0 p/s

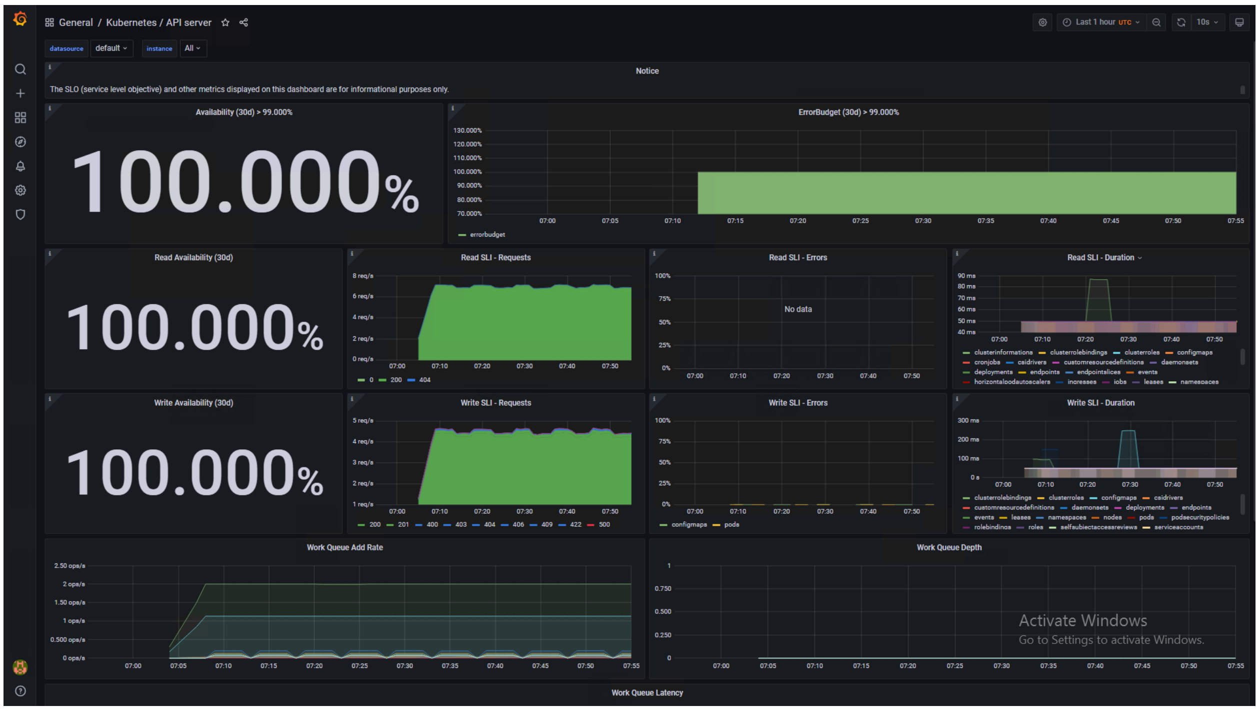
Bandwidth

Receive Bandwidth | Transmit Bandwidth

Packets (2 panels)

Errors (2 panels)

API Server Availability Dashboard



General / Kubernetes / API server

datasource: default | instance: All

Notice

The SLO (service level objective) and other metrics displayed on this dashboard are for informational purposes only.

Availability (30d) > 99.000%: 100.000%

ErrorBudget (30d) > 99.000%

Read Availability (30d): 100.000%

Read SLI - Requests | Read SLI - Errors (No data) | Read SLI - Duration

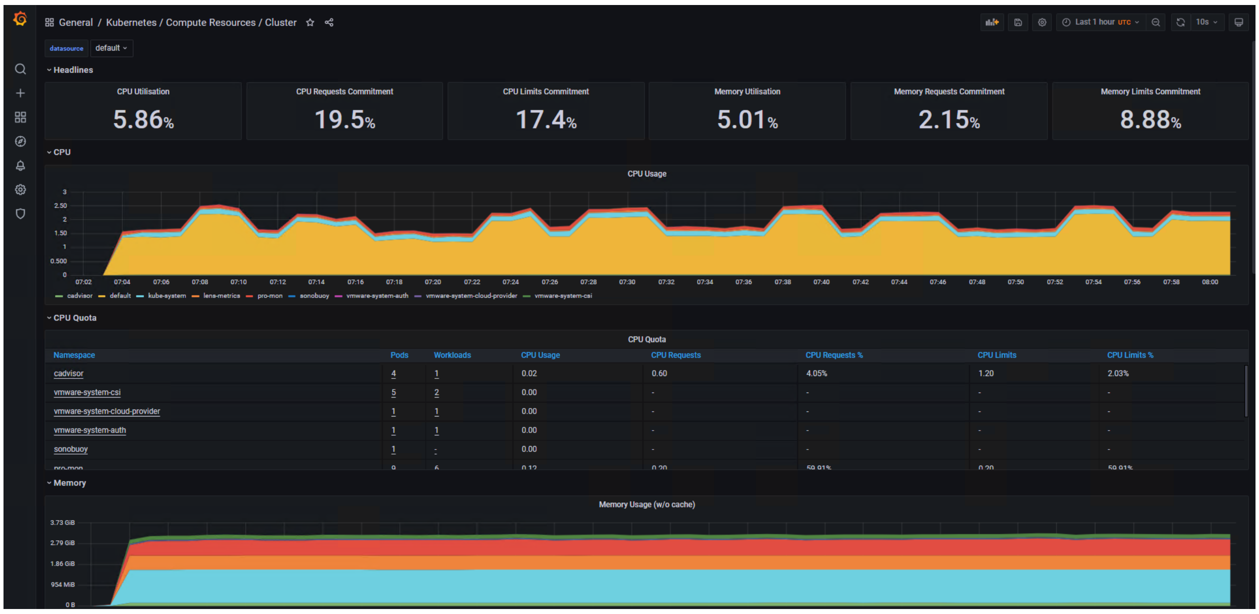
Write Availability (30d): 100.000%

Write SLI - Requests | Write SLI - Errors | Write SLI - Duration

Work Queue Add Rate | Work Queue Depth

Work Queue Latency

Cluster Compute Resources Utilization Dashboard



CPU Utilisation	CPU Requests Commitment	CPU Limits Commitment	Memory Utilisation	Memory Requests Commitment	Memory Limits Commitment
5.86%	19.5%	17.4%	5.01%	2.15%	8.88%

CPU Quota

Namespace	Pods	Workloads	CPU Usage	CPU Requests	CPU Requests %	CPU Limits	CPU Limits %
cadvisor	4	1	0.02	0.60	4.05%	1.20	2.03%
vmware-system-csi	5	2	0.00	-	-	-	-
vmware-system-cloud-provider	1	1	0.00	-	-	-	-
vmware-system-auth	1	1	0.00	-	-	-	-
sonobuoy	1	-	0.00	-	-	-	-
pro-mon	9	6	0.12	0.20	59.91%	0.20	59.91%

Container Registry

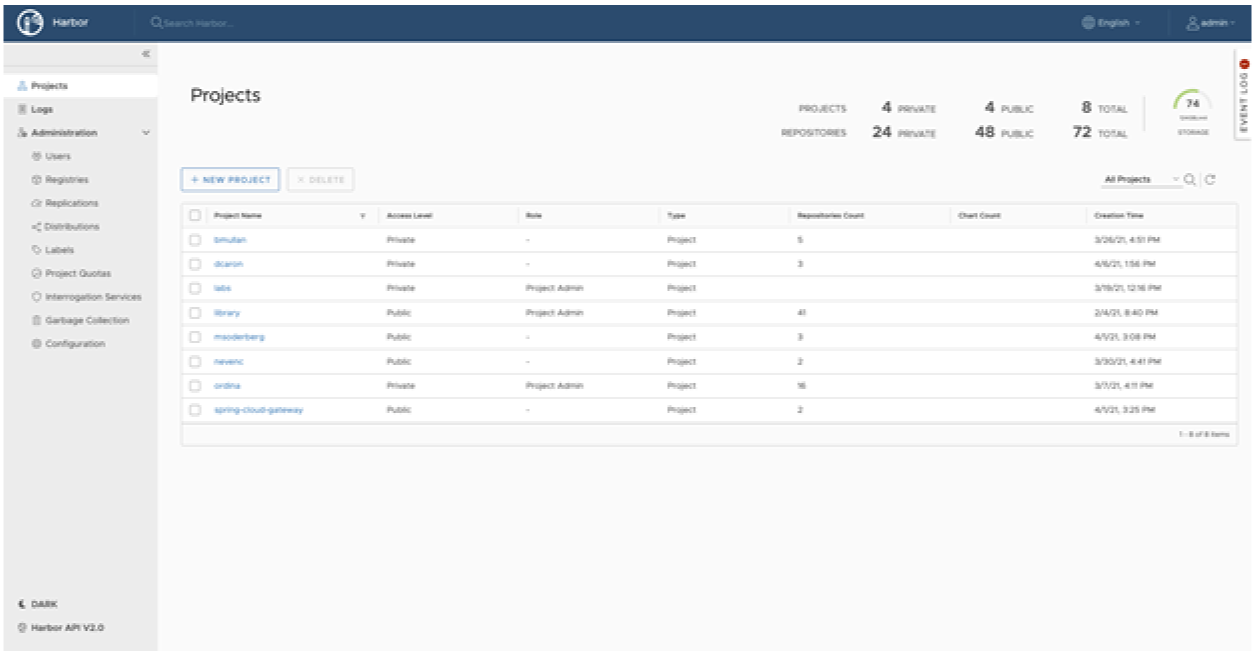
Tanzu Kubernetes Grid includes Harbor as a container registry. Harbor provides a location for pushing, pulling, storing, and scanning container images used in your Kubernetes clusters.

Harbor registry is used for day-2 operations of the Tanzu Kubernetes workload clusters. Typical day-2 operations include tasks such as pulling images from Harbor for application deployment, pushing custom images to Harbor, etc.

You may use one of the following methods to install Harbor:

- **Tanzu Kubernetes Grid Package deployment** - VMware recommends this installation method for general use cases. The Tanzu packages, including Harbor, must either be pulled directly from VMware or be hosted in an internal registry.
- **VM-based deployment using OVA** - VMware recommends this installation method in cases where Tanzu Kubernetes Grid is being installed in an air-gapped or Internet-restricted environment, and no pre-existing image registry exists to host the Tanzu Kubernetes Grid system images. VM-based deployments are only supported by VMware Global Support Services to host the system images for air-gapped or Internet-restricted deployments. Do not use this method for hosting application images.

If you are deploying Harbor without a publicly signed certificate, you must include the Harbor root CA in your Tanzu Kubernetes Grid clusters. To do so, follow the procedure in Trust Custom CA Certificates on Cluster Nodes.



Projects

PROJECTS	4 PRIVATE	4 PUBLIC	8 TOTAL
REPOSITORIES	24 PRIVATE	48 PUBLIC	72 TOTAL

Project Name	Access Level	Role	Type	Repositories Count	Chart Count	Creation Time
bmulan	Private	-	Project	5		3/26/21, 4:51 PM
dcaron	Private	-	Project	3		4/6/21, 1:56 PM
labs	Private	Project Admin	Project			3/19/21, 12:16 PM
library	Public	Project Admin	Project	41		2/4/21, 8:40 PM
msoderberg	Public	-	Project	3		4/1/21, 3:08 PM
nevenc	Public	-	Project	2		3/30/21, 4:41 PM
ordina	Private	Project Admin	Project	16		3/7/21, 4:11 PM
spring-cloud-gateway	Public	-	Project	2		4/1/21, 3:25 PM

1 - 8 of 8 items

Tanzu Kubernetes Grid Logging

Metrics and logs are critical for any system or application as they provide insights into the activities of the system or the application. It is important to have a central place to observe a multitude of metrics and log sources from multiple endpoints.

Log processing and forwarding in Tanzu Kubernetes Grid is provided via Fluent Bit. Fluent bit binaries are available as part of extensions and can be installed on management cluster or in workload cluster. Fluent Bit is a light-weight log processor and forwarder that allows you to collect data and logs from different sources, unify them, and send them to multiple destinations. VMware Tanzu Kubernetes Grid includes signed binaries for Fluent Bit that you can deploy on management clusters and on Tanzu Kubernetes clusters to provide a log-forwarding service.

Fluent Bit makes use of the Input Plug-ins, the filters, and the Output Plug-ins. The Input Plug-ins define the source from where it can collect data, and the Output plug-ins define the destination where it should send the information. The Kubernetes filter will enrich the logs with Kubernetes metadata, specifically labels and annotations. Once you configure Input and Output plug-ins on the Tanzu Kubernetes Grid cluster. Fluent Bit is installed as a user-managed package.

Fluent Bit integrates with logging platforms such as VMware Aria Operations for Logs, Elasticsearch, Kafka, Splunk, or an HTTP endpoint. For more details about configuring Fluent Bit to your logging provider, see Implement Log Forwarding with Fluent Bit.

Bring Your Own Images for Tanzu Kubernetes Grid Deployment

You can build custom machine images for Tanzu Kubernetes Grid to use as a VM template for the management and Tanzu Kubernetes (workload) cluster nodes that it creates. Each custom machine image packages a base operating system (OS) version and a Kubernetes version, along with any additional customizations, into an image that runs on vSphere, Microsoft Azure infrastructure, and AWS (EC2) environments.

A custom image must be based on the operating system (OS) versions that are supported by Tanzu Kubernetes Grid. The table below provides a list of the operating systems that are supported for building custom images for Tanzu Kubernetes Grid.

vSphere	AWS	Azure
Ubuntu 20.04 Ubuntu 18.04 RHEL 8 Photon OS 3 Windows 2019	Ubuntu 20.04 Ubuntu 18.04 Amazon Linux 2	Ubuntu 20.04 Ubuntu 18.04

For additional information on building custom images for Tanzu Kubernetes Grid, see Build Machine Images.

- Linux Custom Machine Images
- Windows Custom Machine Images

Compliance and Security

VMware published Tanzu Kubernetes releases (TKrs), along with compatible versions of Kubernetes and supporting components, use the latest stable and generally-available update of the OS version that they package. They contain all current CVE and USN fixes, as of the day that the image is built. The image files are signed by VMware and have file names that contain a unique hash identifier.

VMware provides FIPS-capable Kubernetes OVA, which can be used to deploy FIPS compliant Tanzu Kubernetes Grid management and workload clusters. Tanzu Kubernetes Grid core components such as Kubelet, Kube-apiserver, Kube-controller manager, Kube-proxy, Kube-scheduler, Kubectl, Etcd, Coredns, Containerd, and Cri-tool are made FIPS compliant by compiling them with the BoringCrypto FIPS modules, an open-source cryptographic library that provides FIPS 140-2 approved algorithms.

Supplemental Information

Cluster Deployment Parameters

```
# NSX Advanced Load Balancer details

AVI_CA_DATA_B64: # NSX Advanced Load Balancer Controller Certificate in base64 encoded
format.
AVI_CLOUD_NAME: # Name of the cloud that you created in your NSX Advanced Load Balance
r deployment.
AVI_CONTROL_PLANE_HA_PROVIDER: "true" # Set to true to enable NSX Advanced Load Balanc
er as the control plane API server endpoint
AVI_CONTROL_PLANE_NETWORK: # Optional. Defines the VIP network of the workload cluste
r’s control plane. Use when you want to configure a separate VIP network for the workl
oad clusters. This field is optional, and if it is left empty, it will use the same ne
twork as AVI_DATA_NETWORK.
AVI_CONTROL_PLANE_NETWORK_CIDR: # Optional. The CIDR of the subnet to use for the work
load cluster’s control plane. Use when you want to configure a separate VIP network fo
r the workload clusters. This field is optional, and if it is left empty, it will use
the same network as AVI_DATA_NETWORK_CIDR.
```

```
AVI_CONTROLLER: # The IP or hostname of the NSX Advanced Load Balancer controller.
AVI_DATA_NETWORK:  # The network which you want to use a VIP network for the applicati
ons deployed in the workload cluster.
AVI_DATA_NETWORK_CIDR: # The CIDR of the network that you have chosen for the applicat
ion load balancing in the workload cluster.
AVI_ENABLE: "true" # Enables NSX Advanced Load Balancer as a load balancer for workloa
ds.
AVI_LABELS: # Optional labels in the format key: value. When set, NSX Advanced Load Ba
lancer is enabled only on workload clusters that have this label.
AVI_MANAGEMENT_CLUSTER_CONTROL_PLANE_VIP_NETWORK_NAME: # The CIDR of the subnet to use
for the management cluster’s control plane. Use when you want to configure a separate
VIP network for the management cluster’s control plane. This field is optional, and if
it is left empty, it will use the same network as AVI_DATA_NETWORK_CIDR.
AVI_MANAGEMENT_CLUSTER_CONTROL_PLANE_VIP_NETWORK_CIDR: # The CIDR of the network that
you have chosen for the control plane HA of the management cluster.
AVI_MANAGEMENT_CLUSTER_SERVICE_ENGINE_GROUP: # Optional. Specifies the name of the Ser
vice Engine group that is to be used by AKO in the management cluster. This field is o
ptional, and if it is left empty, it will use the same network as AVI_SERVICE_ENGINE_G
ROUP.
AVI_MANAGEMENT_CLUSTER_VIP_NETWORK_NAME: # The network that you want to use as load ba
lancer network for any applications deployed in the shared services or management clus
ter.
AVI_MANAGEMENT_CLUSTER_VIP_NETWORK_CIDR: # Subnet CIDR of the VIP network choosen for
application load balancing in the shared services or management cluster.
AVI_NSXT_T1LR: # UUID of the tier-1 gateway in NSX where the logical segment chosen fo
r TKG management network is connected.
AVI_PASSWORD: # Password of the NSX ALB Controller admin user in th base 64 encoded fo
rmat
AVI_SERVICE_ENGINE_GROUP: # Name of the Service Engine Group configured in NSX ALB
AVI_SERVICE_ENGINE_GROUP: # Service Engine group name for the workload clusters.
AVI_USERNAME: admin

# Common Variables

CLUSTER_CIDR: # The CIDR range to use for pods.
SERVICE_CIDR: # The CIDR range to use for the Kubernetes services.
CLUSTER_NAME: # The name of the TKG Management Cluster that must comply with DNS hostn
ame requirements as outlined in https://datatracker.ietf.org/doc/html/rfc952
CLUSTER_PLAN: # Can be set to dev, prod or custom. The dev plan deploys a cluster with
a single control plane node. The prod plan deploys a highly available cluster with thr
ee control plane nodes.
ENABLE_AUDIT_LOGGING: # Audit logging for the Kubernetes API server. The default value
is false. To enable audit logging, set the variable to true.
ENABLE_CEIP_PARTICIPATION: #The default value is true. false opts out of the VMware Cu
stomer Experience Improvement Program.
ENABLE_MHC: "true/false" # When set to true, machine health checks are enabled for man
agement cluster control plane and worker nodes.
IDENTITY_MANAGEMENT_TYPE: <none/oidc/ldap> # Set oidc or ldap when enabling centralize
d authentication for management cluster access.
INFRASTRUCTURE_PROVIDER: # For vSphere platform set this value to vsphere.
DEPLOY_TKG_ON_VSPHERE7: "true" # Set this to true to deploy TKGm on vSphere.

# Node Configuration

OS_ARCH: amd64
OS_NAME: # Defaults to ubuntu for Ubuntu LTS. Can also be photon for Photon OS on vSph
ere
OS_VERSION: "3"
```

```
# Proxy Configuration

TKG_HTTP_PROXY_ENABLED: "true/false" # To send outgoing HTTP(S) traffic from the manag
ement cluster to a proxy, for example in an internet-restricted environment, set this
to true.
TKG_IP_FAMILY: ipv4
VSPHERE_CONTROL_PLANE_ENDPOINT: # If you use NSX Advanced Load Balancer, leave this fi
eld blank.

# Control Plane and Worker VM sizing

VSPHERE_CONTROL_PLANE_DISK_GIB: "40" # The size in gigabytes of the disk for the contr
ol plane node VMs. Include the quotes ("")
VSPHERE_CONTROL_PLANE_MEM_MIB: "16384" # The amount of memory in megabytes for the con
trol plane node VMs
VSPHERE_CONTROL_PLANE_NUM_CPUS: "4" # The number of CPUs for the control plane node VM
s. Include the quotes (""). Must be at least 2.
VSPHERE_WORKER_DISK_GIB: "40" # The size in gigabytes of the disk for the worker node
VMs. Include the quotes ("")
VSPHERE_WORKER_MEM_MIB: "16384" # The amount of memory in megabytes for the worker nod
e VMs. Include the quotes ("")
VSPHERE_WORKER_NUM_CPUS: "4" # The number of CPUs for the worker node VMs. Include the
quotes (””). Must be at least 2.

# vSphere Infrastructure details

VSPHERE_DATACENTER: # The name of the datacenter in which to deploy the TKG management
cluster.
VSPHERE_DATASTORE: # The name of the vSphere datastore where TKG cluster VMs will be s
tored.
VSPHERE_FOLDER: # The name of an existing VM folder in which to place TKG VMs.
VSPHERE_INSECURE: # Optional. Set to true or false to bypass thumbprint verification.
If false, set VSPHERE_TLS_THUMBPRINT
VSPHERE_NETWORK: # The name of an existing vSphere network where TKG management cluste
r control plane and worker VMs will be connected.
VSPHERE_PASSWORD: # The password for the vSphere user account in base64 encoded forma
t.
VSPHERE_RESOURCE_POOL: # The name of an existing resource pool in which to place TKG c
luster.
VSPHERE_SERVER: # The IP address or FQDN of the vCenter Server instance on which to de
ploy the Tanzu Kubernetes cluster.
VSPHERE_SSH_AUTHORIZED_KEY: # Paste in the contents of the SSH public key that you cre
ated in on the bootstrap machine.
VSPHERE_TLS_THUMBPRINT: # if VSPHERE_INSECURE is false. The thumbprint of the vCenter
Server certificate.
VSPHERE_USERNAME: # A vSphere user account, including the domain name, with the requir
ed privileges for Tanzu Kubernetes Grid operation
TKG_CUSTOM_IMAGE_REPOSITORY: # IP address or FQDN of your private registry.
TKG_CUSTOM_IMAGE_REPOSITORY_CA_CERTIFICATE: # Set if your private image registry uses
a self-signed certificate. Provide the CA certificate in base64 encoded format.
TKG_CUSTOM_IMAGE_REPOSITORY_SKIP_TLS_VERIFY: "rue/false" # Optional. Set to true if yo
ur private image registry uses a self-signed certificate and you do not use TKG_CUSTOM
_IMAGE_REPOSITORY_CA_CERTIFICATE. Because the Tanzu connectivity webhook injects the H
arbor CA certificate into cluster nodes, TKG_CUSTOM_IMAGE_REPOSITORY_SKIP_TLS_VERIFY s
hould always be set to false when using Harbor.
```

For a full list of configurable values, see Tanzu CLI Configuration File Variable Reference.

Configure Node Sizes

The Tanzu CLI creates the individual nodes of management clusters and Tanzu Kubernetes clusters according to the settings that you provide in the configuration file.

On vSphere, you can configure all node VMs to have the same predefined configurations, set different predefined configurations for control plane and worker nodes, or customize the configurations of the nodes. By using these settings, you can create clusters that have nodes with different configurations from the management cluster nodes. You can also create clusters in which the control plane nodes and worker nodes have different configurations.

Use Predefined Node Configurations

The Tanzu CLI provides the following predefined configurations for cluster nodes:

Size	CPU	Memory (in GB)	Disk (in GB)
Small	2	4	20
Medium	2	8	40
Large	4	16	40
Extra-large	8	32	80

To create a cluster in which all of the control plane and worker node VMs are the same size, specify the `SIZE` variable. If you set the `SIZE` variable, all nodes are created with the configuration that you set.

- `SIZE: "large"`

To create a cluster in which the control plane and worker node VMs are of different sizes, specify the `CONTROLPLANE_SIZE` and `WORKER_SIZE` options.

- `CONTROLPLANE_SIZE: "medium"`
- `WORKER_SIZE: "large"`

You can combine the `CONTROLPLANE_SIZE` and `WORKER_SIZE` options with the `SIZE` option. For example, if you specify `SIZE: "large"` with `WORKER_SIZE: "extra-large"`, the control plane nodes are set to `large` and worker nodes are set to `extra-large`.

- `SIZE: "large"`
- `WORKER_SIZE: "extra-large"`

Define Custom Node Configurations

You can customize the configuration of the nodes rather than using the predefined configurations.

To use the same custom configuration for all nodes, specify the `VSPHERE_NUM_CPUS`, `VSPHERE_DISK_GIB`, and `VSPHERE_MEM_MIB` options.

- `VSPHERE_NUM_CPUS: 2`
- `VSPHERE_DISK_GIB: 40`
- `VSPHERE_MEM_MIB: 4096`

To define different custom configurations for control plane nodes and worker nodes, specify the `VSPHERE_CONTROL_PLANE_*` and `VSPHERE_WORKER_*`

- `VSPHERE_CONTROL_PLANE_NUM_CPUS: 2`
- `VSPHERE_CONTROL_PLANE_DISK_GIB: 20`
- `VSPHERE_CONTROL_PLANE_MEM_MIB: 8192`
- `VSPHERE_WORKER_NUM_CPUS: 4`
- `VSPHERE_WORKER_DISK_GIB: 40`
- `VSPHERE_WORKER_MEM_MIB: 4096`

NSX Advanced Load Balancer Sizing Guidelines

NSX ALB Controller Sizing Guidelines

Regardless of NSX Advanced Load Balancer Controller configuration, each controller cluster can achieve up to 5000 virtual services, which is a hard limit. For more information, see Sizing Compute and Storage Resources for NSX Advanced Load Balancer Controller(s).

Controller Size	VM Configuration	Virtual Services	NSX Advanced Load Balancer SE Scale
Essentials	4 vCPUs, 24 GB RAM	0-50	0-10
Small	6 vCPUs, 24 GB RAM	0-200	0-100
Medium	10 vCPUs, 32 GB RAM	200-1000	100-200
Large	16 vCPUs, 48 GB RAM	1000-5000	200-400

Service Engine Sizing Guidelines

For guidance on sizing your service engines (SEs), see Sizing Compute and Storage Resources for NSX Advanced Load Balancer Service Engine(s).

Performance metric	1 vCPU core
Throughput	4 Gb/s
Connections/s	40k
SSL Throughput	1 Gb/s
SSL TPS (RSA2K)	~600
SSL TPS (ECC)	2500

Multiple performance vectors or features may have an impact on performance. For instance, to achieve 1 Gb/s of SSL throughput and 2000 TPS of SSL with EC certificates, NSX ALB recommends two cores.

NSX ALB Service Engines may be configured with as little as 1 vCPU core and 1 GB RAM, or up to 36 vCPU cores and 128 GB RAM. Service Engines can be deployed in Active/Active or Active/Standby mode depending on the license tier used. NSX ALB Essentials license doesn’t support Active/Active HA mode for SE.

Summary

Tanzu Kubernetes Grid on vSphere on hyper-converged hardware offers high-performance potential, convenience, and addresses the challenges of creating, testing, and updating on-premises Kubernetes platforms in a consolidated production environment. This validated approach will result in a near-production quality installation with all the application services needed to serve combined or uniquely separated workload types through a combined infrastructure solution.

This plan meets many Day-0 needs for quickly aligning product capabilities to full stack infrastructure, including networking, firewalling, load balancing, workload compute alignment, and other capabilities.

Deploy Tanzu Kubernetes Grid on vSphere with NSX-T Networking in Air-gapped Environment

VMware Tanzu Kubernetes Grid (informally known as TKG) (multi-cloud) provides organizations with a consistent, upstream-compatible, regional Kubernetes substrate that is ready for end-user workloads and ecosystem integrations. It delivers an open source aligned Kubernetes distribution with consistent operations and management to support infrastructure and app modernization.

An air-gapped installation method is used when the Tanzu Kubernetes Grid components (bootstrapper and cluster nodes) are unable to connect to the Internet to download the installation binaries from the public VMware Registry during Tanzu Kubernetes Grid installation or upgrade.

The scope of the document is limited to providing deployment steps based on the reference design in Tanzu Kubernetes Grid on NSX-T Networking and it does not cover deployment procedures for the underlying SDDC components.

Supported Component Matrix

The following table provides the component versions and interoperability matrix supported with the reference design:

Software Components	Version
Tanzu Kubernetes Grid	2.3.0
VMware vSphere ESXi	8.0 U1 or later
VMware vCenter (VCSA)	8.0 U1 or later
NSX Advanced Load Balancer	22.1.3
VMware NSX	4.1.0.2

For the latest information about software versions that can be used together, see the Interoperability Matrix.

Prepare the Environment for Deployment of Tanzu Kubernetes Grid

Before deploying Tanzu Kubernetes Grid in the your VMware NSX environment, ensure that your environment is set up as described in the following sections:

- General Requirements

- Network Requirements
- Firewall Requirements

General Requirements

- A vCenter with NSX backed environment.
- Ensure that the following NSX configurations are complete:

  > The following configurations provide only a high-level overview of the required NSX configuration. For more information, see NSX Data Center Installation Guide and NSX Data Center Product Documentation.

  - NSX manager instance is deployed and configured with Advanced or higher license.
  - vCenter Server that is associated with the NSX Data Center is configured as Compute Manager.
  - Required overlay and vLAN Transport Zones are created.
  - IP pools for host and edge tunnel endpoints (TEP) are created.
  - Host and edge uplink profiles are in place.
  - Transport node profiles are created. This is not required if you are configuring the NSX data center on each host instead of the cluster.
  - NSX data center configured on all hosts part of the vSphere cluster or clusters.
  - Edge transport nodes and at least one edge cluster is created.
  - Tier-0 uplink segments and tier-0 gateway is created.
  - Tier-0 router is peered with uplink L3 switch.
  - DHCP profile is created in NSX.
- SDDC environment has the following objects are available:
  - A vSphere cluster with at least three hosts on which vSphere DRS is enabled and NSX is successfully configured.
  - A dedicated resource pool to deploy the following Tanzu Kubernetes management cluster, shared services cluster, and workload clusters. The number of required resource pools depends on the number of workload clusters to be deployed.
  - VM folders to collect the Tanzu Kubernetes Grid VMs.
  - A datastore with sufficient capacity for the control plane and worker node VM files.
  - Network time protocol (NTP) service is running on all hosts and vCenter.
  - A host, server, or VM based on Linux, macOS, or Windows which acts as your bootstrap machine which has docker installed. For this deployment, a virtual machine based on Photon OS will be used.
  - Depending on the OS flavor of the bootstrap VM, download and configure the following packages from Broadcom Support. To configure required packages on the Cent OS

machine, see Deploy and Configure Bootstrap Machine:

- Tanzu CLI 2.3.0
- Kubectl cluster CLI 1.26.5
- A vSphere account with permissions as described in Required Permissions for the vSphere Account.
- Download and import NSX Advanced Load Balancer 22.1.3 OVA to Content Library.
- Download the following OVA files from Broadcom Support and import to vCenter. Convert the imported VMs to templates:
- Photon v3 Kubernetes v1.26.5 OVA and/or
- Ubuntu 2004 Kubernetes v1.26.5 OVA

> You can also download supported older versions of Kubernetes from Broadcom Support and import them to deploy workload clusters on the intended Kubernetes versions.
>
> In Tanzu Kubernetes Grid nodes, it is recommended not to use hostnames with “.local” domain suffix. For more information, see KB article.

Resource Pools and VM Folders

The sample entries of the resource pools and folders that need to be created are as follows.

Resource Type	Sample Resource Pool Name	Sample Folder Name
NSX ALB Components	`tkg-vsphere-alb-components`	`tkg-vsphere-alb-components`
TKG Management components	`tkg-management-components`	`tkg-management-components`
TKG Shared Service Components	`tkg-vsphere-shared-services`	`tkg-vsphere-shared-services`
TKG Workload components	`tkg-vsphere-workload`	`tkg-vsphere-workload`

Network Requirements

Create logical segments in NSX for deploying Tanzu Kubernetes Grid components as per Network Requirements defined in the reference architecture.

Firewall Requirements

Ensure that the firewall is set up as described in Firewall Requirements.

Subnet and CIDR Examples

For this demonstration, we used the following CIDR for Tanzu Kubernetes Grid deployment. Change the values to reflect your environment:

Network Type	Segment Name	Gateway CIDR	DHCP Pool in NSXT	NSX ALB IP Pool
NSX ALB Management Network	sfo01-w01-vds01-albmanagement	172.16.10.1/24	N/A	172.16.10.100 - 172.16.10.200
TKG Cluster VIP Network	sfo01-w01-vds01-tkgclustervip	172.16.80.1/24	N/A	172.16.80.100 - 172.16.80.200
TKG Management Network	sfo01-w01-vds01-tkgmanagement	172.16.40.1/24	172.16.40.100 - 172.16.40.200	N/A
TKG Shared Service Network	sfo01-w01-vds01-tkgshared	172.16.50.1/24	172.16.50.100 - 172.16.50.200	N/A
TKG Workload Network	sfo01-w01-vds01-tkgworkload	172.16.60.1/24	172.16.60.100-172.16.60.200	N/A
TKG Workload VIP Network	sfo01-w01-vds01-tkgworkloadvip	172.16.70.1/24	N/A	172.16.70.100-172.16.70.200

Tanzu Kubernetes Grid Deployment Workflow

Here are the high-level steps for deploying Tanzu Kubernetes Grid on NSX networking in an air-gapped environment:

- Configure T1 Gateway and Logical Segments in NSX Data Center
- Deploy and Configure NSX Advanced Load Balancer
- Configure Bastion Host
- Install Harbor Image Registry
- Configure Bootstrap Virtual machine
- Deploy Tanzu Kubernetes Grid Management Cluster
- Deploy Tanzu Kubernetes Grid Shared Service Cluster
- Deploy Tanzu Kubernetes Grid Workload Cluster
- Deploy User-Managed Packages on Tanzu Kubernetes Grid Clusters

Configure T1 Gateway and Logical Segments in NSX-T Data Center

As a prerequisite, an NSX-T backed vSphere environment must be configured with at least one tier-0 gateway. A tier-0 gateway performs the functions of a tier-0 logical router. It processes traffic between the logical and physical networks. For more information about creating and configuring a tier-0 gateway, see NSX documentation.

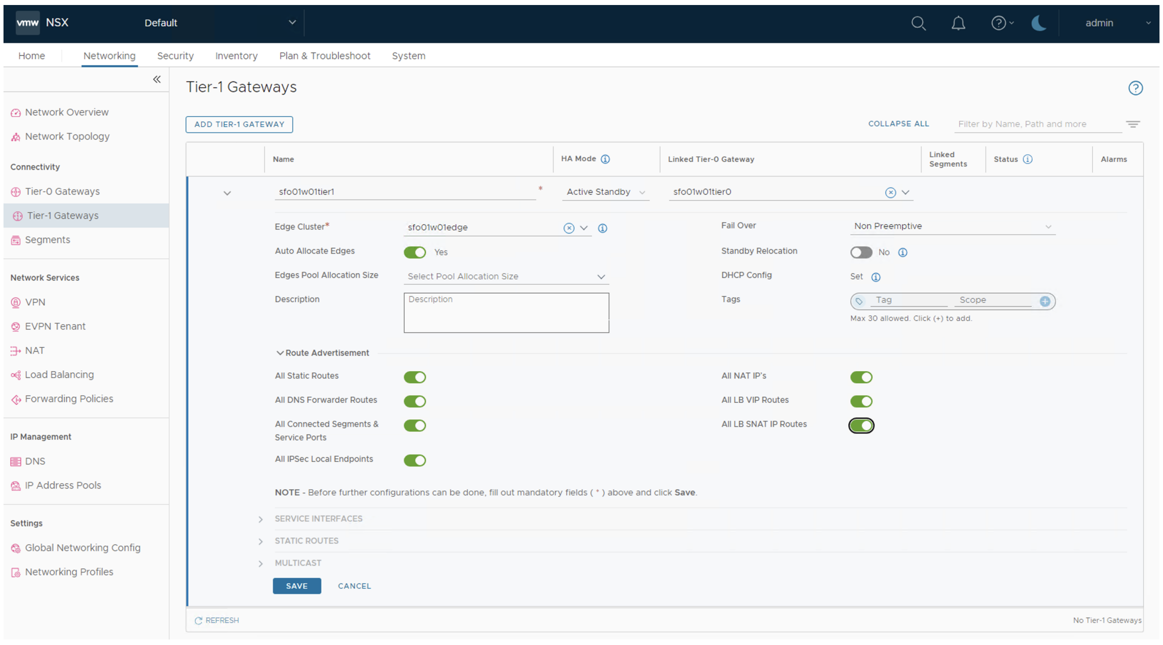
This procedure comprises the following tasks:

1. Add two Tier-1 Gateway
2. Create Overlay-Backed Segments

Add a Tier-1 Gateway

The tier-1 logical router must be connected to the tier-0 logical router to get the northbound physical router access. The following procedure provides the minimum required configuration to create a tier-1 gateway, which is adequate to successfully deploy the Tanzu for Kubernetes Operations stack. For a more advanced configuration, see the NSX documentation.

1. With admin privileges, log in to NSX Manager.
2. Select **Networking** > **Tier-1 Gateways**.
3. Click **Add Tier-1 Gateway**.
4. Enter a name for the gateway.
5. Select a tier-0 gateway to connect to this tier-1 gateway to create a multi-tier topology.
6. Select an NSX Edge cluster. This is required for this tier-1 gateway to host stateful services such as NAT, load balancer, or firewall.
7. (Optional) In the **Edges** field, select **Auto Allocated** or manually set the edge nodes.
8. Select a failover mode or accept the default. The default option is **Non-preemptive**.
9. Select **Enable Standby Relocation**.
10. Click **Route Advertisement** and ensure that following routes are selected:
    - **All DNS Forwarder Routes**
    - **All Connected Segments and Service Ports**
    - **All IPSec Local Endpoints**
    - **All LB VIP Routes**
    - **All LB SNAT IP Routes**



Tier-1 Gateways

Name	HA Mode	Linked Tier-0 Gateway	Linked Segments	Status	Alarms
sfo01w01tier1 *	Active Standby	sfo01w01tier0			

Edge Cluster*	sfo01w01edge	Fail Over	Non Preemptive
Auto Allocate Edges	Yes	Standby Relocation	No
Edges Pool Allocation Size	Select Pool Allocation Size	DHCP Config	Set
Description	Description	Tags	Tag / Scope — Max 30 allowed. Click (+) to add.

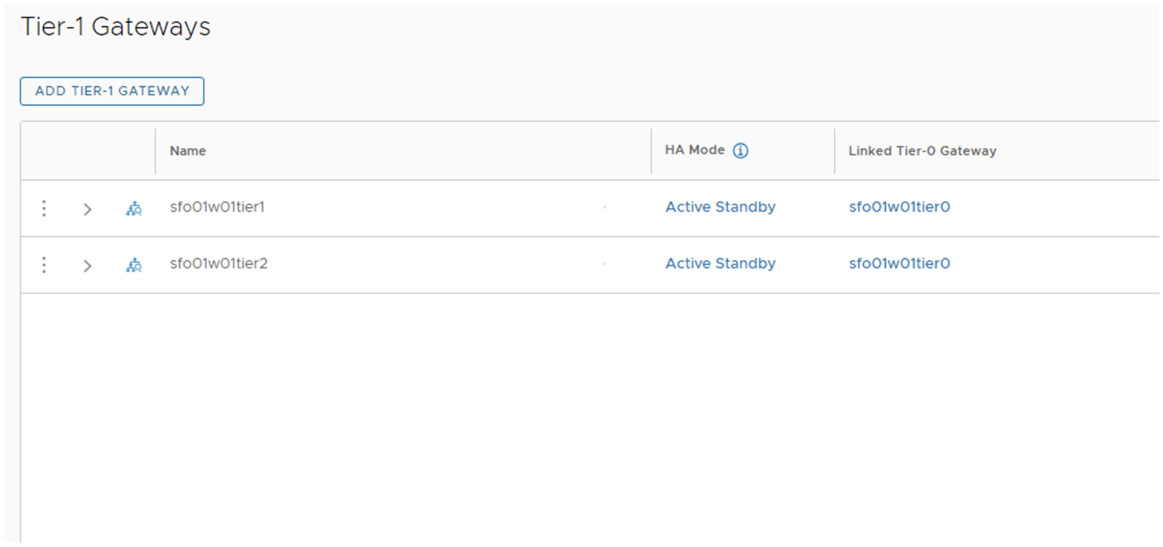
Route Advertisement

All Static Routes	✓	All NAT IP's	✓
All DNS Forwarder Routes	✓	All LB VIP Routes	✓
All Connected Segments & Service Ports	✓	All LB SNAT IP Routes	✓
All IPSec Local Endpoints	✓		

NOTE - Before further configurations can be done, fill out mandatory fields (*) above and click **Save**.

No Tier-1 Gateways

11. Click **Save**.
12. Repeat steps from 1-11 and create another **Tier-1** gateway.

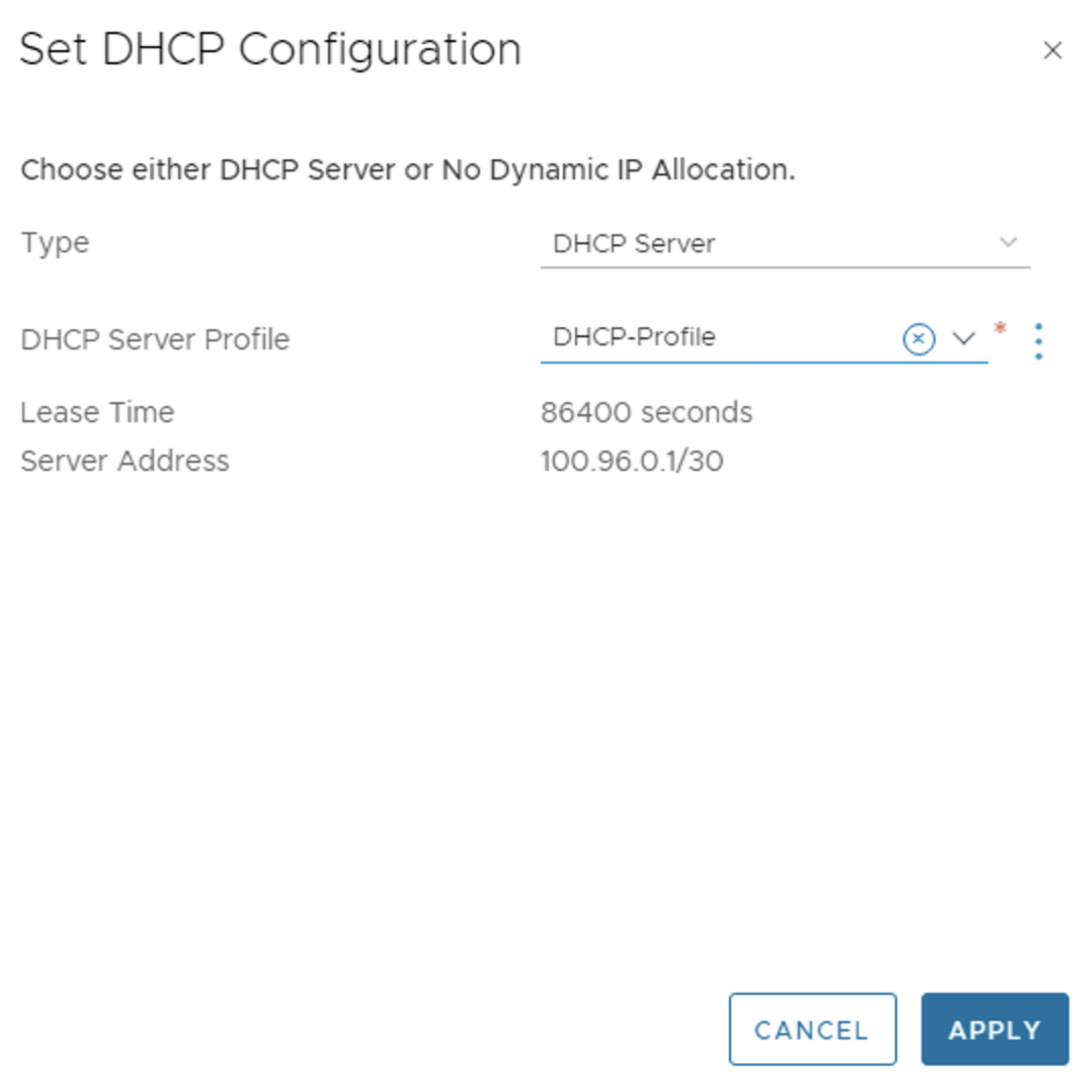


Name	HA Mode	Linked Tier-0 Gateway
sfo01w01tier1	Active Standby	sfo01w01tier0
sfo01w01tier2	Active Standby	sfo01w01tier0

DHCP configuration on Tier-1 Gateway

Complete the following steps to set the DHCP configuration in the tier-1 gateway:

1. With admin privileges, log in to NSX Manager.
2. Select **Networking** > **Tier-1 Gateways**.
3. On the tier-1 gateway that you created earlier, click the three dots and select **Edit**.
4. Next to DHCP Config, click **Set**.
5. In the Set DHCP Configuration dialog box, set **Type** to DHCP Server and select the DHCP profile that you created as part of the prerequisites.


Set DHCP Configuration

Choose either DHCP Server or No Dynamic IP Allocation.

Type: DHCP Server

DHCP Server Profile: DHCP-Profile

Lease Time: 86400 seconds

Server Address: 100.96.0.1/30

6. Click **Save**.

Create Overlay-Backed Segments

VMware NSX provides the option to add two kinds of segments: overlay-backed segments and VLAN-backed segments. Segments are created as part of a transport zone. There are two types of transport zones: VLAN transport zones and overlay transport zones. A segment created in a VLAN transport zone is a VLAN-backed segment and a segment created in an overlay transport zone is an overlay-backed segment.

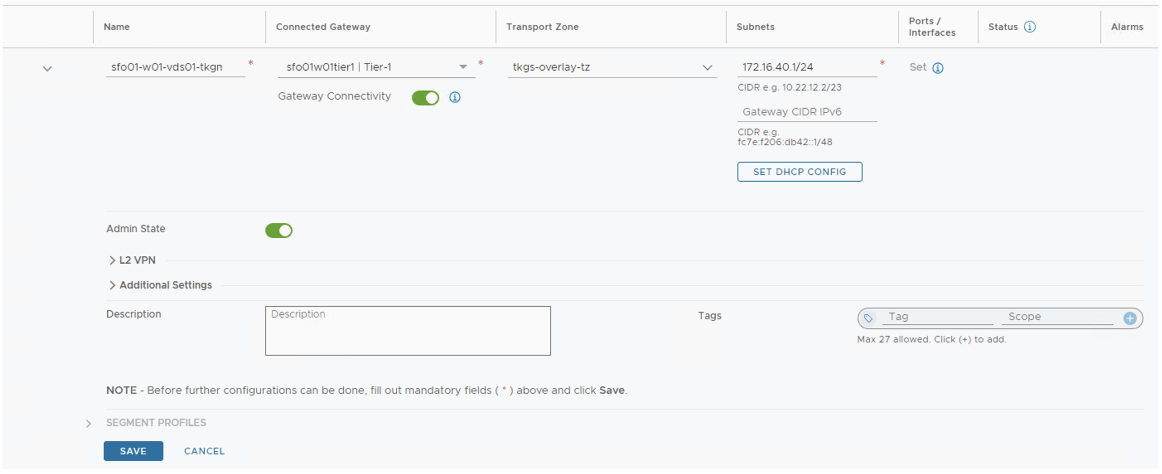
Create the overlay backed logical segments as shown in the Overlay backed segments CIDR example. All these segments will be a part of the same overlay transport zone and they must be connected to the tier-1 gateway.

NSX ALB Management Network, TKG Cluster VIP Network, TKG Management Network & TKG Shared Service Network must be connected to **sfo01w01tier1** while TKG Workload Network and TKG Workload VIP Network should be connected to **sfo01w01tier2**.

If you want to use TKG Cluster VIP Network to be used for applications deployed in workload cluster, connect all network segments to **sfo01w01tier1** tier-1 gateway.

The following procedure provides details to create one such network which is required for the Tanzu for Kubernetes Operations deployment:

1. With admin privileges, log in to NSX Manager.
2. Select **Networking** > **Segments**.
3. Click **ADD SEGMENT** and enter a name for the segment. For example, `sfo01-w01-vds01-tkgmanagement`.
4. Under **Connected Gateway**, select the tier-1 gateway that you created earlier.
5. Under **Transport Zone**, select a transport zone that will be an overlay transport zone.
6. Under **Subnets**, enter the gateway IP address of the subnet in the CIDR format. For example, `172.16.40.1/24`



Name	Connected Gateway	Transport Zone	Subnets	Ports / Interfaces	Status	Alarms
sfo01-w01-vds01-tkgn *	sfo01w01tier1 | Tier-1 *	tkgs-overlay-tz	172.16.40.1/24 *	Set		
	Gateway Connectivity		CIDR e.g. 10.22.12.2/23			
			Gateway CIDR IPv6			
			CIDR e.g. fc7e:f206:db42::1/48			

Admin State

> L2 VPN

> Additional Settings

Description: Description

Tags: Tag Scope

Max 27 allowed. Click (+) to add.

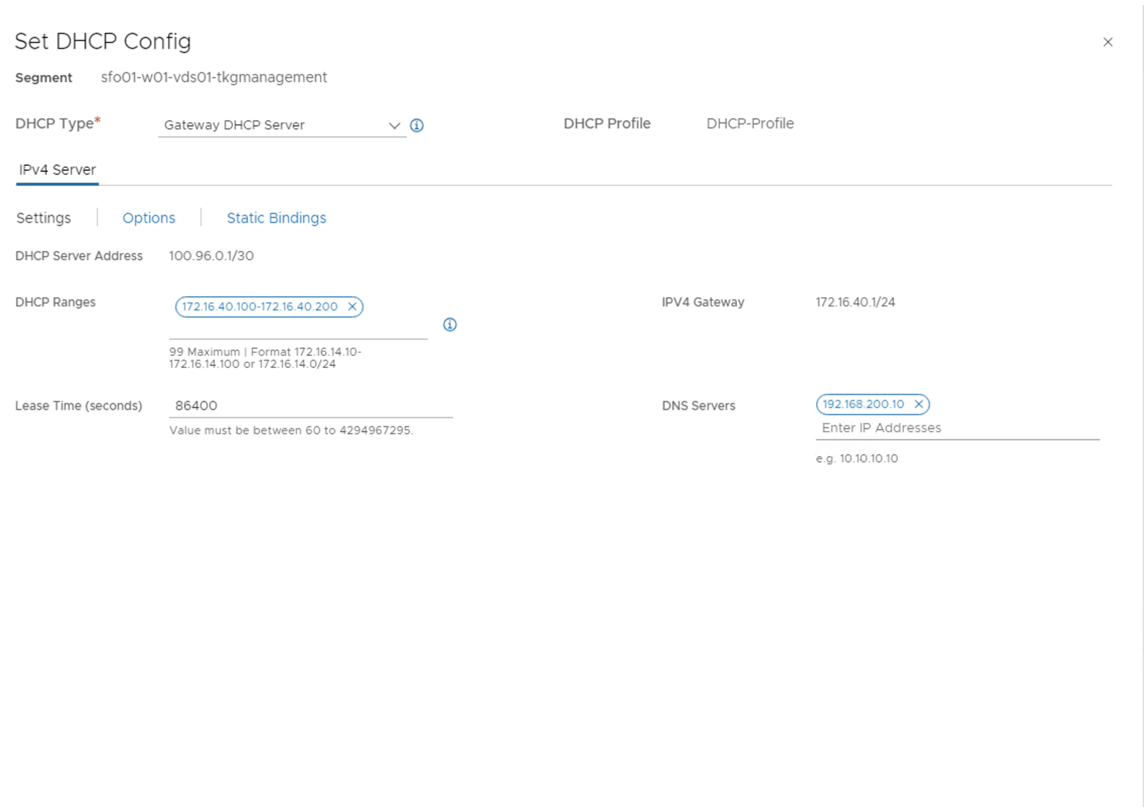
NOTE - Before further configurations can be done, fill out mandatory fields (*) above and click **Save**.

The following step is required only for Tanzu Kubernetes Grid management network, shared services network, and workload network.

7. Click **SET DHCP CONFIG**.

   DHCP Type field is set to Gateway DHCP Server and **DHCP Profile** is set to the profile created while creating the tier-1 gateway.

   1. Click **Settings**, select **Enable DHCP Config**, and enter the DHCP range and DNS server information.



Set DHCP Config

Segment sfo01-w01-vds01-tkgmanagement

DHCP Type* Gateway DHCP Server | DHCP Profile DHCP-Profile

IPv4 Server

Settings | Options | Static Bindings

DHCP Server Address 100.96.0.1/30

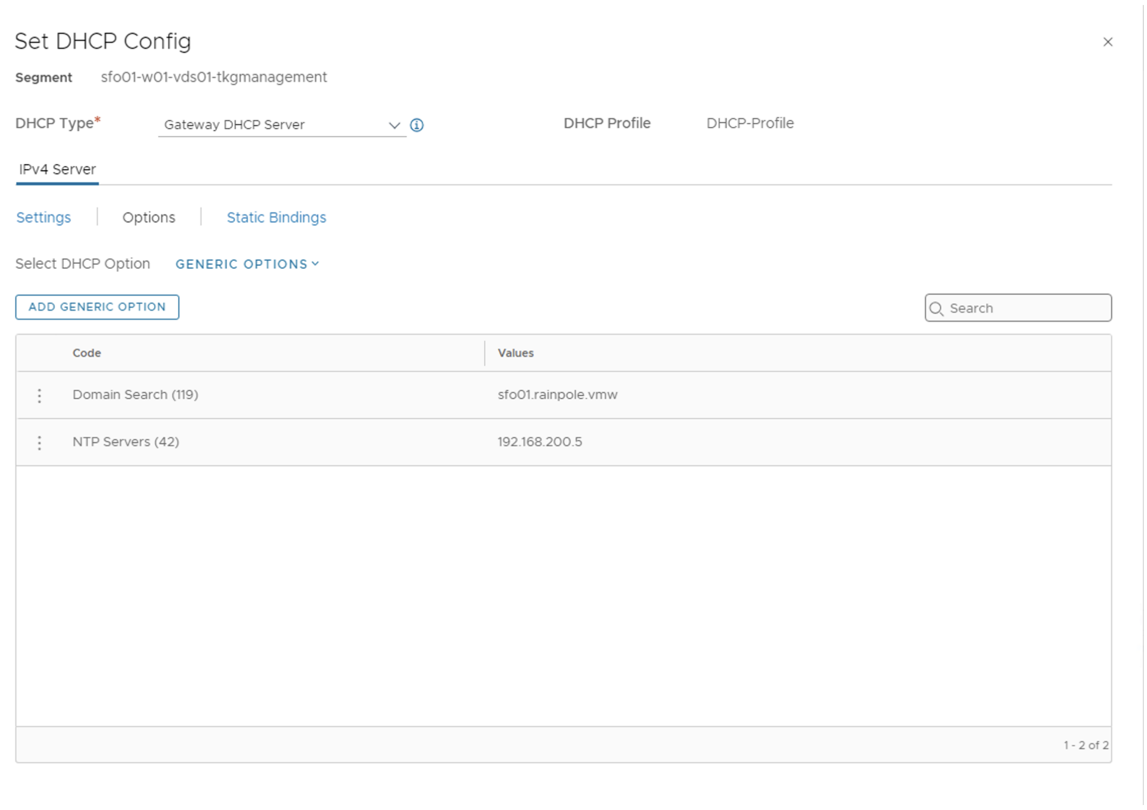
DHCP Ranges 172.16.40.100-172.16.40.200
99 Maximum | Format 172.16.14.10-172.16.14.100 or 172.16.14.0/24

IPV4 Gateway 172.16.40.1/24

Lease Time (seconds) 86400
Value must be between 60 to 4294967295.

DNS Servers 192.168.200.10
Enter IP Addresses
e.g. 10.10.10.10

1. Click **Options** and under **Select DHCP Options**, select **GENERIC OPTIONS**.
2. Click **ADD GENERIC OPTION**, add **NTP servers (42)** and **Domain Search (119)**.



Set DHCP Config

Segment sfo01-w01-vds01-tkgmanagement

DHCP Type* Gateway DHCP Server | DHCP Profile DHCP-Profile

IPv4 Server

Settings | Options | Static Bindings

Select DHCP Option GENERIC OPTIONS

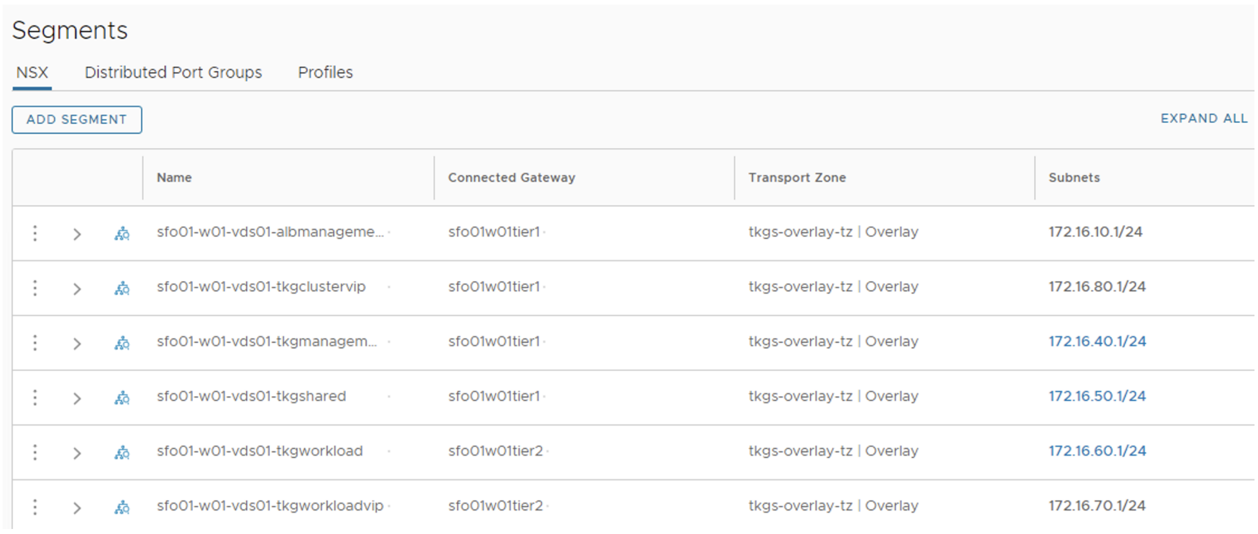
ADD GENERIC OPTION

Code	Values
Domain Search (119)	sfo01.rainpole.vmw
NTP Servers (42)	192.168.200.5

1 - 2 of 2

1. Click **Save** to create the logical segment.

Repeat steps 1-7 to create all other required overlay-backed segments. Once completed, you should see an output similar to:



Name	Connected Gateway	Transport Zone	Subnets
sfo01-w01-vds01-albmanageme...	sfo01w01tier1	tkgs-overlay-tz | Overlay	172.16.10.1/24
sfo01-w01-vds01-tkgclustervip	sfo01w01tier1	tkgs-overlay-tz | Overlay	172.16.80.1/24
sfo01-w01-vds01-tkgmanagem...	sfo01w01tier1	tkgs-overlay-tz | Overlay	172.16.40.1/24
sfo01-w01-vds01-tkgshared	sfo01w01tier1	tkgs-overlay-tz | Overlay	172.16.50.1/24
sfo01-w01-vds01-tkgworkload	sfo01w01tier2	tkgs-overlay-tz | Overlay	172.16.60.1/24
sfo01-w01-vds01-tkgworkloadvip	sfo01w01tier2	tkgs-overlay-tz | Overlay	172.16.70.1/24

Additionally, you can create the required inventory groups and firewall rules. For more information, see the NSX Data Center Product Documentation.

Deploy and Configure NSX Advanced Load Balancer

NSX Advanced Load Balancer (ALB) is an enterprise-grade integrated load balancer that provides L4- L7 load balancer support.

NSX Advanced Load Balancer is deployed in Write Access Mode in the vSphere Environment backed by NSX-T. This mode grants NSX Advanced Load Balancer controllers full write access to the vCenter or NSX which helps in automatically creating, modifying, and removing service engines (SEs) and other resources as needed to adapt to changing traffic needs.

The sample IP address and FQDN set for the NSX Advanced Load Balancer controllers are as follows:

Controller Node	IP Address	FQDN
Node 1 Primary	172.16.10.11	`sfo01albctlr01a.sfo01.rainpole.local`
Node 2 Secondary	172.16.10.12	`sfo01albctlr01b.sfo01.rainpole.local`
Node 3 Secondary	172.16.10.13	`sfo01albctlr01c.sfo01.rainpole.local`
HA Address	172.16.10.10	`sfo01albctlr01.sfo01.rainpole.local`

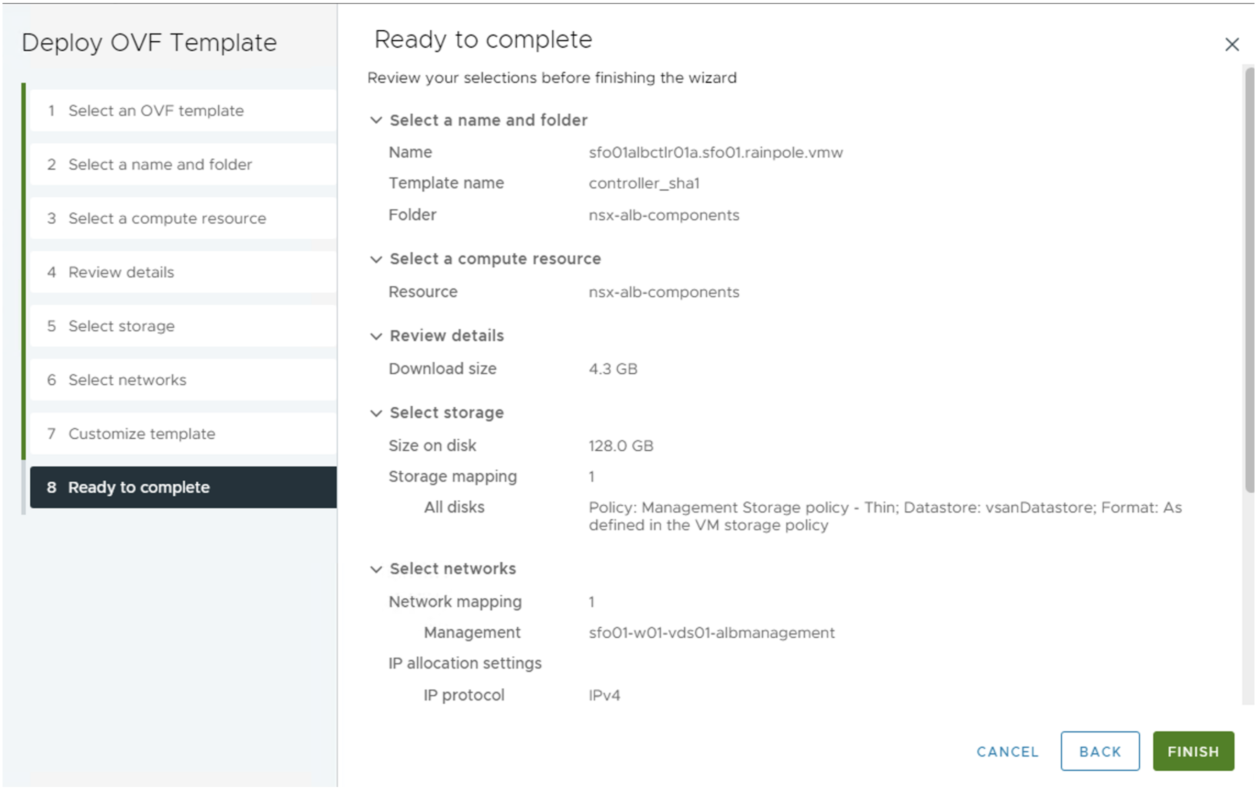
Deploy NSX Advanced Load Balancer

As a prerequisite, you must have the NSX Advanced Load Balancer 22.1.3 OVA downloaded and imported to the content library. Deploy the NSX Advanced Load Balancer under the resource pool **tkg-vsphere-alb-components** and place it under the folder **tkg-vsphere-alb-components**.

To deploy NSX Advanced Load Balancer, complete the following steps:

1. Log in to **vCenter** and navigate to **Home** > **Content Libraries**.
2. Select the content library under which the NSX-ALB OVA is placed.
3. Click on **OVA & OVF Templates**.
4. Right-click the NSX Advanced Load Balancer image and select **New VM from this Template**.

5. On the Select name and folder page, enter a name and select a folder for the NSX Advanced Load Balancer VM as **tkg-vsphere-alb-components**.
6. On the Select a compute resource page, select the resource pool **tkg-vsphere-alb-components**.
7. On the Review details page, verify the template details and click **Next**.
8. On the Select storage page, select a storage policy from the VM Storage Policy drop-down menu and choose the datastore location where you want to store the virtual machine files.
9. On the Select networks page, select the network **sfo01-w01-vds01-albmanagement** and click **Next**.
10. On the Customize template page, provide the NSX Advanced Load Balancer management network details such as IP address, subnet mask, and gateway, and then click **Next**.
11. On the Ready to complete page, review the provided information and click **Finish**.



Ready to complete

Review your selections before finishing the wizard

Select a name and folder

Name	sfo01albctlr01a.sfo01.rainpole.vmw
Template name	controller_sha1
Folder	nsx-alb-components

Select a compute resource

Resource	nsx-alb-components

Review details

Download size	4.3 GB

Select storage

Size on disk	128.0 GB
Storage mapping	1
All disks	Policy: Management Storage policy - Thin; Datastore: vsanDatastore; Format: As defined in the VM storage policy

Select networks

Network mapping	1
Management	sfo01-w01-vds01-albmanagement
IP allocation settings	
IP protocol	IPv4

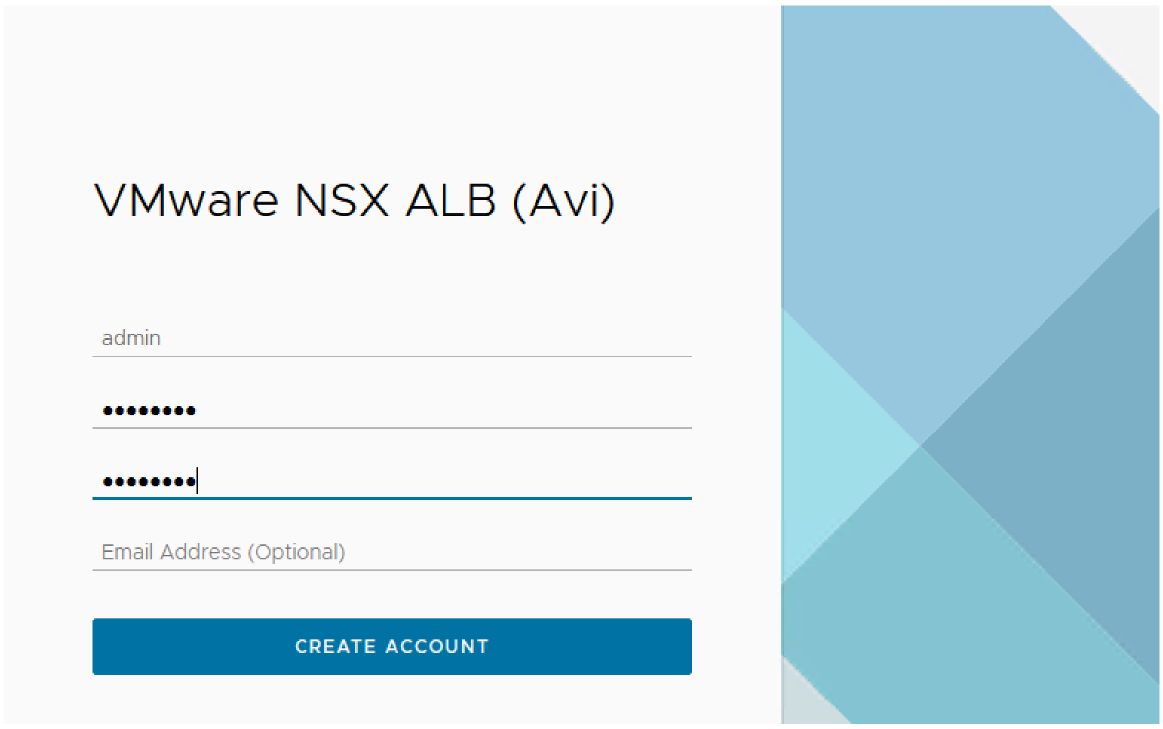
A new task for creating the virtual machine appears in the **Recent Tasks** pane. After the task is complete, the NSX Advanced Load Balancer virtual machine is created on the selected resource. Power on the virtual machine and give it a few minutes for the system to boot. Upon successful boot up, navigate to NSX Advanced Load Balancer on your browser.

While the system is booting up, a blank web page or a 503 status code might appear.

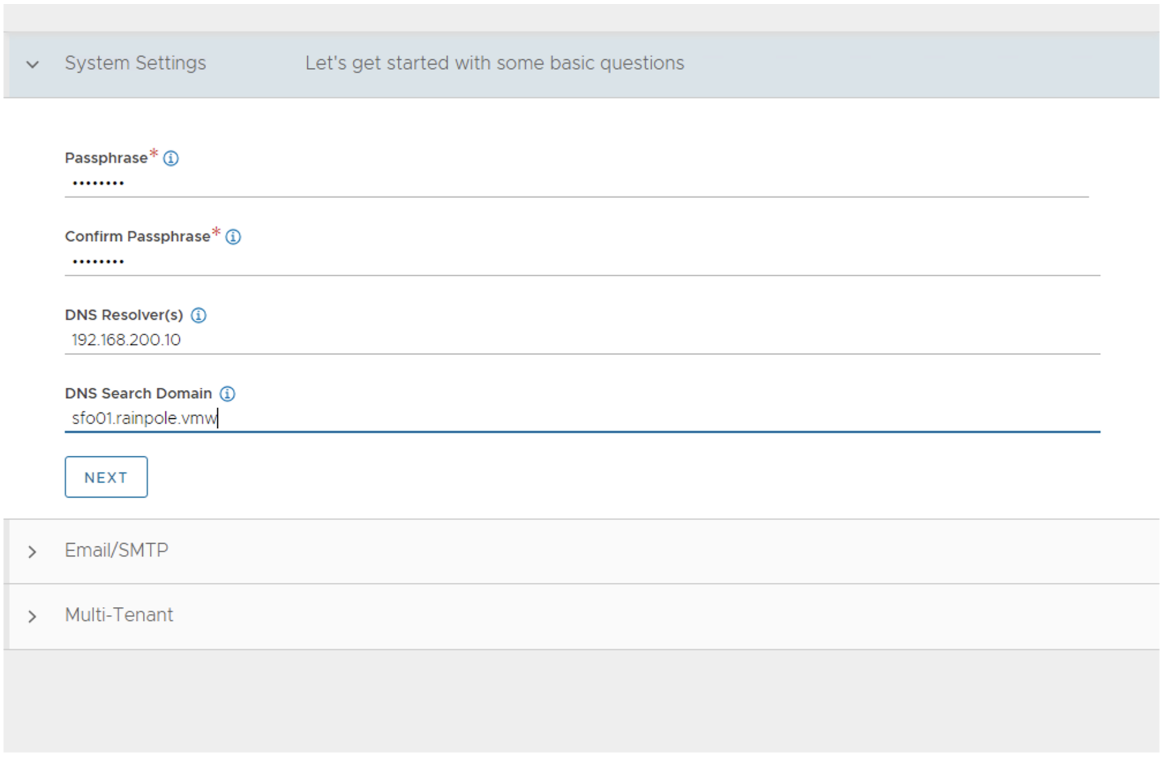
NSX Advanced Load Balancer: Initial Setup

Once NSX Advanced Load Balancer is successfully deployed and running, navigate to NSX Advanced Load Balancer on your browser using the URL https://*<IP/FQDN>* and configure the basic system settings:

1. Set admin password and click on **Create Account**.



2. On the Welcome page, under **System Settings**, set backup passphrase and provide DNS information, and then click **Next**.



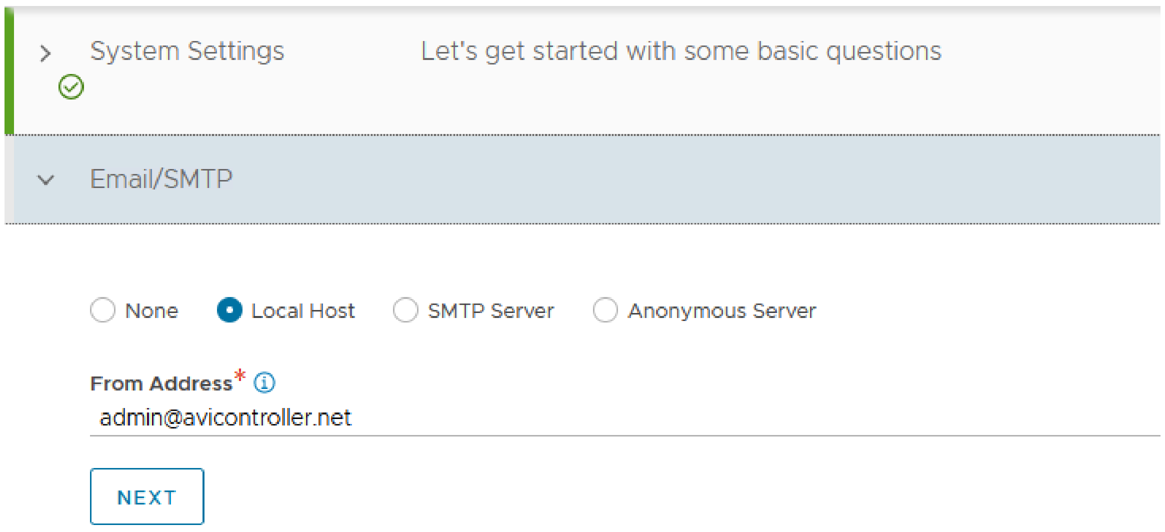
Passphrase* ••••••••

Confirm Passphrase* ••••••••

DNS Resolver(s) 192.168.200.10

DNS Search Domain sfo01.rainpole.vmw

3. Under **Email/SMTP**, provide email and SMTP information, and then click **Next**.

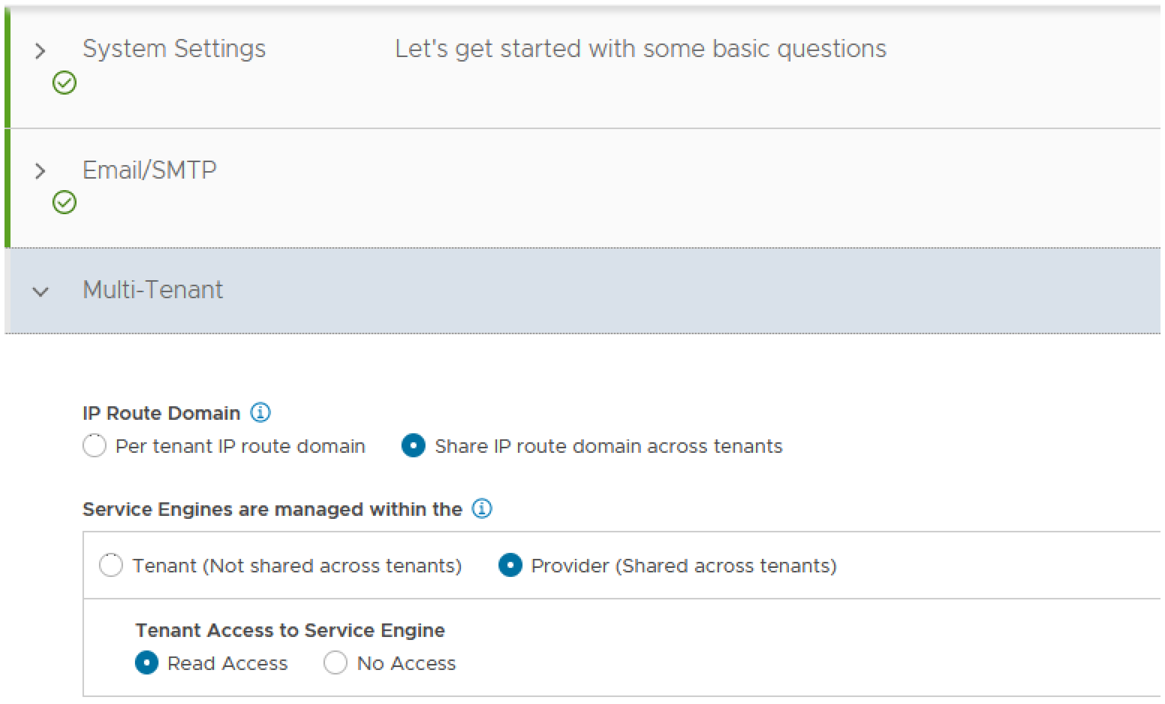

System Settings — Let's get started with some basic questions

Email/SMTP

None [✓ Local Host] SMTP Server Anonymous Server

From Address* admin@avicontroller.net

4. Under **Multi-Tenant**, configure settings as follows and click **Save**.
   - IP Route Domain: Share IP route domain across tenants
   - Service Engines are managed within the: Provider (Shared across tenants)
   - Tenant Access to Service Engine: Read


System Settings — Let's get started with some basic questions

Email/SMTP

Multi-Tenant

IP Route Domain: Per tenant IP route domain [✓ Share IP route domain across tenants]

Service Engines are managed within the: Tenant (Not shared across tenants) [✓ Provider (Shared across tenants)]

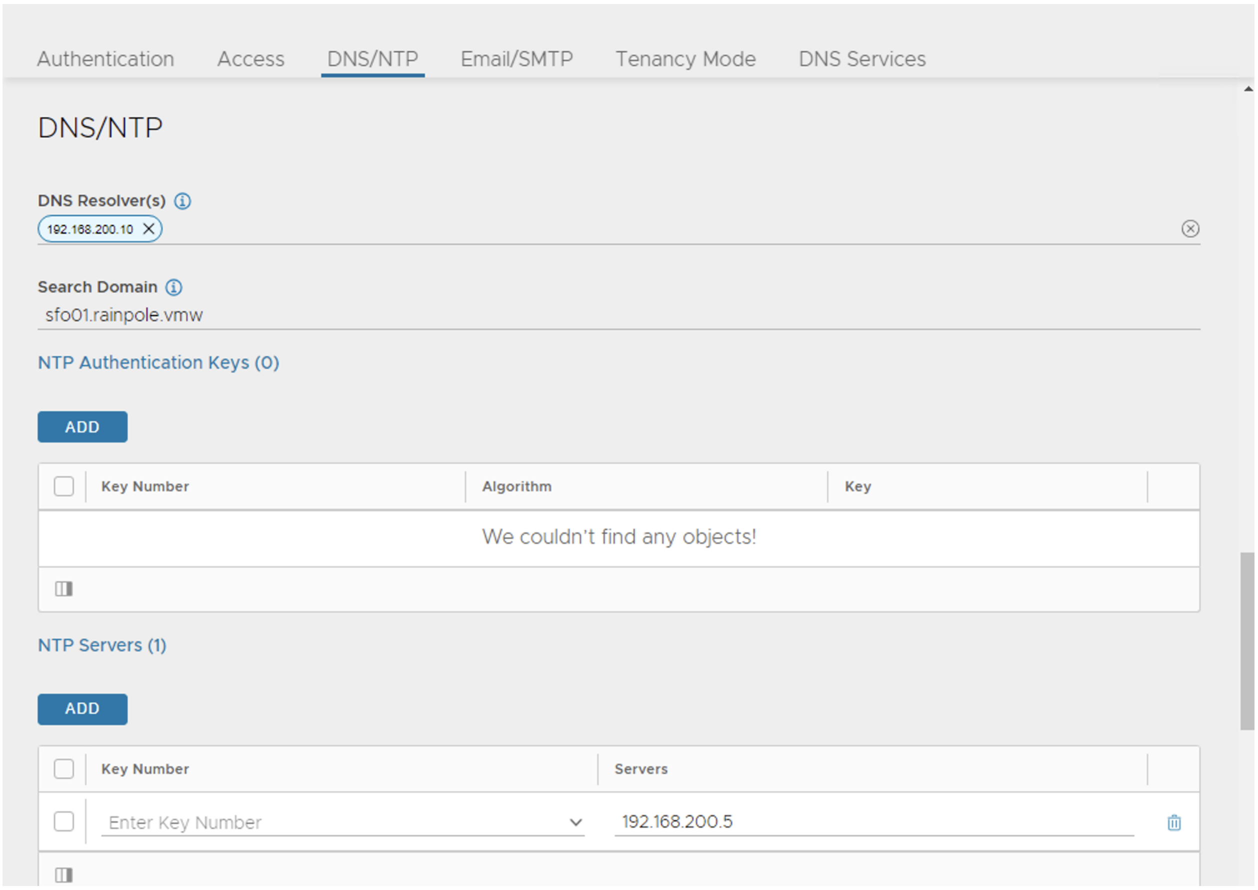
Tenant Access to Service Engine: [✓ Read Access] No Access

If you did not select the **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.

NSX Advanced Load Balancer: NTP Configuration

To configure NTP, navigate to **Administration** > **System Settings** > edit the System Settings and select **DNS/NTP**. Add your NTP server details and then click **Save.**

You might also delete the default NTP servers.



DNS/NTP

DNS Resolver(s): 192.168.200.10

Search Domain: sfo01.rainpole.vmw

NTP Authentication Keys (0)

Key Number	Algorithm	Key
We couldn't find any objects!		

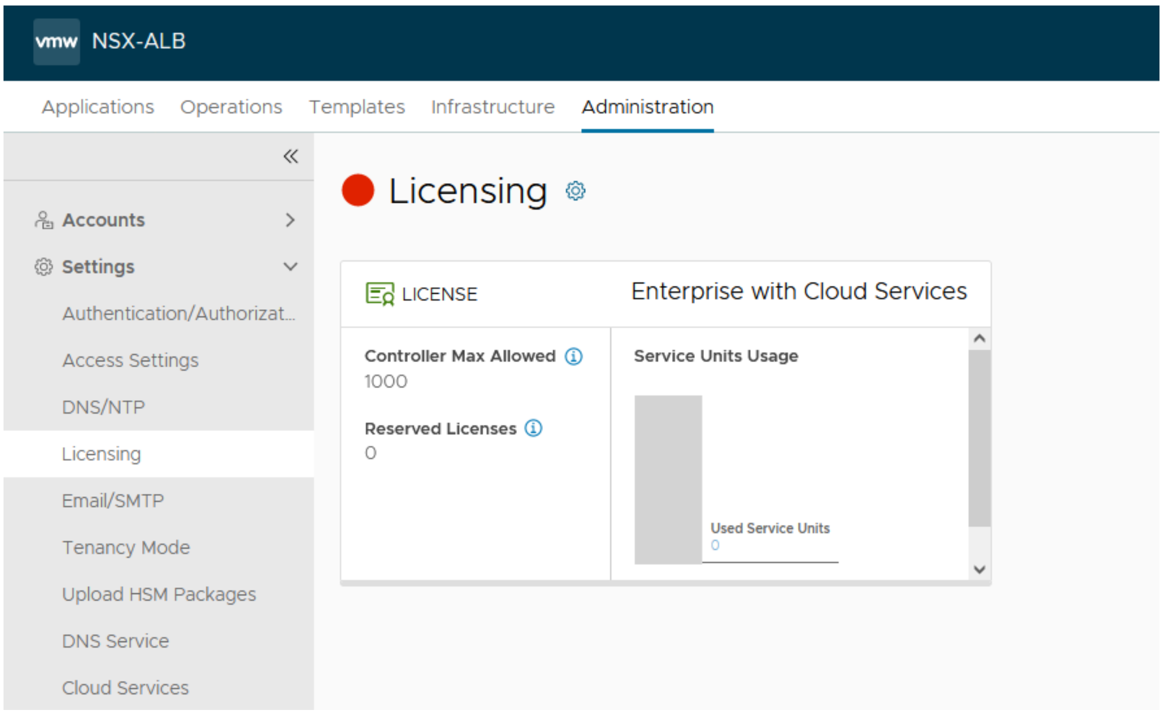
NTP Servers (1)

Key Number	Servers
Enter Key Number	192.168.200.5

NSX Advanced Load Balancer: Licensing

This document focuses on enabling NSX Advanced Load Balancer using the license model: **Enterprise License (VMware NSX ALB Enterprise)**.

1. To configure licensing, navigate to **Administration** > **Licensing**, and click on the gear icon to change the license type to Enterprise.



Licensing

LICENSE Enterprise with Cloud Services

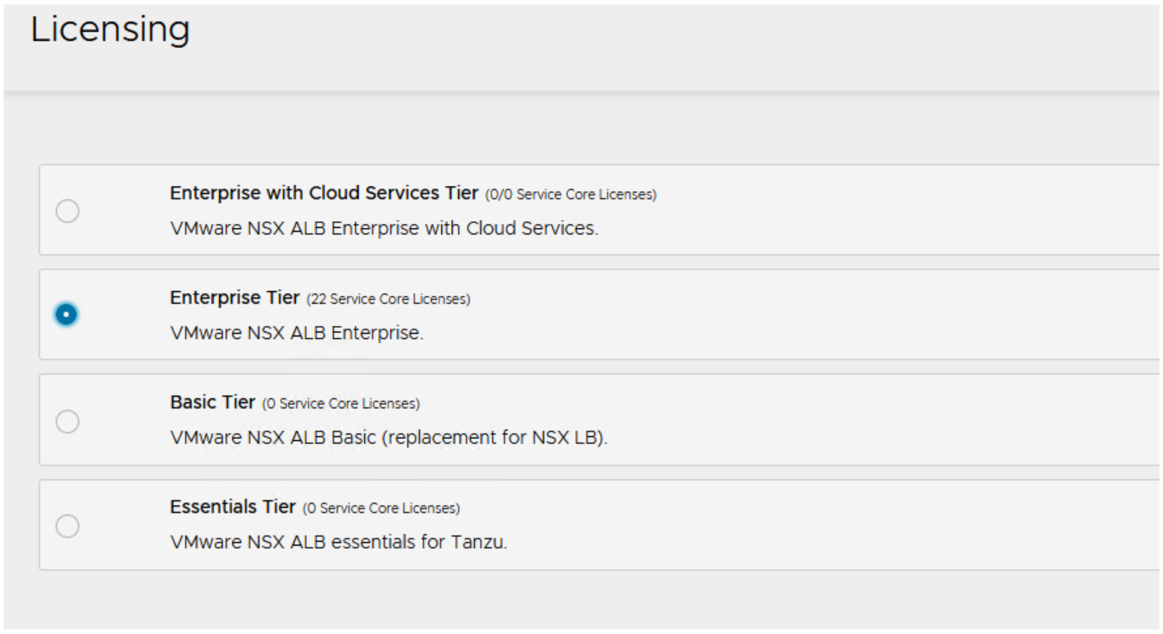
Controller Max Allowed
1000

Reserved Licenses
0

Service Units Usage

Used Service Units
0

2. Select **Enterprise Tier** radio button as the license type and click **Save**.



Licensing

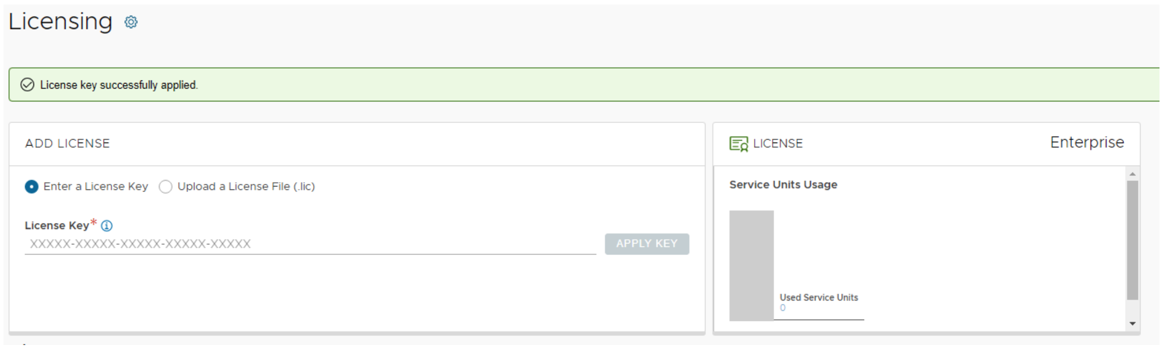
( ) **Enterprise with Cloud Services Tier** (0/0 Service Core Licenses)
VMware NSX ALB Enterprise with Cloud Services.

[✓] **Enterprise Tier** (22 Service Core Licenses)
VMware NSX ALB Enterprise.

( ) **Basic Tier** (0 Service Core Licenses)
VMware NSX ALB Basic (replacement for NSX LB).

( ) **Essentials Tier** (0 Service Core Licenses)
VMware NSX ALB essentials for Tanzu.

3. Once the license tier is changed, apply the NSX Advanced Load Balancer Enterprise license key. If you have a license file instead of a license key, apply the license by selecting the **Upload a License File** option.



Licensing

License key successfully applied.

ADD LICENSE

[✓ Enter a License Key] Upload a License File (.lic)

License Key*
XXXXX-XXXXX-XXXXX-XXXXX-XXXXX

LICENSE Enterprise

Service Units Usage

Used Service Units
0

NSX Advanced Load Balancer: Controller High Availability

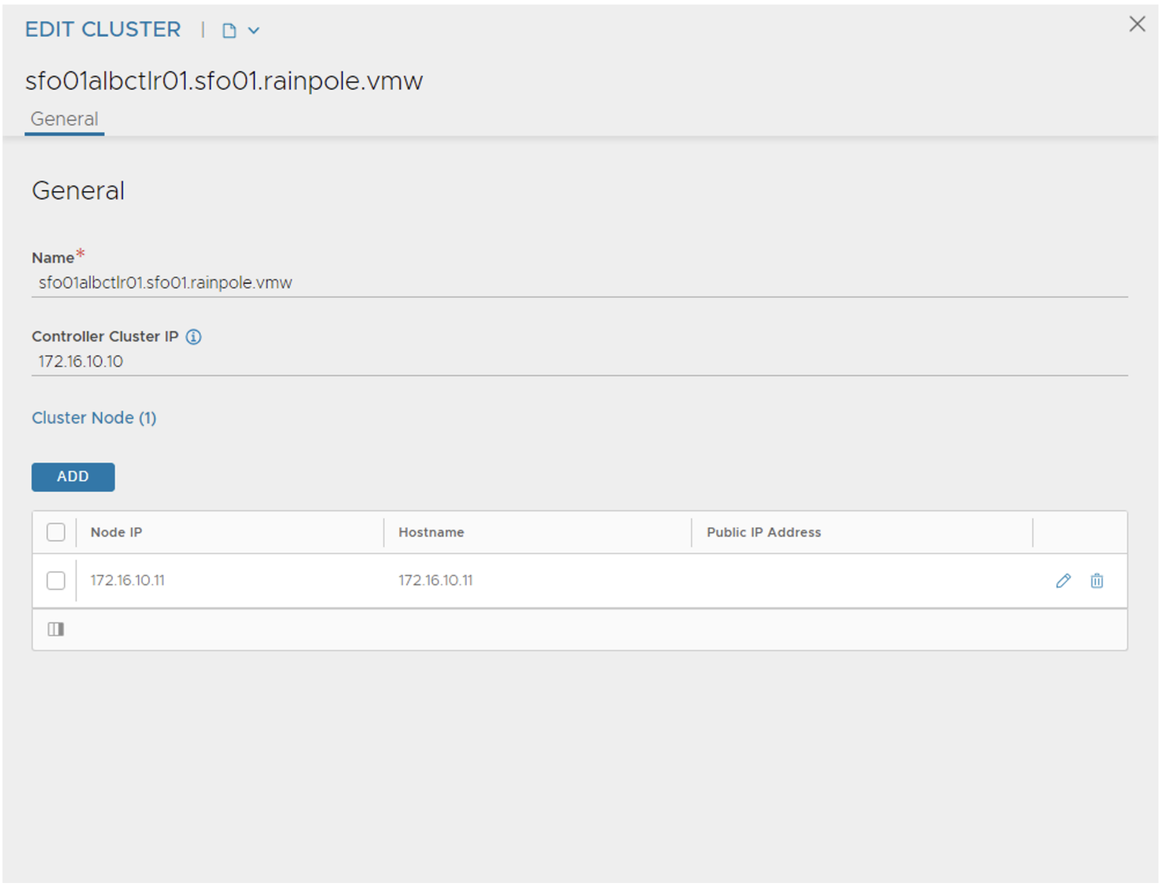
In a production environment, it is recommended to deploy additional controller nodes and configure the controller cluster for high availability and disaster recovery. Adding 2 additional nodes to create a 3-node cluster provides node-level redundancy for the controller and also maximizes performance for CPU-intensive analytics functions.

> To run a 3-node controller cluster, you deploy the first node and perform the initial configuration, and set the cluster IP address. After that, you deploy and power on two more controller VMs, but you must not run the initial configuration wizard or change the admin password for these controllers VMs. The configuration of the first controller VM is assigned to the two new controller VMs.

The first controller of the cluster receives the Leader role. The second and third controllers work as the Follower.

Complete the following steps to configure NSX Advanced Load Balancer cluster:

1. Log in to the primary NSX Advanced Load Balancer controller and navigate to **Administrator** > **Controller** > **Nodes**, and then click **Edit**.
2. Specify **Name** and **Controller Cluster IP**, and then click **Save**. This IP address must be from the NSX ALB management network.



EDIT CLUSTER

sfo01albctlr01.sfo01.rainpole.vmw

General

General

Name*
sfo01albctlr01.sfo01.rainpole.vmw

Controller Cluster IP
172.16.10.10

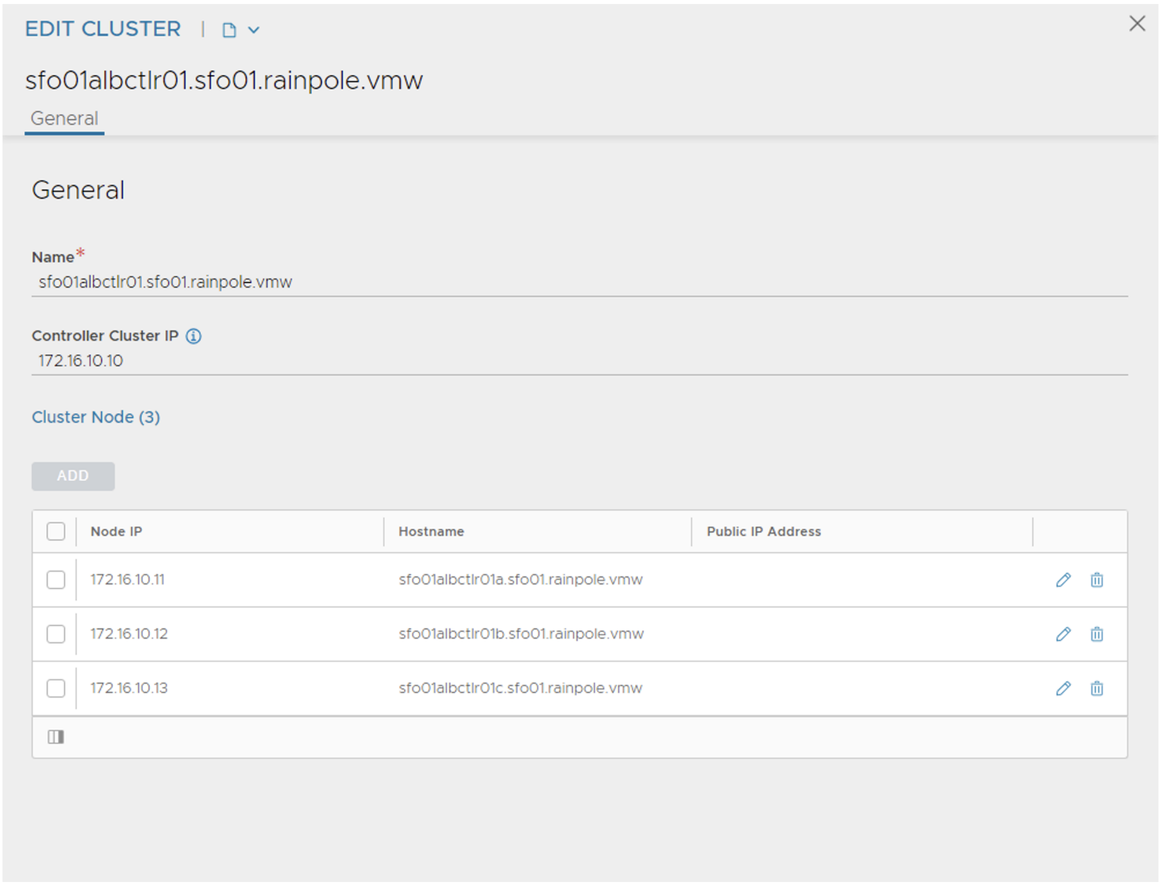
Cluster Node (1)

ADD

Node IP	Hostname	Public IP Address
172.16.10.11	172.16.10.11	

3. Deploy the 2nd and 3rd NSX Advanced Load Balancer controller nodes by using steps in Deploy NSX Advanced Load Balancer.

4. Log into the primary NSX Advanced Load Balancer controller using the Controller Cluster IP/FQDN and navigate to **Administrator** > **Controller** > **Nodes**, and then click **Edit**. The Edit Controller Configuration popup appears.
5. In the **Cluster Nodes** field, enter the IP address for the 2nd and 3rd controller, and then click **Save**.


EDIT CLUSTER

sfo01albctlr01.sfo01.rainpole.vmw

General

Name*: sfo01albctlr01.sfo01.rainpole.vmw

Controller Cluster IP: 172.16.10.10

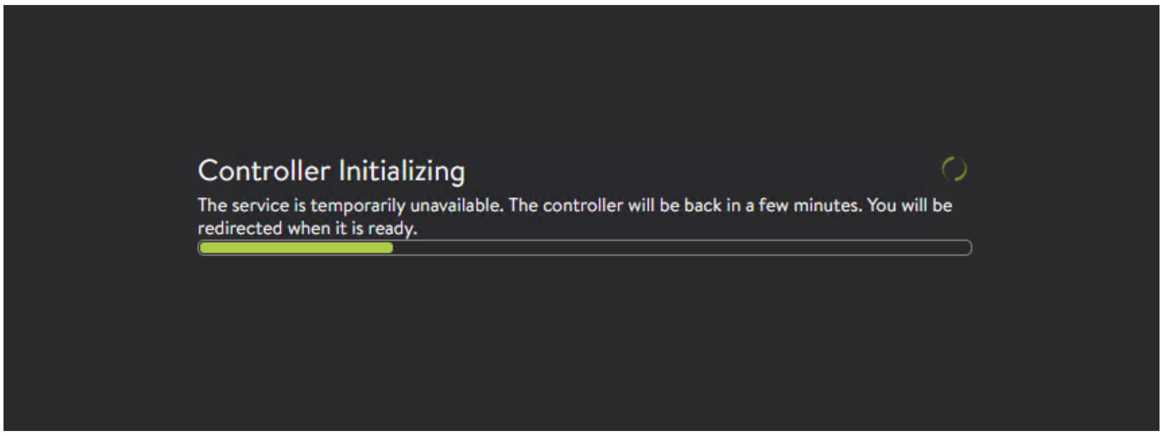
Cluster Node (3)

ADD

Node IP	Hostname	Public IP Address
172.16.10.11	sfo01albctlr01a.sfo01.rainpole.vmw	
172.16.10.12	sfo01albctlr01b.sfo01.rainpole.vmw	
172.16.10.13	sfo01albctlr01c.sfo01.rainpole.vmw	

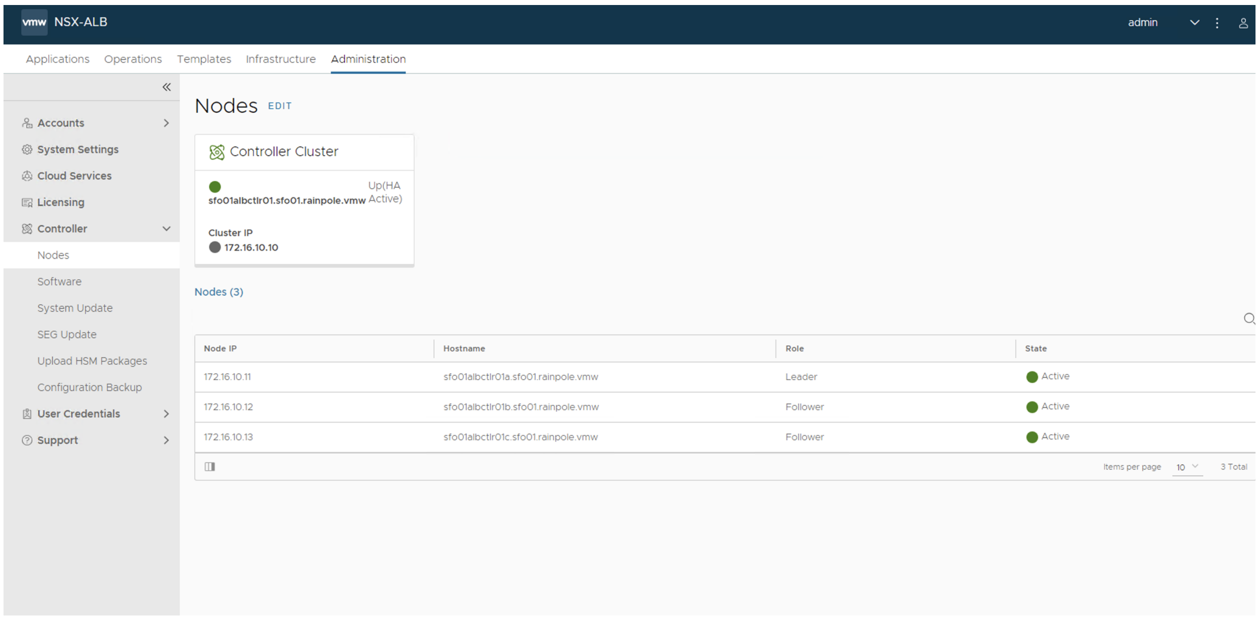
After you complete these steps, the primary NSX Advanced Load Balancer controller becomes the leader for the cluster and invites the other controllers to the cluster as members.

NSX Advanced Load Balancer then performs a warm reboot of the cluster. This process can take approximately 10-15 minutes. You will be automatically logged out of the controller node where you are currently logged in. On entering the cluster IP address in the browser, you can see details about the cluster formation task.


Controller Initializing

The service is temporarily unavailable. The controller will be back in a few minutes. You will be redirected when it is ready.

The configuration of the primary (leader) controller is synchronized to the new member nodes when the cluster comes online following the reboot. Once the cluster is successfully formed, you can see the following status:



Nodes

Controller Cluster

sfo01albctlr01.sfo01.rainpole.vmw Up(HA Active)

Cluster IP
172.16.10.10

Nodes (3)

Node IP	Hostname	Role	State
172.16.10.11	sfo01albctlr01a.sfo01.rainpole.vmw	Leader	● Active
172.16.10.12	sfo01albctlr01b.sfo01.rainpole.vmw	Follower	● Active
172.16.10.13	sfo01albctlr01c.sfo01.rainpole.vmw	Follower	● Active

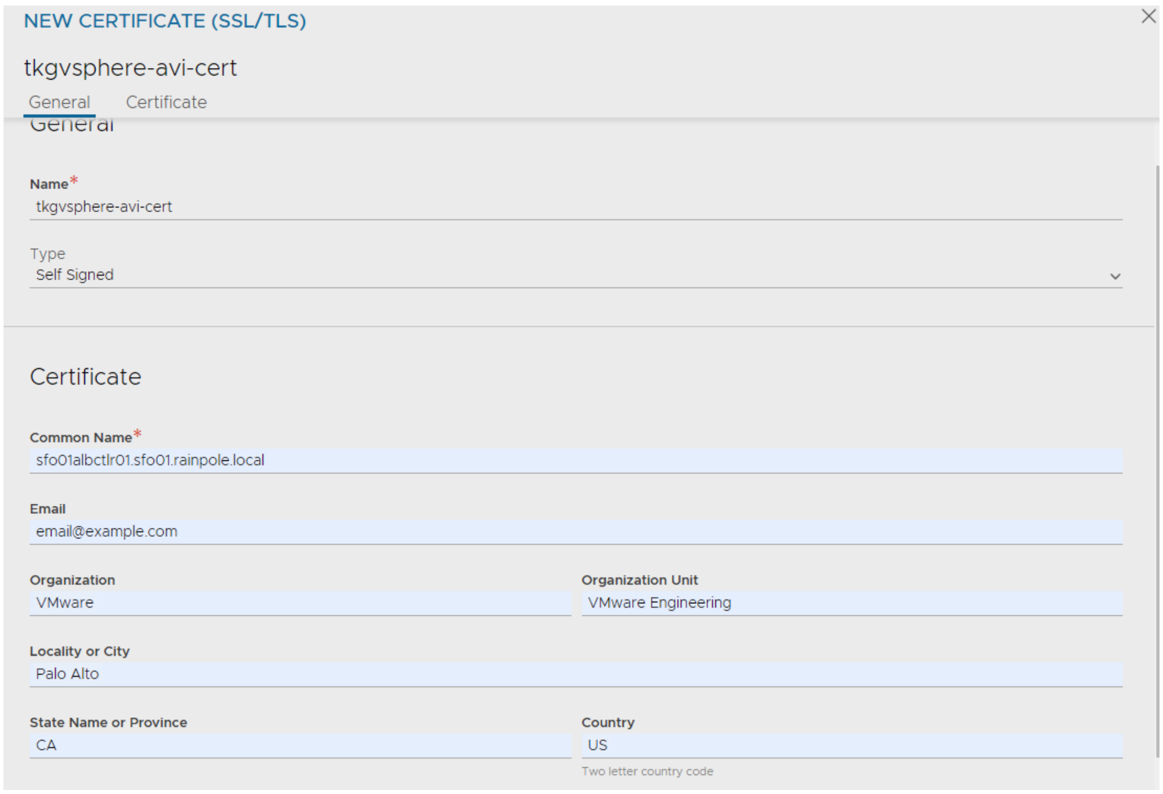
Items per page 10 3 Total

In the following tasks, all NSX Advanced Load Balancer configurations are done by connecting to the NSX Advanced Load Balancer Controller Cluster IP/FQDN.

NSX Advanced Load Balancer: Certificate Management

The default system-generated controller certificate generated for SSL/TSL connections will not have the required subject alternate name (SAN) entries. Complete the following steps to create a controller certificate:

1. Log in to the NSX Advanced Load Balancer controller and navigate to **Templates** > **Security** > **SSL/TLS Certificates**.
2. Click **Create** and select **Controller Certificate**. You can either generate a self-signed certificate, generate CSR, or import a certificate. For the purpose of this document, a self-signed certificate will be generated.
3. Provide all required details as per your infrastructure requirements and in the **Subject Alternate Name (SAN)** field, provide IP address and FQDN of all NSX Advanced Load Balancer controllers including NSX Advanced Load Balancer cluster IP and FQDN, and then click **Save**.



NEW CERTIFICATE (SSL/TLS)

tkgvsphere-avi-cert

General Certificate

General

Name*
tkgvsphere-avi-cert

Type
Self Signed

Certificate

Common Name*
sfo01albctlr01.sfo01.rainpole.local

Email
email@example.com

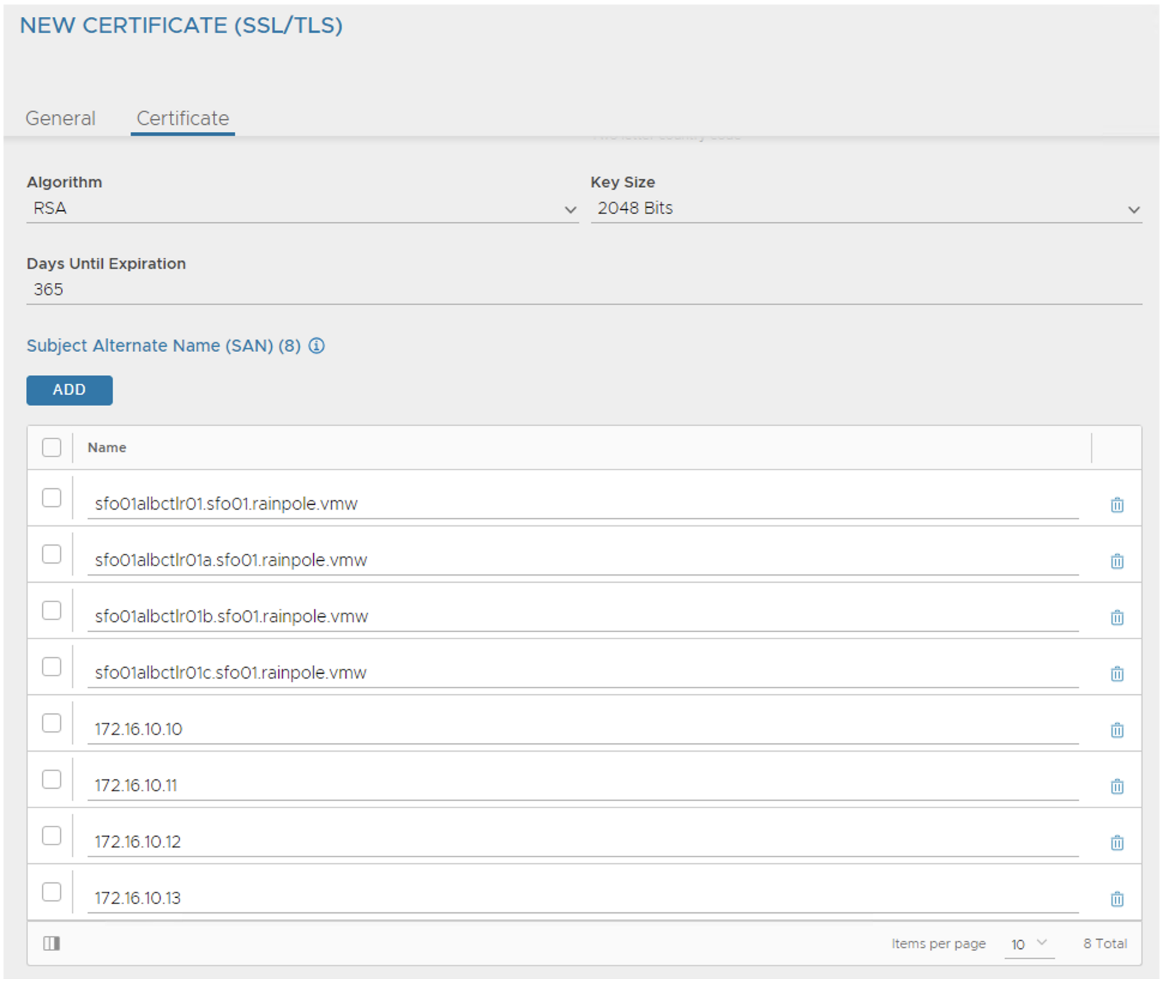
Organization
VMware

Organization Unit
VMware Engineering

Locality or City
Palo Alto

State Name or Province
CA

Country
US
Two letter country code



NEW CERTIFICATE (SSL/TLS)

General Certificate

Algorithm
RSA

Key Size
2048 Bits

Days Until Expiration
365

Subject Alternate Name (SAN) (8)

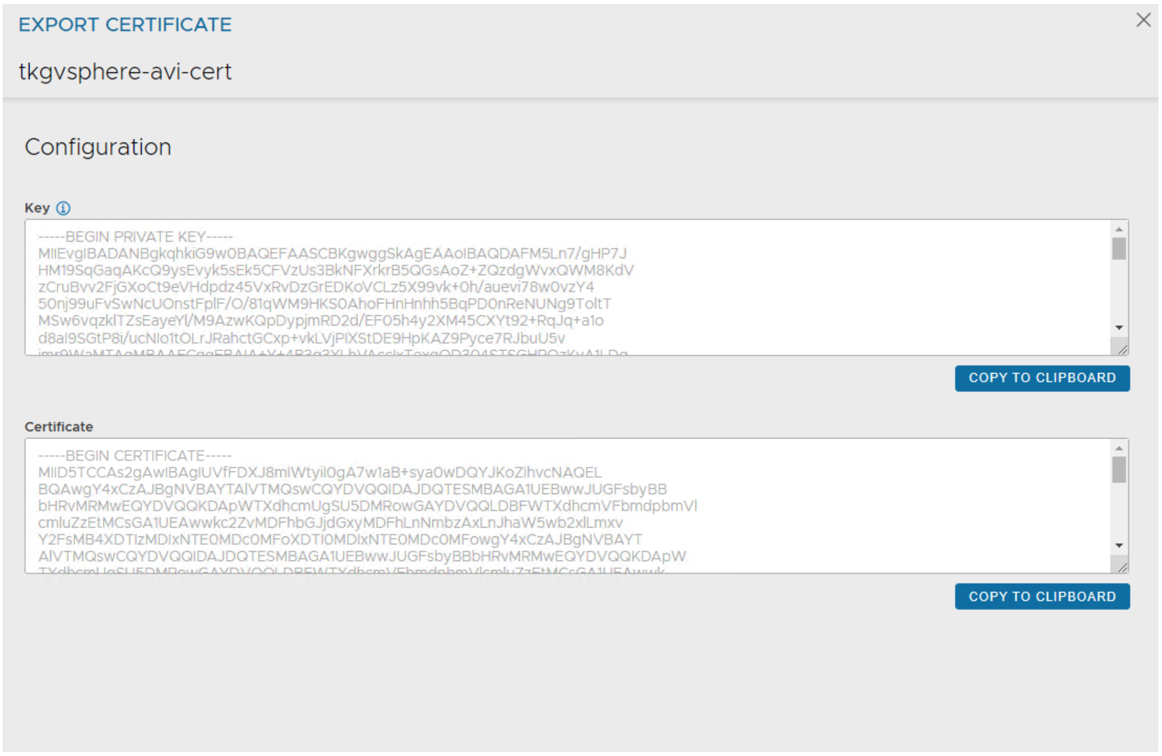
ADD

Name
sfo01albctlr01.sfo01.rainpole.vmw
sfo01albctlr01a.sfo01.rainpole.vmw
sfo01albctlr01b.sfo01.rainpole.vmw
sfo01albctlr01c.sfo01.rainpole.vmw
172.16.10.10
172.16.10.11
172.16.10.12
172.16.10.13

Items per page 10 8 Total

4. Once the certificate is created, capture the certificate contents as this is required while deploying the Tanzu Kubernetes Grid management cluster. To capture the certificate content, click on the

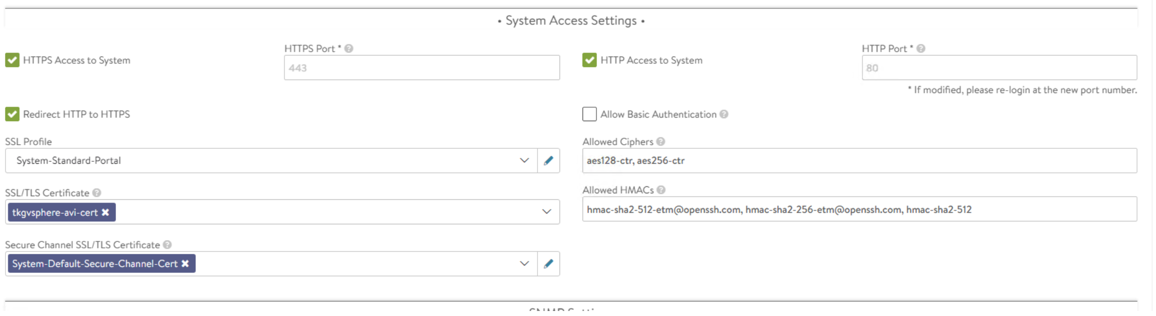
Download icon next to the certificate, and then click **Copy to clipboard** under **Certificate**.



```
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQDAFM5Ln7/gHP7J
HM19SqGaqAKcQ9ysEvyk5sEk5CFVzUs3BkNFXrkrB5QGsAoZ+ZQzdgWvxQWM8KdV
zCruBvv2FjGXoCt9eVHdpdz45VxRvDzGrEDKoVCLz5X99vk+0h/auevi78w0vzY4
50nj99uFvSwNcUOnstFplF/O/81qWM9HKS0AhoFHnHnhh5BqPD0nReNUNg9ToltT
MSw6vqzklTZsEayeYl/M9AzwKQpDypjmRD2d/EF05h4y2XM45CXYt92+RqJq+a1o
d8al9SGtP8i/ucNlo1tOLrJRahctGCxp+vkLVjPlXStDE9HpKAZ9Pyce7RJbuU5v
```

```
-----BEGIN CERTIFICATE-----
MIID5TCCAs2gAwIBAgIUVfFDXJ8mlWtyil0gA7w1aB+sya0wDQYJKoZIhvcNAQEL
BQAwgY4xCzAJBgNVBAYTAlVTMQswCQYDVQQIDAJDQTESMBAGA1UEBwwJUGFsbyBB
bHRvMRMwEQYDVQQKDApWTXdhcmUgSU5DMRowGAYDVQQLDBFWTXdhcmVFbmdpbmVl
cmluZzEtMCsGA1UEAwwkc2ZvMDFhbGJjdGxyMDFhLnNmbzAxLnJhaW5wb2xlLmxv
Y2FsMB4XDTIzMDIxNTE0MDc0MFoXDTI0MDIxNTE0MDc0MFowgY4xCzAJBgNVBAYT
AlVTMQswCQYDVQQIDAJDQTESMBAGA1UEBwwJUGFsbyBBbHRvMRMwEQYDVQQKDApW
```

5. To replace the certificate, navigate to **Administration** > **Settings** > **Access Settings**, and click the pencil icon at the top right to edit the system access settings, and then replace the SSL/TSL certificate and click **Save**.



✓ HTTPS Access to System	HTTPS Port *: 443	✓ HTTP Access to System	HTTP Port *: 80
			* If modified, please re-login at the new port number.
✓ Redirect HTTP to HTTPS		☐ Allow Basic Authentication	
SSL Profile: System-Standard-Portal		Allowed Ciphers: aes128-ctr, aes256-ctr	
SSL/TLS Certificate: tkgvsphere-avi-cert		Allowed HMACs: hmac-sha2-512-etm@openssh.com, hmac-sha2-256-etm@openssh.com, hmac-sha2-512	
Secure Channel SSL/TLS Certificate: System-Default-Secure-Channel-Cert			

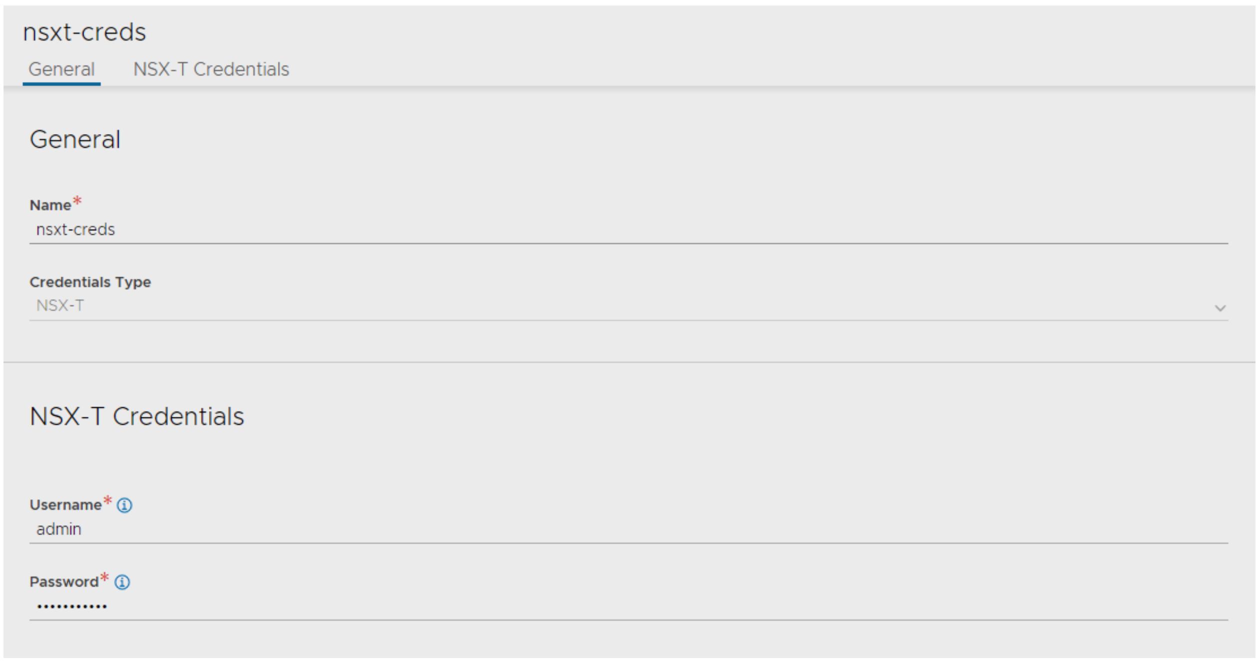
6. Log out and log in to NSX Advanced Load Balancer.

Create Credentials

NSX Advanced Load Balancer requires credentials of VMware NSX and vCenter Server to authenticate with these endpoints. These credentials need to be created before configuring NSX Cloud.

To create a new credential, navigate to **Administration** > **User Credentials** and click **Create**.

1. Create NSX Credential: Select the **Credential Type** as NSX-T from the drop-down menu, and provide a name for the credential. Under the section NSX-T Credentials, specify the username and password that NSX Advanced Load Balancer will use to authenticate with VMware NSX.


nsxt-creds

General | NSX-T Credentials

General

Name* nsxt-creds

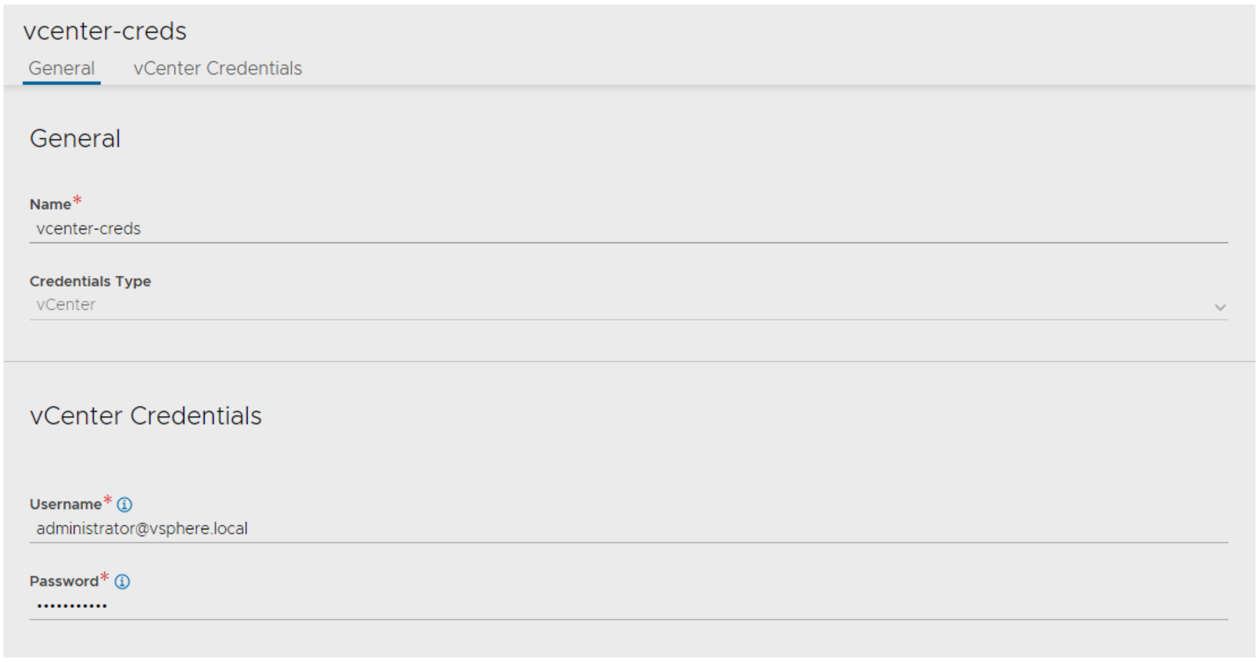
Credentials Type NSX-T

NSX-T Credentials

Username* admin

Password* ••••••••••

1. Create vCenter Credential: Select the **Credential type** as vCenter from the drop-down menu and provide a name for the credential. Under the section vCenter Credentials, specify the username and password that NSX Advanced Load Balancer will use to authenticate with the vCenter server.


vcenter-creds

General | vCenter Credentials

General

Name* vcenter-creds

Credentials Type vCenter

vCenter Credentials

Username* administrator@vsphere.local

Password* ••••••••••

Create NSX Cloud and Service Engine Groups

NSX Advanced Load Balancer can be deployed in multiple environments for the same system. Each environment is called a cloud. The following procedure provides steps to create a VMware NSX cloud. As per the architecture, two service engine (SE) groups will be created.

Service Engine Group 1: Service engines associated with this service engine group hosts:

- Virtual services that load balances control plane nodes of Management Cluster and Shared services cluster.

- Virtual services for all load balancer functionalities requested by Tanzu Kubernetes Grid management cluster and Shared services cluster.

Service Engine Group 2: Service engines part of this service engine group hosts virtual services that load balances control plane nodes and virtual services for all load balancer functionalities requested by the workload clusters mapped to this SE group.

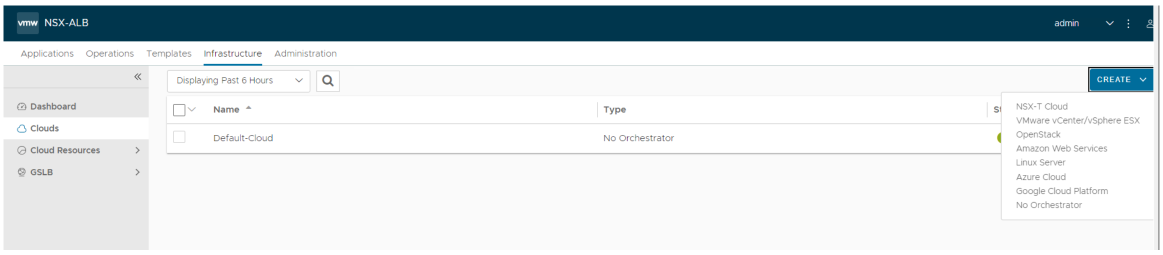
> - Based on your requirements, you can create additional SE groups for the workload clusters. - Multiple workload clusters can be mapped to a single SE group. - A Tanzu Kubernetes Grid cluster can be mapped to only one SE group for application load balancer services. - Control plane VIP for the workload clusters will be placed on the respective Service Engine group assigned through AKO Deployment Config (ADC) during cluster creation.

For information about mapping a specific service engine group to Tanzu Kubernetes Grid workload cluster, see Configure NSX Advanced Load Balancer in Tanzu Kubernetes Grid Workload Cluster.

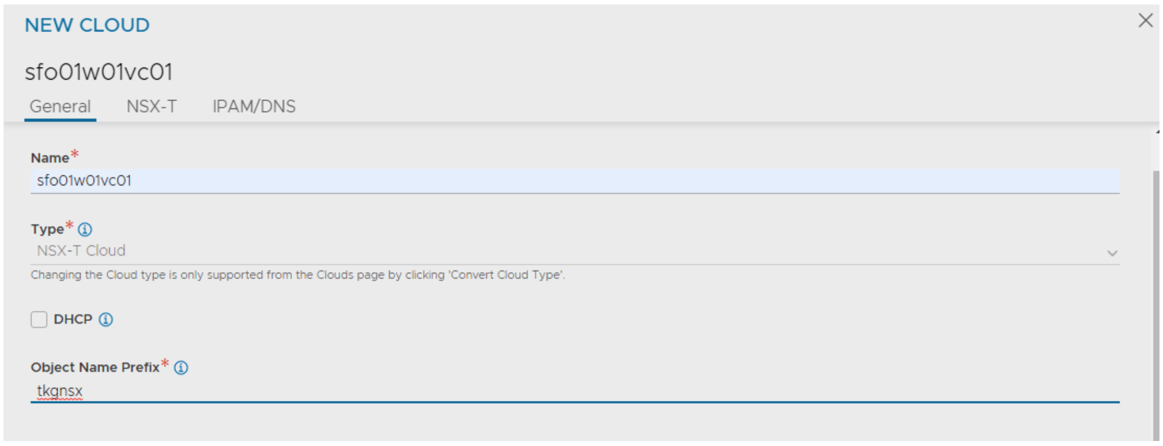
The following components are created in NSX Advanced Load Balancer.

Object	Sample Name
NSX Cloud	sfo01w01vc01
Service Engine Group 1	sfo01m01segroup01
Service Engine Group 2	sfo01w01segroup01

1. Log in to NSX Advanced Load Balancer and navigate to **Infrastructure** > **Clouds** > **Create** > **NSX-T Cloud**.



2. Enter cloud name and provide a object name prefix. Click **CHANGE CREDENTIALS** to connect NSX Advanced Load Balancer with VMware NSX.


NEW CLOUD

sfo01w01vc01

General NSX-T IPAM/DNS

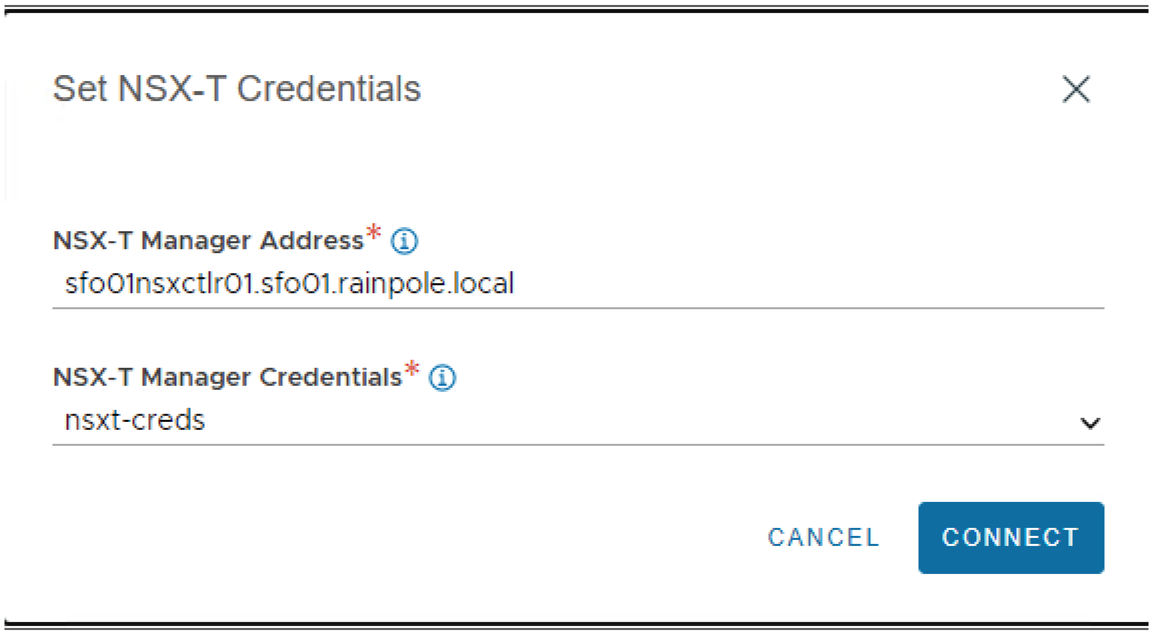
Name*
sfo01w01vc01

Type*
NSX-T Cloud
Changing the Cloud type is only supported from the Clouds page by clicking 'Convert Cloud Type'.

DHCP

Object Name Prefix*
tkgnsx

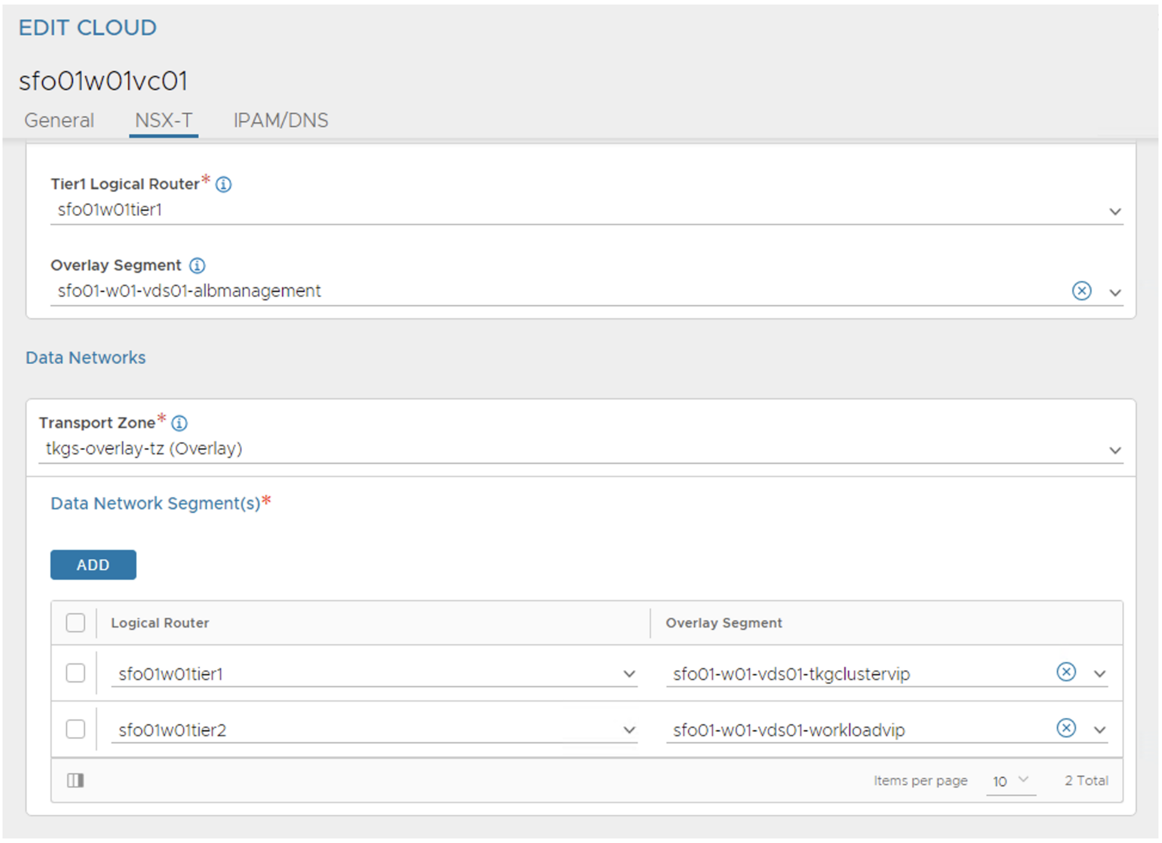
3. Specify NSX-T Manager Address and select the NSX-T credential that you created earlier.



NSX-T Manager Address*: sfo01nsxctlr01.sfo01.rainpole.local

NSX-T Manager Credentials*: nsxt-creds

4. Under the **Management Network** pane, select the following options:
   - Transport Zone: Overlay transport zone where you connected your NSX Advanced Load Balancer management network.
   - Tier-1 Router: Tier-1 gateway where Advanced Load Balancer management network is connected.
   - Overlay Segment: Logical segment that you have created for Advanced Load Balancer management.
5. Under the **Data Networks** pane, select the following:
   - Transport Zone: Overlay transport zone where you connected your Tanzu Kubernetes Grid VIP networks.
   - Tier-1 Router: Tier-1 gateway **sfo01w01tier1** TKG Cluster VIP network is connected.
   - Overlay Segment: Logical segment that you have created for TKG Cluster VIP Network.
   - Tier-1 Router: Tier-1 gateway **sfo01w01tier2** where TKG Workload VIP Network is connected.
   - Overlay Segment: Logical segment that you have created for TKG Workload VIP Network.



EDIT CLOUD

sfo01w01vc01

General | NSX-T | IPAM/DNS

Tier1 Logical Router*: sfo01w01tier1

Overlay Segment: sfo01-w01-vds01-albmanagement

Data Networks

Transport Zone*: tkgs-overlay-tz (Overlay)

Data Network Segment(s)*

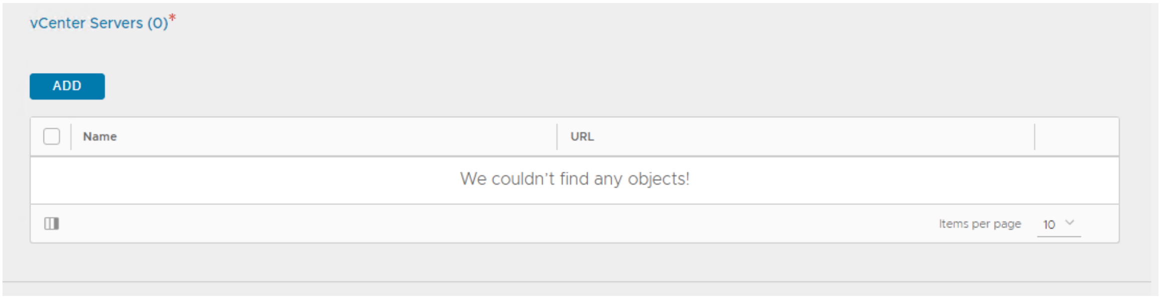
ADD

Logical Router	Overlay Segment
sfo01w01tier1	sfo01-w01-vds01-tkgclustervip
sfo01w01tier2	sfo01-w01-vds01-workloadvip

Items per page 10 — 2 Total

For a single VIP network architecture, do not add **sfo01w01tier2** tier-1 gateway under Data Network Segments and associated Overlay Segment. TKG Workload cluster will use the TKG Cluster VIP for both control plane and data plane network.

1. Under **vCenter Servers** pane, click **ADD**.



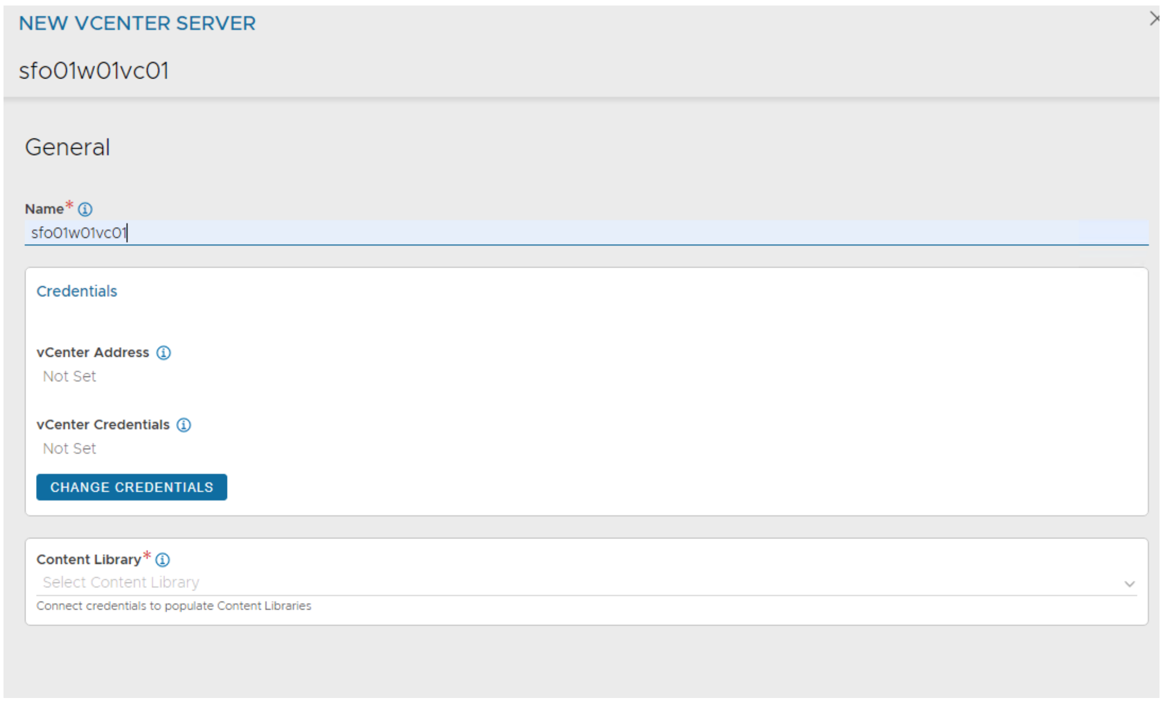
vCenter Servers (0)*

ADD

Name	URL
We couldn't find any objects!	

Items per page 10

2. Specify a name for the vCenter server and click **CHANGE CREDENTIALS** to connect NSX Advanced Load Balancer with the vCenter server.


NEW VCENTER SERVER

sfo01w01vc01

General

Name*
sfo01w01vc01

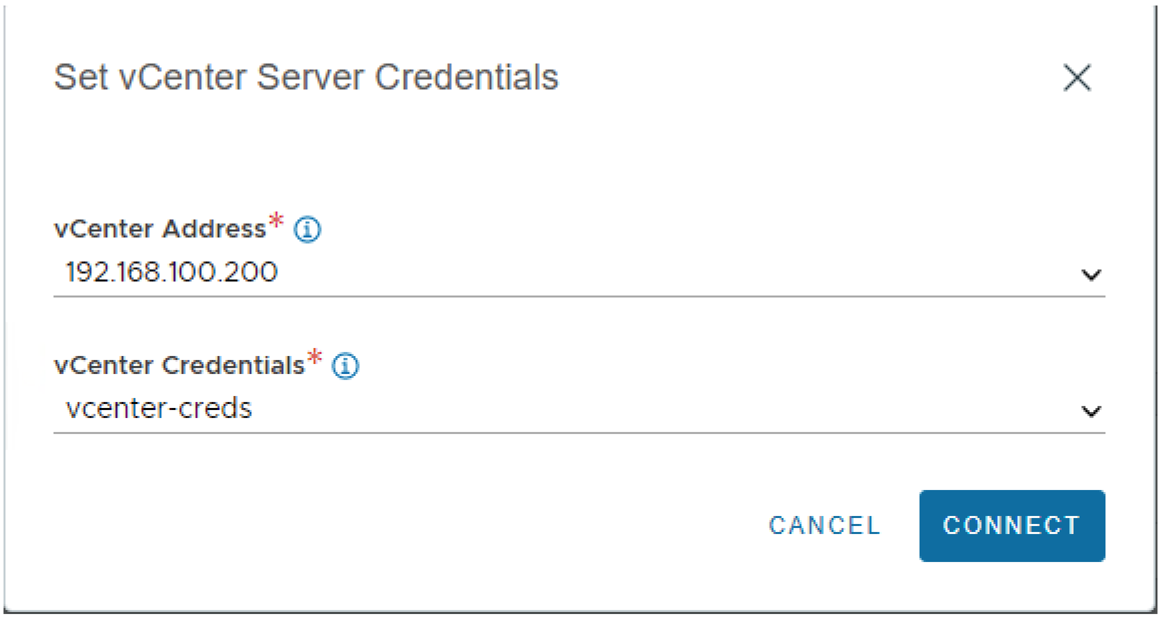
Credentials

vCenter Address
Not Set

vCenter Credentials
Not Set

Content Library*
Select Content Library
Connect credentials to populate Content Libraries

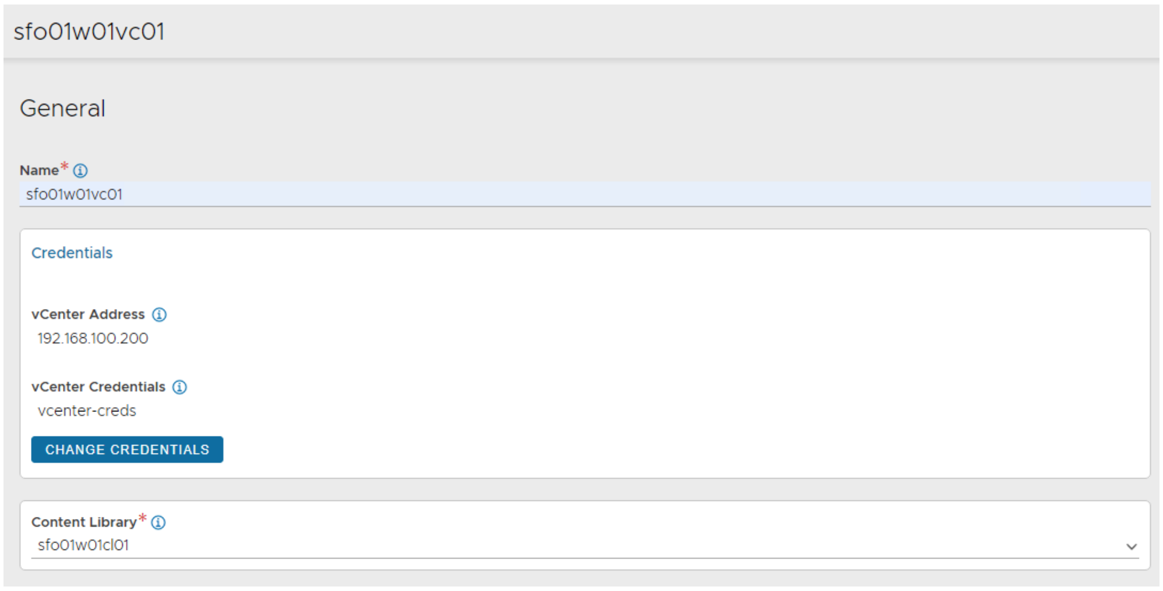
3. Select the vCenter server from the drop down and select the vCenter credential which you have created earlier.


Set vCenter Server Credentials

vCenter Address*
192.168.100.200

vCenter Credentials*
vcenter-creds

4. Select the Content Library where Service Engine templates will be stored by NSX Advanced Load Balancer.



sfo01w01vc01

General

Name* ⓘ
sfo01w01vc01

Credentials

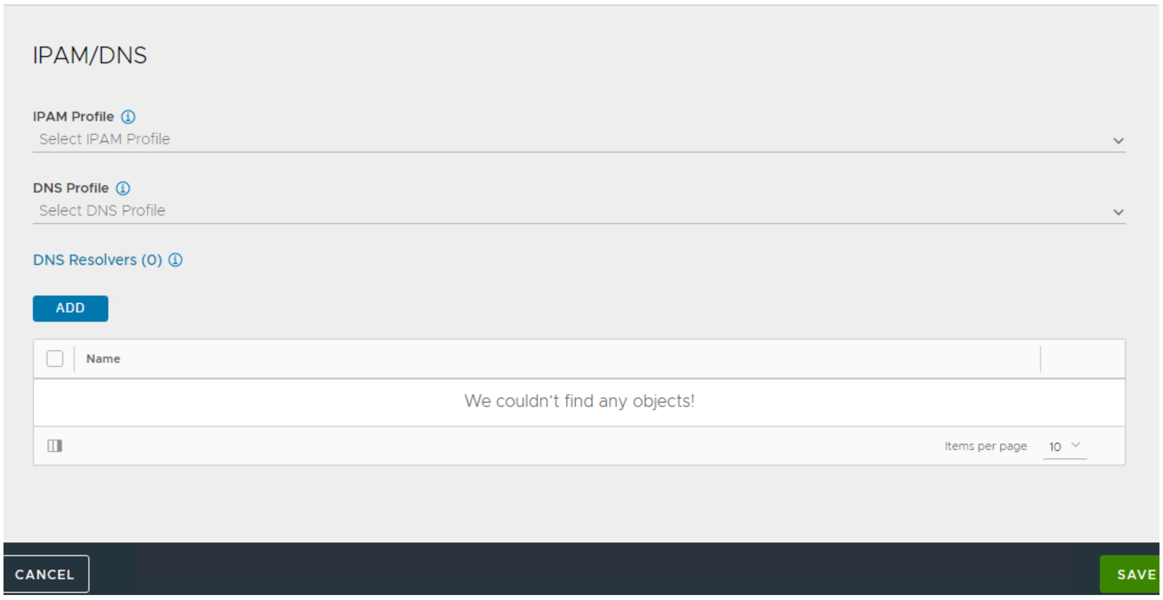
vCenter Address ⓘ
192.168.100.200

vCenter Credentials ⓘ
vcenter-creds

CHANGE CREDENTIALS

Content Library* ⓘ
sfo01w01cl01

5. Leave the IPAM/DNS profile section empty as this will be populated later, once you have created the profiles. Click **SAVE** to finish the NSX-T cloud configuration.



IPAM/DNS

IPAM Profile ⓘ
Select IPAM Profile

DNS Profile ⓘ
Select DNS Profile

DNS Resolvers (0) ⓘ

ADD

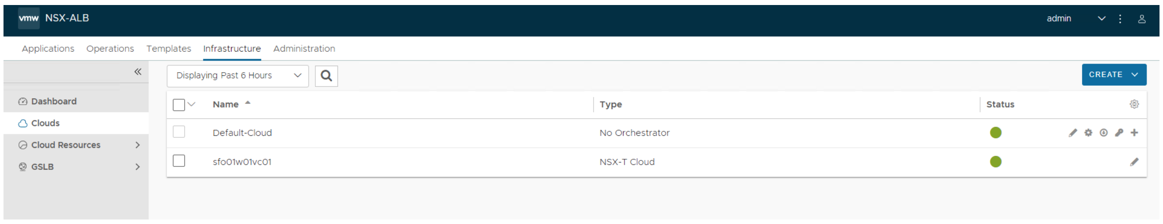
	Name
	We couldn't find any objects!

Items per page 10

CANCEL

SAVE

6. Ensure that status of the NSX-T cloud is Green post creation.



Name	Type	Status
Default-Cloud	No Orchestrator	●
sfo01w01vc01	NSX-T Cloud	●

7. Create a service engine group for Tanzu Kubernetes Grid management clusters:

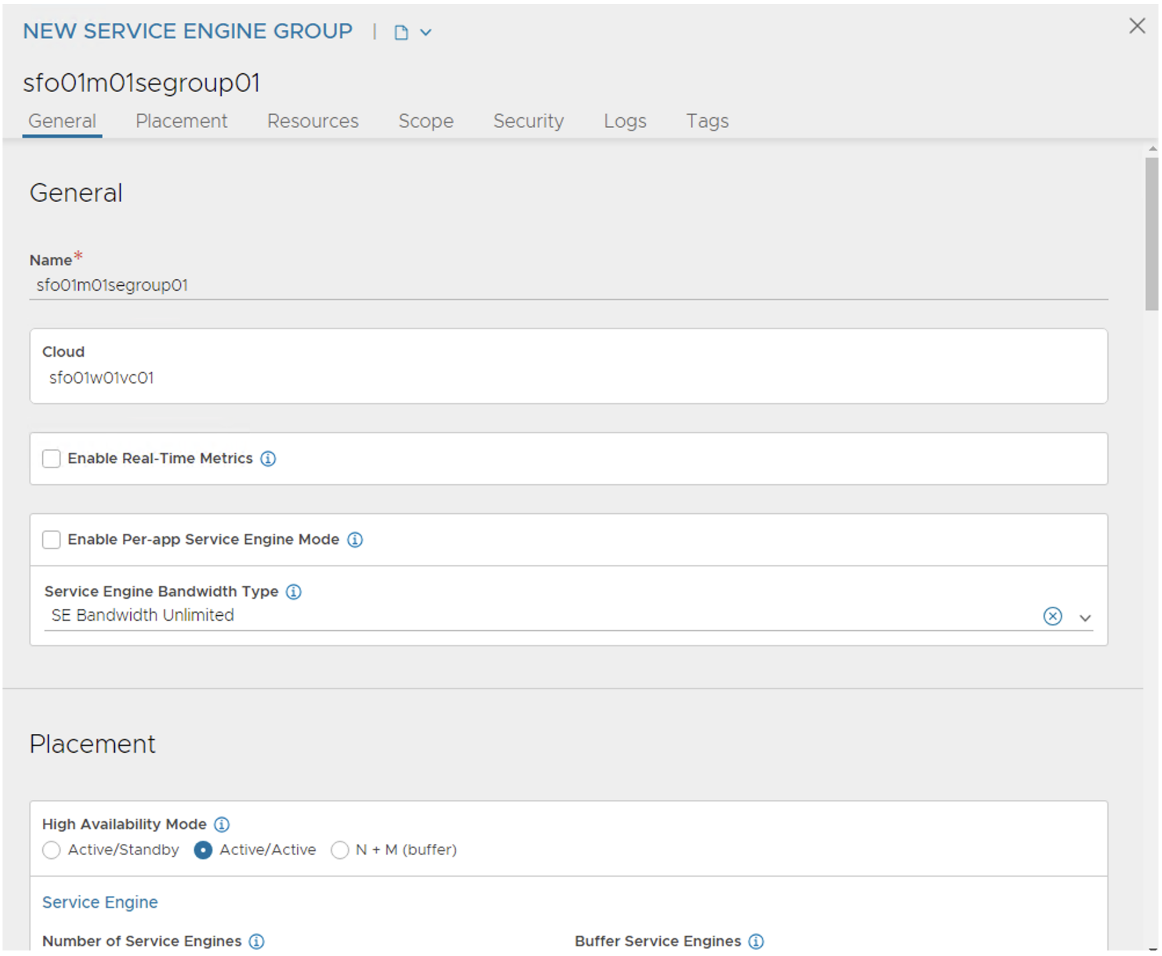
   1. Click on the **Service Engine Group** tab.

   2. Under Select Cloud, choose the cloud created in the previous step, and click **Create**.

8. Enter a name for the Tanzu Kubernetes Grid management service engine group and set the following parameters:

Parameter	Value
High availability mode	Active/Active
VS Placement	Compact
Memory per Service Engine	4
vCPU per Service Engine	2

Use the default values for the rest of the parameters.



General

Name*: sfo01m01segroup01

Cloud: sfo01w01vc01

☐ Enable Real-Time Metrics

☐ Enable Per-app Service Engine Mode

Service Engine Bandwidth Type: SE Bandwidth Unlimited

Placement

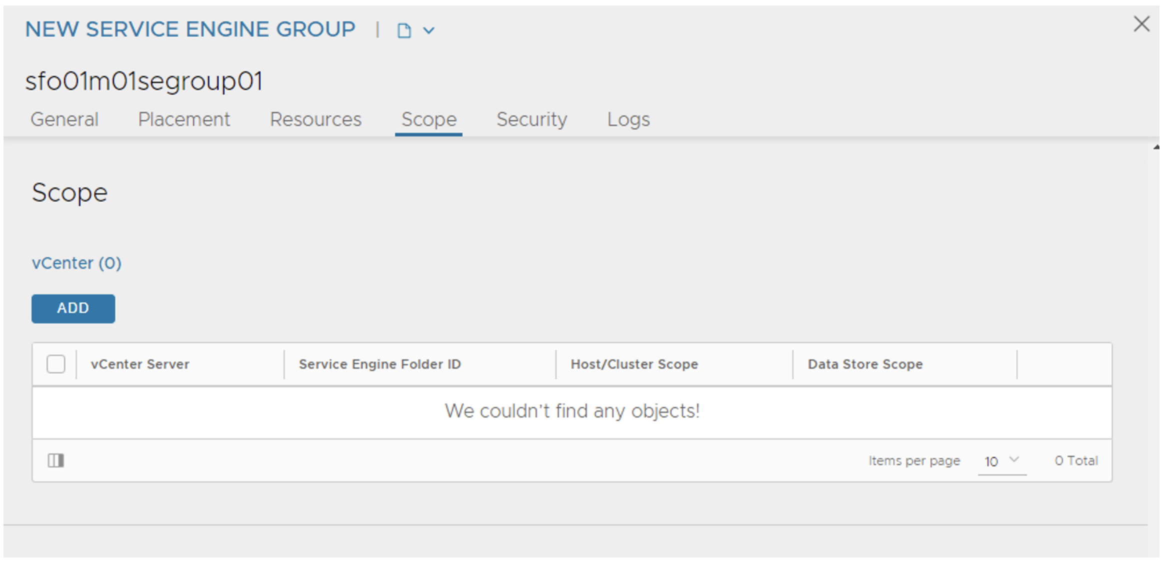
High Availability Mode: Active/Standby [✓ Active/Active] N + M (buffer)

Service Engine

Number of Service Engines

Buffer Service Engines

Under **Scope** tab, Specify the vCenter server endpoint by clicking on the Add option.



NEW SERVICE ENGINE GROUP

sfo01m01segroup01

General | Placement | Resources | Scope | Security | Logs

Scope

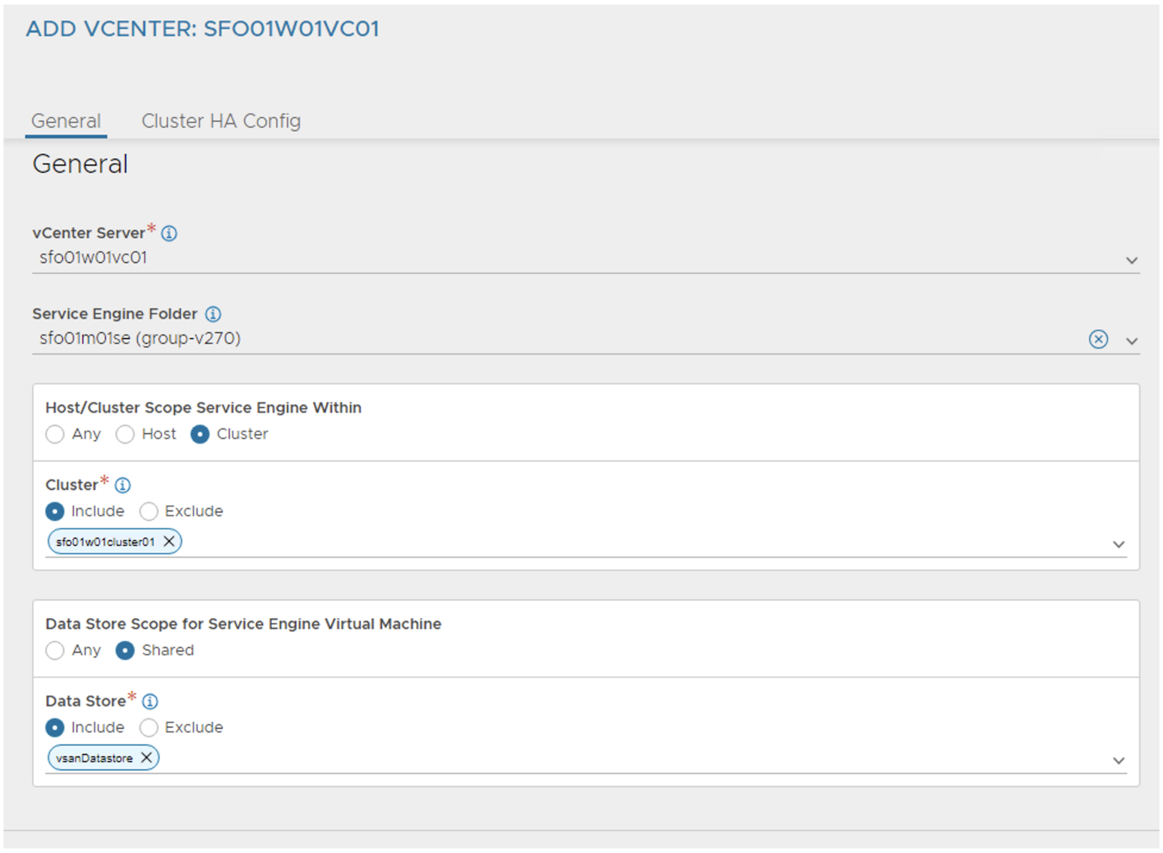
vCenter (0)

ADD

vCenter Server	Service Engine Folder ID	Host/Cluster Scope	Data Store Scope
We couldn't find any objects!			

Items per page 10 0 Total

Select the vCenter server from the dropdown, Service Engine Folder, vSphere cluster, and datastore for service engine placement, and then click **Save**.



ADD VCENTER: SFO01W01VC01

General | Cluster HA Config

General

vCenter Server*: sfo01w01vc01

Service Engine Folder: sfo01m01se (group-v270)

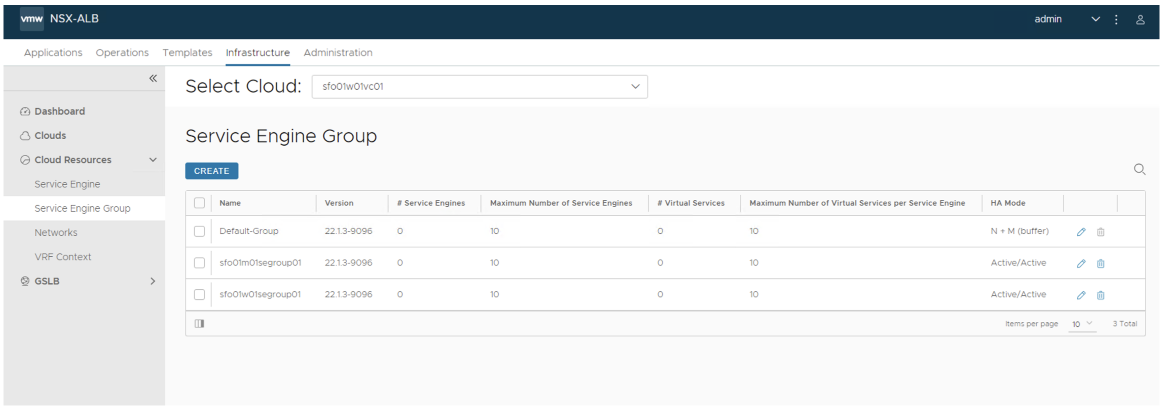
Host/Cluster Scope Service Engine Within: ( ) Any ( ) Host [✓ Cluster]

Cluster*: [✓ Include] ( ) Exclude — sfo01w01cluster01

Data Store Scope for Service Engine Virtual Machine: ( ) Any [✓ Shared]

Data Store*: [✓ Include] ( ) Exclude — vsanDatastore

9. Repeat steps 12 and 13 to create another service engine group for Tanzu Kubernetes Grid workload clusters. Once complete, there must be two service engine groups created.



Select Cloud: sfo01w01vc01

Service Engine Group

Name	Version	# Service Engines	Maximum Number of Service Engines	# Virtual Services	Maximum Number of Virtual Services per Service Engine	HA Mode
Default-Group	22.1.3-9096	0	10	0	10	N + M (buffer)
sfo01m01segroup01	22.1.3-9096	0	10	0	10	Active/Active
sfo01w01segroup01	22.1.3-9096	0	10	0	10	Active/Active

Items per page 10 3 Total

Configure Network and IPAM Profile

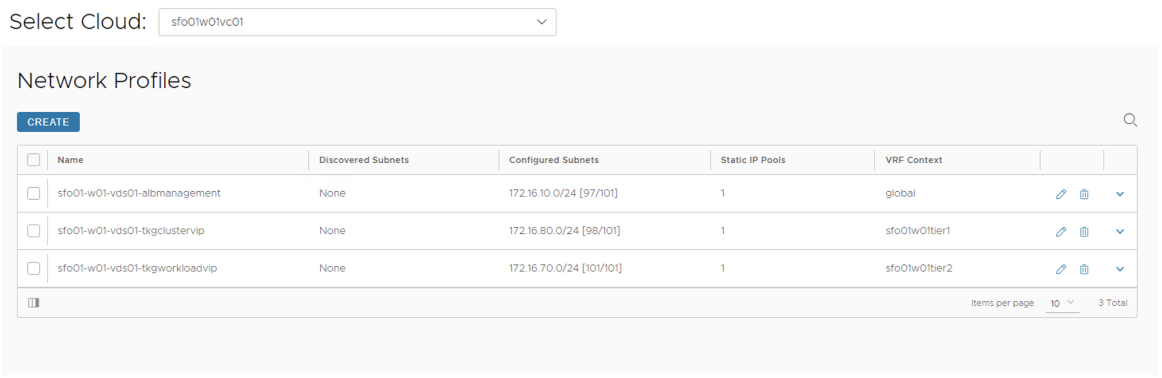
As part of the cloud creation, NSX Advanced Load Balancer management and Tanzu Kubernetes Grid VIP networks have been configured in NSX Advanced Load Balancer. Since DHCP was not selected as the IP address management method in the cloud configuration, you have to specify pool of IP addresses that can be assigned to the service engine NICs and the virtual services that will be created in future.

To configure IP address pools for the networks, perform the following steps:

1. Navigate to **Infrastructure** > **Cloud Resources** > **Networks** and select the cloud that you have created earlier. Click on the **Edit** icon next for the network and configure as follows. Change the provided details as per your SDDC configuration.

Network Name	DHCP	Subnet	Static IP Pool
sfo01-w01-vds01-albmanagement	No	172.16.10.0/24	172.16.10.100 - 172.16.10.200
sfo01-w01-vds01-tkgclustervip	No	172.16.80.0/24	172.16.80.100 - 172.16.80.200
sfo01-w01-vds01-tkgworkloadvip	No	172.16.70.0/24	172.16.70.100 - 172.16.70.200

Once the networks are configured, the configuration must look like the following image.



Select Cloud: sfo01w01vc01

Network Profiles

Name	Discovered Subnets	Configured Subnets	Static IP Pools	VRF Context
sfo01-w01-vds01-albmanagement	None	172.16.10.0/24 [97/101]	1	global
sfo01-w01-vds01-tkgclustervip	None	172.16.80.0/24 [98/101]	1	sfo01w01tier1
sfo01-w01-vds01-tkgworkloadvip	None	172.16.70.0/24 [101/101]	1	sfo01w01tier2

Items per page 10 3 Total

For a single VIP network architecture, do not configure sfo01-w01-vds01-tkgworkloadvip network. The sfo01-w01-vds01-tkgclustervip segment is used for control plane and data network of TKG workload cluster.

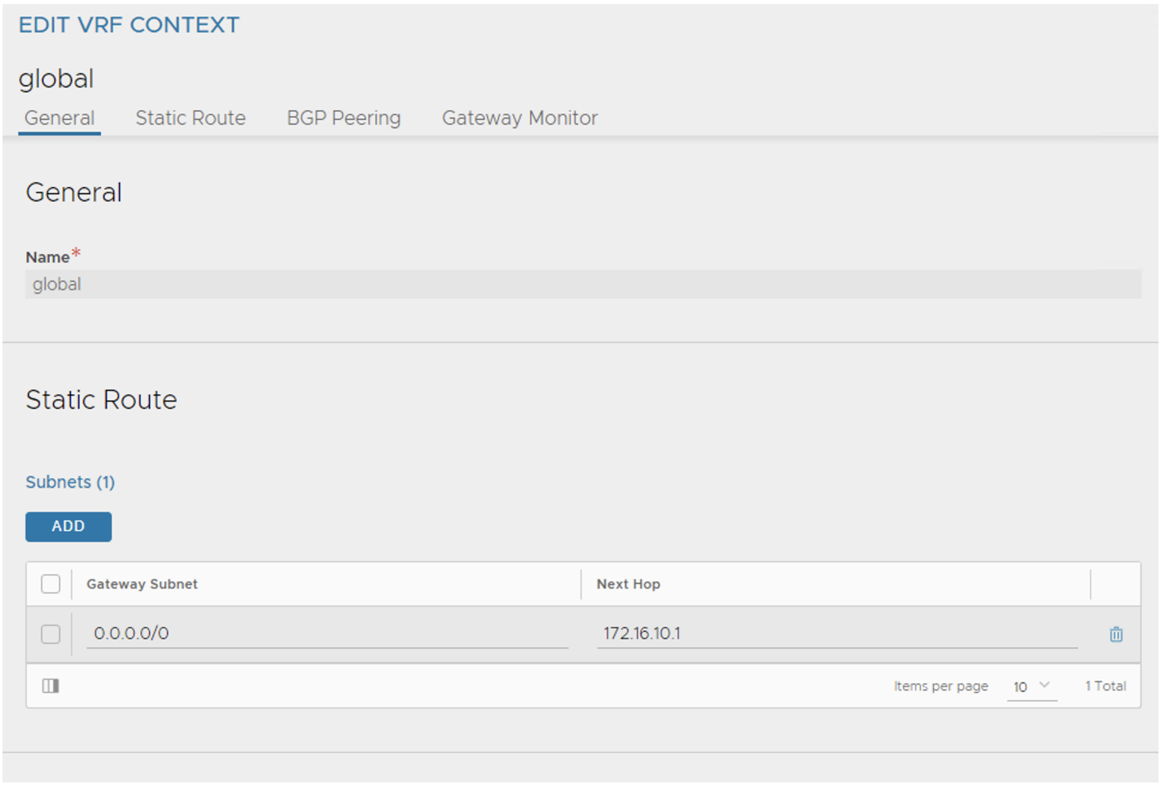
2. Once the networks are configured, set the default routes for the networks by navigating to **Infrastructure** > **Cloud Resource** > **Routing**.

- Ensure that VRF Context for `sfo01-w01-vds01-albmanagement` network is set to `Global`.

- Ensure that VRF Context for `sfo01-w01-vds01-tkgclustervip` network is set to NSX tier-1 gateway `sfo01w01tier1`.

- Ensure that VRF Context for sfo01-w01-vds01-tkgworkloadvip network is set to NSX tier-1 gateway `sfo01w01tier2`.

To set the default gateway for the `asfo01-w01-vds01-albmanagement` network, click **ADD** Static Route under the global VRF context and set the default gateway to gateway of the NSX Advanced Load Balancer management subnet.

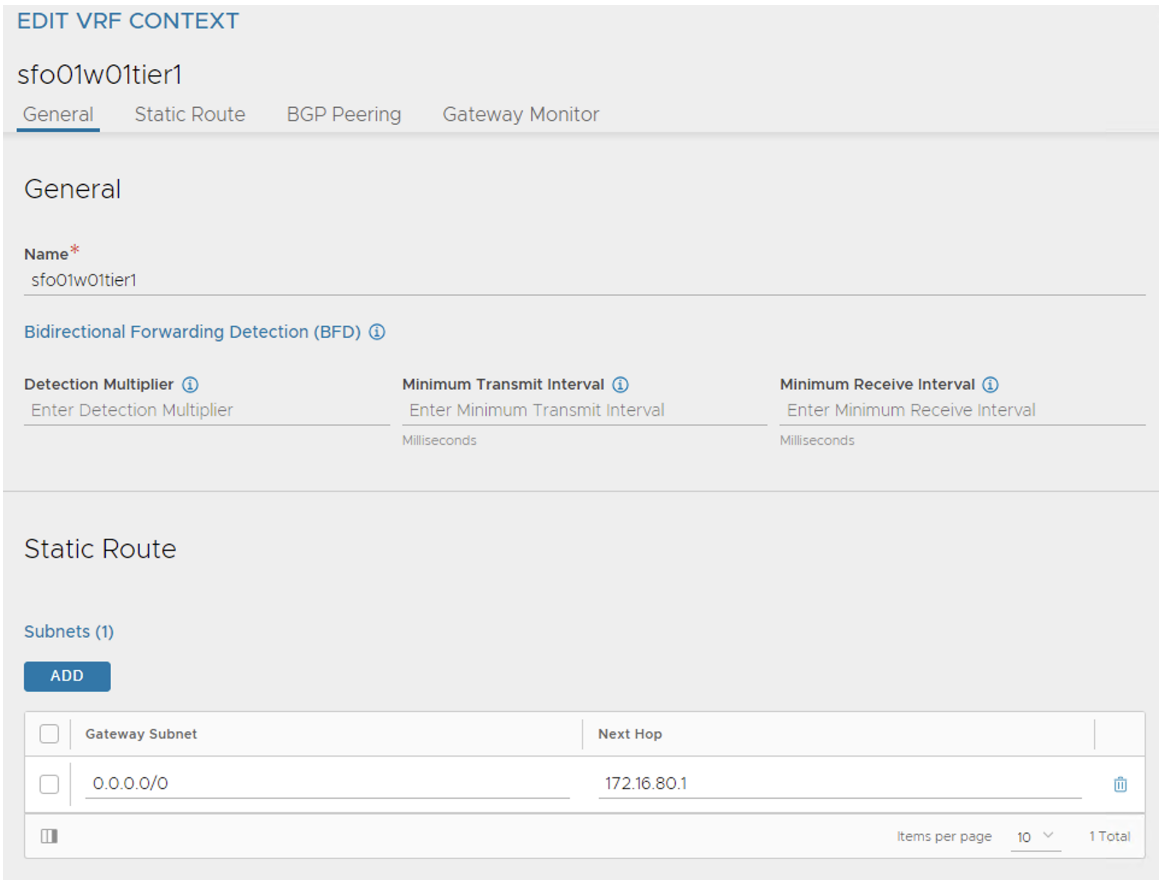


Name* global

Gateway Subnet	Next Hop
0.0.0.0/0	172.16.10.1

Items per page 10 1 Total

To set the default gateway for the `sfo01-w01-vds01-tkgclustervip` network, click **CREATE** under the tier-1 gateway VRF context and set the default gateway to gateway of the VIP network subnet.


EDIT VRF CONTEXT

sfo01w01tier1

General | Static Route | BGP Peering | Gateway Monitor

General

Name*
sfo01w01tier1

Bidirectional Forwarding Detection (BFD)

Detection Multiplier	Minimum Transmit Interval	Minimum Receive Interval
Enter Detection Multiplier	Enter Minimum Transmit Interval	Enter Minimum Receive Interval
	Milliseconds	Milliseconds

Static Route

Subnets (1)

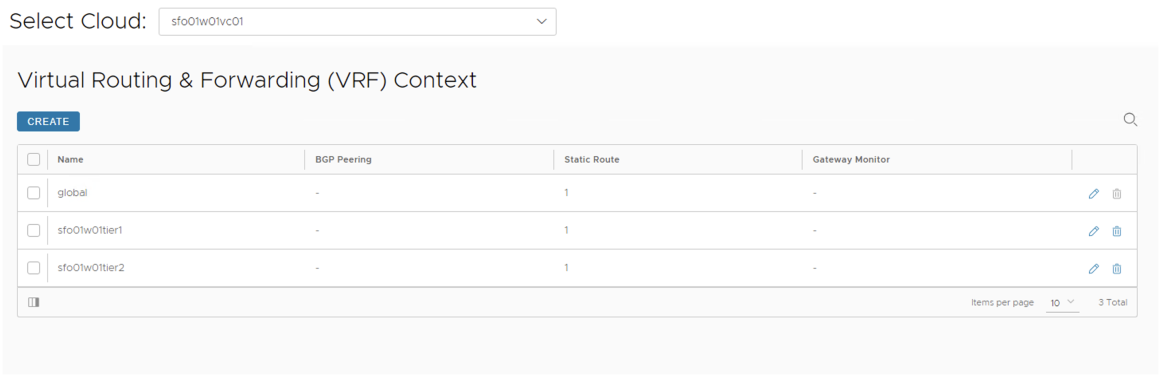
ADD

Gateway Subnet	Next Hop
0.0.0.0/0	172.16.80.1

Items per page 10 1 Total

To set the default gateway for the `sfo01-w01-vds01-tkgworkloadvip` network, click **Create** under the tier-1 gateway `sfo01w01tier2` VRF context and set the default gateway to the gateway of the VIP network subnet.

The final configuration is shown below:


Select Cloud: sfo01w01vc01

Virtual Routing & Forwarding (VRF) Context

CREATE

Name	BGP Peering	Static Route	Gateway Monitor
global	-	1	-
sfo01w01tier1	-	1	-
sfo01w01tier2	-	1	-

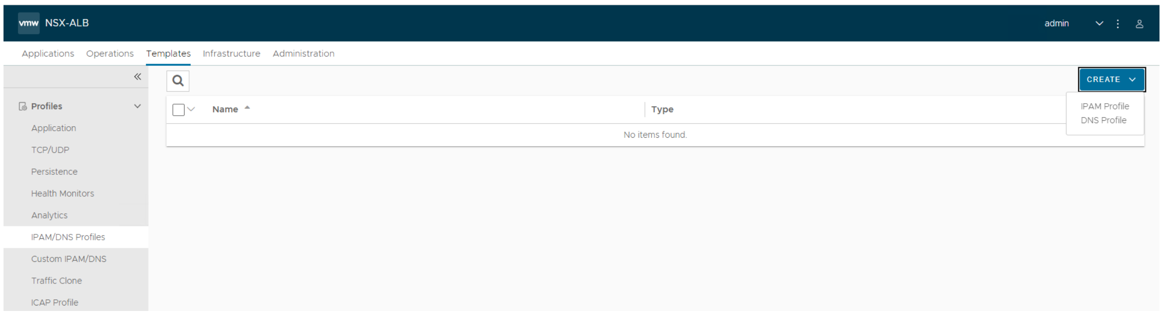
Items per page 10 3 Total

Create IPAM Profile in NSX Advanced Load Balancer and Attach to Cloud

At this point, all the required networks related to Tanzu functionality are configured in NSX Advanced Load Balancer. NSX Advanced Load Balancer provides IPAM service for TKG Cluster VIP, TKG Workload VIP and NSX ALB management network.

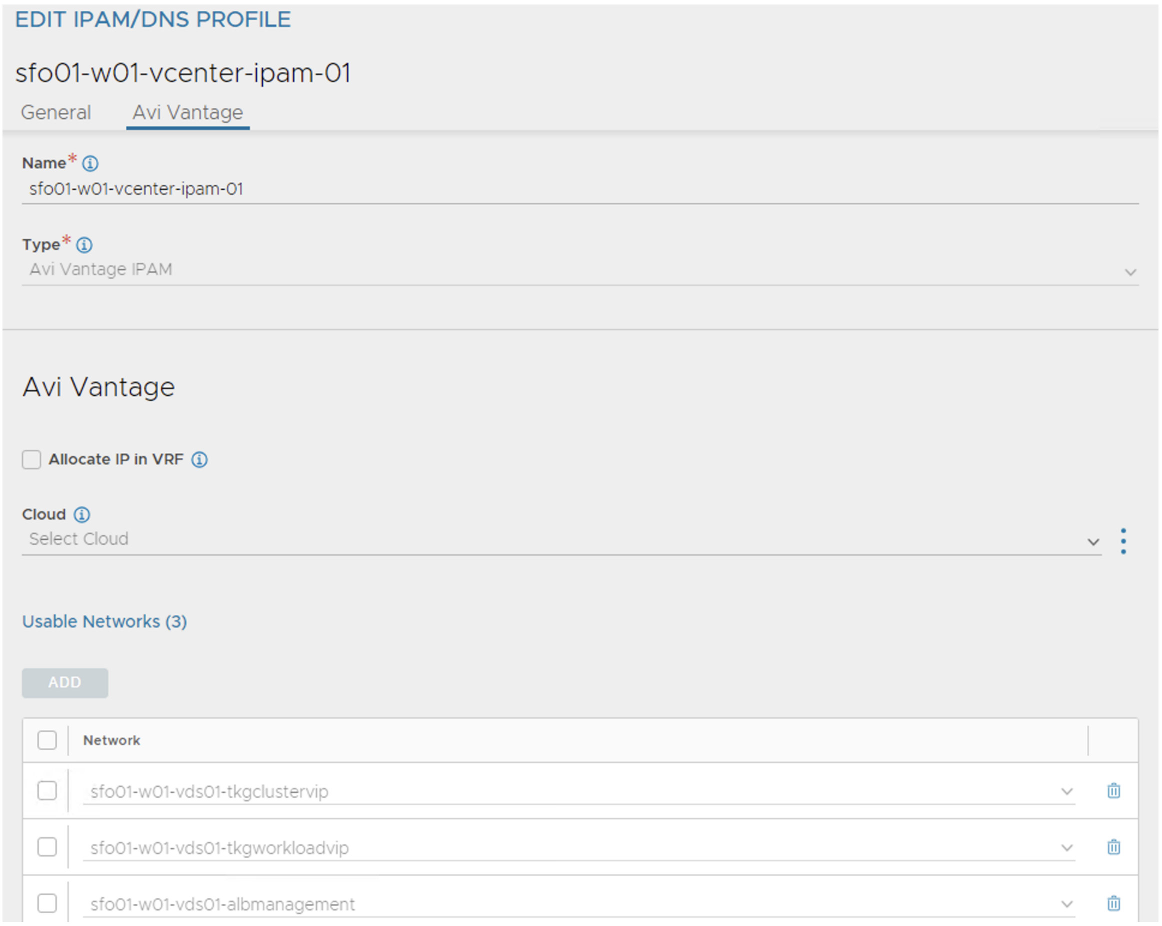
Perform the following steps to create an IPAM profile, and attach it to the NSX-T cloud created earlier:

1. Log in to NSX Advanced Load Balancer and navigate to **Templates** > **IPAM/DNS Profiles** > **Create** > **IPAM Profile**.



Specify the following details, and then click **Save**.

Parameter	Value
Name	sfo01-w01-vcenter-ipam01
Type	AVI Vintage IPAM
Cloud for Usable Networks	sfo01w01vc01
Usable Networks	sfo01-w01-vds01-management sfo01-w01-vds01-tkgworkloadvip sfo01-w01-vds01-tkgworkloadvip



EDIT IPAM/DNS PROFILE

sfo01-w01-vcenter-ipam-01

General   Avi Vantage

Name*
sfo01-w01-vcenter-ipam-01

Type*
Avi Vantage IPAM

Avi Vantage

Allocate IP in VRF

Cloud
Select Cloud

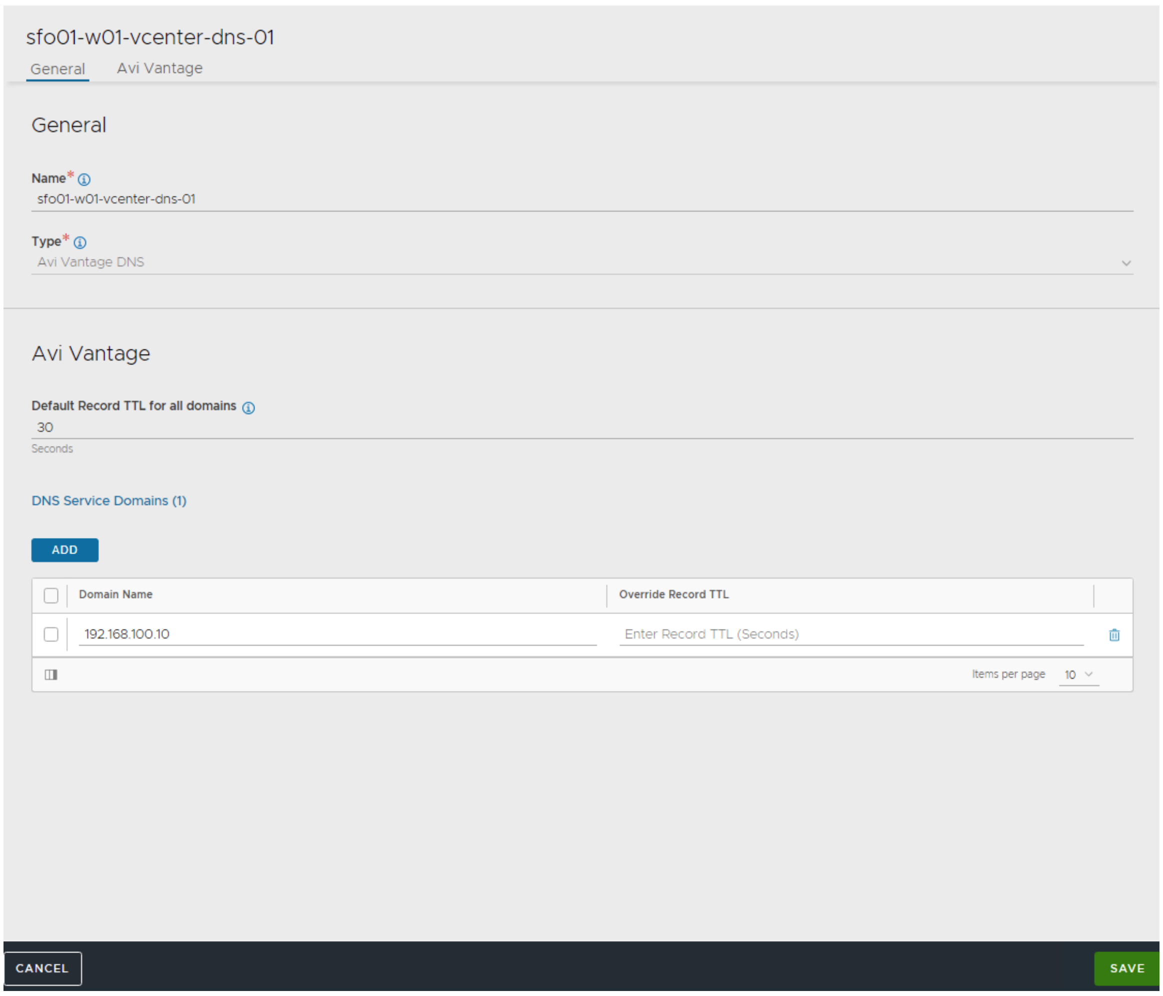
Usable Networks (3)

ADD

Network
sfo01-w01-vds01-tkgclustervip
sfo01-w01-vds01-tkgworkloadvip
sfo01-w01-vds01-albmanagement

For a single VIP network architecture, do not add sfo01-w01-vds01-tkgworkloadvip network segment to the IPAM profile.

2. Click **Create** > **DNS Profile** and provide the domain name.



sfo01-w01-vcenter-dns-01

General | Avi Vantage

General

Name*: sfo01-w01-vcenter-dns-01

Type*: Avi Vantage DNS

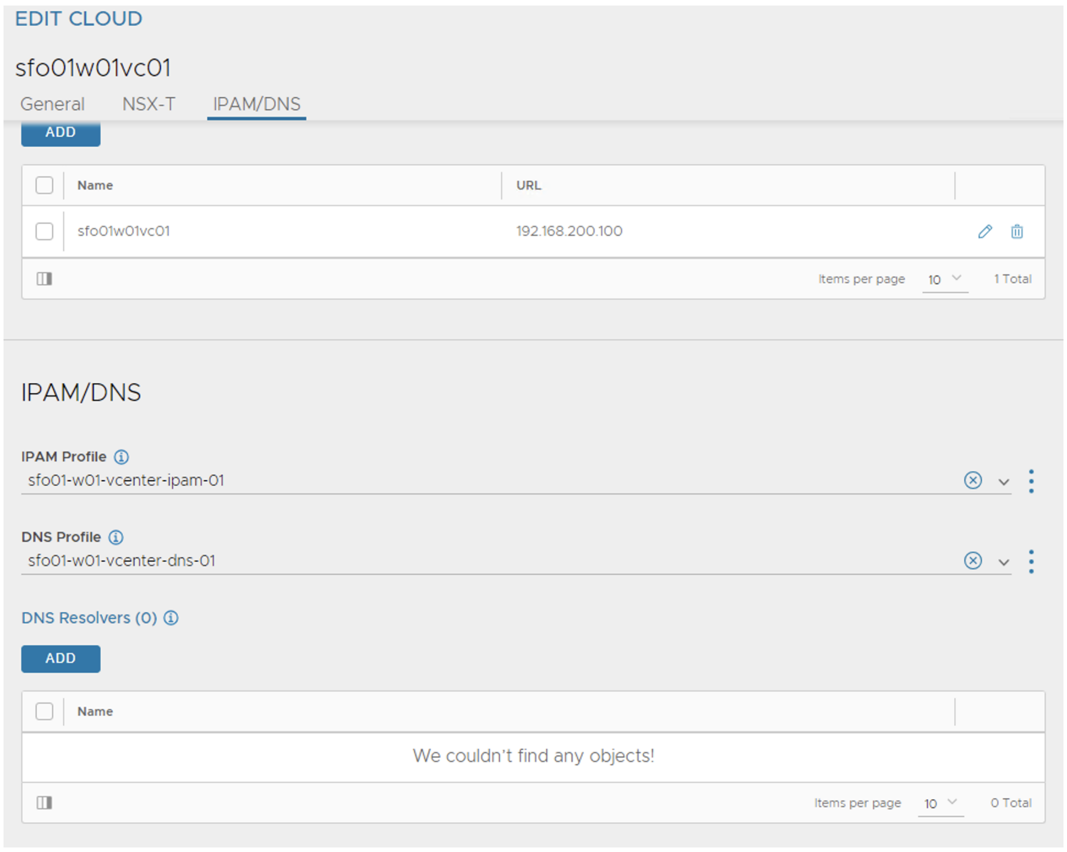
Avi Vantage

Default Record TTL for all domains: 30 Seconds

DNS Service Domains (1)

Domain Name	Override Record TTL
192.168.100.10	Enter Record TTL (Seconds)

3. Attach the IPAM and DNS profiles to the NSX-T cloud:
   1. Navigate to **Infrastructure** > **Clouds**.
   2. Edit the `sfo01w01vc01` cloud.
   3. Under IPAM/DNS section, choose the IPAM and DNS profiles created earlier, and save the updated configuration.


EDIT CLOUD

sfo01w01vc01

General NSX-T IPAM/DNS

ADD

Name	URL
sfo01w01vc01	192.168.200.100

Items per page 10 1 Total

IPAM/DNS

IPAM Profile
sfo01-w01-vcenter-ipam-01

DNS Profile
sfo01-w01-vcenter-dns-01

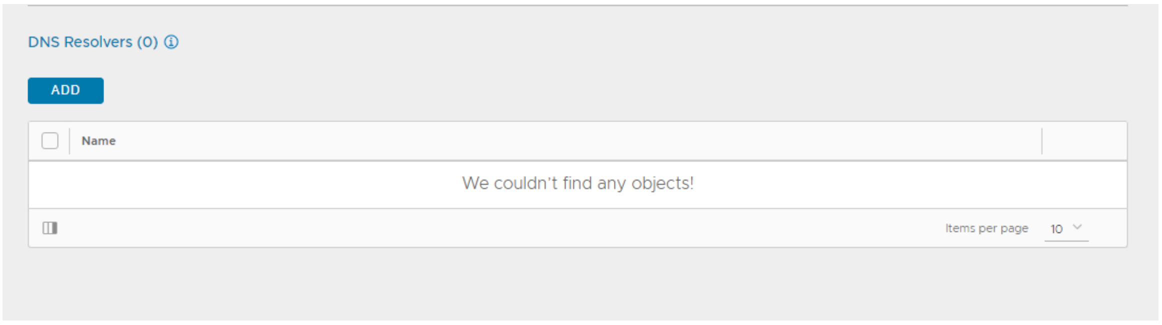
DNS Resolvers (0)

ADD

Name
We couldn't find any objects!

Items per page 10 0 Total

4. Under the section **DNS Resolvers**, click **ADD** to add the DNS server that NSX-T cloud will use to resolve the hostname or FQDN of the components that will be created later.

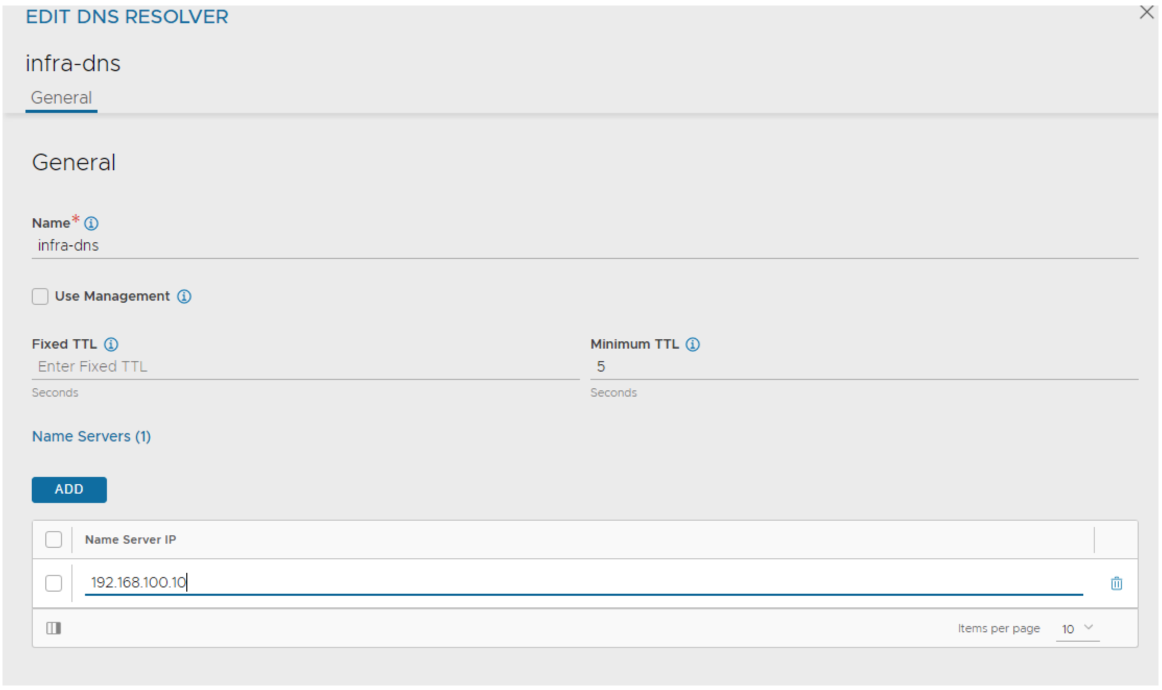

DNS Resolvers (0)

ADD

Name
We couldn't find any objects!

Items per page 10

Specify a name for the DNS and click **ADD** under **Name Servers** to add your infrastructure DNS servers. Optionally, you can specify the TTL for the DNS.


EDIT DNS RESOLVER

infra-dns

General

General

Name* infra-dns

☐ Use Management

Fixed TTL: Enter Fixed TTL (Seconds)

Minimum TTL: 5 (Seconds)

Name Servers (1)

ADD

Name Server IP
192.168.100.10

Items per page 10

This completes the NSX Advanced Load Balancer configuration. The next step is to deploy and configure a bootstrap machine which will be used to deploy and manage Tanzu Kubernetes clusters.

Deploy and Configure Bastion Host

Bastion host is the physical or virtual machine where you download the required installation images or binaries for Tanzu Kubernetes Grid installation from the Internet. The downloaded items then need to be shipped to the bootstrap machine which is inside the air-gapped environment. The bastion host needs to have a browser installed to download the binaries from the Internet.

The bastion host needs to be deployed with the following hardware configuration:

- CPU: 1
- Memory: 4 GB
- Storage (HDD): 160 GB or greater.

The following instructions are for CentOS 7. If you are using any other operating system for your bastion host, change the commands accordingly.

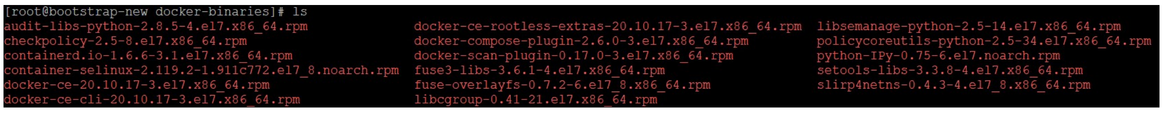
Download Binaries Required for Configuring Bastion Host

1. Download Docker Engine and associated dependencies binaries using the steps provided below

```
### Create a directory for collecting docker installation binaries:

mkdir docker-binaries && cd docker-binaries

### Add docker repository to the yum command:

yum install yum-utils -y
```

```
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-c
e.repo

### Download docker and associated dependencies:

yumdownloader --resolve docker-ce docker-ce-cli containerd.io docker-compose-pl
ugin
```

The `yumdownloader` command downloads the following binaries.



```
[root@bootstrap-new docker-binaries]# ls
audit-libs-python-2.8.5-4.el7.x86_64.rpm              docker-ce-rootless-extras-20.10.17-3.el7.x86_64.rpm  libsemanage-python-2.5-14.el7.x86_64.rpm
checkpolicy-2.5-8.el7.x86_64.rpm                      docker-compose-plugin-2.6.0-3.el7.x86_64.rpm         policycoreutils-python-2.5-34.el7.x86_64.rpm
containerd.io-1.6.6-3.1.el7.x86_64.rpm                docker-scan-plugin-0.17.0-3.el7.x86_64.rpm           python-IPy-0.75-6.el7.noarch.rpm
container-selinux-2.119.2-1.911c772.el7_8.noarch.rpm  fuse3-libs-3.6.1-4.el7.x86_64.rpm                    setools-libs-3.3.8-4.el7.x86_64.rpm
docker-ce-20.10.17-3.el7.x86_64.rpm                   fuse-overlayfs-0.7.2-6.el7_8.x86_64.rpm              slirp4netns-0.4.3-4.el7_8.x86_64.rpm
docker-ce-cli-20.10.17-3.el7.x86_64.rpm               libcgroup-0.41-21.el7.x86_64.rpm
```

2. Download Harbor OVA from the Broadcom Support.
3. Download the NSX Advanced Load Balancer OVA from Broadcom Support.
4. Download Tanzu CLI, Kubectl, and the Kubernetes OVA images from the Tanzu Kubernetes Grid product download page. Tanzu CLI and plug-ins need to be installed on the bastion host and the bootstrap machine.
5. Download the yq installation binary from mikefarah / yq GitHub repository.

Configure Bastion Host

1. Install Tanzu CLI.

```
tar -xvf tanzu-cli-linux-amd64.tar
cd ./v0.90.1/
install tanzu-cli-linux_amd64 /usr/local/bin/tanzu
chmod +x /usr/local/bin/tanzu
```

Run the `tanzu version` command to check that the correct version of tanzu is installed and executable.

```
# tanzu version

version: v0.90.1
buildDate: 2023-06-29
sha: 8945351c
```

2. Install the Tanzu CLI plug-ins.

```
tanzu plugin group search
[i] Reading plugin inventory for "projects.registry.vmware.com/tanzu_cli/plugin
s/plugin-inventory:latest", this will take a few seconds.
GROUP                DESCRIPTION      LATEST
vmware-tkg/default   Plugins for TKG  v2.3.0

tanzu plugin install --group vmware-tkg/default
[i] Installing plugin 'isolated-cluster:v0.30.1' with target 'global'
[i] Installing plugin 'management-cluster:v0.30.1' with target 'kubernetes'
[i] Installing plugin 'package:v0.30.1' with target 'kubernetes'
[i] Installing plugin 'pinniped-auth:v0.30.1' with target 'global'
[i] Installing plugin 'secret:v0.30.1' with target 'kubernetes'
[i] Installing plugin 'telemetry:v0.30.1' with target 'kubernetes'
[ok] successfully installed all plugins from group 'vmware-tkg/default:v2.3.0'
```

```
#Accept EULA
tanzu config eula accept
[ok] Marking agreement as accepted.
```

3. Download the Images.

   Before performing this step, ensure that the disk partition where you download the images has 45 GB of available space.

   ```
   tanzu isolated-cluster download-bundle --source-repo <SOURCE-REGISTRY> --tkg-ve
   rsion <TKG-VERSION> --ca-certificate <SECURITY-CERTIFICATE>
   ```

   - SOURCE-REGISTRY is the IP address or the hostname of the registry where the images are stored.
   - TKG-VERSION is the version of Tanzu Kubernetes Grid that you want to deploy in the proxied or the air-gapped environment.
   - SECURITY-CERTIFICATE is the security certificate of the registry where the images are stored. To bypass the security certificate validation, use –insecure, instead of –ca-certificate. Both the strings are optional. If you do not specify any value, the system validates the default server security certificate.

   ```
   tanzu isolated-cluster download-bundle --source-repo projects.registry.vmware.c
   om/tkg --tkg-version v2.3.0
   ```

   The image bundle in the form of TAR files, along with the publish-images-fromtar.yaml file, is downloaded. The YAML file defines the mapping between the images and the TAR files.

4. Download the Tanzu CLI plug-ins.

   Download the plugin-inventory image along with all selected plug-in images as a tar.gz file on the local disk of a machine which has internet access using the tanzu plugin download-bundle command.

   ```
   tanzu plugin download-bundle --group vmware-tkg/default:v2.3.0 --to-tar plugin_
   bundle_tkg_latest.tar.gz
   ```

5. Copy the Files to the bootstrap Machine after bootstrap Machine deployment.

   Copy the following files to the offline machine, which is the bootstrap machine in the proxied or air-gapped environment, through a USB thumb drive or other medium: * The Image TAR files. * The YAML files * Tanzu CLI plugins * Tanzu CLI, Kubectl & Carvel Tools - kbld, kapp, ytt and imgpkg

Install Harbor Image Registry

Install Harbor only if you don’t have any existing image repository in your environment.

To install Harbor, deploy an operating system of your choice with the following hardware configuration:

- vCPU: 4
- Memory: 8 GB
- Storage (HDD): 160 GB

Follow the instructions provided in Deploy an Offline Harbor Registry on vSphere to deploy and configure Harbor.

Note: This VM-based harbor deployment is only supported for hosting TKG system images in an internet-restricted or air-gapped environment. To deploy a scalable and highly-available Harbor that can manage large numbers of images for hosted apps in a production environment, deploy the Harbor package to TKG clusters as described in Install Harbor for Service Registry in Creating and Managing TKG 2.3 Workload Clusters with the Tanzu CLI.

Deploy and Configure Bootstrap Machine

The deployment of the Tanzu Kubernetes Grid management and workload clusters is facilitated by setting up a bootstrap machine where you install the Tanzu CLI and Kubectl utilities which are used to create and manage the Tanzu Kubernetes Grid instance. This machine also keeps the Tanzu Kubernetes Grid and Kubernetes configuration files for your deployments. The bootstrap machine can be a laptop, host, or server running on Linux, macOS, or Windows that you deploy management and workload clusters from.

The bootstrap machine runs a local `kind` cluster when Tanzu Kubernetes Grid management cluster deployment is started. Once the `kind` cluster is fully initialized, the configuration is used to deploy the actual management cluster on the backend infrastructure. After the management cluster is fully configured, the local `kind` cluster is deleted and future configurations are performed with the Tanzu CLI.

For this deployment, a Photon-based virtual machine is used as the bootstrap machine. For more information about configuring a macOS or a Windows machine, see Install the Tanzu CLI and Other Tools.

The bootstrap machine must meet the following prerequisites:

- A minimum of 6 GB of RAM, 2-core CPU, 160 Storage GB.
- System time is synchronized with a Network Time Protocol (NTP) server.
- Docker and containerd binaries are installed. For instructions on how to install Docker, see the Docker documentation.
- Ensure that the bootstrap VM is connected to Tanzu Kubernetes Grid management network, `sfo01-w01-vds01-tkgmanagement`.

To install Tanzu CLI, Tanzu Plug-ins, and Kubectl utility on the bootstrap machine, follow the instructions below: 1. Copy Files to bootstrap Machine.

Copy the following files downloaded in Bastion Host through a USB thumb drive or other medium: * Image TAR files * YAML files * Tanzu CLI Plugins

1. Copy following Linux CLI packages from Bastion Host to the bootstrap machine:
    - VMware Tanzu CLI v0.90.1 for Linux
    - kubectl cluster CLI v1.26.5 for Linux
    - Carvel tools - kbld, kapp, ytt & imgpkg
2. Run the following commands to install Tanzu Kubernetes Grid CLI, kubectl CLIs, and Carvel tools:

```
## Install required packages
install tar, zip, unzip, wget

## Install Tanzu Kubernetes Grid CLI
```

```
tar -xvf tanzu-cli-linux-amd64.tar
cd ./v0.90.1/
install tanzu-cli-linux_amd64 /usr/local/bin/tanzu
chmod +x /usr/local/bin/tanzu

## Verify Tanzu CLI version

# tanzu version

version: v0.90.1
buildDate: 2023-06-29
sha: 8945351c
```

3. Log in to the private registry on the offline machine.

```
docker login <URL>

docker login harbor.sfo01.rainpole.vmw
Username: admin
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

If your private registry uses a self-signed certificate, save the CA certificate of the registry in `/etc/docker/certs.d/registry.example.com/ca.crt`.

4. Upload the Images to the Private Registry.

```
tanzu isolated-cluster upload-bundle --source-directory <SOURCE-DIRECTORY> --de
stination-repo <DESTINATION-REGISTRY> --ca-certificate <SECURITY-CERTIFICATE>
```

   - SOURCE-DIRECTORY is the path to the location where the image TAR files are stored.
   - DESTINATION-REGISTRY is the path to the private registry where the images will be hosted in the air-gapped environment.
   - SECURITY-CERTIFICATE is the security certificate of the private registry where the images will be hosted in the proxied or the air-gapped environment.

```
Example:- t Example: tanzu isolated-cluster upload-bundle --source-directory /h
ome/test/tkg-images/ --destination-repo harbor.sfo01.rainpole.vmw/tkgm-images -
-ca-certificate /etc/docker/certs.d/harbor.sfo01.rainpole.vmw/harbor.sfo01.rain
pole.vmw-ca.crt
```

5. Upload the CLI plug-ins bundle to harbor repository.

```
tanzu plugin upload-bundle --tar ./plugin_bundle_tkg_latest.tar.gz --to-repo ha
rbor.sfo01.rainpole.vmw/tkgm-images/
```

6. Run tanzu plugin source command to set default discovery source to the images uploaded in internal harbor registry.

```
tanzu plugin source update default --uri harbor.sfo01.rainpole.vmw/tkgm-images/
plugin-inventory:latest
```

> You can skip step 4 and 5 if your Bastion host has direct access to the private registry. You can directly upload the files from Bastion to the private registry.

7. Install the kubectl utility.

```
gunzip kubectl-linux-v1.26.5+vmware.2.gz
mv gunzip kubectl-linux-v1.26.5+vmware.2 /usr/local/bin/kubectl && chmod +x /us
r/local/bin/kubectl

Run the kubectl version --short=true to check that the correct version of kubec
tl is installed and executable.
```

8. Configure the environment variables.

   In an air-gapped environment, if you run the `tanzu init` or `tanzu plugin sync` commands, the command hangs and times out after some time with the following error:

   By default, the Tanzu global config file, `config.yaml`, which gets created when you first run `tanzu init` command, points to the repository URL https://projects.registry.vmware.com to fetch the Tanzu plugins for installation. Since there is no Internet in the environment, the commands fails after some time.

   To ensure that Tanzu Kubernetes Grid always pulls images from the local private registry, run the Tanzu `export` command to add `TKG_CUSTOM_IMAGE_REPOSITORY` to the global Tanzu CLI configuration file, `~/.config/tanzu/config.yaml`.

   If your image registry is configured with a public signed CA certificate, set the following environment variables:

```
export TKG_CUSTOM_IMAGE_REPOSITORY=custom-image-repository.io/yourproject

export TKG_CUSTOM_IMAGE_REPOSITORY_SKIP_TLS_VERIFY=false
```

   If your registry solution uses self-signed certificates, also add TKG_CUSTOM_IMAGE_REPOSITORY_CA_CERTIFICATE in base64-encoded format to the global Tanzu CLI configuration file. For self-signed certificates, set the following environment variables:

```
export TKG_CUSTOM_IMAGE_REPOSITORY=custom-image-repository.io/yourproject

export TKG_CUSTOM_IMAGE_REPOSITORY_SKIP_TLS_VERIFY=false

export TKG_CUSTOM_IMAGE_REPOSITORY_CA_CERTIFICATE LS0t[...]tLS0tLQ==

# Example
root@photon-a17e54311cf [ ~ ]# export TKG_CUSTOM_IMAGE_REPOSITORY=harbor.sfo01.
```

```
rainpole.vmw/tkgm-images
root@photon-a17e54311cf [ ~ ]# export TKG_CUSTOM_IMAGE_REPOSITORY_SKIP_TLS_VERI
FY=false
root@photon-a17e54311cf [ ~ ]# export TKG_CUSTOM_IMAGE_REPOSITORY_CA_CERTIFICAT
E=LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUZnekNDQTJ1Z0F3SUJBZ0lVUXhvVVV5VnpwU
VlkdUlXbStwL3dxZ0JDSFVrd0RRWUpLb1pJaHZjTkFRRUwKQlFBd1VURUxNQWtHQTFVRUJoTUNRMDR4
RERBS0JnTlZCQWdNQTFCRlN6RVFNQTRHQTFVRUJ3d0hRbVZwU21sdQpaekVQTUEwR0ExVUVDZ3dHVms
xM1lYSmxNUkV3RHdZRFZRUUREQWhJWVhKaWIzSkRRVEFlRncweU16QTRNRGN3Ck5qVTBOVGRhRncwek
16QTRNRFF3TmpVME5UZGFNRkV4Q3pBSkJnTlZCQVlUQWtOT01Rd3dDZ1lEVlFRSURBTlEKUlVzeEVEQ
U9CZ05WQkFjTUIwSmxhVXBwYm1jeER6QU5CZ05WQkFvTUJsWk5kMkZ5WlRFUk1BOEdBMVVFQXd3SQpT
R0Z5WW05eVEwRXdnZ0lpTUEwR0NTcUdTSIzRFFFQkFRVUFBNElDRHdBd2dnSUtBb0lDQVFDckFtajB
VR1lUCjVobm5OeTNIMVdkcUhIejBHbFphWVpGQlhvQVhlWitZa2ljeXJhOFpFeVJCTFBKZVNOcldzMW
45dFp1RUg2WDgKM3JJbzUwdzhrZTYwYU9OU80VlVXZExxZKzYxYUxrNlpkNGkzcU9IQTVKSFU1cE5De
nU0bThaQ1F1bUp2SzFVSQpwQ1lQNnFtNGxSQUFvWFVzWGZ0S24vRkJ4bGdReTNhUjJ1Y0IzdXA5UndD
RllDLzA5TVd5ZjErUmhja3ZvWExRCmppUWx4aHZ0NFpxeG12b09KMi9lbUorTHBqbENZaXBRVkNwN3N
peVM1ZGIvRmw5U1VWSFVlRDhpdzgxKzZTaEEKenRpZXY1U20zbGE0UGw1cUU4Vm9EVEJTUXRpMmQzUX
B2M05IR0Z4UUNWMjVTM1BVcWxPQ1Z0dDFtSHRjZEhSUQpTOEN2SE9DVlczRHd5ZXB2Y3M4STJFYlU3M
0VQN3JKOEJtL21GQUFwd2F1NXpjMmtxRGJRclMrTjhEeWFpcGY4Ckgxa3FFVlU2VmEvbCtjZHFSbkc1
ZWVmM21vT3dKMDBPa2ZqSWtCUExxSOC9iZVdDR0R4RVE1RkZlRXhxSTdhNUIKcm1FK25vTTBjMFRQSFp
5Z3c4UnZVU3hQbGJ3MFZkVzA5MHdQOHB3WGFGYVBPMmhtT2lvVGJiVkpkMVhqaGRFZAoxWTdmQktYKz
l6UUxjb3dlcmF5cDRaSUlSTTNZnRaZkc5bjErN3pmcVBpamVoTGdiUkUrcHVaaUQwbi9xUXIvClNzO
WU5MXZQR1U2NW82VjYvL2Y5MVZmVHJRUmVJV0JWUWNubCtobGpPOVpUeHRRMlVwRnU3TDRwT2RJZ1JH
RTgKcDVjZE91TDF1cWltTi8rTW5hckV3K2JFalc0ajF3cnFQUUlEQVFBQm8xTXdVVEFkQmdOVkhRNEV
GZ1FVdEozbQpvR3RrcVJiVGl2ZFkzQWxnMTZUOWpCZ3dId1lEVlIwakJCZ3dGb0FVdEozbW9HdGxUm
JUaXZkWTNBbGcxNlQ5CmpCZ3dEd1lEVlIwVEFFRSC9CQVV3QXdFQi96QU5CZ2txaGtpRzl3MEJBUXNGQ
UFPQ0FnRUFUMFBGcUduUjVRdEoKZm1PMTdwSmVDK0IwM3NOVGozS0VzZjUybmJXUnprWm9xNjMrRDZn
VURtU3FMMm1RSExxRUy9WNWhadmIzZTFCTApDSXpEZmF5cG14K2k3MnlIbVRRNGRBUW5NT2hCUm8rMEN
VOGpZa0t4TjllNlNhOTlERk0yVE5Bb1pzcEs4ZEVUCmlUa0p2bHlHS2tyellkNEFvczN3dU9uVE85VW
xCK1FqQmI4TUVDc3lMR2U4VGxJMk4vOFdwMFMySG1QUHVNeU8KemJ0RUUrSWJXaW1qT1lLRU44cHlUY
3plelVnZHpGcXJ3bjVKdWZSejdockV6MnE5ZG9sYm0vTGRNK3pnUHVzZwo0a3lnVjlxWU5KcldvVExxX
bFJKNzRyQmFOZFpYM3BOV1VGRGtjQ3JkSkloWFFFSWdWUDNWa2xJZ21zUUhTU3IrCktJYSt5R1p3MU5
hODZML2R0djNrR1ovN2VRMHNHVzVpS3R3VmY5UnBqYTdXL3ZhTTA1OTdFWGNSSGZ2cHRxeFQKcjNOOcn
FTQmZkTlJtNStXOVh3c0RxNDl3dFdERE52OHNDS2JrenI2Q0JHYUxXSHFGRWtCOHpiTGlJQVBYd2Vqa
ApSOTh5TnY1ZjBzb0ltZlg1R3REY2RMZjd3Gg0UGlvRGloZklrRXVzd0twVGN1WWo3clR0SnFYTFV2
b25jZkV0Ck93cytHa2c5L0ZHd0p6ZkJlYUNrNHVXbGw5bC9JanZ6azdydkc0cC0Z1VXa2tMVi9SdmVzbFl
EZVlSUXBnYzRybzgKdFFpMThvb3V4RGZuMTlSS2JPVjNtNm5uTlYwdzlHdjZiUGxxbjlRaDB0MWJOaU
xwZThJeWhGb0VPOFpVYTBnSApCZ29PbXlGZHQ0VTlQclIvZTdNcWZzM2tQVjZkbmtzPQotLS0tLUVOR
CBDRVJUSUZJQ0FURS0tLS0t
```

If we reboot the VM , this setting will go to default

9. Install the Tanzu CLI plug-ins.

```
root@photon-a17e54311cf [ ~ ]# tanzu plugin group search
GROUP                DESCRIPTION      LATEST
vmware-tkg/default   Plugins for TKG  v2.3.0

root@photon-a17e54311cf [ ~ ]# tanzu plugin install --group vmware-tkg/default
[i] Installing plugin 'isolated-cluster:v0.30.1' with target 'global'
[i] Plugin binary for 'isolated-cluster:v0.30.1' found in cache
[i] Installing plugin 'management-cluster:v0.30.1' with target 'kubernetes'
[i] Plugin binary for 'management-cluster:v0.30.1' found in cache
[i] Installing plugin 'package:v0.30.1' with target 'kubernetes'
[i] Plugin binary for 'package:v0.30.1' found in cache
```

```
[i] Installing plugin 'pinniped-auth:v0.30.1' with target 'global'
[i] Plugin binary for 'pinniped-auth:v0.30.1' found in cache
[i] Installing plugin 'secret:v0.30.1' with target 'kubernetes'
[i] Plugin binary for 'secret:v0.30.1' found in cache
[i] Installing plugin 'telemetry:v0.30.1' with target 'kubernetes'
[i] Plugin binary for 'telemetry:v0.30.1' found in cache
[ok] successfully installed all plugins from group 'vmware-tkg/default:v2.3.0'
```

After installing the tanzu plug-ins, run the `tanzu plugin list` command to check the plug-in version and installation status.

10. Install Carvel tools.

    Tanzu Kubernetes Grid uses the following tools from the Carvel open-source project:

    - ytt - a command-line tool for templating and patching YAML files. You can also use ytt to collect fragments and piles of YAML into modular chunks for easy re-use.
    - kapp - the application deployment CLI for Kubernetes. It allows you to install, upgrade, and delete multiple Kubernetes resources as one application.
    - kbld - an image-building and resolution tool.
    - imgpkg - a tool that enables Kubernetes to store configurations and the associated container images as OCI images, and to transfer these images.
    - Install ytt.

      ```
      gunzip ytt-linux-amd64-v0.45.0+vmware.2.gz

      chmod ugo+x ytt-linux-amd64-v0.45.0+vmware.2 &&  mv ./ytt-linux-amd64-v0.
      45.0+vmware.2 /usr/local/bin/ytt
      ```

      Run `ytt --version` to check that the correct version of ytt is installed and executable.
    - Install kapp.

      ```
      gunzip kapp-linux-amd64-v0.55.0+vmware.2.gz

      chmod ugo+x kapp-linux-amd64-v0.55.0+vmware.2 && mv ./kapp-linux-amd64-v
      0.55.0+vmware.2 /usr/local/bin/kapp
      ```

      Run `kapp --version` to check that the correct version of kapp is installed and executable.
    - Install kbld.

      ```
      gunzip kbld-linux-amd64-v0.37.0+vmware.2.gz

      chmod ugo+x kbld-linux-amd64-v0.37.0+vmware.2 && mv ./kbld-linux-amd64-v
      0.37.0+vmware.2 /usr/local/bin/kbld
      ```

      Run `kbld --version` to check that the correct version of kbld is installed and executable.
    - Install imgpkg.

      ```
      gunzip imgpkg-linux-amd64-v0.36.0+vmware.2.gz
      chmod ugo+x imgpkg-linux-amd64-v0.36.0+vmware.2 && mv ./imgpkg-linux-amd6
      4-v0.36.0+vmware.2 /usr/local/bin/imgpkg
      ```

Run `imgpkg --version` to check that the correct version of imgpkg is installed and executable.

11. Install yq.

    yq a lightweight and portable command-line YAML processor. Click here to download yq.

    ```
    tar -zxvf yq_linux_amd64.tar.gz

    mv yq_linux_amd64 /usr/local/bin/
    ```

    Run the `yq -V` command to check that the correct version of yq is installed and executable.

12. Run the following commands to start the Docker service and enable it to start at boot. Photon OS has Docker installed by default.

    ```
    ## Check Docker service status
    systemctl status docker

    ## Start Docker Service
    systemctl start docker

    ## To start Docker Service at boot
    systemctl enable docker
    ```

13. Execute the following commands to ensure that the bootstrap machine uses cgroup v1.

    ```
    docker info | grep -i cgroup

    ## You should see the following
    Cgroup Driver: cgroupfs
    ```

14. Create an SSH key-pair.

    This is required for Tanzu CLI to connect to vSphere from the bootstrap machine. The public key part of the generated key will be passed during the Tanzu Kubernetes Grid management cluster deployment.

    ```
    ### Generate public/Private key pair.

    ssh-keygen -t rsa -b 4096 -C "email@example.com"

    ### Add the private key to the SSH agent running on your machine and enter the
    password you created in the previous step

    ssh-add ~/.ssh/id_rsa

    ### If the above command fails, execute "eval $(ssh-agent)" and then rerun the
    command.
    ```

    Make a note of the public key from the file **$home/.ssh/id_rsa.pub**. You need this while creating a config file for deploying the Tanzu Kubernetes Grid management cluster.

15. If your bootstrap machine runs Linux or Windows Subsystem for Linux, and it has a Linux kernel built after the May 2021 Linux security patch, for example Linux 5.11 and 5.12 with Fedora, run the following command.

```
sudo sysctl net/netfilter/nf_conntrack_max=131072
```

Import Base Image Template for Tanzu Kubernetes Grid Cluster Deployment

Before you proceed with the management cluster creation, ensure that the base image template is imported into vSphere and is available as a template. To import a base image template into vSphere:

1. Go to the Tanzu Kubernetes Grid downloads page and download a Tanzu Kubernetes Grid OVA for the cluster nodes.
   - For the management cluster, this must be either Photon or Ubuntu based Kubernetes v1.26.5 OVA.

     > Custom OVA with a custom Tanzu Kubernetes release (TKr) is also supported, as described in Build Machine Images.

   - For workload clusters, OVA can have any supported combination of OS and Kubernetes version, as packaged in a Tanzu Kubernetes release.

   > Ensure that you download the most recent OVA base image templates in the event of security patch releases. You can find updated base image templates that include security patches on the Tanzu Kubernetes Grid product download page.

2. In the vSphere client, right-click an object in the vCenter Server inventory and select **Deploy OVF template**.
3. Select Local file, click the button to upload files, and go to the downloaded OVA file on your local machine.
4. Follow the installer prompts to deploy a VM from the OVA.
5. Click **Finish** to deploy the VM. When the OVA deployment finishes, right-click the VM and select **Template** > **Convert to Template**.

   > Do not power on the VM before you convert it to a template.

6. **If using non administrator SSO account**: In the VMs and Templates view, right-click the new template, select **Add Permission**, and assign the **tkg-user** to the template with the **TKG role**.

For more information about creating the user and role for Tanzu Kubernetes Grid, see Required Permissions for the vSphere Account.

Management Cluster Configuration Template

The templates include all of the options that are relevant to deploying management clusters on vSphere. You can copy this template and use it to deploy management clusters to vSphere.

- The environment variables that you have set, override values from a cluster configuration file. To use all settings from a cluster configuration file, unset any conflicting environment variables before you deploy the management cluster from the CLI.

- Image repository configuration is very important details which will not be part of default config file when we are creating from TKG UI.

- Insert the key `AVI_NSXT_T1LR`. The value of this key is the tier-1 gateway where you have connected the `tkg management network` network.

```
#! ---------------------------------------------------------------------
#! Basic cluster creation configuration
#! ---------------------------------------------------------------------

CLUSTER_NAME:
CLUSTER_PLAN: <dev/prod>
INFRASTRUCTURE_PROVIDER: vsphere
ENABLE_CEIP_PARTICIPATION: <true/false>
ENABLE_AUDIT_LOGGING: <true/false>
CLUSTER_CIDR: 100.96.0.0/11
SERVICE_CIDR: 100.64.0.0/13
# CAPBK_BOOTSTRAP_TOKEN_TTL: 30m

#! ---------------------------------------------------------------------
#! vSphere configuration
#! ---------------------------------------------------------------------

VSPHERE_SERVER:
VSPHERE_USERNAME:
VSPHERE_PASSWORD:
VSPHERE_DATACENTER:
VSPHERE_RESOURCE_POOL:
VSPHERE_DATASTORE:
VSPHERE_FOLDER:
VSPHERE_NETWORK: <tkg-management-network>
VSPHERE_CONTROL_PLANE_ENDPOINT: #Leave blank as VIP network is configured in NSX ALB a
nd IPAM is configured with VIP network

# VSPHERE_TEMPLATE:

VSPHERE_SSH_AUTHORIZED_KEY:
VSPHERE_TLS_THUMBPRINT:
VSPHERE_INSECURE: <true/false>
DEPLOY_TKG_ON_VSPHERE7: true

#! ---------------------------------------------------------------------
#! Node configuration
#! ---------------------------------------------------------------------

# SIZE:
# CONTROLPLANE_SIZE:
# WORKER_SIZE:
# OS_NAME: ""
# OS_VERSION: ""
# OS_ARCH: ""
# VSPHERE_NUM_CPUS: 2
# VSPHERE_DISK_GIB: 40
# VSPHERE_MEM_MIB: 4096
# VSPHERE_CONTROL_PLANE_NUM_CPUS: 2
# VSPHERE_CONTROL_PLANE_DISK_GIB: 40
# VSPHERE_CONTROL_PLANE_MEM_MIB: 8192
# VSPHERE_WORKER_NUM_CPUS: 2
# VSPHERE_WORKER_DISK_GIB: 40
# VSPHERE_WORKER_MEM_MIB: 4096

#! ---------------------------------------------------------------------
#! NSX Advanced Load Balancer configuration
#! ---------------------------------------------------------------------

AVI_ENABLE: true
AVI_CONTROL_PLANE_HA_PROVIDER: true
AVI_CONTROLLER:
AVI_USERNAME: ""
AVI_PASSWORD: ""
AVI_CLOUD_NAME:
AVI_SERVICE_ENGINE_GROUP:
AVI_MANAGEMENT_CLUSTER_SERVICE_ENGINE_GROUP:
AVI_DATA_NETWORK:
AVI_DATA_NETWORK_CIDR:
AVI_MANAGEMENT_CLUSTER_VIP_NETWORK_NAME:
AVI_MANAGEMENT_CLUSTER_VIP_NETWORK_CIDR:
AVI_CA_DATA_B64: ""
AVI_LABELS: ""
AVI_NSXT_T1LR: ""

# AVI_DISABLE_STATIC_ROUTE_SYNC: true
# AVI_INGRESS_DEFAULT_INGRESS_CONTROLLER: false
# AVI_INGRESS_SHARD_VS_SIZE: ""
# AVI_INGRESS_SERVICE_TYPE: ""
# AVI_INGRESS_NODE_NETWORK_LIST: ""
# AVI_NAMESPACE: "tkg-system-networking"
# AVI_DISABLE_INGRESS_CLASS: true
# AVI_AKO_IMAGE_PULL_POLICY: IfNotPresent
# AVI_ADMIN_CREDENTIAL_NAME: avi-controller-credentials
# AVI_CA_NAME: avi-controller-ca
#AVI_CONTROLLER_VERSION:# Required for NSX Advanced Load Balancer (ALB) v21.1.x.

#! ---------------------------------------------------------------------
#! Image repository configuration
#! ---------------------------------------------------------------------

TKG_CUSTOM_IMAGE_REPOSITORY: ""
TKG_CUSTOM_IMAGE_REPOSITORY_SKIP_TLS_VERIFY: false
TKG_CUSTOM_IMAGE_REPOSITORY_CA_CERTIFICATE: ""

#! ---------------------------------------------------------------------
#! Machine Health Check configuration
#! ---------------------------------------------------------------------

ENABLE_MHC:
ENABLE_MHC_CONTROL_PLANE: <true/false>
ENABLE_MHC_WORKER_NODE: <true/flase>

#! ---------------------------------------------------------------------
#! Identity management configuration
#! ---------------------------------------------------------------------

IDENTITY_MANAGEMENT_TYPE: "none"

#! Settings for IDENTITY_MANAGEMENT_TYPE: "oidc"
# CERT_DURATION: 2160h
# CERT_RENEW_BEFORE: 360h
# OIDC_IDENTITY_PROVIDER_CLIENT_ID:
# OIDC_IDENTITY_PROVIDER_CLIENT_SECRET:
# OIDC_IDENTITY_PROVIDER_GROUPS_CLAIM: groups
# OIDC_IDENTITY_PROVIDER_ISSUER_URL:
# OIDC_IDENTITY_PROVIDER_SCOPES: "email,profile,groups"
# OIDC_IDENTITY_PROVIDER_USERNAME_CLAIM: email

#! Settings for IDENTITY_MANAGEMENT_TYPE: "ldap"
# LDAP_BIND_DN:
# LDAP_BIND_PASSWORD:
# LDAP_HOST:
# LDAP_USER_SEARCH_BASE_DN:
# LDAP_USER_SEARCH_FILTER:
# LDAP_USER_SEARCH_USERNAME: userPrincipalName
# LDAP_USER_SEARCH_ID_ATTRIBUTE: DN
# LDAP_USER_SEARCH_EMAIL_ATTRIBUTE: DN
# LDAP_USER_SEARCH_NAME_ATTRIBUTE:
# LDAP_GROUP_SEARCH_BASE_DN:
# LDAP_GROUP_SEARCH_FILTER:
# LDAP_GROUP_SEARCH_USER_ATTRIBUTE: DN
# LDAP_GROUP_SEARCH_GROUP_ATTRIBUTE:
# LDAP_GROUP_SEARCH_NAME_ATTRIBUTE: cn
# LDAP_ROOT_CA_DATA_B64:
```

To create the Management Cluster, run the following command:

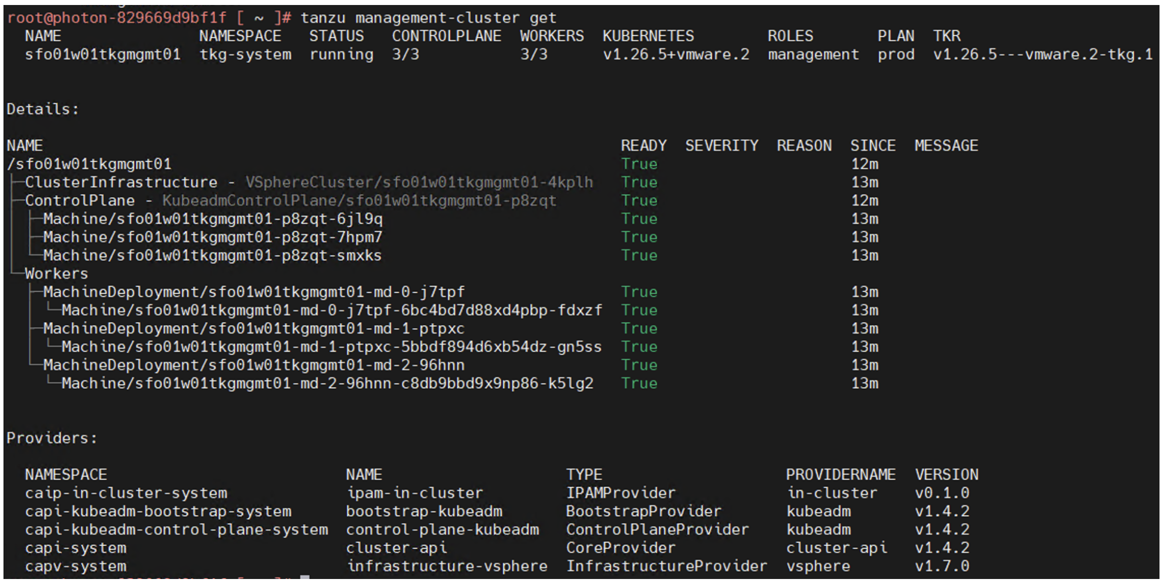
```
tanzu management-cluster create --file config.yaml
```

- For a full list of configurable values and to know more about the fields present in the template file, see Create a Management Cluster Configuration File.
- Create a file using the values provided in the template and save the file with the `.yaml` extension. A sample yaml file used for management cluster deployment is provided in the Appendix section for your reference.
- After you have created or updated the cluster configuration file, you can deploy a management cluster by running the `tanzu mc create --file CONFIG-FILE` command, where `CONFIG-FILE` is the name of the configuration file.
- The cluster deployment logs are streamed in the terminal when you run the `tanzu mc create` command. The first run of `tanzu mc create` takes longer than subsequent runs because it has to pull the required Docker images into the image store on your bootstrap machine. Subsequent runs do not require this step, and thus the process is faster.
- While the cluster is being deployed, you will find that a virtual service will be created in NSX Advanced Load Balancer and new SEs will be deployed in vCenter by NSX ALB and the service engines will be mapped to the SE group `sfo01m01segroup01`.
- Now, you can access the Tanzu Kubernetes Grid management cluster from the bootstrap machine and perform additional tasks such as verifying the management cluster health and deploying the

workload clusters, and so on.

- To get the status of the Tanzu Kubernetes Grid management cluster, run the following command:

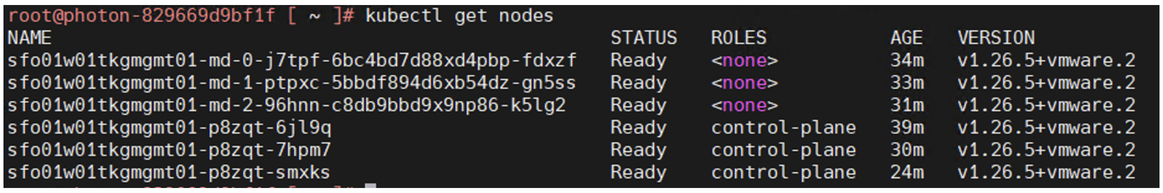
```
tanzu management-cluster get
```



```
root@photon-829669d9bf1f [ ~ ]# tanzu management-cluster get
  NAME                 NAMESPACE   STATUS   CONTROLPLANE  WORKERS  KUBERNETES         ROLES       PLAN  TKR
  sfo01w01tkgmgmt01   tkg-system  running  3/3           3/3      v1.26.5+vmware.2  management  prod  v1.26.5---vmware.2-tkg.1


Details:

NAME                                                                 READY  SEVERITY  REASON  SINCE  MESSAGE
/sfo01w01tkgmgmt01                                                   True                     12m
├─ClusterInfrastructure - VSphereCluster/sfo01w01tkgmgmt01-4kplh     True                     13m
├─ControlPlane - KubeadmControlPlane/sfo01w01tkgmgmt01-p8zqt         True                     12m
│ ├─Machine/sfo01w01tkgmgmt01-p8zqt-6jl9q                            True                     13m
│ ├─Machine/sfo01w01tkgmgmt01-p8zqt-7hpm7                            True                     13m
│ └─Machine/sfo01w01tkgmgmt01-p8zqt-smxks                            True                     13m
└─Workers
  ├─MachineDeployment/sfo01w01tkgmgmt01-md-0-j7tpf                   True                     13m
  │ └─Machine/sfo01w01tkgmgmt01-md-0-j7tpf-6bc4bd7d88xd4pbp-fdxzf    True                     13m
  ├─MachineDeployment/sfo01w01tkgmgmt01-md-1-ptpxc                   True                     13m
  │ └─Machine/sfo01w01tkgmgmt01-md-1-ptpxc-5bbdf894d6xb54dz-gn5ss    True                     13m
  └─MachineDeployment/sfo01w01tkgmgmt01-md-2-96hnn                   True                     13m
    └─Machine/sfo01w01tkgmgmt01-md-2-96hnn-c8db9bbd9x9np86-k5lg2     True                     13m


Providers:

  NAMESPACE                          NAME                    TYPE                    PROVIDERNAME  VERSION
  caip-in-cluster-system             ipam-in-cluster         IPAMProvider            in-cluster    v0.1.0
  capi-kubeadm-bootstrap-system      bootstrap-kubeadm       BootstrapProvider       kubeadm       v1.4.2
  capi-kubeadm-control-plane-system  control-plane-kubeadm   ControlPlaneProvider    kubeadm       v1.4.2
  capi-system                        cluster-api             CoreProvider            cluster-api   v1.4.2
  capv-system                        infrastructure-vsphere  InfrastructureProvider  vsphere       v1.7.0
```

- Use `kubectl get nodes` command to get the status of the Tanzu Kubernetes Grid management cluster nodes.

```
kubectl get nodes
```



```
root@photon-829669d9bf1f [ ~ ]# kubectl get nodes
NAME                                              STATUS   ROLES           AGE   VERSION
sfo01w01tkgmgmt01-md-0-j7tpf-6bc4bd7d88xd4pbp-fdxzf   Ready    <none>          34m   v1.26.5+vmware.2
sfo01w01tkgmgmt01-md-1-ptpxc-5bbdf894d6xb54dz-gn5ss   Ready    <none>          33m   v1.26.5+vmware.2
sfo01w01tkgmgmt01-md-2-96hnn-c8db9bbd9x9np86-k5lg2    Ready    <none>          31m   v1.26.5+vmware.2
sfo01w01tkgmgmt01-p8zqt-6jl9q                      Ready    control-plane   39m   v1.26.5+vmware.2
sfo01w01tkgmgmt01-p8zqt-7hpm7                      Ready    control-plane   30m   v1.26.5+vmware.2
sfo01w01tkgmgmt01-p8zqt-smxks                      Ready    control-plane   24m   v1.26.5+vmware.2
```

The Tanzu Kubernetes Grid management cluster is successfully deployed and now you can proceed with creating shared services and workload clusters.

Configure AKO Deployment Config (ADC) for Workload Clusters

Tanzu Kubernetes Grid management clusters with NSX Advanced Load Balancer are deployed with 2 AKODeploymentConfigs.

- `install-ako-for-management-cluster`: default configuration for management cluster
- `install-ako-for-all`: default configuration for all workload clusters. By default, all the workload clusters refer to this file for their virtual IP networks and service engine (SE) groups. This ADC configuration does not enable NSX L7 Ingress by default.

As per this Tanzu deployment, create two more ADCs:

- `tanzu-ako-for-shared`: Used by shared services cluster to deploy the virtual services in `TKG Mgmt SE Group` and the loadbalancer applications in `TKG Cluster VIP Network`.

- `tanzu-ako-for-workload-L7-ingress`: Use this ADC only if you would like to enable NSX Advanced Load Balancer L7 ingress on workload cluster. Otherwise, leave the cluster labels empty to apply the network configuration from default ADC `install-ako-for-all`.

Configure AKODeploymentConfig (ADC) for Shared Services Cluster

As per the defined architecture, shared services cluster uses the same control plane and data plane network as the management cluster. Shared services cluster control plane endpoint uses `TKG Cluster VIP Network`, application loadbalancing uses `TKG Cluster VIP Network` and the virtual services are deployed in `sfo01m01segroup01` SE group. This configuration is enforced by creating a custom AKO Deployment Config (ADC) and applying the respective `AVI_LABELS` while deploying the shared services cluster.

The format of the AKODeploymentConfig YAML file is as follows:

```
apiVersion: networking.tkg.tanzu.vmware.com/v1alpha1
kind: AKODeploymentConfig
metadata:
  finalizers:
     - ako-operator.networking.tkg.tanzu.vmware.com
  generation: 2
  name: <Unique name of AKODeploymentConfig>
spec:
  adminCredentialRef:
    name: nsx-alb-controller-credentials
    namespace: tkg-system-networking
  certificateAuthorityRef:
    name: nsx-alb-controller-ca
    namespace: tkg-system-networking
  cloudName: <NAME OF THE CLOUD in ALB>
  clusterSelector:
    matchLabels:
      <KEY>: <VALUE>
  controlPlaneNetwork:
    cidr: <TKG-Cluster-VIP-CIDR>
    Name: <TKG-Cluster-VIP-Network>
  controller: <NSX ALB CONTROLLER IP/FQDN>
  dataNetwork:
    cidr: <TKG-Mgmt-Data-VIP-CIDR>
    name: <TKG-Mgmt-Data-VIP-Name>
  extraConfigs:
   cniPlugin: antrea
   disableStaticRouteSync: true
   ingress:
      defaultIngressController: false
      disableIngressClass: true
      nodeNetworkList:
      - networkName: <TKG-Mgmt-Network>
  serviceEngineGroup: <Mgmt-Cluster-SEG>
```

The sample AKODeploymentConfig with sample values in place is as follows. You must add the respective NSX ALB label `type=shared-services` while deploying shared services cluster to enforce this network configuration.

- cloud: `sfo01w01vc01`

- service engine group: `sfo01m01segroup01`
- Control Plane network: `sfo01-w01-vds01-tkgclustervip`
- VIP/data network: `sfo01-w01-vds01-tkgclustervip`
- Node Network: `sfo01-w01-vds01-tkgmanagement`

```
apiVersion: networking.tkg.tanzu.vmware.com/v1alpha1
kind: AKODeploymentConfig
metadata:
  generation: 3
  name: tanzu-ako-for-shared
spec:
  adminCredentialRef:
    name: avi-controller-credentials
    namespace: tkg-system-networking
  certificateAuthorityRef:
    name: avi-controller-ca
    namespace: tkg-system-networking
  cloudName: sfo01w01vc01
  clusterSelector:
    matchLabels:
      type: shared-services
  controlPlaneNetwork:
    cidr: 172.16.80.0/24
    name: sfo01-w01-vds01-tkgclustervip
  controller: 172.16.10.10
  controllerVersion: 22.1.3
  dataNetwork:
    cidr: 172.16.80.0/24
    name: sfo01-w01-vds01-tkgclustervip
  extraConfigs:
    disableStaticRouteSync: false
    ingress:
      defaultIngressController: false
      disableIngressClass: true
      nodeNetworkList:
      - networkName: sfo01-w01-vds01-tkgmanagement
    networksConfig:
      nsxtT1LR: /infra/tier-1s/sfo01w01tier1
  serviceEngineGroup: sfo01m01segroup01
```

For a single VIP Network Architecture, see Single VIP Nertwork Architecture - Shared Service Cluster ADC file.

After you have the AKO configuration file ready, use the `kubectl` command to set the context to Tanzu Kubernetes Grid management cluster and create the ADC:

```
# kubectl config use-context sfo01w01tkgmgmt01-admin@sfo01w01tkgmgmt01
Switched to context "sfo01w01tkgmgmt01-admin@sfo01w01tkgmgmt01".


# kubectl apply -f ako-shared-services.yaml
akodeploymentconfig.networking.tkg.tanzu.vmware.com/tanzu-ako-for-shared created
```

Use the following command to list all AKODeploymentConfig created under the management cluster:

```
# kubectl get adc
NAME                                 AGE
install-ako-for-all                  21h
install-ako-for-management-cluster   21h
tanzu-ako-for-shared                 113s
```

Configure AKO Deployment Config (ADC) for Workload Cluster to Enable NSX ALB L7 Ingress with NodePortLocal Mode

VMware recommends using NSX Advanced Load Balancer L7 ingress with NodePortLocal mode for the L7 application load balancing. This is enabled by creating a custom ADC with ingress settings enabled, and then applying the `AVI_LABELS` while deploying the workload cluster.

As per the defined architecture, workload cluster cluster control plane endpoint uses `TKG Cluster VIP Network`, application loadbalancing uses `TKG Workload VIP Network` and the virtual services are deployed in `sfo01w01segroup01` SE group.

Below are the changes in ADC Ingress section when compare to the default ADC.

- **disableIngressClass**: Set to `false` to enable NSX ALB L7 Ingress.
- **nodeNetworkList**: Provide the values for TKG workload network name and CIDR.
- **serviceType**: L7 Ingress type, recommended to use `NodePortLocal`
- **shardVSSize**: Virtual service size

The format of the AKODeploymentConfig YAML file for enabling NSX ALB L7 Ingress is as follows:

```
apiVersion: networking.tkg.tanzu.vmware.com/v1alpha1
kind: AKODeploymentConfig
metadata:
  name: <unique-name-for-adc>
spec:
  adminCredentialRef:
    name: NSXALB-controller-credentials
    namespace: tkg-system-networking
  certificateAuthorityRef:
    name: NSXALB-controller-ca
    namespace: tkg-system-networking
  cloudName: <cloud name configured in nsx alb>
  clusterSelector:
    matchLabels:
      <KEY>: <value>
  controller: <ALB-Controller-IP/FQDN>
  controlPlaneNetwork:
    cidr: <TKG-Cluster-VIP-Network-CIDR>
    name: <TKG-Cluster-VIP-Network-CIDR>
  dataNetwork:
    cidr: <TKG-Workload-VIP-network-CIDR>
    name: <TKG-Workload-VIP-network-CIDR>
  extraConfigs:
    cniPlugin: antrea
    disableStaticRouteSync: false                               # required
    ingress:
      disableIngressClass: false                                # required
      nodeNetworkList:                                       # required
        - networkName: <TKG-Workload-Network>
          cidrs:
            - <TKG-Workload-Network-CIDR>
      serviceType: NodePortLocal                             # required
      shardVSSize: MEDIUM                                    # required
  serviceEngineGroup: <Workload-Cluster-SEG>
```

The AKODeploymentConfig with sample values in place is as follows. You should add the respective NSX ALB label `workload-l7-enabled=true` while deploying shared services cluster to enforce this network configuration.

- cloud: `sfo01w01vc01`
- service engine group: `sfo01w01segroup01`
- Control Plane network: `sfo01-w01-vds01-tkgclustervip`
- VIP/data network: `sfo01-w01-vds01-tkgworkloadvip`
- Node Network: `sfo01-w01-vds01-tkgworkload`

```
apiVersion: networking.tkg.tanzu.vmware.com/v1alpha1
kind: AKODeploymentConfig
metadata:
  generation: 3
  name: tanzu-ako-for-workload-L7-ingress
spec:
  adminCredentialRef:
    name: avi-controller-credentials
    namespace: tkg-system-networking
  certificateAuthorityRef:
    name: avi-controller-ca
    namespace: tkg-system-networking
  cloudName: sfo01w01vc01
  clusterSelector:
    matchLabels:
      workload-l7-enabled: "true"
  controlPlaneNetwork:
    cidr: 172.16.80.0/24
    name: sfo01-w01-vds01-tkgclustervip
  controller: 172.16.10.11
  controllerVersion: 22.1.3
  dataNetwork:
    cidr: 172.16.70.0/24
    name: sfo01-w01-vds01-tkgworkloadvip
  extraConfigs:
    disableStaticRouteSync: true
    ingress:
      defaultIngressController: true
      disableIngressClass: false
      serviceType: NodePortLocal
      shardVSSize: MEDIUM
      nodeNetworkList:
      - networkName: sfo01-w01-vds01-tkgworkload
        cidrs:
        - 172.16.60.0/24
    networksConfig:
      nsxtT1LR: /infra/tier-1s/sfo01w01tier2
  serviceEngineGroup: sfo01w01segroup01
```

For a single VIP Network Architecture, see Single VIP Nertwork Architecture - Workload Cluster ADC file.

Use the `kubectl` command to set the context to Tanzu Kubernetes Grid management cluster and create the ADC:

```
# kubectl config use-context sfo01w01tkgmgmt01-admin@sfo01w01tkgmgmt01
Switched to context "sfo01w01tkgmgmt01-admin@sfo01w01tkgmgmt01".

# kubectl apply -f workload-adc-l7.yaml
akodeploymentconfig.networking.tkg.tanzu.vmware.com/tanzu-ako-for-workload-l7-ingress
created
```

Use the following command to list all AKODeploymentConfig created under the management cluster:

```
# kubectl get adc
NAME                                 AGE
install-ako-for-all                  22h
install-ako-for-management-cluster   22h
tanzu-ako-for-shared                 82m
tanzu-ako-for-workload-l7-ingress    25s
```

Now that you have successfully created the AKO deployment config, you need to apply the cluster labels while deploying the workload clusters to enable NSX Advanced Load Balancer L7 Ingress with the NodePortLocal mode.

Deploy Tanzu Kubernetes Grid Shared Services Cluster

Each Tanzu Kubernetes Grid instance can have only one shared services cluster. Create a shared services cluster if you intend to deploy Harbor.

The procedures for deploying a shared services cluster and workload cluster are almost the same. A key difference is that for the shared service cluster you add the `tanzu-services` label to the shared services cluster, as its cluster role. This label identifies the shared services cluster to the management cluster and workload clusters.

Shared services cluster uses the custom ADC `tanzu-ako-for-shared` created earlier to apply the network settings similar to the management cluster. This is enforced by applying the AVI_LABEL `type:shared-services` while deploying the shared services cluster.

Deployment of the shared services cluster is done by creating a YAML file and invoking the `tanzu cluster create -f <file-name>` command. The YAML file used for shared services deployment is smaller compared to the YAML used for the management cluster deployment because, you don’t need to define the AVI fields except `AVI_CONTROL_PLANE_HA_PROVIDER` & `AVI_LABELS` in the YAML.

Image repository configuration is very important details.

A sample yaml for shared services cluster deployment is given below:

```
CLUSTER_NAME: sfo01w01tkgshared01
CLUSTER_PLAN: prod
INFRASTRUCTURE_PROVIDER: vsphere
ENABLE_CEIP_PARTICIPATION: "true"
ENABLE_AUDIT_LOGGING: "true"
CLUSTER_CIDR: 100.96.0.0/11
SERVICE_CIDR: 100.64.0.0/13
VSPHERE_SERVER: sfo01w01vc01.sfo01.rainpole.local
VSPHERE_USERNAME: administrator@vsphere.local
VSPHERE_PASSWORD: <encoded:Vk13YXJlMSE=>
VSPHERE_DATACENTER: /sfo01w01dc01
VSPHERE_RESOURCE_POOL: /sfo01w01dc01/host/sfo01w01cluster01/Resources/tkg-sharedsvc-co
mponents
VSPHERE_DATASTORE: /sfo01w01dc01/datastore/vsanDatastore
VSPHERE_FOLDER: /sfo01w01dc01/vm/tkg-sharedsvc-components
VSPHERE_NETWORK: /sfo01w01dc01/network/sfo01-w01-vds01-tkgshared
VSPHERE_CONTROL_PLANE_ENDPOINT: #Leave blank as VIP network is configured in NSX ALB a
nd IPAM is configured with VIP network
VSPHERE_SSH_AUTHORIZED_KEY: ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDrPqkVaPpNxHcKxukYro
V6LcCTuRK9NDyygbsAr/P73jEeWIcC+SU4tRpOZks2+BoduUDzdrsfm/Uq/0uj9LuzqIZKAzA1iQ5DtipVzROq
eTuAXJVCMZc6RPgQSZofLBo1Is85M/IrBS20OMALwjukMdwotKKFwL758l51FVsKOT+MUSW/wJLKTv3l0KPObg
SRTMUQdQpoG7ONcMNG2VkBMfgaK44cL7vT0/0Mv/Fmf3Zd59ZaWvX28ZmGEjRx8kOm1j/os61Y+kOvl1MTv8wc
85rYusRuP2Uo5UM4kUTdhSTFasw6TLhbSWicKORPi3FYklvS70jkQFse2WsvmtFG5xyxE/rzDGHloud9g2bQ7T
x0rtWWoRCCC8Sl/vzCjgZfDQXwKXoMP0KbcYHZxSA3zY2lXBlhNtZtyKlynnhr97EaWsm3b9fvhJMmKW5ylkmk
7+4Bql7frJ4bOOR4+hHv57Q8XFOYdLGQPGv03RUFQwFE6a0a6qWAvmVmoh8+BmlGOfx7WYpp8hkyGOdtQz8ZJe
SOyMT6ztLHbY/WqDwEvKpf1dJy93w8fDmz3qXHpkpdnA0t4TiCfizlBk15ZI03TLi4ELoFvso9We13dGClHDDy
v0Dm87uaACC+fyAT5JPbZpAcCw8rm/yTuZ8awtR0LEzJUqNJjX/5OX7Bf45h9w== email@example.com
VSPHERE_TLS_THUMBPRINT: DC:FA:81:1D:CA:08:21:AB:4E:15:BD:2B:AE:12:2C:6B:CA:65:49:B8
VSPHERE_INSECURE: "false"
OS_NAME: photon
OS_VERSION: "3"
OS_ARCH: amd64
VSPHERE_CONTROL_PLANE_NUM_CPUS: 2
VSPHERE_CONTROL_PLANE_DISK_GIB: 40
VSPHERE_CONTROL_PLANE_MEM_MIB: 8192
VSPHERE_WORKER_NUM_CPUS: 2
VSPHERE_WORKER_DISK_GIB: 40
VSPHERE_WORKER_MEM_MIB: 8192
AVI_CONTROL_PLANE_HA_PROVIDER: "true"
AVI_LABELS: |
    'type': 'shared-services'
ENABLE_MHC: "true"
IDENTITY_MANAGEMENT_TYPE: "none"
TKG_CUSTOM_IMAGE_REPOSITORY: harbor.sfo01.rainpole.vmw/tkgm-images
TKG_CUSTOM_IMAGE_REPOSITORY_SKIP_TLS_VERIFY: false
TKG_CUSTOM_IMAGE_REPOSITORY_CA_CERTIFICATE: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSU
ZnekNDQTJ1Z0F3SUJBZ0lVUXhvVVV5VnpwUVlkdUlXbStwL3dxZ0JDSFVrd0RRWUpLb1pJaHZjTkFRRUwKQlFB
d1VURUxNQWtHQTFVRUJoTUNRMDR4RERBS0JnTlZCQWdNQTFCRlN6RVFNQTRHQTFVRUJ3d0hRbVZwU21sdQpaek
VQTUEwR0ExVUVDZ3dHVmsxM1lYSmxNUkV3RHdZRFZRUUREQWhJWVhKaWIzSkRRVEFlRncweU16QTRNRGN3Ck5q
VTBOVGRhRncek16QTRNRFF3TmpVME5UZGFNRkV4Q3pBSkJnTlZCQVlUQWtOT01Rd3dDZ1lEVlFRSURBTlEKUl
VzeEVEQU9CZ05WQkFjTUIwSmxhVXBwYm1jeER6QU5CZ05WQkFvTUJsWk5kMkZ5WlRFUk1BOEdBMVVFQXd3SQpT
R0Z5WW05eVEwRXdnZ0lpTUEwR0NTcUdTSIzRFFFQkFRVUFBNElDRHdBd2dnSUtBb0lDQVFDckFtajBVR1lUCj
```

```
Vobm5OeTNIMVdkcUhIejBHbFphWVpGQlhvQVhlWitZa2ljeXJhOFpFeVJCTFBKZVNOcldzMW45dFp1RUg2WDgK
M3JJbzUwdzhrZTYwYU9OOU80VlVXZExZKzYxYUxrNlpkNGkzcU9IQTVKSFU1cE5DenU0bThaQ1F1bUp2SzFVSQ
pwQ1lQNnFtNGxSQUFvWFVzWGZ0S24vRkJ4bGdReTNhUjJ1Y0IzdXA5UndDRllDLzA5TVd5ZjErUmhja3ZvWExxR
CmppUWx4aHZ0NFpxeG12b09KMi9lbUorTHBqbENZaXBRVkNwN3NpeVM1ZGIvRmw5U1VWSFVlRDhpdzgxKzZTaE
EKenRpZXY1U20zbGE0UGw1cUU4Vm9EVEJTUXRpMmQzUXB2M05IR0Z4UUNWMjVTM1BVcWxPQ1Z0dDFtSHRjZEhS
UQpTOEN2SE9DVlczRHd5ZXB2Y3M4STJFYlU3M0VQN3JKOEJtL21GQUFwd2F1NXpjMmtxRGJRclMrTjhEeWFpcG
Y4Ckgxa3FFVlU2VmEvbCtjZHFSbkc1ZWVmM21vT3dKMDBPa2ZqSWtCUExSOC9iZVdDR0R4RVE1RkZlRXhxSTdh
NUIKcm1FK25vTTBjMFRQSFp5Z3c4UnZVU3hQbGJ3MFZkVzA5MHdQOHB3WGFGYVBPMmhtT2lvVGJiVkpkMVhqaG
RFZAoxWTdmQktYKzl6UUxjb3dlcmF5cDRaSUlSTTNTZnRaZkc5bjErN3pmcVBpamVoTGdiUkUrcHVaaUQwbi9x
UXIvClNzOWU5MXZQR1U2NW82VjYvL2Y5MVZmVHJRUmVJV0JWUWNubCtobGpPOVpUeHRRMlVwRnU3TDRwT2RJZ1
JHRTgKcDVjZE91TDF1cWltTi8rTW5hckV3K2JFalc0ajF3cnFQUUlEQVFBQm8xTXdVVEFkQmdOVkhRNEVGZ1FV
dEozbQpvR3RrcVJiVGl2ZFkzQWxnMTZUOWpCZ3dId1lEVlIwakJCZ3dGb0FVdEozbW9HdGtxUmJUaXZkWTNBbG
cxNlQ5CmpCZ3dEd1lEVlIwVEFRSC9CQVV3QXdFQi96QU5CZ2txaGtpRzl3MEJBUXNGQUFPQ0FnRUFUUFBGcUdu
UjVRdEoKZm1PMTdwSmVDK0IwM3NOVGozS0VzZjUybmJXUnprWm9xNjMrRDZnVURtU3FMMm1RSExRUy9WNWhadm
IzZTFCTApDSXpEZmF5cG14K2k3MnlIbVRRNGRBUW5NT2hCUm8rMENVOGpZa0t4TjllN1NhOTlERk0yVE5Bb1pz
cEs4ZEVUCmlUa0p2bHlHS2tyellkNEFvczN3dU9uVE85VWxCK1FqQmI4TUVDc3lMR2U4VGxJMk4vOFdwMFMySG
1QUHVNeU8KemJ0RUUrSWJXaW1qT1lLRU44cHlUY3plelVnZHpGcXJ3bjVKdWZSejdockV6MnE5ZG9sYm0vTGRN
K3pnUHVzZwo0a3lnVjlxWU5KcldvVExXbFJKNzRyQmFOZFpYM0OV1VGRGtjQ3JkSkloWFFESWdWUDNWa2xJZ2
1zUUhTU3IrCktJYSt5R1p3MU5hODZML2R0djNrR1ovN2VRMHNHVzVpS3R3VmY5UnBqYTdXL3ZhTTA1OTdFWGNS
SGZ2cHRxeFQKcjNOcnFTQmZkTlJtNStXOVh3c0RxNDl3dFdERE52OHNDS2JrenI2Q0JHYUxXSHFGRWtCOHpiTG
lJQVBYd2VqaApSOTh5TnY1ZjBzb0ltZlg1R3REY2RMZjd3dGg0UGlvRGloZklrRXVzd0twVGN1WWo3clR0SnFY
TFV2b25jZkV0Ck93cytHa2c5L0ZHd0p6ZkJlYUNrNHVXbGw5bC9JanZ6azdydkc0Z1VXa2tMVi9SdmVzbFlEZV
lSUXBnYzRybzgKdFFpMThvb3V4RGZuMTlSS2JPVjNtNTluTlYwdzlHdjZiUGxqbjlRaDB0MWJOaUxwZThJeWhG
b0VPOFpVYTBnSApCZ29PbXlGZHQ0VTlQclIvZTdNcWZzM2tQVjZkbmtzPQotLS0tLUVORCBDRVJUSUZJQ0FURS
0tLS0t
```

To create Workload Cluster, run the following command:

```
tanzu cluster create --file config.yaml
```

Cluster creation takes approximately 15-20 minutes to complete. Verify the health of the cluster and validate the cluster labels applied.

1. Connect to the Tanzu Management Cluster context and verify the cluster labels for the Shared Service cluster.

```
## verify the Shared Service cluster creation

tanzu cluster list
NAME                  NAMESPACE  STATUS   CONTROLPLANE  WORKERS  KUBERNETES
ROLES   PLAN  TKR
sfo01w01tkgshared01  default    running  3/3           3/3      v1.26.5+vmware.
2  <none>  prod  v1.26.5---vmware.2-tkg.1


## Connect to tkg management cluster

kubectl config use-context sfo01w01tkgmgmt01-admin@sfo01w01tkgmgmt01

## Add the tanzu-services label to the shared services cluster as its cluster r
ole. In the following command "sfo01w01tkgshared01” is the name of the shared s
ervice cluster

kubectl label cluster.cluster.x-k8s.io/sfo01w0tkgshared01 cluster-role.tkg.tanz
u.vmware.com/tanzu-services="" --overwrite=true
cluster.cluster.x-k8s.io/sfo01w0tkgshared01 labeled

## Validate AVI_LABELS applied to shared serice cluster
```

```
kubectl get cluster sfo01w0tkgshared01 --show-labels
NAME                   PHASE         AGE    VERSION   LABELS

sfo01w0tkgshared01   Provisioned   105m             cluster-role.tkg.tanzu.vmwa
re.com/tanzu-services=,networking.tkg.tanzu.vmware.com/avi=tanzu-ako-for-share
d,tanzuKubernetesRelease=v1.26.5---vmware.2-tkg.1,tkg.tanzu.vmware.com/cluster-
name=sfo01w0tkgshared01,type=shared-services
```

2. Connect to admin context of the Shared Service cluster using the following commands and validate the ako pod status.

```
## Use the following command to get the admin context of Service Service Cluste
r.

tanzu cluster kubeconfig get sfo01w0tkgshared01 --admin

Credentials of cluster 'sfo01w0tkgshared01' have been saved
You can now access the cluster by running 'kubectl config use-context sfo01w0tk
gshared01-admin@sfo01w0tkgshared01'


## Use the following command to use the context of Shared Service Cluster

kubectl config use-context sfo01w0tkgshared01-admin@sfo01w0tkgshared01

Switched to context "sfo01w0tkgshared01-admin@sfo01w0tkgshared01".

# Verify that ako pod gets deployed in avi-system namespace

kubectl get pods -n avi-system
NAME    READY   STATUS    RESTARTS   AGE
ako-0   1/1     Running   0          73m

# verify the nodes and pods status by running the command:
kubectl get nodes -o wide

kubectl get pods -A
```

Now, the shared services cluster is successfully created.

Deploy Tanzu Kubernetes Grid Workload Cluster

The workload cluster is deployed by using a YAML file similar to the shared services cluster YAML file but customized for the workload cluster placement objects.

The following is a sample YAML for deploying the workload cluster:

```
CLUSTER_CIDR: 100.96.0.0/11
SERVICE_CIDR: 100.64.0.0/13
CLUSTER_PLAN: prod
ENABLE_CEIP_PARTICIPATION: 'false'
ENABLE_MHC: 'true'
IDENTITY_MANAGEMENT_TYPE: none
INFRASTRUCTURE_PROVIDER: vsphere
TKG_HTTP_PROXY_ENABLED: 'false'
```

```
AVI_CONTROL_PLANE_HA_PROVIDER: "true"
CLUSTER_NAME: sfo01w01tkgworkload01
DEPLOY_TKG_ON_VSPHERE7: 'false'
OS_ARCH: amd64
OS_NAME: photon
OS_VERSION: "3"
AVI_LABELS: |
    'workload-l7-enabled': 'true'
VSPHERE_DATACENTER: /sfo01w01dc01
VSPHERE_DATASTORE: /sfo01w01dc01/datastore/vsanDatastore
VSPHERE_FOLDER: /sfo01w01dc01/vm/tkg-workload01-components
VSPHERE_INSECURE: "true"
VSPHERE_NETWORK: /sfo01w01dc01/network/sfo01-w01-vds01-tkgworkload
VSPHERE_PASSWORD: <encoded:Vk13YXJlMSE=>
VSPHERE_RESOURCE_POOL: /sfo01w01dc01/host/sfo01w01cluster01/Resources/tkg-workload01-c
omponents
VSPHERE_SERVER: sfo01w01vc01.sfo01.rainpole.local
VSPHERE_SSH_AUTHORIZED_KEY: ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDrPqkVaPpNxHcKxukYro
V6LcCTuRK9NDyygbsAr/P73jEeWIcC+SU4tRpOZks2+BoduUDzdrsfm/Uq/0uj9LuzqIZKAzA1iQ5DtipVzROq
eTuAXJVCMZc6RPgQSZofLBo1Is85M/IrBS20OMALwjukMdwotKKFwL758l51FVsKOT+MUSW/wJLKTv3l0KPObg
SRTMUQdQpoG7ONcMNG2VkBMfgaK44cL7vT0/0Mv/Fmf3Zd59ZaWvX28ZmGEjRx8kOm1j/os61Y+kOvl1MTv8wc
85rYusRuP2Uo5UM4kUTdhSTFasw6TLhbSWicKORPi3FYklvS70jkQFse2WsvmtFG5xyxE/rzDGHloud9g2bQ7T
x0rtWWoRCCC8Sl/vzCjgZfDQXwKXoMP0KbcYHZxSA3zY2lXBlhNtZtyKlynnhr97EaWsm3b9fvhJMmKW5ylkmk
7+4Bql7frJ4bOOR4+hHv57Q8XFOYdLGQPGv03RUFQwFE6a0a6qWAvmVmoh8+BmlGOfx7WYpp8hkyGOdtQz8ZJe
SOyMT6ztLHbY/WqDwEvKpf1dJy93w8fDmz3qXHpkpdnA0t4TiCfizlBk15ZI03TLi4ELoFvso9We13dGClHDDy
v0Dm87uaACC+fyAT5JPbZpAcCw8rm/yTuZ8awtR0LEzJUqNJjX/5OX7Bf45h9w== email@example.com
VSPHERE_TLS_THUMBPRINT: ""
VSPHERE_USERNAME: administrator@vsphere.local
ENABLE_AUDIT_LOGGING: true
ENABLE_DEFAULT_STORAGE_CLASS: true
ENABLE_AUTOSCALER: false
VSPHERE_CONTROL_PLANE_NUM_CPUS: 2
VSPHERE_CONTROL_PLANE_DISK_GIB: 40
VSPHERE_CONTROL_PLANE_MEM_MIB: 8192
VSPHERE_WORKER_NUM_CPUS: 2
VSPHERE_WORKER_DISK_GIB: 40
VSPHERE_WORKER_MEM_MIB: 8192
WORKER_MACHINE_COUNT: 3
TKG_CUSTOM_IMAGE_REPOSITORY: harbor.sfo01.rainpole.vmw/tkgm-images
TKG_CUSTOM_IMAGE_REPOSITORY_SKIP_TLS_VERIFY: false
TKG_CUSTOM_IMAGE_REPOSITORY_CA_CERTIFICATE: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSU
ZnekNDQTJ1Z0F3SUJBZ0lVT2hDeStrNUtDRGRKSUl2QS9IV1ZXKzdiYUVrd0RRWUpLb1pJaHZjTkFRRUwKQlFB
d1VURUxNQWtHQTFVRUJoTUNRMDR4RERBS0JnTlZCQWdNQTFCRlN6RVFNQTRHQTFVRUJ3d0hRbVZwU21sdQpaek
VQTUEwR0ExVUVDZ3dHVmsxM1lYSmxNUkV3RHdZRFZRUUREQWhJWVhKaWIzSkRRVEFlRncweU16QTNNVGd4Ck5q
VXdORFZhRncwek16QTNNVFV4TmpVd05EVmFNRkV4Q3pBSkJnTlZCQVlUQWtOT01Rd3dDZ1lEVlFRSURBTlEKUl
VzeEVEQU9CZ05WQkFjTUIwSmxhVXBwYm1jeER6QU5CZ05WQkFvTUJsWk5kMkZ5WlRFUk1BOEdBMVVFQXd3SQpT
R0Z5WW05eVEwRXdnZ0lpTUEwR0NTcUdTSWIzRFFFQkFRVUFBNElDRHdBd2dnSUtBb0lDQVFDUVlWMFhQTWFlCl
ZDQU5SdUMzMFBEeERvMVFIRVRQdmUyQVhUQWsxVXkra09JOUpWQzFUek9RUW42YjFtbmNLUzBsM2cyR3JDNnUK
eFNKR25NNEJyMWxsalZwNHh5TU1LcndXdTVkZzRQZXBqempLeVFuaVNkcEx4TThOYnNlU0JjN3U0YWNHNWVRcg
pxSHZRTy8zK0k1cTI1SXE1Mi9EaEZiSjdZbllVTTRuc2c3TUZEU2xqRnVlYTNTTytORms5OFI5bmh3RnpOTXRY
CkFFM28rL0t3SjRlTmwyRTJGUHhOUW5nTDhMWG8wUWE4OXFCbXdjMmx6eGFCc1N3VDNpOWlMQ2JLK0pabzAvZH
AKaXRwZGcrZHNpZ1dUcWRFekErMG12aCtlNW9iaG43UXo2VzJpb3pRMWZGcVRKZ09Jd1VFZ2srdG5SZWxhZFBF
QQpKaUcvQVBncGdQY0ZleDExeUFJYUdUNnRNOVlPV3VMRk5mV1FCNTBzVWRRZlJaUlk5cUxheG1uR1c2TWhDME
srCkFIcmgvUmxJa2g5cVFVZmoxTUpLdFNrZzJBQVRRZ3RVWkY2TG1nei9XK1p6aU0xOG1HY1pRU3UzUWVHWElS
L1IKWjF5VzcrTWhUQ2Q0ZHRiU1FzRFBGZEhaazZVVXhlOE40MjV4cTFBeG5PNCtSR3dBTG96RS9YWk1zbnNZaD
hiego5UHpXSFIxN2E5UGJ1L1l6c2NDNE9qN1dJVlY4UGtWREptK0xaelhDUU5IL2g1Y1phSzZhSXU4d0NLSG5K
dE42CnA4RUpsWkliZ3RTOEl5eVJqQStycUZ2QUtXS2tjYVEwUVduWjB6MjBTVk9vYVg2bHNlbnBaZmpyS1RnV3
hUdncKS01tTkQyc0NkVU91MnlkWHpMVWRRUDBDb25VN3V3bnViUUlEQVFBQm8xTXdVVEFkQmdOVkhRNEVGZ1FV
S1FGegpOcjdwYzZndHdOUitCVXhud1h1TVY5NHdId1lEVlIwakJCZ3dGb0FVS1FGek5yN3BjNmd0d05SK0JVeG
```

```
53WHVNClY5NHdEd1lEVlIwVEFRSC9CQVV3QXdFQi96QU5CZ2txaGtpRzl3MEJBUXNGQUFPQ0FnRUFEay85UjVD
Ykhrek8KZGdQbXFPVVlqUnVUcWdtcWdqbUlhM2VWNG51NzdmNGpRZWNwcnJJTzcrbk1vN0RrTHNBQURyNVdjL0
xFdTBlTAp2N0c2U3Rsbzd6dVVpRlVmbERvUndsaFRnMXdyRDBxcmNxendUTUFTWVh2dWxLOWgvU05ySEdmR2hJ
L1Q5TXBPCm1leS85WkY1ZmdOTC80b2lvQ0lPVXFkQ2luUWhCaEFGQWJ5YWUrNHI4dXkzcXB6VDQ2ZXVRY040Ym
VTczBoaFUKNnpwRWpRU05mWjJOUWRSbHhISWlPT1pKZ1pmYXlZa3Ard2hqT3ZTL09Vdm5NR0tNNk1hckk4L3Qz
VVBhNHE1ZQpLVU9PS01Ma3RRWjA5a3o1VndZWmFuNk4yL010U09hT28zdmx6b0ZXSC9LdFdUZFhxcmppRmN6Um
FORElxZEExCkJXR2xYcUJsUTdHWjRTazBHSzE1cENJUU9QZTZ4dEprVW9GSnV4SDJ5TW9tUWlMbFdQdE9obllS
WkhEWUpjMkcKem81SFVkM0lhNnlROUMySlZzcGdXcFhYeWtsVmJPb0treVFvWVorWU1rdGt5dDJnaG52S1ZNTm
5PUXdCM2dpcgpBMVZTY29UUS82ZEREcFl4MytwVTZGSXpRdXgzdDRkZUZOWU8zd3BjSksrbS9PSUZFZlNGbXdP
TFJ0NTMzU1RiCk94dzJKdzZPQ0tobzl0TUZBNGR6czBUS3R6NE9QUTlTb2RxZDlsNlZNbnprY1JlcFdHd0wzQ0
JjRHBCdlFQNjAKWVQydHAwRGEwSmZ3cExSdk55WE5xQXZZZUUydHEvTG9EcG1lcnoyYUZHR3c1VDZsTDhwMXZY
eUdEYmw2bEVIawpJRXBQUWExWk8vVW5HKzdUeEZ1OHhHV21YVXVFcEQ0PQotLS0tLUVORCBDRVJUSUZJQ0FURS
0tLS0t
```

To create Workload Cluster, run the following command:

```
tanzu cluster create --file config.yaml
```

The Cluster creation takes 15-20 minutes to complete. Verify the health of the cluster and apply the labels.

After the Workload cluster is created, verify the cluster labels and ako pod status.

1. Connect to the Tanzu Management Cluster context and verify the cluster labels for the workload cluster.

   ```
   ## verify the workload cluster creation

   tanzu cluster list
   NAME                  NAMESPACE  STATUS   CONTROLPLANE  WORKERS  KUBERNETES
   ROLES   PLAN  TKR

   sfo01w01tkgshared01    default    running  3/3           3/3      v1.26.5+vmwar
   e.2  <none>  prod   v1.26.5---vmware.2-tkg.1

   sfo01w01tkgworkload01  default    running  3/3           3/3      v1.26.5+vmwar
   e.2  <none>  prod   v1.26.5---vmware.2-tkg.1

   ## Connect to tkg management cluster

   kubectl config use-context sfo01w01tkgmgmt01-admin@sfo01w01tkgmgmt01

   ## Validate AVI_LABELS applied to workload cluster

   kubectl get cluster sfo01w01workload01 --show-labels
   NAME                    PHASE         AGE    VERSION   LABELS

   sfo01w01tkgworkload01   Provisioned   105m             networking.tkg.tanzu.vmw
   are.com/avi=tanzu-ako-for-workload-l7-ingress,tanzuKubernetesRelease=v1.26.5---
   vmware.2-tkg.1,tkg.tanzu.vmware.com/cluster-name=sfo01w01tkgworkload01,workload
   -l7-enabled=true
   ```

2. Connect to admin context of the workload cluster using the following commands and validate the ako pod status.

```
## Use the following command to get the admin context of workload Cluster.

tanzu cluster kubeconfig get sfo01w01tkgworkload01 --admin

Credentials of cluster 'sfo01w01tkgworkload01' have been saved
You can now access the cluster by running 'kubectl config use-context sfo01w01t
kgworkload01-admin@sfo01w01workload01'


## Use the following command to use the context of workload Cluster

kubectl config use-context sfo01w01tkgworkload01-admin@sfo01w01workload01

Switched to context "sfo01w01tkgworkload01-admin@sfo01w01workload01".

# Verify that ako pod gets deployed in avi-system namespace

kubectl get pods -n avi-system
NAME     READY   STATUS    RESTARTS   AGE
ako-0    1/1     Running   0          73m

# verify the nodes and pods status by running the command:
kubectl get nodes -o wide

kubectl get pods -A
```

You can see that the workload cluster is successfully deployed and the AKO pod is deployed on the cluster. You can now deploy user-managed packages on this cluster.

Deploy User-Managed Packages

User-managed packages are installed after workload cluster creation. These packages extend the core functionality of Kubernetes clusters created by Tanzu Kubernetes Grid.

Tanzu Kubernetes Grid includes the following user-managed packages. These packages provide in-cluster and shared services to the Kubernetes clusters that are running in your Tanzu Kubernetes Grid environment.

Installing and Managing Packages with the Tanzu CLI

Function	Package	Location
Certificate Management	Cert Manager	Workload and shared services cluster
Container registry	Harbor	Shared services cluster
Ingress control	Contour	Workload and shared services cluster
Log forwarding	Fluent Bit	Workload cluster
Monitoring	Grafana Prometheus	Workload cluster

User-managed packages can be installed via CLI by invoking the `tanzu package install` command. Before installing the user-managed packages, ensure that you have switched to the context of the cluster where you want to install the packages.

Also, ensure that the tanzu-standard repository is configured on the cluster where you want to install the packages.

You can run the command `tanzu package repository list -A` to verify this. Also, ensure that the repository status is `Reconcile succeeded`.

```
# Add Private Registry to the workload Cluster

tanzu package repository add tanzu-standard --url harbor.sfo01.rainpole.vmw/tkgm-image
s/packages/standard/repo -n tkg-system

# tanzu package repository list -A

NAMESPACE   NAME            SOURCE
STATUS
tkg-system  tanzu-standard  (imgpkg) harbor.sfo01.rainpole.vmw/tkgm-images/packages/st
andard/repo  Reconcile succeeded

#tanzu package available list -A

NAMESPACE   NAME                                            DISPLAY-NAME
tkg-system  cert-manager.tanzu.vmware.com                   cert-manager
tkg-system  contour.tanzu.vmware.com                        contour
tkg-system  external-csi-snapshot-webhook.tanzu.vmware.com  external-csi-snapshot-webh
ook
tkg-system  external-dns.tanzu.vmware.com                   external-dns
tkg-system  fluent-bit.tanzu.vmware.com                     fluent-bit
tkg-system  fluxcd-helm-controller.tanzu.vmware.com         Flux Helm Controller
tkg-system  fluxcd-kustomize-controller.tanzu.vmware.com    Flux Kustomize Controller
tkg-system  fluxcd-source-controller.tanzu.vmware.com       Flux Source Controller
tkg-system  grafana.tanzu.vmware.com                        grafana
tkg-system  harbor.tanzu.vmware.com                         harbor
tkg-system  multus-cni.tanzu.vmware.com                     multus-cni
tkg-system  prometheus.tanzu.vmware.com                     prometheus
tkg-system  whereabouts.tanzu.vmware.com                    whereabouts
```

Install Cert Manager

The first package that you should install on your cluster is the **cert-manager** package which adds certificates and certificate issuers as resource types in Kubernetes clusters, and simplifies the process of obtaining, renewing and using those certificates.

1. Capture the available Cert Manager package versions.

```
NAMESPACE   NAME                           VERSION                 RELEASED-AT
tkg-system  cert-manager.tanzu.vmware.com  1.1.0+vmware.1-tkg.2    2020-11-24 1
8:00:00 +0000 UTC
tkg-system  cert-manager.tanzu.vmware.com  1.1.0+vmware.2-tkg.1    2020-11-24 1
8:00:00 +0000 UTC
tkg-system  cert-manager.tanzu.vmware.com  1.11.1+vmware.1-tkg.1   2023-01-11 1
2:00:00 +0000 UTC
tkg-system  cert-manager.tanzu.vmware.com  1.5.3+vmware.2-tkg.1    2021-08-23 1
7:22:51 +0000 UTC
tkg-system  cert-manager.tanzu.vmware.com  1.5.3+vmware.4-tkg.1    2021-08-23 1
7:22:51 +0000 UTC
tkg-system  cert-manager.tanzu.vmware.com  1.5.3+vmware.7-tkg.1    2021-08-23 1
7:22:51 +0000 UTC
```

```
tkg-system  cert-manager.tanzu.vmware.com  1.5.3+vmware.7-tkg.3   2021-08-23 1
7:22:51 +0000 UTC
tkg-system  cert-manager.tanzu.vmware.com  1.7.2+vmware.1-tkg.1   2021-10-29 1
2:00:00 +0000 UTC
tkg-system  cert-manager.tanzu.vmware.com  1.7.2+vmware.3-tkg.1   2021-10-29 1
2:00:00 +0000 UTC
tkg-system  cert-manager.tanzu.vmware.com  1.7.2+vmware.3-tkg.3   2021-10-29 1
2:00:00 +0000 UT
```

2. Install the `cert-manager` package.

   Capture the latest version from the previous command. If there are multiple versions available, you must check the “RELEASED-AT” to collect the version of the latest one. This document uses the version 1.7.2+vmware.3-tkg.3 for installation.

   The following command installs the `cert-manager` package:

```
tanzu package install cert-manager --package cert-manager.tanzu.vmware.com --na
mespace cert-manager-package --version <AVAILABLE-PACKAGE-VERSION>

# tanzu package install cert-manager --package cert-manager.tanzu.vmware.com --
namespace cert-manager-package --version 1.7.2+vmware.3-tkg.3
8:05:31AM: Creating service account 'cert-manager-cert-manager-package-sa'
8:05:31AM: Creating cluster admin role 'cert-manager-cert-manager-package-clust
er-role'
8:05:31AM: Creating cluster role binding 'cert-manager-cert-manager-package-clu
ster-rolebinding'
8:05:31AM: Creating overlay secrets
8:05:31AM: Creating package install resource
8:05:31AM: Waiting for PackageInstall reconciliation for 'cert-manager'
8:05:31AM: Fetch started (1s ago)
8:05:32AM: Fetching
        | apiVersion: vendir.k14s.io/v1alpha1
        | directories:
        | - contents:
        |   - imgpkgBundle:
        |       image: harbor.sfo01.rainpole.vmw/tkgm-images/packages/standard/
repo@sha256:cac4e2d8a3e98be121a86e687b57d8058dba5f0ba240f3db5008bc85e5ac04cf
        |     path: .
        |   path: "0"
        | kind: LockConfig
        |
8:05:32AM: Fetch succeeded
8:05:33AM: Template succeeded (1s ago)
```

3. Confirm that the `cert-manager` package has been installed successfully, and the status is `Reconcile succeeded`.

```
]# tanzu package installed get cert-manager -n cert-manager-package
NAME:                      cert-manager
PACKAGE-NAME:              cert-manager.tanzu.vmware.com
PACKAGE-VERSION:           1.7.2+vmware.3-tkg.3
STATUS:                    Reconcile succeeded
CONDITIONS:                [{ReconcileSucceeded True  }]
USEFUL-ERROR-MESSAGE:
```

Install Contour

Contour is an open-source Kubernetes ingress controller providing the control plane for the Envoy edge and service proxy. Tanzu Kubernetes Grid includes signed binaries for Contour and Envoy, which you can deploy into workload clusters to provide ingress control services in those clusters.

After you have set up the cluster, you must first create the configuration file that is used when you install the Contour package and then install the package.

Package installation can be customized by entering the user-configurable values in YAML format. Following is an example YAML for customizing Contour installation:

```
---
infrastructure_provider: vsphere
namespace: tanzu-system-ingress
contour:
 configFileContents: {}
 useProxyProtocol: false
 replicas: 2
 pspNames: "vmware-system-restricted"
 logLevel: info
envoy:
 service:
   type: LoadBalancer
   annotations: {}
   nodePorts:
     http: null
     https: null
   externalTrafficPolicy: Cluster
   disableWait: false
 hostPorts:
   enable: true
   http: 80
   https: 443
 hostNetwork: false
 terminationGracePeriodSeconds: 300
 logLevel: info
 pspNames: null
certificates:
 duration: 8760h
 renewBefore: 360h
```

For a full list of user-configurable values, see Configure the Contour Extension.

1. Capture the available Contour package versions.

   ```
   # tanzu package available list contour.tanzu.vmware.com -A

   NAMESPACE   NAME                        VERSION               RELEASED-AT
   tkg-system  contour.tanzu.vmware.com  1.24.4+vmware.1-tkg.1  2023-04-28 00:00:0
   0 +0000 UTC
   ```

   Capture the latest version from the previous command. If there are multiple versions available, check the “RELEASED-AT” to collect the version of the latest one. This document makes use of version 1.24.4+vmware.1-tkg.1 for installation.

2. Install the Contour package.

```
tanzu package install contour --package contour.tanzu.vmware.com --version <AVA
ILABLE-PACKAGE-VERSION> --values-file <path to contour-data-values.yaml> --name
space tanzu-contour-ingress

# kubectl create namespace tanzu-system-ingress
# kubectl create namespace tanzu-contour-ingress
# tanzu package install contour --package contour.tanzu.vmware.com --version 1.
24.4+vmware.1-tkg.1 --values-file contour-data-values.yaml --namespace tanzu-co
ntour-ingress

8:12:04AM: Creating service account 'contour-tanzu-contour-ingress-sa'
8:12:04AM: Creating cluster admin role 'contour-tanzu-contour-ingress-cluster-r
ole'
8:12:04AM: Creating cluster role binding 'contour-tanzu-contour-ingress-cluster
-rolebinding'
8:12:04AM: Creating secret 'contour-tanzu-contour-ingress-values'
8:12:04AM: Creating overlay secrets
8:12:04AM: Creating package install resource
8:12:04AM: Waiting for PackageInstall reconciliation for 'contour'
8:12:04AM: Fetch started (1s ago)
8:12:05AM: Fetching
        | apiVersion: vendir.k14s.io/v1alpha1
        | directories:
        | - contents:
        |   - imgpkgBundle:
        |       image: harbor.sfo01.rainpole.vmw/tkgm-images/packages/standard/
repo@sha256:20db584c146086a789ab29e3efd24a8b406054a945607322abd134f38c603013
        |     path: .
        |   path: "0"
        | kind: LockConfig
        |
8:12:05AM: Fetch succeeded
8:12:06AM: Template succeeded
```

3. Confirm that the Contour package has been installed and the status is `Reconcile succeeded`.

```
# tanzu package installed get contour --namespace tanzu-contour-ingress

NAME:                    contour
PACKAGE-NAME:            contour.tanzu.vmware.com
PACKAGE-VERSION:         1.24.4+vmware.1-tkg.1
STATUS:                  Reconcile succeeded
CONDITIONS:              [{ReconcileSucceeded True   }]
USEFUL-ERROR-MESSAGE:
```

Install Harbor

Harbor is an open-source container registry. Harbor Registry may be used as a private registry for container images that you want to deploy to Tanzu Kubernetes clusters.

Tanzu Kubernetes Grid includes signed binaries for Harbor, which you can deploy into:

- A workload cluster to provide container registry services for that clusters.
- A shared services cluster to provide container registry services for other Tanzu Kubernetes (workload) clusters.

When deployed as a shared service, Harbor is available to all of the workload clusters in a given Tanzu Kubernetes Grid instance.

Follow this procedure to deploy Harbor into a workload cluster or a shared services cluster.

1. Confirm that the Harbor package is available in the cluster and retrieve the version of the available package.

   ```
   # tanzu package available list harbor.tanzu.vmware.com -A

   NAMESPACE   NAME                         VERSION               RELEASED-AT
   tkg-system  harbor.tanzu.vmware.com  2.8.2+vmware.2-tkg.1  2023-06-08 10:18:00
   +0000 UTC
   ```

2. Create a configuration file named `harbor-data-values.yaml` by executing the following commands:

   ```
   image_url=$(kubectl -n tkg-system get packages harbor.tanzu.vmware.com.2.8.2+vm
   ware.2-tkg.1 -o jsonpath='{.spec.template.spec.fetch[0].imgpkgBundle.image}')

   imgpkg pull -b $image_url -o /tmp/harbor-package --registry-ca-cert-path /etc/d
   ocker/certs.d/harbor.tanzu.lab/ca.crt

   cp /tmp/harbor-package/config/values.yaml harbor-data-values.yaml
   ```

3. Set the mandatory passwords and secrets in the `harbor-data-values.yaml` file.

   ```
   bash /tmp/harbor-package/config/scripts/generate-passwords.sh harbor-data-value
   s.yaml
   ```

4. Edit the `harbor-data-values.yaml` file and configure the values for the following mandatory parameters:

   - namespace
   - port
   - harborAdminPassword
   - secretKey

   You can also change the values for other parameters to meet the requirements for your deployment. For the full list of the user-configurable values, see [https://techdocs.broadcom.com/us/en/vmware-tanzu/cli/tanzu-packages/latest/tnz-packages/packages-harbor-mc.html#deploy).

5. Remove the comments in the `harbor-data-values.yaml` file.

   ```
   yq -i eval '... comments=""' harbor-data-values.yaml
   ```

6. Install the Harbor package by executing the following command:

   ```
   # kubectl create namespace tanzu-system-registry
   # kubectl create namespace tanzu-harbor-registry
   # tanzu package install harbor --package-name harbor.tanzu.vmware.com --version
   2.8.2+vmware.2-tkg.1 --values-file harbor-data-values.yaml --namespace tanzu-ha
   rbor-registry
   ```

```
 8:01:14AM: Creating service account 'harbor-tanzu-system-registry-sa'
 8:01:14AM: Creating cluster admin role 'harbor-tanzu-system-registry-cluster-r
ole'
 8:01:15AM: Creating cluster role binding 'harbor-tanzu-system-registry-cluster
-rolebinding'
 8:01:15AM: Creating secret 'harbor-tanzu-system-registry-values'
 8:01:15AM: Creating overlay secrets
 8:01:15AM: Creating package install resource
 8:01:15AM: Waiting for PackageInstall reconciliation for 'harbor'
 8:01:15AM: Fetch started (6s ago)
        | 8:04:50AM:  L ongoing: waiting on pod/harbor-registry-78c99df744-v8ps
j (v1) namespace: tanzu-system-registry
        | 8:04:50AM:     ^ Condition Ready is not True (False)
        | 8:04:52AM: ok: reconcile deployment/harbor-registry (apps/v1) namespa
ce: tanzu-system-registry
        | 8:04:52AM: ---- applying complete [50/50 done] ----
        | 8:04:52AM: ---- waiting complete [50/50 done] ----
        | Succeeded
8:04:52AM: Deploy succeeded
```

7. Confirm that the Harbor package has been installed and the status is `Reconcile succeeded`.

```
# tanzu package installed get harbor --namespace tanzu-system-registry


NAME:                        harbor
PACKAGE-NAME:                harbor.tanzu.vmware.com
PACKAGE-VERSION:             2.8.2+vmware.2-tkg.1
STATUS:                      Reconcile succeeded
CONDITIONS:                  [{ReconcileSucceeded True   }]
USEFUL-ERROR-MESSAGE:
```

Install Prometheus

Prometheus is a system and service monitoring system. It collects metrics from configured targets at given intervals, evaluates rule expressions, displays the results, and can trigger alerts if some condition is observed to be true. The Alertmanager handles alerts generated by Prometheus and routes them to their receiving endpoints.

Do the following to deploy Prometheus into a workload cluster:

1. Capture the available Prometheus version.

```
# tanzu package available list prometheus.tanzu.vmware.com -A

NAMESPACE   NAME                         VERSION                 RELEASED-AT
tkg-system  prometheus.tanzu.vmware.com  2.27.0+vmware.1-tkg.1  2021-05-12 18:0
0:00 +0000 UTC
tkg-system  prometheus.tanzu.vmware.com  2.27.0+vmware.2-tkg.1  2021-05-12 18:0
0:00 +0000 UTC
tkg-system  prometheus.tanzu.vmware.com  2.36.2+vmware.1-tkg.1  2022-06-23 18:0
0:00 +0000 UTC
tkg-system  prometheus.tanzu.vmware.com  2.37.0+vmware.1-tkg.1  2022-10-25 18:0
0:00 +0000 UTC
tkg-system  prometheus.tanzu.vmware.com  2.37.0+vmware.2-tkg.1  2022-10-25 18:0
0:00 +0000 UTC
```

```
tkg-system  prometheus.tanzu.vmware.com  2.37.0+vmware.3-tkg.1  2022-10-25 18:0
0:00 +0000 UTC
tkg-system  prometheus.tanzu.vmware.com  2.43.0+vmware.2-tkg.1  2023-03-21 18:0
0:00 +0000 UTC
```

Capture the latest version from the previous command. If there are multiple versions available check the “RELEASED-AT” to collect the version of the latest one. This document makes use of version 2.43.0+vmware.2-tkg.1 for installation.

2. Retrieve the template of the Prometheus package’s default configuration:

```
image_url=$(kubectl -n tkg-system get packages prometheus.tanzu.vmware.com.2.4
3.0+vmware.2-tkg.1 -o jsonpath='{.spec.template.spec.fetch[0].imgpkgBundle.imag
e}')

imgpkg pull -b $image_url -o /tmp/prometheus-package-2.43.0+vmware.2-tkg.1 --re
gistry-ca-cert-path /etc/docker/certs.d/harbor.tanzu.lab/ca.crt

cp /tmp/prometheus-package-2.43.0+vmware.2-tkg.1/config/values.yaml prometheus-
data-values.yaml
```

This creates a configuration file named `prometheus-data-values.yaml` that you can modify.

3. To customize the Prometheus installation, modify the following values:

Key	Default Value	Modified value
Ingress.tlsCertificate.tls.crt	Null	Note: This is optional.
ingress.tlsCertificate.tls.key	Null	<Cert Key provided in Input file Note: This is optional.
ingress.enabled	false	true
ingress.virtual_host_fqdn	prometheus.system.tanzu	prometheus.your-domain

To see a full list of user configurable configuration parameters, see Prometheus Package Configuration Parameters.

4. After you make any changes needed to the `prometheus-data-values.yaml` file, remove all comments in the file:

```
yq -i eval '... comments=""' prometheus-data-values.yaml
```

5. Install Prometheus package.

```
# kubectl create namespace tanzu-system-monitoring
# kubectl create namespace tanzu-prometheus-monitoring
# tanzu package install prometheus --package-name prometheus.tanzu.vmware.com -
-version 2.43.0+vmware.2-tkg.1 --values-file prometheus-data-values.yaml --name
space tanzu-prometheus-monitoring

8:20:09AM: Creating service account 'prometheus-tanzu-system-monitoring-sa'
8:20:09AM: Creating cluster admin role 'prometheus-tanzu-system-monitoring-clus
```

```
ter-role'
8:20:09AM: Creating cluster role binding 'prometheus-tanzu-system-monitoring-cl
uster-rolebinding'
8:20:09AM: Creating secret 'prometheus-tanzu-system-monitoring-values'
8:20:09AM: Creating overlay secrets
8:20:09AM: Creating package install resource
8:20:09AM: Waiting for PackageInstall reconciliation for 'prometheus'

        | 8:22:02AM:  L ok: waiting on replicaset/alertmanager-56f6ccfc64 (app
s/v1) namespace: tanzu-system-monitoring
        | 8:22:02AM:  L ok: waiting on pod/alertmanager-56f6ccfc64-h5tl9 (v1) n
amespace: tanzu-system-monitoring
        | 8:22:03AM: ok: reconcile deployment/alertmanager (apps/v1) namespace:
tanzu-system-monitoring
        | 8:22:03AM: ---- waiting on 1 changes [35/36 done] ----
        | 8:22:23AM: ok: reconcile deployment/prometheus-server (apps/v1) names
pace: tanzu-system-monitoring
        | 8:22:23AM: ---- applying complete [36/36 done] ----
        | 8:22:23AM: ---- waiting complete [36/36 done] ----
        | Succeeded
8:22:23AM: Deploy succeeded (1s ago)
```

6. Confirm that the Prometheus package has been installed successfully and the status is `Reconcile succeeded`.

```
# tanzu package installed get prometheus -n tanzu-prometheus-monitoring

NAME:                    prometheus
PACKAGE-NAME:            prometheus.tanzu.vmware.com
PACKAGE-VERSION:         2.43.0+vmware.2-tkg.1
STATUS:                  Reconcile succeeded
CONDITIONS:              [{ReconcileSucceeded True  }]
USEFUL-ERROR-MESSAGE:
```

Install Grafana

Grafana allows you to query, visualize, alert on, and explore metrics no matter where they are stored. Grafana provides tools to form graphs and visualizations from application data.

Grafana is configured with Prometheus as a default data source. If you have customized the Prometheus deployment namespace and it is not deployed in the default namespace, `tanzu-system-monitoring`, you need to change the Grafana data source configuration in the following code.

1. Retrieve the version of the available package.

```
# tanzu package available list grafana.tanzu.vmware.com -A

NAMESPACE   NAME                       VERSION               RELEASED-AT
tkg-system  grafana.tanzu.vmware.com  7.5.16+vmware.1-tkg.1  2022-05-19 18:00:0
0 +0000 UTC
tkg-system  grafana.tanzu.vmware.com  7.5.17+vmware.1-tkg.2  2022-05-19 18:00:0
0 +0000 UTC
```

```
tkg-system  grafana.tanzu.vmware.com  7.5.7+vmware.1-tkg.1   2021-05-19 18:00:0
0 +0000 UTC
tkg-system  grafana.tanzu.vmware.com  7.5.7+vmware.2-tkg.1   2021-05-19 18:00:0
0 +0000 UTC
tkg-system  grafana.tanzu.vmware.com  9.5.1+vmware.2-tkg.1   2022-05-19 18:00:0
0 +0000 UTC
```

Capture the latest version from the previous command. If there are multiple versions available check the “RELEASED-AT” to collect the version of the latest one. This document uses the version 9.5.1+vmware.2-tkg.1 for installation.

2. Retrieve the template of the Grafana package’s default configuration.

```
image_url=$(kubectl -n tkg-system get packages grafana.tanzu.vmware.com.9.5.1+v
mware.2-tkg.1 -o jsonpath='{.spec.template.spec.fetch[0].imgpkgBundle.image}')

imgpkg pull -b $image_url -o /tmp/grafana-package-9.5.1+vmware.2-tkg.1 --regist
ry-ca-cert-path /etc/docker/certs.d/harbor.tanzu.lab/ca.crt

cp /tmp/grafana-package-9.5.1+vmware.2-tkg.1/config/values.yaml grafana-data-va
lues.yaml
```

This creates a configuration file named `grafana-data-values.yaml` that you can modify. For a full list of user-configurable values, see Grafana Package Configuration Parameters.

3. Edit grafana-data-values.yaml and replace the following with your custom values.

Key	Default Value	Modified value
secret.admin_password	Null	Your password in Base64 encoded format.
grafana.service.type	LoadBalancer	NodePort
ingress.virtual_host_fqdn	grafana.system.tanzu	User-Provided FQDN from Input File
ingress.tlsCertificate.tls.crt	Null	Full chain cert provided in Input file
ingress.tlsCertificate.tls.key	Null	Full chain cert provided in Input file

4. (Optional) Modify the Grafana data source configuration.

   Grafana is configured with Prometheus as a default data source. If you have customized the Prometheus deployment namespace and it is not deployed in the default namespace, `tanzu-system-monitoring`, you need to change the Grafana data source configuration in `grafana-data-values.yaml`.

```
datasources:
        - name: Prometheus
          type: prometheus
          url: prometheus-server.<change-to-prometheus-namespace>.svc.cluster.l
ocal
```

5. Remove all comments from `grafana-data-values.yaml` file.

```
yq -i eval '... comments=""' grafana-data-values.yaml
```

6. Install Grafana.

```
# kubectl create namespace tanzu-system-dashboards
# kubectl create namespace tanzu-grafana-dashboards
#  tanzu package install grafana --package-name grafana.tanzu.vmware.com --vers
ion 9.5.1+vmware.2-tkg.1 --values-file grafana-data-values.yaml --namespace tan
zu-grafana-dashboards

8:12:41AM: Creating service account 'grafana-tanzu-system-dashboards-sa'
8:12:42AM: Creating cluster admin role 'grafana-tanzu-system-dashboards-cluster
-role'
8:12:42AM: Creating cluster role binding 'grafana-tanzu-system-dashboards-clust
er-rolebinding'
8:12:42AM: Creating secret 'grafana-tanzu-system-dashboards-values'
8:12:42AM: Creating overlay secrets
8:12:42AM: Creating package install resource
8:12:42AM: Waiting for PackageInstall reconciliation for 'grafana'
        | 8:14:19AM: ongoing: reconcile deployment/grafana (apps/v1) namespace:
tanzu-system-dashboards
        | 8:14:19AM:  ^ Waiting for 1 unavailable replicas
        | 8:14:19AM:  L ok: waiting on replicaset/grafana-58656c5f9b (apps/v1)
namespace: tanzu-system-dashboards
        | 8:14:19AM:  L ongoing: waiting on pod/grafana-58656c5f9b-mjphv (v1) n
amespace: tanzu-system-dashboards
        | 8:14:19AM:     ^ Condition Ready is not True (False)
        | 8:14:31AM: ok: reconcile deployment/grafana (apps/v1) namespace: tanz
u-system-dashboards
        | 8:14:31AM: ---- applying complete [18/18 done] ----
        | 8:14:31AM: ---- waiting complete [18/18 done] ----
        | Succeeded
8:14:31AM: Deploy succeeded
```

7. Confirm that the Grafana package has been installed and the status is `Reconcile succeeded`.

```
# tanzu package installed get grafana -n tanzu-grafana-dashboards

NAME:                    grafana
PACKAGE-NAME:            grafana.tanzu.vmware.com
PACKAGE-VERSION:         9.5.1+vmware.2-tkg.1
STATUS:                  Reconcile succeeded
CONDITIONS:              [{ReconcileSucceeded True  }]
USEFUL-ERROR-MESSAGE:
```

Install Fluent Bit

Fluent Bit is a lightweight log processor and forwarder that allows you to collect data and logs from different sources, unify them, and send them to multiple destinations.

The current release of Fluent Bit allows you to gather logs from management clusters or Tanzu Kubernetes clusters running in vSphere, Amazon EC2, and Azure. You can then forward them to a log storage provider such as Elastic Search, Kafka, Splunk, or an HTTP endpoint.

The example shown in this document uses HTTP endpoint `vRealize Log Insight` for forwarding logs from Tanzu Kubernetes clusters.

1. Retrieve the version of the available package.

```
# tanzu package available list fluent-bit.tanzu.vmware.com -A
```

```
NAMESPACE   NAME                           VERSION                RELEASED-AT
tkg-system  fluent-bit.tanzu.vmware.com  1.7.5+vmware.1-tkg.1   2021-05-13 18:0
0:00 +0000 UTC
tkg-system  fluent-bit.tanzu.vmware.com  1.7.5+vmware.2-tkg.1   2021-05-13 18:0
0:00 +0000 UTC
tkg-system  fluent-bit.tanzu.vmware.com  1.8.15+vmware.1-tkg.1  2022-05-24 18:0
0:00 +0000 UTC
tkg-system  fluent-bit.tanzu.vmware.com  1.9.5+vmware.1-tkg.2   2022-06-23 18:0
0:00 +0000 UTC
tkg-system  fluent-bit.tanzu.vmware.com  2.1.2+vmware.1-tkg.1   2022-06-23 18:0
0:00 +0000 UTC
```

Capture the latest version from the previous command. If there are multiple versions available, check the "RELEASED-AT" to collect the version of the latest one. This document uses the version 2.1.2+vmware.1-tkg.1 for installation.

2. Retrieve the template of the Fluent Bit package's default configuration.

```
image_url=$(kubectl -n tkg-system get packages fluent-bit.tanzu.vmware.com.2.1.
2+vmware.1-tkg.1  -o jsonpath='{.spec.template.spec.fetch[0].imgpkgBundle.imag
e}')

imgpkg pull -b $image_url -o /tmp/fluent-bit-2.1.2+vmware.1-tkg.1 --registry-ca
-cert-path /etc/docker/certs.d/harbor.tanzu.lab/ca.crt

cp /tmp/fluent-bit-2.1.2+vmware.1-tkg.1/config/values.yaml fluentbit-data-value
s.yaml
```

3. Modify the resulting `fluentbit-data-values.yaml` file and configure the endpoint as per your choice. A sample endpoint configuration for sending logs to vRealize Log Insight Cloud over HTTP is shown in the following example.

```
[OUTPUT]
        Name                  syslog
        Match                 *
        Host                  vrli.lab.vmw
        Port                  514
        Mode                  udp
        Syslog_Format         rfc5424
        Syslog_Hostname_key   tkg_cluster
        Syslog_Appname_key    pod_name
        Syslog_Procid_key     container_name
        Syslog_Message_key    message
        Syslog_SD_key         k8s
        Syslog_SD_key         labels
        Syslog_SD_key         annotations
        Syslog_SD_key         tkg
```

4. Deploy Fluent Bit.

```
# kubectl create namespace tanzu-system-logging
# kubectl create namespace tanzu-fluent-bit-logging

 tanzu package install fluent-bit --package-name fluent-bit.tanzu.vmware.com --
version 2.1.2+vmware.1-tkg.1 --namespace tanzu-fluent-bit-logging --values-file
fluent-bit-data-values.yaml
```

```
ℹ   Installing package 'fluent-bit.tanzu.vmware.com'
ℹ   Getting package metadata for 'fluent-bit.tanzu.vmware.com'
ℹ   Creating service account 'fluent-bit-tanzu-fluent-bit-logging-sa'
ℹ   Creating cluster admin role 'fluent-bit-tanzu-fluent-bit-logging-cluster-rol
e'
ℹ   Creating cluster role binding 'fluent-bit-tanzu-fluent-bit-logging-cluster-r
olebinding'
ℹ   Creating package resource
ℹ   Waiting for 'PackageInstall' reconciliation for 'fluent-bit'
ℹ   'PackageInstall' resource install status: Reconciling
ℹ   'PackageInstall' resource install status: ReconcileSucceeded
ℹ
Added installed package 'fluent-bit'
```

5. Confirm that the Fluent Bit package has been installed and the status is `Reconcile succeeded`.

```
# tanzu package installed get fluent-bit --namespace tanzu-fluent-bit-logging

NAME:                    fluent-bit
PACKAGE-NAME:            fluent-bit.tanzu.vmware.com
PACKAGE-VERSION:         2.1.2+vmware.1-tkg.1
STATUS:                  Reconcile succeeded
CONDITIONS:              [{ReconcileSucceeded True  }]
USEFUL-ERROR-MESSAGE:
```

Appendix

Appendix A - Management Cluster Configuration File

```
AVI_CA_DATA_B64: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUVJVENDQXdtZ0F3SUJBZ0lVZVUrT
TMvUHNlN2hkS3kwWG5WMGxFTk5jdHRBd0RRWUpLb1pJaHZjTkFRRUwKQlFBd2dhd3hDekFKQmdOVkJBWVRBbFZ
UTVFzd0NRWURWUVFJREFKRFFURVNNQkFHQTFVRUJ3d0pVR0ZzYnlCQgpiSFJ2TVE4d0RRWURWUVFLREFaV1RYZ
GhjbVV4R3pBWkJnTlZCQXNNRWxaTmQyRnlaU0JGYm1kcGJtVmxjbWx1Clp6RXNNQ29HQTFVRUF3d2pjMlp2TUR
GaGJHSmpkR3h5TURFdWMyWnZNREV1Y21GcGJuQnZiR1V1Ykc5allXd3gKSURBZUJna3Foa2lHOXcwQkNRRVdFV
1Z0WVdsc1FHVjRZVzF3YkdVdVkyOXRNQjRYRFRJek1ETXhOakEyTkRVeApORm9YRFRJME1ETXhOVEEyTkRVeE5
Gb3dnYXd4Q3pBSkJnTlZCQVlUQWxWVE1Rc3dDUVlEVlFRSURBSkRRVEVTCk1CQUdBMVVFQnd3SlVHRnNieUJCY
khSdk1ROHdEUVlEVlFRS0RBWldUWGRoY21VeEd6QVpCZ05WQkFzTUVsWk4KZDJGeVpTQkZibWRwYm1WbGNtbHV
aekVzTUNvR0ExVUVBd3dqYzJadk1ERmhiR0pqZEd4eU1ERXVjMlp2TURFdQpjbUZwYm5CdmJHVXViRzlqWVd3e
ElEQWVCZ2txaGtpRzl3MEJDUUVXRVdWdFlXbHNRR1Y0WVcxd2JHVXVZMjl0Ck1JSUJJakFOQmdrcWhraUc5dzB
CQVFFRkFBT0NBUThBTUlJQkNnS0NBUUVBcU1ZbmoyMWRETklveVdyMlJSYkEKdzhjei9DM0tTMU5WcXozTXJsW
ktNckQvUHhnVVJvc3c4Rzl5UkZ0aEZIRUJERXZYODhKY1gyYUtiRHFpaXRKVApiL2UvUHNuTkl4UlMvUG94T3U
wQTFTMGJRZ3d1ZUpOMjNXc1NTeUJUR0RmeE1MT3p3NitONXFmZkdlY3Y1c2hlCmVRQjljaXZvTTJOZTJZVTFDV
kFkcll5WFNJbWljdmFPZ2kvUlE1ZGpYY0VEMDg5TGRnQkV6TTgvckxCM1g0ZWQKMG05Qm10c1NQYUFja3JLZnR
BZFBxMUlodHpFSGZaeno0VkF6VXFjdjkyQlR1eUM1cDVwUTRpSVZlSTlHZGxTQQoxL0kxNHQ5NTQ0M29saGVYW
UY4bHR2Q2FkOGtsTFRIY1V0VjN2U0pUUm9hdFh2SzdMSmt3cmVZeTFKclQ5WTU1ClJ3SURBUUFCb3prd056QTF
CZ05WSFJFRUxqQXNnaVJ6Wm04d01XRnNZbU4wYkhJd01XRXVjMlp2TURFdWNtRnAKYm5CdmJHVXViRzlqWVd5S
EJLd1FDaEl3RFFZSktvWklodmNOQVFFTEJRQURnZ0VCQUJZV0xZdzZqaHFzU1pDVQp0WnBDYXB1OGl3RU9Ma3J
0Q3N1WUkvd0dWUGtJRnIramJiRTFzMFkvTlBRb1VsalJwbDNRSFp4bzYveFFCQllMCnI4RHFxMk1UOHI0U0t1T
XNmRGl6cTB6RHJUSUhwUjQ1RlBWV0dXRU05Q2hqeWJMRGJWazhyY1ZYZ2F4L3JIZUYKZlJGZmZka1lTdzRYNlp
CbitYMmlYL1p0eHZxSStDbzFrdFNGZG9DSm5kbVRwdFAxaks3S3E4dGZiM05EYk9QaAp0dlRpY0s4SDRJY01SK
2JwQUNKdTFxdHI3eThyTHZpa1l3S0dTLzR5WGh0Ky8zU3QxZVdFRWljYnpyTk1VN0FBCkhkZVNhYzlTc0Y1TDB
sVkVnbVpNZUFKT1NrOE9tTVFzUnB5M29pRjMwQXZ5SDZqa0Jhc2QwcUNJUVlySjluUVQKMGoweU9Pbz0KLS0tL
S1FTkQgQ0VSVElGSUNBVEUtLS0tLQ==
AVI_CLOUD_NAME: sfo01w01vc01
AVI_CONTROL_PLANE_HA_PROVIDER: "true"
AVI_CONTROL_PLANE_NETWORK: sfo01-w01-vds01-tkgclustervip
```

```
AVI_CONTROL_PLANE_NETWORK_CIDR: 172.16.80.0/24
AVI_CONTROLLER: sfo01albctlr01.sfo01.rainpole.local
AVI_DATA_NETWORK: sfo01-w01-vds01-tkgworkloadvip
AVI_DATA_NETWORK_CIDR: 172.16.70.0/24
AVI_NSXT_T1LR: /infra/tier-1s/sfo01w01tier1
AVI_ENABLE: "true"
AVI_LABELS: |
    'type': 'management'
AVI_MANAGEMENT_CLUSTER_CONTROL_PLANE_VIP_NETWORK_CIDR: 172.16.80.0/24
AVI_MANAGEMENT_CLUSTER_CONTROL_PLANE_VIP_NETWORK_NAME: sfo01-w01-vds01-tkgclustervip
AVI_MANAGEMENT_CLUSTER_SERVICE_ENGINE_GROUP: sfo01m01segroup01
AVI_MANAGEMENT_CLUSTER_VIP_NETWORK_CIDR: 172.16.80.0/24
AVI_MANAGEMENT_CLUSTER_VIP_NETWORK_NAME: sfo01-w01-vds01-tkgclustervip
AVI_PASSWORD: <encoded:Vk13YXJlMSE=>
AVI_SERVICE_ENGINE_GROUP: sfo01w01segroup01
AVI_USERNAME: admin
TKG_CUSTOM_IMAGE_REPOSITORY: "harbor.tanzu.lab/tanzu-170"
TKG_CUSTOM_IMAGE_REPOSITORY_SKIP_TLS_VERIFY: false
TKG_CUSTOM_IMAGE_REPOSITORY_CA_CERTIFICATE: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSU
ZzVENDQTVtZ0F3SUJBZ0lKQUtvLzY4U3RGSC8vTUEwR0NTcUdTSWIzRFFFQkRRVUFNRzh4Q3pBSkJnTlYKQkFZ
VEFrTk9NUkl3RUFZRFZRUUlEQWxMWVhKdVlYUmhhMkV4RWpBUUJnTlZCQWNNQ1VKaGJtZGhiRzl5WlRFUApNQT
BHQTFVRUNnd0dWazEzWVhKbE1Rd3dDZ1lEVlFRTERBTldUVU14R1RBWEJnTlZCQU1NRUdoaGNtSnZjaTUwClllX
NTZkUzVzWVdJd0hoY05Nak13TXpJd01UUTBOakV5V2hjTk16TXdNekUzTVRRME5qRXlXakJ2TVFzd0NRWUQKVVl
FRR0V3SkRUakVTTUJBR0ExVUVDQXdKUzJGeWJtRjBZV3RoTVJJd0VBWURWUVFIREFsQ1lXNW5ZV3h2Y21VeApE
ekFOQmdOVkJBb01CbFpOZDJGeVpURU1NQW9HQTFVRUN3d0RWazFETVJrd0Z3WURWUVFEREJCb1lYSmliM0l1Cm
RHRnVlblV1YkdGaU1JSUNJakFOQmdrcWhraUc5dzBCQVFFRkFBT0NBZzhBTUlJQ0NnS0NBZ0VBMSs1NEJXNU0K
b0xIZ3BJRUZEUk9OTngyR2FtVkxPdEh0MlNVSWhmcmtlWXJ0Nlc4dWNoM3B3WUVOS3BuSGtaWU85a09kcnpIag
pETENmSzR6dDdxY0J4R2pZSEJPY3pablkxL0oxWTRLU1l3UHc2ZzU2YU82MThFMGY3RDc4T05udkVZVjcxQlZJ
Cmd3SmFoa0g1bCtTM3dURlZYd00xNGMxMTVNYitmV1Bmb1RSSGdwVnM2N1IzR0pFMGlBWEJDbHJRS3JsbDdCQW
kKRDIxdVZxcnA1anEvd2xHUHN6ZkdDRUF3N21Wc2Z5RWIwWm1vV21jdnExRER1T3hLQjlFdjcyaUloRURmeDhJ
Ugo1OWVTVHFvbzg2Q0xEaWxPMnduRVZuaEVBQy8wL0hGYUdmeWhLQ09neE50RWtER1c3ei9KWHNNb2Q3N3Bhdz
N4CkpqNEFSQWkvZ2ZlYVlIMjlGZnBWTHF2VitNazJydTF0ZHdING5RUTlra21CYlh2NDNIL3krSVBySDZxMUpF
NlAKeGV0Lzh6UGE5RHd6eWs1Y0NPbHU0SDRLOSs5eGVnN0lKRk5jQk8reE1XQ2VtaUhlVXZ3SlBBRUcyQmFUSV
duQwpxNGNFNEY5ck1ybHFHeW9sZkhURzFCSTR1dDl3NWRVbFViUlJuM0x5SHpZcG9tc1BvZ3lIb29QbXBiWk9h
bTY4CjV4N1NZdEFka2ZnYW1kV2VPYU9Qc09Ob1VXenA1UTZIT0lHaVI3Z2ZiU3Yxa3owNzBnYXMvRlJVK1F4Zk
VkWncKeGhOWjlvUENNBYWJFMVJPRkhhTGszNFUrZ2lHclNiZmNzUUdZcFBYZUR0TS9GSUlsTGdiSGMrZEEZINWNU
bmcwLworR2RUR01qdkI5UHJTc05VekJTMGdNK3dTT3luKzZqWXJtMENBd0VBQWFOOUU1FNHdIUVlEVlIwT0JCWU
VGTHl3CnErVENNpRDZxNllvTy9IMnNmRnE1cE1ERE1COEdBMVVkSXdRWU1CYUFGTHl3cStUQ2lENnE2WW9PL0gy
c2ZGcTUKcE1ERE1Bd0dBMVVkRXdRRk1BTUJBZjh3RFFZSktvWklodmNOQVFFTkJRQURnZ0lCQUJ4dkxQWUg4dT
FudTZVMQpScEhMQ1JLVy91eGlBS3ZHR1UyWkR0ZVArS243Qml1RlorNlBWZE9EaVI0QjRqcmxCc241TkhvanBo
a2tTL2QrCmJkRFB3NlRtb2RTWjFhcjdNQUNSODFRdlpLcmtSUm05VklZdVJwMTExalliRWowY1dHbUp6eVhZQW
ZxREMydFMKUCs4NWYzazlrT2ZpclFhWld3SkJGcDViSklLNmhKam5DN3NwdmJaRHRqTGGvZTBRbXdzNHRveWRG
OTRMaEdrZAp2cXB4OWJuSXIrV1cyTkNKbzRGa3ppMnhJcDRnVG50N1Z2bkI0MnFLaW80a0FxbXpDN1VSU3daVW
YrT1NKUkpFCm5RNWpLbEF0MllZZnE0b0F4OVN6dm5wWjRQSkhKT1FseEJJTNmplWkh6UzRjdS9qMHIwbHRkdDh4
MWo0eEkwQSsKK01oSXRpdVVlWEZMUHNJYjdFRU9zUjlZblUxOVdTd0lUYTAyMVFVVVVHTWlvRlhhYno0VkpEY2
V1K0YvK1d2Rgp6RndXMDBvbTZ0dWREbm9qVFpzcEc4cUhkS2NiLytSRzJVak14NGFmVMRCtrRmdsZGozc3hOZWwx
dldhMWNnK0hzCKd5UFl5ejJlUlFyRG5yU21QMFN5QlFpallZV1QwNTd4TUUwK1M0ZzdZQXRKQWFPNDdtbW41dW
RxQ2lmNmZlcHMKYW5Sd3hnOHY3N1Zvb2IvZUdxZFBjZmpNQXZEdm1Ed0VhZW1OWnJOM1g2ODlJdllTdkF2R1Aw
Y2ovKzc1VVJBNApsOTFYU3FPdUMrd0RzRXVUUlNsQ1ZDMzhZNzJpQmNnK2VtUjg2dWc1M1Y4WVQyVTRwWVEwdk
RMR3NZNWduWGlFCmVpSEFmK0M1TGhrMlc2ZnRBUElnRUdRV3NET0kKLS0tLS1FTkQgQ0VSVElGSUNBVEUtLS0t
LQo=
CLUSTER_ANNOTATIONS: 'description:,location:'
CLUSTER_CIDR: 100.96.0.0/11
CLUSTER_NAME: sfo01w01tkgmgmt01
CLUSTER_PLAN: prod
ENABLE_AUDIT_LOGGING: "true"
ENABLE_CEIP_PARTICIPATION: "true"
ENABLE_MHC: "true"
IDENTITY_MANAGEMENT_TYPE: none
```

```
INFRASTRUCTURE_PROVIDER: vsphere
LDAP_BIND_DN: ""
LDAP_BIND_PASSWORD: ""
LDAP_GROUP_SEARCH_BASE_DN: ""
LDAP_GROUP_SEARCH_FILTER: ""
LDAP_GROUP_SEARCH_GROUP_ATTRIBUTE: ""
LDAP_GROUP_SEARCH_NAME_ATTRIBUTE: cn
LDAP_GROUP_SEARCH_USER_ATTRIBUTE: DN
LDAP_HOST: ""
LDAP_ROOT_CA_DATA_B64: ""
LDAP_USER_SEARCH_BASE_DN: ""
LDAP_USER_SEARCH_FILTER: ""
LDAP_USER_SEARCH_NAME_ATTRIBUTE: ""
LDAP_USER_SEARCH_USERNAME: userPrincipalName
OIDC_IDENTITY_PROVIDER_CLIENT_ID: ""
OIDC_IDENTITY_PROVIDER_CLIENT_SECRET: ""
OIDC_IDENTITY_PROVIDER_GROUPS_CLAIM: ""
OIDC_IDENTITY_PROVIDER_ISSUER_URL: ""
OIDC_IDENTITY_PROVIDER_NAME: ""
OIDC_IDENTITY_PROVIDER_SCOPES: ""
OIDC_IDENTITY_PROVIDER_USERNAME_CLAIM: ""
OS_ARCH: amd64
OS_NAME: photon
OS_VERSION: "3"
SERVICE_CIDR: 100.64.0.0/13
TKG_HTTP_PROXY_ENABLED: "false"
VSPHERE_CONTROL_PLANE_CUSTOM_VMX_KEYS: ""
VSPHERE_CONTROL_PLANE_DISK_GIB: "40"
VSPHERE_CONTROL_PLANE_ENDPOINT: ""
VSPHERE_CONTROL_PLANE_HARDWARE_VERSION: ""
VSPHERE_CONTROL_PLANE_MEM_MIB: "8192"
VSPHERE_CONTROL_PLANE_NUM_CPUS: "2"
VSPHERE_CONTROL_PLANE_PCI_DEVICES: ""
VSPHERE_DATACENTER: /sfo01w01dc01
VSPHERE_DATASTORE: /sfo01w01dc01/datastore/vsanDatastore
VSPHERE_FOLDER: /sfo01w01dc01/vm/tkg-management-components
VSPHERE_IGNORE_PCI_DEVICES_ALLOW_LIST: ""
VSPHERE_INSECURE: "false"
VSPHERE_NETWORK: /sfo01w01dc01/network/sfo01-w01-vds01-tkgmanagement
VSPHERE_PASSWORD: <encoded:Vk13YXJlMSE=>
VSPHERE_RESOURCE_POOL: /sfo01w01dc01/host/sfo01w01cluster01/Resources/tkg-management-c
omponents
VSPHERE_SERVER: sfo01w01vc01.sfo01.rainpole.local
VSPHERE_SSH_AUTHORIZED_KEY: ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQC0mAnVUaKqvQQilprtq5
WMQL3Xf6MVxhKzxvZwqS6thXDD4ET2hqcmztAJorKBqwI6aX5mVM8cIy5lUmLYcsW6PaRt24+wY2JInqtPxr/0
6VIp2CZts+46d8PZ4fQCoqI59yPwE0AU8EOvS9k9AxUPIJwLMbsYMFgu2yveU5sTG0jgYuGodkghMul1ohfMh8
WpKKSnq4QM6Pll0t2k1CfOviwmRDJECET/K34XgFigLx1CbI5HfyTeG84TFPBJs8OiWZMjCEG3++xW1nV9kpLP
2JyOBmDCDVJNS2ZqlGXUkCWaaXQ/VJnPOJIEQHCoJ2GT2mUrjW4kUcrlHkmO04a1Fu3q7RmVxDLKAVxZOc1tkK
HqhUKqbdRYc71vftpI8n9os/hxU4N0uKLuW4ymw2n3+LNiUWVU9UKVOZ3LUT+rm2JlIi/ooTwkwJM+48BeKQ1u
C0jqIAcafpSk9PibXj1BR1Qp7PpB/97d3hUVYfNoiT3zDfyQbDfbwgVMr/DG4Bhow3n34iV72nRLdqPm81ckUU
ZEvbftX9ylYsqm2U4X1zTVAZDXAXUlylyV2bGdt5usoKtp5lK3xI+WlqTEP+WQ65WRBP+gJkjxoe1QkYjYJS8u
3sysy8M/sxJLElutVvAzwBfitYYRAm49A2gTTuekDtELjmKIjgNGALikk1qB/Q== email@example.com
VSPHERE_TLS_THUMBPRINT: 7F:BA:25:AC:DD:B9:89:DD:04:EB:89:B8:76:74:18:F4:23:EC:75:17
VSPHERE_USERNAME: administrator@vsphere.local
VSPHERE_WORKER_CUSTOM_VMX_KEYS: ""
VSPHERE_WORKER_DISK_GIB: "40"
VSPHERE_WORKER_HARDWARE_VERSION: ""
VSPHERE_WORKER_MEM_MIB: "8192"
VSPHERE_WORKER_NUM_CPUS: "2"
```

```
VSPHERE_WORKER_PCI_DEVICES: ""
WORKER_ROLLOUT_STRATEGY: ""
```

Appendix B - Single VIP Nertwork Architecture - Shared Service Cluster ADC file

```
apiVersion: networking.tkg.tanzu.vmware.com/v1alpha1
kind: AKODeploymentConfig
metadata:
  generation: 3
  name: tanzu-ako-for-shared
spec:
  adminCredentialRef:
    name: avi-controller-credentials
    namespace: tkg-system-networking
  certificateAuthorityRef:
    name: avi-controller-ca
    namespace: tkg-system-networking
  cloudName: sfo01w01vc01
  clusterSelector:
    matchLabels:
      type: shared-services
  controlPlaneNetwork:
    cidr: 172.16.80.0/24
    name: sfo01-w01-vds01-tkgclustervip
  controller: 172.16.10.10
  controllerVersion: 22.1.3
  dataNetwork:
    cidr: 172.16.80.0/24
    name: sfo01-w01-vds01-tkgclustervip
  extraConfigs:
    disableStaticRouteSync: false
    ingress:
      defaultIngressController: false
      disableIngressClass: true
      nodeNetworkList:
      - networkName: sfo01-w01-vds01-tkgmanagement
    networksConfig:
      nsxtT1LR: /infra/tier-1s/sfo01w01tier1
  serviceEngineGroup: sfo01m01segroup01
```

Appendix C - Single VIP Nertwork Architecture - Workload Cluster ADC file

```
apiVersion: networking.tkg.tanzu.vmware.com/v1alpha1
kind: AKODeploymentConfig
metadata:
  generation: 3
  name: install-ako-for-workload-02
spec:
  adminCredentialRef:
    name: avi-controller-credentials
    namespace: tkg-system-networking
  certificateAuthorityRef:
    name: avi-controller-ca
    namespace: tkg-system-networking
  cloudName: sfo01w01vc01
  clusterSelector:
    matchLabels:
      type: workload-02
  controlPlaneNetwork:
    cidr: 172.16.80.0/24
    name: sfo01-w01-vds01-tkgclustervip
  controller: 172.16.10.10
  controllerVersion: 22.1.3
  dataNetwork:
    cidr: 172.16.80.0/24
    name: sfo01-w01-vds01-tkgclustervip
  extraConfigs:
    disableStaticRouteSync: true
    ingress:
      defaultIngressController: true
      disableIngressClass: false
      serviceType: NodePortLocal
      shardVSSize: MEDIUM
      nodeNetworkList:
      - networkName: sfo01-w01-vds01-tkgworkload
        cidrs:
        - 172.16.60.0/24
    networksConfig:
      nsxtT1LR: /infra/tier-1s/sfo01w01tier1
  serviceEngineGroup: sfo01w01segroup01
```