Bachelor's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Abel Weldu Abraha
Mehari Berhe Zerai
Mohammed Anas Rihan

Kubernetes in VMware and NSX-T

Bachelor's thesis in Digital Infrastructure and Cyber Security
Supervisor: Ernst Gunnar Gran
May 2022

NTNU
Kunnskap for en bedre verden

Abel Weldu Abraha
Mehari Berhe Zerai
Mohammed Anas Rihan

Kubernetes in VMware and NSX-T

Bachelor's thesis in Digital Infrastructure and Cyber Security
Supervisor: Ernst Gunnar Gran
May 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Preface

The authors of this bachelor's thesis are Abel Weldu, Mehari Berhe, and Mohammed Anas. We would like to thank our supervisor, Ernst Gunnar Gran, for close collaboration and a lot of good advice and guidance throughout the project. We would also like to thank HDO for this inspiring and educational project. We would especially like to thank Morten Singstad from HDO for his close follow-up and good cooperation.

Abstract

The tendency shifts from a monolithic architecture to a microservice architecture has increased the usage of containers. As a result, today's application stacks are often made up of many containers, which again has caused the need for a container orchestration tool like Kubernetes. Helsetjenestens Driftsorganisasjon (HDO) wants to take advantage of such technology to increase its capacity and improve the operation of existing services using Kubernetes. Since HDO provides an environment to host health-related services, which are very critical to be maintained, Kubernetes can be used to optimize HDO's resource usage and improve the availability and scalability of these services.

HDO has virtualized its infrastructure and aims to expand its platform. In 2020, HDO acquired a modern virtualization infrastructure from VMware, which includes technologies such as vSphere and NSX-T. To support HDO in expanding its virtualized platform, the thesis group has installed and configured vSphere with Tanzu on NSX-T. It has been demonstrated how a Kubernetes application can be deployed, patched, and monitored on this virtualized platform.

Sammendrag

Skiftet fra monolittisk arkitektur til mikrotjeneste arkitektur har ført til økt bruk av kontainere. Som et resultat består dagens applikasjonsstakker ofte av mange kontainere, noe som igjen har forårsaket behovet for et kontainerorkestreringsverktøy som Kubernetes. Helsetjenestens Driftsorganisasjon (HDO) ønsker å dra nytte av slik teknologi for å øke kapasiteten og forbedre driften av eksisterende tjenester ved hjalp av Kubernetes. Siden HDO kan brukes som vertsmiljø for helserelaterte tjenester som er svært kritiske for å opprettholdes, kan Kubernetes brukes til å optimalisere HDOs ressursbruk og forbedre tilgjengeligheten og skalerbarheten til disse tjenestene.

HDO har virtualisert infrastrukturen sin og har som mål å utvide plattformen sin. I 2020 kjøpte HDO en moderne virtualiseringsinfrastruktur fra VMware, som inkluderer teknologier som vSphere og NSX-T. For å støtte HDO i å utvide sin virtualiserte plattform, har bachelorgruppen installert og konfigurert vSphere med Tanzu på NSX-T. Det har blitt demonstrert hvordan et Kubernetes applikasjon kan deployes, patches, oppdateres og overvåkes på denne virtualiserte plattformen.

Contents

Figures

Acronyms

ALB Advanced Load Balancer. 26

BIOS Basic Input/Output System. 22

CVE The Common Vulnerabilities and Exposure. 71

DFW Distributed firewall. 72

HDO Helsetjenestens driftsorganisasjon. 1

HTTP HyperText Transfer Protocol. 16

HTTPS Hypertext Transfer Protocol Secure. 16

IPAM IP Address Management. 31

K8s Kubernetes. 11

OVA open virtualization appliance. 27, 29

PSB PodSecurityPolicy. 69

PVC persistent volume claim. 9

RBAC Role-Based Access Control. 69

SDDC software-defined data center. 21

SSH Secure Shell. 16

TCP Transmission Control Protocol. 14, 74

TKGS Tanzu Kubernetes Grid Service. 23

VDS vSphere Distributed Switch. 9

VM Virtual Machine. 8, 12

WQL Wavefront Query Language. 62

Glossary

ESXi A virtualization platform or a type one hypervisor that allows users to create and manage virtual machines in an infrastructure. 7

kubectl a Kubernetes command-line tool users can use to interact with a cluster. 17, 49

Micro-segmentation The process of dividing all components in the network into logical segments and defining security rules and policies for each segment [1]. 72

Microservices are components of an application that are designed to run independently. 14

Monolithic architectures Traditional way of organizing applications. Applications are implemented using a single stack. 14

Type-1 hypervisor Sometimes called bare-metal hypervisor is a virtualization software technology that can be installed directly on the physical machine.. 7

Type-2 hypervisor Type-2 hypervisor is virtualization software technology that hosts virtual machines on top of existing operating system in the physical machine.. 7

vCenter server A service in a virtualized environment that allows users to manage, administer and make changes to multiple hosts.. 7

Chapter 1

Introduction

1.1 Background and objectives

Helsetjenestens Driftsorganisasjon (HDO) is an operating center whose mission is to provide municipalities and health trusts with safe and secure communication solutions [2]. HDO was established in 2013 and is today owned by the four regional health trusts. Each of the four regional health trusts has an ownership share of 20 percent, except Helse Sør-Øst which has 40 percent. The company's primary and prioritized tasks are to deliver stable, cost-effective, and nationwide communication solutions used by Norwegian emergency services. HDO's main services also include maintaining emergency numbers 113 and 116117. Other tasks include contributing to the development of the health sector's emergency call service and providing 24-hour operation and customer support. The users of HDO's services are health personnel in regional healthcare companies and all municipalities in Norway.

Since HDO provides an environment to host health-related services, which are very critical to be maintained, the company needs to be forward-looking when it comes to the operation of new and existing services. HDO is concerned that its services should be scalable, cost-effective, and user-friendly. To achieve this, an operations model that allows a secure, centralized, scalable, and automated method of managing services is required. This is where Tanzu, a VMware product, comes into the picture [3].

1.2 Purpose

Microservice architecture has become a mainstream approach to structuring applications. In a microservice architecture, applications are structured as a collection of autonomous services modeled around a business model [4]. The usage of microservices has led to increased usage of containers. This has resulted in applications comprised of many containers, which again has caused the need for a container orchestration tool like Kubernetes. These types of orchestration tools help manage containers across hosts. HDO wants to take advantage of these types of technologies to improve the operation of new and existing applications. The purpose of this thesis is to evaluate the use of Tanzu, which is a platform for managing Kubernetes clusters.

1.3 Task description

For a long time, deploying a service on a physical server was a common approach. However, in the modern technological era, this approach has been shown to be inefficient since deploying and provisioning applications on a physical server can take a long time. Furthermore, the need for reliability and scalability to manage increasing workloads and network traffic is crucial in today's working world. As a result, HDO intends to expand their environment and virtualize it completely.

In 2020, HDO acquired a modern virtualization infrastructure from VMware, which includes technologies such as vSphere and NSX-T. HDO provides an environment to host health-related services, and one of HDO's objectives is to deliver efficient services to their consumers. This bachelor's thesis will thus pursue this objective by developing an infrastructure with the capability to enhance the capacity, stability, scalability, and security. This infrastructure will be constructed using VMware Tanzu along with vSphere and NSX-T technologies.

This thesis will look more deeply into VMware Tanzu installation and configuration in NSX-T and vSphere, and will further look into numerous criteria for deploying, administering, orchestrating containers, and applying best practices to secure a Tanzu-based service. The thesis is organized into four aspects:

- Installing and configuring Tanzu Kubernetes with vSphere and NSX-T:
  Creating the project's environment by setting up and installing vSphere with Tanzu and configuring networking for this environment using NSX-T.
- Orchestration:
  This includes container and service management, deploying a new service on Tanzu Kubernetes, security patching, and application upgrades.
- Monitoring:
  Set up monitoring for the applications to check that they are functioning properly as well as to monitor the application load.
- vSphere with Tanzu on NSX-T Security:
  This includes best practices for protecting and hardening the whole platform, including container images, Tanzu Kubernetes, deployed services, and vSphere with NSX-T.

1.4 Limitations

There are numerous modifications, improvements, and other related things that can be made in the virtualized-world of computers using several types of technologies and software in a company like HDO. These modifications may have an impact on the services provided to HDO's customers. As a result, it is important to ensure that all modifications or improvements made during the process have limitations to avoid misunderstandings and negative consequences.

Therefore, based on the requirements and expected expansion of HDO's implementation on the newly acquired virtualization platform from VMware and Dell, the new infrastructure should include administering and orchestrating containers using vSphere with Tanzu on NSX-T. It will not consider other alternative forms of orchestration and administration, for example, docker orchestration. In addition to that, the concepts of supervision and security of the running applications in the containers or Kubernetes clusters will be considered. Supervision will include load balancing and monitoring of applications in containers.

To say a whole infrastructure is secured if and only if all its components are secured, or if it is only secured as if its weakest link is secured, Hence, security will be limited to configurations of tools, security at the container level, and runtime applications. The security of the whole infrastructure is beyond the limitations of the thesis.

1.5 Target group

The target group for this bachelor's thesis is mainly HDO. This report is also relevant for others who want to get started with Kubernetes and are interested in knowing how it works in vSphere. For this report to be of great use, it would be advantageous to have some knowledge about containers. It is also an advantage to know about VMware's virtualization products such as vSphere, NSX-T, and Tanzu.

1.6 The groups background

All group members are studying for a bachelor's degree in Digital Infrastructure and Cyber Security at NTNU Gjøvik. The study has provided the group members with knowledge in a variety of topic areas, which are relevant to this thesis. The following courses are related to this thesis:
IIKG3005 - Infrastructure as Code
DCSG2003 - Robust and scalable services
DCSG2001 - Interconnected Networks and Network Security

DCSG1005 - Infrastructure: secure core services
DCSG2005 - Risk Management

No one in the group has prior knowledge of Kubernetes, vSphere, or NSX-T. Therefore, all members must acquire a solid understanding and knowledge of these technologies to begin work on this thesis.

1.7 Methodology

vSphere, NSX-T and Tanzu Kubernetes are new topics for the group. Therefore, understanding all the concepts behind is very important before jumping into how to configure and setup the vSphere with Tanzu on NSX-T and deploying applications. Hence, qualitative approach of collecting and gathering relevant information about all the technologies and concepts supposed to be used in this dissertation is necessary. Moreover, the group agreed to read about the different topics and collect relevant sources in the first two weeks of the project that could help in making the assignments set by the contractor feasible.

The group agreed to use agile scrum model. The project will be broken into iterations, with each iteration consisting of a two-week sprint. The iteration will help the group to divide the features incrementally and assess the work done ongoing. In addition to that, GitLab Issue Board will be used to keep track of the different work distributions and issue identifications.

The dissertation does not involve data analysis; rather, it is practical. Therefore, collecting relevant information will be based somehow on secondary sources of information gathering. It will not involve primary sources of collecting information or data, such as interviewing people from HDO. Documentation from VMware will be the primary online source of information. In addition to that, best practices that are provided by VMware will be used in almost all aspects of configuration, implementation, deployment, and security of the self-selected applications.

1.8 Structure

This thesis is divided into 7 chapters. A brief description of the chapters is given below. This thesis has clickable links to other chapters, sections, and figures.

Chapter 1 Introduction:
This chapter contains the task description, purpose, and limitations of this project.

Chapter 2 Theory:
Chapter 2 explains the basic theory behind the technologies used throughout this thesis.

Chapter 3 Install and Configure Tanzu:
This chapter explains how the project group has installed and configured vSphere with Tanzu on NSX-T.

Chapter 4 Application Management:
Chapter 4 explains how an application can be deployed, scaled, updated, and patched in Tanzu Kubernetes.

Chapter 5 Monitoring:
This chapter explains how the group has implemented monitoring and how it can be used.

Chapter 6 Security:
This chapter provides an overview of relevant security features used to secure vSphere with Tanzu on NSX-T.

Chapter 7 Discussion and Conclusion:
This chapter contains the result of the work, discussions, and recommendations.

Chapter 2

Theory

This chapter explains the basic theory behind the technologies used throughout this thesis. vSphere, NSX-T, Tanzu, and Kubernetes are all extensive topics that would take more than this whole thesis to go through. The focus has been on explaining the basics of vSphere, NSX-T, Kubernetes, and Tanzu.

2.1 Virtualization and containerization

Virtualization is the process of creating virtual computers that are totally isolated from the hosting environment or other virtual machines, while sharing a certain amount of the physical resources of the host or physical machine [5]. It also includes network and storage virtualization.

Containerization is the process of running an application as a result of packaging different resources like application codes, libraries, dependencies, and other necessary elements that make up that application and running it in an isolated container [6].

2.2 VMware

VMware is a software company that deals in virtualization and cloud computing products, and it is based in California. According to the chief executive officer of VMware, Raghu Raghuram, VMware's mission is "*to deliver the trusted software foundation that accelerates customers' innovation*" [7]. This means that, VMware is dedicated to automation, modernization, and simplification of cloud computing software and virtualization for customer innovation. VMware uses ESXi Type-1 hypervisor on the physical server. This hypervisor enables a user to host multiple virtual machines like in a conventional Type-2 hypervisor vSphere, NSX-T, vCenter server, ESXi, etc. are some of the technologies from VMware, and are covered in depth in this thesis. Figure 2.1 shows a simplified setup of the above-mentioned technologies in a VMware infrastructure. vSphere clients are a web-based HTML-5 interface application provided by the vCenter server. It lets the user to connect remotely to vCenter server and gives an administrator interface to access ESXi hosts.

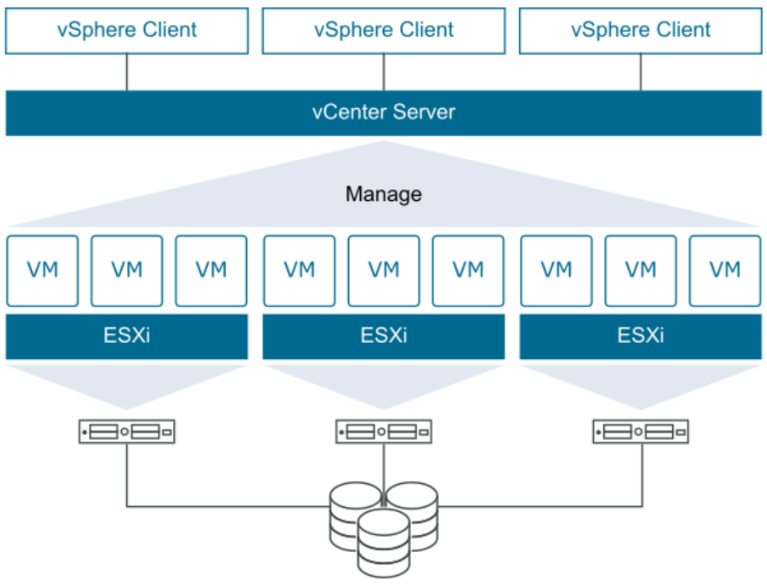


Figure 2.1: A simple VMware infrastructure [8]

2.2.1 vSphere

vSphere is a VMware's cloud computing virtualization platform that reforms different physical infrastructures and all their components into an assembled computing infrastructure [8]. vSphere provides users with different tools that help them manage and administer resources that are in the physical environment, like CPUs, network resources, data storage, etc. A vSphere environment has many components and features. The core vSphere components used throughout this thesis include:

vCenter

A vCenter provides centralized management of a vSphere environment, including ESXi hosts and virtual machines. It is used to monitor and manage thousands of VMs and ESXi hosts. For this project, HDO has set up a vSphere client that will be used to access the vCenter for management purposes.

vSphere Distributed Switch

A vSphere Distributed Swicth (VDS) is a virtual switch that connects hosts within a vSphere cluster to allow centralized management of the network configurations in the vSphere environment. All hosts within the vSphere cluster are attached to this vSphere distributed switch. Distributed switches are created using the vCenter through the vSphere Client. With a VDS in place, there is no need to configure switches on each of the ESXi hosts to have the same configuration. It is configured once by adding the hosts one wishes to have VDS connected to. VDS automates and centralizes the virtual machine networking in the vSphere environment and enables Software Defined Networking (SDN) by setting up switching for the entire data center. This eliminates the manual process of configuring each host in the vSphere environment.

Distributed Port Group

Distributed port groups are required by NSX-T to provide connectivity to virtual machines. Distributed port groups are created and assigned to the vSphere distributed switch. Port groups work by aggregating ports under a common configuration and providing connection points for virtual machines. These port groups are identified by network labels that are unique to the data center.

Storage policies

Storage policies are policies that regulate which type of storage is provided for the virtual machine as well as how the virtual machine is placed within storage. They also pin down which data services in the storage device the VM can access [9]. Storage policies can help to map data from where it originates, it can be the VMs to the physical media in the system. In other words, they are like channels that act as bridges in between the VMs and the storage devices. In VMware, one can either use the default storage policy or define their own.

StorageClass

In the case of Tanzu Kubernetes clusters, if a application requires access to accessible data that can be used or retrieved after, for example, a pod dies in the cluster, one must have a persistent volume claim (PVC). To create a persistent volume claim, a storageClass must be in place. vSphere administrators use a storageClass to describe the different storage they provide in the storage classes. These classes could correspond to different aspects of service levels, backup policies, or any other policies set by cluster administrator [10].

Content libraries

Content libraries are object packages that hold VMs and vApp templates[1] as well as other types of files like ISO images, text files, and so on. The templates in the library can be used to deploy virtual machines and vApps in the vSphere inventory. Content libraries can also be used to share content between vCenter Server instances that are in the same or different locations. When deploying workloads, sharing templates and files ensures consistency, compliance, efficiency, and automation in the deployment [11].

2.2.2 NSX-T

NSX-T is a software-based consolidated networking platform that connects any type of application, whether it's running in VMs, clusters, containers, pods, or on any traditional server [12]. It can be characterized as a single abstraction of networks and can be deployed in any type of cloud, private or public. In NSX-T, a single console can be used to manage both north-south and east-west traffic between VMs, containers, and pods. As in any other network virtualization solution , with NSX-T network virtualization in VMware systems, one can create, delete, restore, or make changes to any virtual networks in the system. One of the many reasons why NSX-T is so powerful is that it can run an application and a web browser on the same ESXi hypervisor. And if one wants to route traffic from for example, a web browser to an application or vice versa in the same hypervisor, then the traffic does not need to cross all the way over to the physical router in the infrastructure. It can simply be routed through the DLR[2] directly to the application.

NSX-T components

NSX-T includes many components that are used to provide connectivity, security and availability. Some of NSX-T core components includes:

NSX Edge Nodes

NSX edge nodes have a lot in common with physical routers because they both provide routing and connectivity to external networks. HDO has deployed two edge nodes that are VMs and provide connectivity to the internet. These two NSX Edge nodes is deployed together to form an NSX-T edge cluster. This helps ensure that at least one NSX edge node is available at any given time.

[1]**vApp template:** is a virtual machine image that can be loaded when the operating system, application, and data are loaded.

[2]**DLR:** A distributed logical router(DLR) is a virtual implementation that consists of the routing control plane and distributes the data plane to each hypervisor host via kernel modules [13].

NSX Manager

The purpose of the NSX Manager is to provide users with a graphical user interface(GUI) and a REST API to manage the virtualized network infrastructure. The main functionalities of the NSX manager are to provide a REST API for creating, monitoring, and configuring logical switches, nodes, edge services, and other NSX components [14].

Segments(Logical Switches)

Logical switches are network virtualization devices that correspond to layer two networking switches in a physical network and provide east-west network connections between pods, VMs, and Kubernetes clusters in an infrastructure, as well as connecting a system to a common gateway.

Logical Routing (Tier-0 and Tier-1 Gateways)

A logical router is an NSX-T virtual networking object that connects two different host hypervisors in an infrastructure. It can be configured through the NSX manager. As a result, a logical router are created in each hypervisor [15]. Tier-0 and Tier-1 are two logical routers that will be used throughout this thesis. A Tier-0 router is the top-tier gateway and is used to connect logical segments of the external networks to the physical network through the NSX-T edges. Tier-1 is the bottom-tier gateway and is connected to the Tier-0 gateway. Tier-1 is used for layer 3 routing by adding segments to it.

Compute Manager

The Compute Manager manages resources such as ESXi hosts and VMs. NSX-T polls this compute manager to collect information about changes made to host VMs and uses this information to update its inventory accordingly [16]. The purpose of this inventory is to retrieve settings used for vCenter so that trusted communication between vCenter and the NSX-T can be established.

2.3 Kubernetes

Kubernetes, also known as K8s, is a Google-developed open-source container orchestration platform. It manages containers across different platforms, like physical machines, virtual machines, and cloud environments. Kubernetes is a platform for automatically deploying and managing containerized applications using different infrastructure types. Kubernetes does this by abstracting the underlying infrastructure used to run containers and grouping containers into logical units. This section covers the fundamentals of Kubernetes operation as well as its primary components and services.

2.3.1 Kubernetes Architecture

Kubernetes follows a client-server architecture and is used to deploy, scale, compose, and manage containers across hosts. A Kubernetes cluster is a collection of nodes that run containerized applications [17]. A node, also called a worker node, is either a VM or a physical server that has its own Linux environment. As shown in Figure 2.2, each node in a Kubernetes cluster runs multiple pods that each run one or more containers. A cluster consists of at least one or more nodes and a control plane responsible for all the managing processes in the cluster.

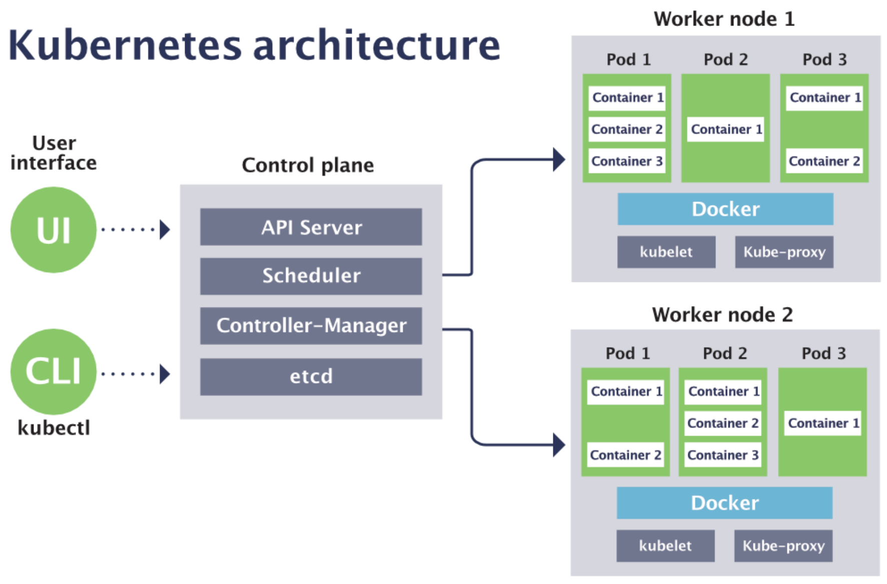


Figure 2.2: A typical Kubernetes cluster
[18]

Node

Nodes in a Kubernetes cluster are used to run containerized applications. A Kubernetes cluster can have hundreds of nodes. Three processes must be installed on every node: a container runtime, a Kubelet, and a Kube proxy. A container runtime can be Docker or another type of container runtime. Kubelet is a process of Kubernetes that interacts with the container runtime and the node itself. This process is responsible for all the communications between the nodes and the control plane. Kubelet is also responsible for starting a pod with a container inside and checking the health of the container. The third process that must be installed on every node is the Kube proxy. This type of process is responsible for forwarding the requests from services to pods.

Control plane

A control plane, also called a manager node, does all the managing processes in the cluster. The components of a control plane include an API server, etcd, a scheduler, and a control manager. Every manager node must have these four services running.

The Kubernetes API server is the front end of the Kubernetes control plane that DevOps engineers can access from outside of the cluster. The API server is a core management component of Kubernetes and is responsible for the communication between all the Kubernetes components. It also functions as a gatekeeper for authentication to ensure that only authenticated requests get through the cluster. When DevOps engineers run a Kubectl command to create, update, or delete a Kubernetes object such as a pod, the API server validates requests and then forwards the request to other processes responsible for creating or updating that object.

The Scheduler, or Kube-scheduler as it is sometimes called, is the process that assigns a node to newly created pods with no node assigned.

The control manager is the process that runs control processes such as detecting cluster state changes. When a pod crashes, the controller manager needs to detect what happened and recover the cluster's state as quickly as possible. The control manager does this by communicating with the scheduler to reschedule the crashed pods.

Etcd is another process that needs to run on every manager node. This process is responsible for storing cluster changes. This process stores information about available resources or cluster state changes, but the actual application data is not stored in etcd.

2.3.2 Why Kubernetes?

The trend from Monolithic architectures to Microservices has increased the usage of containers. Containers are commonly used in today's applications. Managing these containers across multiple environments, like virtual machines and physical servers, using scripts or other tools adds to the complexity. This has necessitated the usage of container orchestration tools like Kubernetes.

Kubernetes offers high availability, meaning that if one container fails, it automatically starts another container to make sure that there is no downtime. Scalability is another advantage of Kubernetes. Containerized applications can thus be scaled to meet changing demands while maintaining high performance. Build-in automated rollouts and rollbacks are another Kubernetes capability. With automated rollouts, applications are updated with no downtime. If applications do not work as expected after an update, an automated rollback is used to roll back that change.

2.3.3 Components

Kubernetes is composed of numerous components and services. This section describes only the components needed for deploying a working Kubernetes cluster. The following are the Kubernetes components that are frequently alluded to in this thesis:

pod

A pod is an abstraction over a container and is the smallest unit of Kubernetes [19]. The purpose of pods is to abstract away the container runtime, which means developers do not have to work directly with Docker or any other container runtime. A pod consists of one or more containers that share network and storage resources. Containers running inside a pod run a specific image. Kubernetes pods communicate with each other using an internal IP address automatically assigned by Kubernetes. Pods are created using a controller, which is a component of a Kubernetes cluster that manages the pod's lifecycle. It is recommended to create pods using a set of identical pods called "ReplicaSets" instead of creating pods directly.

Service

A service in Kubernetes is a way of exposing an interface to a set of pods using an IP address or DNS name that can be used to access the pods. It defines a policy to access a set of pods using a port number and a service type, for instance, TCP, along with the automatically assigned static IP address. There are four types of Kubernetes services: ClusterIP, NodePort, LoadBalancer, and ExternalName. ClusterIP is the default type of service used for internal access to the cluster.

NodePorts are the ports that are open on each node in the cluster. Traffic is routed to the service via a NodePort [20]. When a NodePort service is used, the service becomes accessible from outside of the cluster using the static ports that are open on each node.

A load balancer is needed if a cluster is running on cloud providers like Azure, OpenStack, or AWS. The service is accessible externally when a LoadBalancer service is used.

An ExternalName is a service type used to map a service to a local DNS name that is used to call the service.

Deployment

Deployment is an abstraction of pods used to create or modify pods that run containerized applications. This type of abstraction automates the manual processes involved in deploying, scaling, or updating containerized applications [21]. Deployments are the recommended way of creating pods and managing how applications within a cluster run. Deployments are created using the declarative programming language YAML or JSON. A typical use case for deployment is to rollout a ReplicaSet [22].

ReplicaSet

A replicaset is a component of Kubernetes that runs multiple instances of a single pod. The purpose of replicaset is to guarantee high availability and fault tolerance by running multiple instances of pods. If one pod fails or is not available, Kubernetes starts another pod to maintain high availability. Deployment is used to specify the number of pod replicas running.

Namespaces

A namespace is Kubernetes' way of organizing clusters into virtual clusters. Namespaces are used to separate and isolate Kubernetes objects such as deployments, services, and pods across nodes. It is still possible to access Kubernetes objects that are in other namespaces. Kubernetes does this by allowing these objects to share the same cluster. A typical use case for a namespace is when separating a production deployment from a testing deployment using different namespaces. Namespaces are also useful when setting permissions for DevOps engineers. An administrator can give a DevOps engineer a role that sets permissions for the user inside a given namespace.

Ingress

Ingress is a component of Kubernetes responsible for routing traffic into the cluster. This component exposes HTTP and HTTPS for handling external access to services within the cluster [23]. The purpose of ingress is to manage the access of external users to services in a Kubernetes cluster using routing rules. As shown in Figure 2.3, client requests are routed first to ingress, which then routes the request to the service, instead of routing the requests directly to the service. Ingress is also used to provide services with externally available URLs.

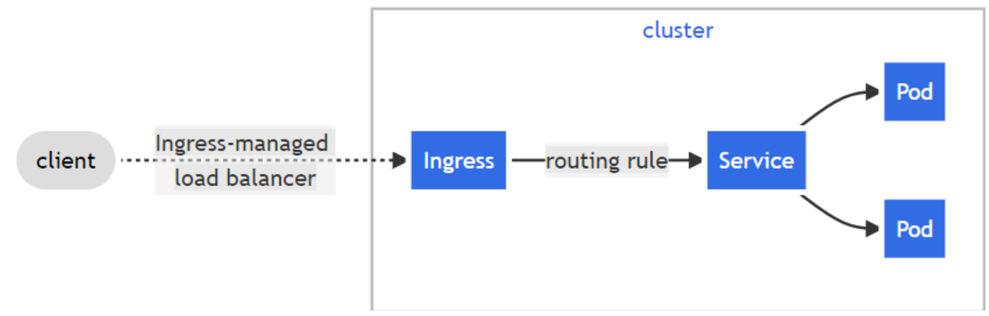


Figure 2.3: External access to services using Ingress [23]

Workloads

Workloads in Kubernetes are containerized applications running across a group of pods. In Kubernetes, "workload" is not an object or component, it is a term used for generalizing applications and services running on a cluster. Kubernetes workloads are managed and configured using other components of Kubernetes like deployments, services, and replicasets.

Persistent Volume Storage

Persistent volume storage in Kubernetes is a resource in a cluster used to store data. It uses physical storage such as hard drives, cloud storage such as Google cloud storage, or network file systems to provide DevOps engineers with storage that will always be available. Persistent volume storage is created using a YAML file by specifying storage capacity and which type of physical storage to use.

Secret

A secret is a Kubernetes object used to store sensitive data such as passwords and SSH keys. Like other Kubernetes resources, secrets are created using a YAML file and stored in the etcd. Secrets are by default stored unencrypted and can be accessed by anyone who has access to the API server. Enabling encryption for secrets in the API server is the best practice.

2.3.4 How a Kubernetes cluster works

For a Kubernetes cluster to work, a DevOps engineer must first define the cluster's desired state using YAML or JSON. This includes defining container images to be pulled and deployed across nodes, resources for applications to use, the number of replicas needed (replicaset), and the applications and workloads. DevOps engineers use kubectl, a Kubernetes command-line tool, to define the desired state. DevOps engineers interact with the control plane to set the desired state, which then communicates the desired state to the worker nodes. After defining the desired state, Kubernetes automatically manages the cluster to match the desired state. Kubernetes clusters can be created in a lot of different ways. The purpose of this thesis is to use Tanzu to deploy Kubernetes clusters.

2.4 Tanzu

Tanzu is a VMware solution that offers a full set of functionalities for building modern applications and simplifies running a standard Kubernetes platform across multiple clouds [24]. Tanzu provides developers with the capabilities to run and manage multiple Kubernetes clusters across both public and private clouds. Tanzu also enables the fast and secure development and deployment of applications, ensuring that better applications are delivered to production on a continuous basis.

2.4.1 VMware Tanzu Editions

Tanzu Basic Edition, Tanzu Advanced Edition, Tanzu Standard Edition, and Tanzu Community Edition are the four VMware Tanzu editions [24].
In terms of pricing, these four editions differ, and one should select a Tanzu edition based on the type of containers to be run, such as prepackaged and customized containers. Tanzu Basic Edition is the type of Tanzu edition Kubernetes clusters that will be used throughout this thesis. It's a full-featured platform that enables Kubernetes integration with vSphere.

2.4.2 What is vSphere with Tanzu?

VMware offers vSphere with Tanzu solution, which is utilized to improve procedure transparency between IT operations and developers. After Tanzu has been installed and configured, vSphere becomes a Kubernetes platform for running workloads on the hypervisor layer. This means that Kubernetes workloads can run directly on ESXi hosts, which improves workload performance [25].

2.4.3 What does vSphere with Tanzu offer?

Today's application stack is typically made up of various microservices, which each run on several Kubernetes pods. This stack is comprised of three layers. The first layer is the underlying virtual environment. The second layer is the Kubernetes infrastructure, which is deployed within VMs provided by the first layer. The third layer consists of Kubernetes pods that run within the workers provided by the second layer. [25]. Figure 2.4 illustrates these three layers.

Working on today's application stack presents numerous challenges. One of these challenges is that this type of stack requires the use of different roles to operate and manage. As a result, roles such as application developers, DevOps engineers, and vSphere administrators are needed to manage this stack as shown in Figure 2.4.

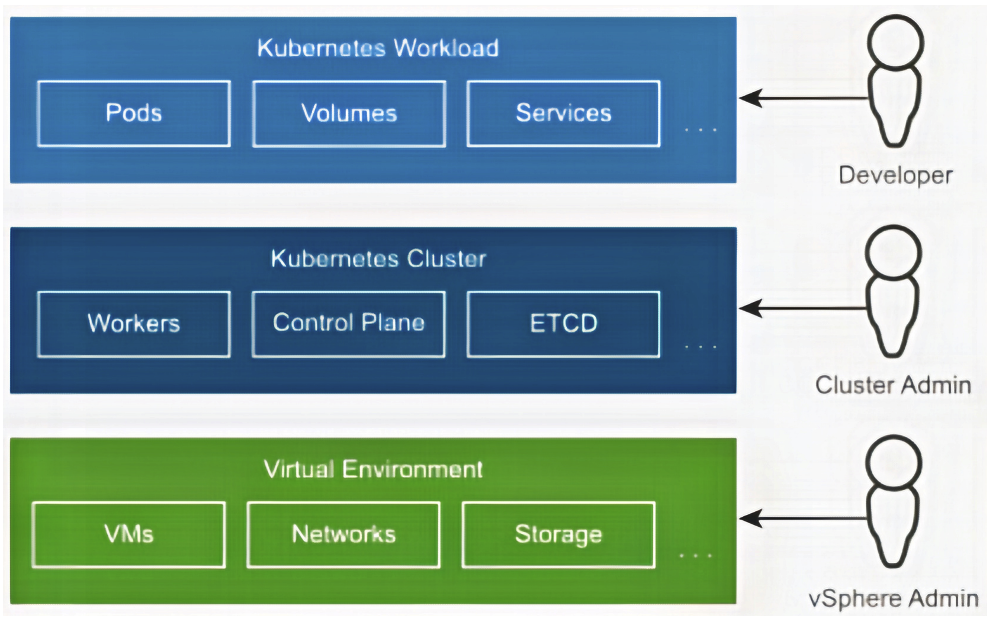


Figure 2.4: Today's Application Stack
[25]

Full-stack operations might be problematic since they necessitate communication between all three roles. While application developers can run and deploy Kubernetes pods and applications, they have no visibility or control over the underlying Kubernetes Cluster that these pods and applications are running in. The DevOps engineer team faces the same difficulty because they can control the Kubernetes infrastructure layer but lack visibility into the virtual environment. As a result, they are unable to address or resolve any resource-related problems. Similarly, vSphere administrators have comprehensive visibility and control over their responsibilities within the underlying virtual environment, but they have no control or visibility over the second or third layers, which run Kubernetes pods and applications.
vSphere with Tanzu offers a solution to these challenges. This solution, shown in Figure 2.5, integrates the different layers of the previous stack by enabling vSphere with Tanzu to create a Kubernetes control plane directly on the hypervisor layer. As a result, the vSphere cluster supports the creation of a Kubernetes layer within the ESXi hosts.

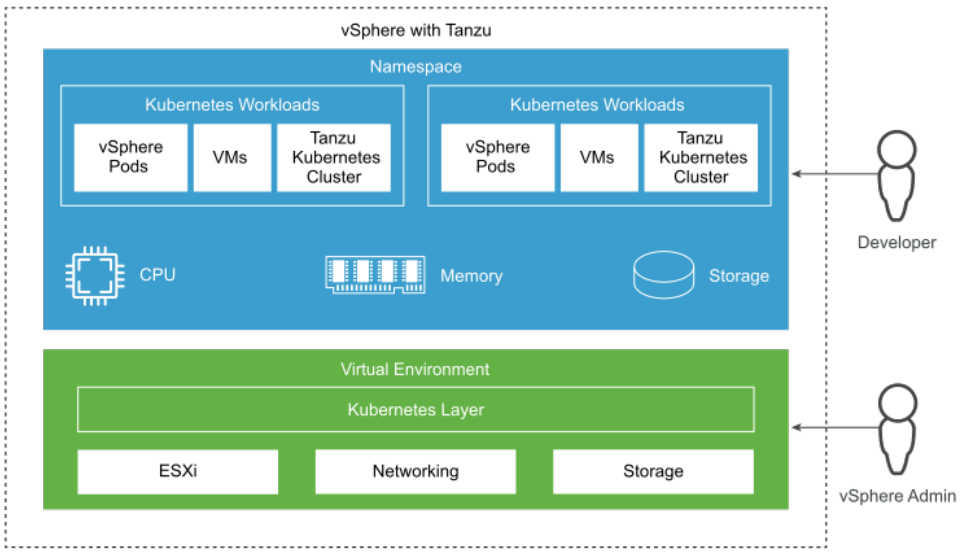


Figure 2.5: vSphere with Tanzu
[25]

The presence of a Kubernetes control plane running on the hypervisor layer provides vSphere with many capabilities. This involves making collaboration and communication between vSphere administrators and DevOps engineers much easier.
The vSphere administrator team can build and configure vSphere Namespaces with a given quantity of resources (CPU, RAM, and Storage). These vSphere namespaces will afterwards be delivered to the DevOps engineer team to deploy multiple Kubernetes containers on these vSphere namespaces. The DevOps engineer team may also deploy and control Tanzu Kubernetes clusters inside a vSphere namespace by using the Tanzu Kubernetes services.
The vSphere administrator team will be provided with a vSphere Client interface to monitor and manage vSphere Pods and Tanzu Kubernetes clusters. As a consequence of that, the vSphere administrator team has thorough visibility over vSphere Pods and Tanzu Kubernetes clusters operating under diverse namespaces.

2.4.4 vSphere with Tanzu Architecture and Components

vSphere with Tanzu is made up of several different components and services. The vSphere with Tanzu components that are frequently mentioned in this thesis will be described in this section.

Supervisor Cluster

The Supervisor Cluster is the cluster that has been enabled for vSphere with Tanzu [26]. This cluster runs on top of an SDDC (software-defined data center) layer that provides networking, computing, and shared storage management. The Supervisor Cluster is the foundation of the vSphere with Tanzu supplying the components and resources for running workloads such as VMs, Tanzu Kubernetes clusters, and vSphere Pods.

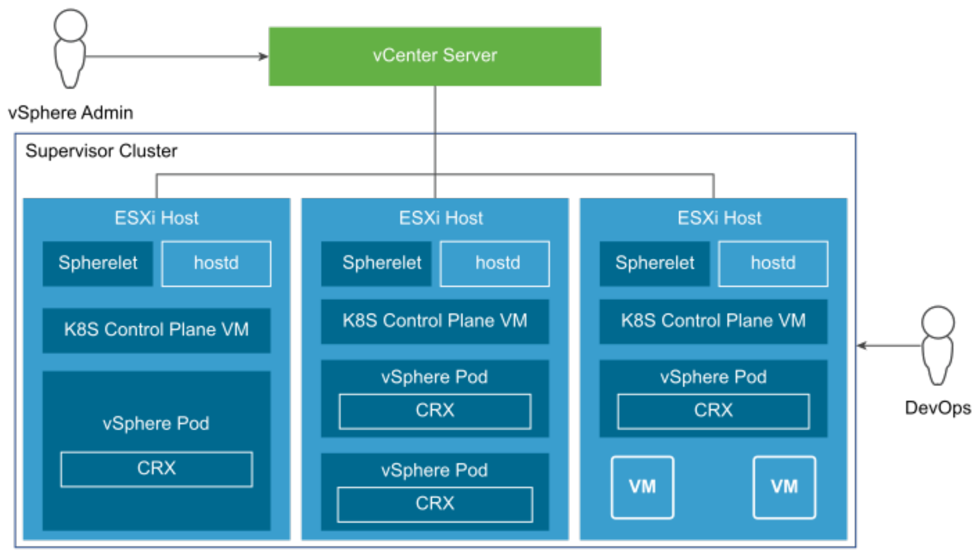


Figure 2.6: Supervisor Cluster Architecture
[26]

The Supervisor Cluster Architecture, as shown in Figure 2.6, includes the following processes and services:

- **Kubernetes control plane VM**
  During the Workload Management setup procedure, three VMs will be deployed within the Supervisor Cluster [26]. These three VMs make up the Control Plane of the vSphere with Tanzu. Each one of these VMs has its own IP address, enabling them to be load balanced. One of them will also be allocated a floating IP address to serve as the leader in the Supervisor Cluster. If the Control Plane leader fails, this floating IP address will be assigned to another Control Plane node to maintain communication between the Supervisor Cluster and the vSphere center..

- **vSphere Pod**
  A vSphere Pod is similar to a Kubernetes pod and allows for running containers natively on ESXi hosts. The vSphere Pod is a small-footprint virtual machine that is optimized to run one or more Linux containers. It is in charge of allocating the resources (storage, memory, and CPU resources) that are needed to run the workloads. The vSphere Pod enables workload features such as strong isolation, resource management, high performance, and diagnostics [27].

- **Spherelet**
  The Spherelet works as a Kubernetes kubelet, with features that enable each ESXi host to serve as a kubernetes worker node [28]. This Spherelet component is created on each ESXi host and can run directly on them, allowing Spherelet to easily interface with ESXi.

- **Container Runtime Executive (CRX)**
  CRX is a vSphere specialized VM that features an optimized Linux kernel. This optimized Linux kernel is optimized to operate with hypervisors seamlessly. The CRX does not load a complete Linux guest OS; instead, it loads a super-thin, optimized kernel and a special type of main init process known as CRX Init. This CRX Init enables the Linux guest to initiate the main init process without requiring to go through kernel initialization. This is accomplished by employing a direct boot technique that bypasses the BIOS. As a consequence, installing and booting a new vSphere Pod is as fast as traditional containers [28].

vSphere Namespaces

vSphere namespaces are namespaces that defined by the vSphere administrator in the Supervisor Cluster, each with its own resources, specifications, permission, and limitations for the developer or user. They can be set up with either the vSphere networking stack or with NSX-T. The network configuration and capabilities of each setup differs. Namespaces created on Supervisor Clusters configured with

NSX-T support the full set of Workload Management platform capabilities i.e. it supports vSphere Pods, VMs, and Tanzu Kubernetes clusters, etc. vSphere network stack, on the other hand, only supports Tanzu Kubernetes clusters and VMs, not vSphere Pods [29].

Tanzu Kubernetes Clusters

Tanzu Kubernetes cluster is a VMware product which is provided as an open source for container orchestration platform and handles application workloads across compatible platforms. These application workloads are managed by the management cluster. It is also provisioned by Tanzu Kubernetes Grid Service on the Supervisor Cluster. Depending on the requirements of the application workload they run, they are compatible with various versions of Kubernetes [30].

Tanzu Kubernetes Grid Service

Tanzu Kubernetes Grid Service (TKGS) is a VMware product module that provisions Tanzu Kubernetes clusters in a self-service manner. It also allows the vSphere administrator to create, administer, and manage Tanzu Kubernetes clusters in a declarative approach. A declarative approach requires the user to describe what they want to make or do. For example, using a YAML file. The Tanzu Kubernetes Grid Service enables the deployment of Tanzu Kubernetes clusters in software-defined Data Centers as well as the most widely available private and public cloud environments [31].

2.4.5 Why Tanzu with NSX-T?

To connect with the different segments in a VMware system, like supervisor cluster, namespace, and all its components such as vSphere pod, Tanzu Kubernetes cluster, VMs, etc. vSphere Tanzu depends upon the connectivity and network configuration between all these components. This network configuration can be installed and set up with the help of NSX-T. Therefore, NSX-T is an important tool or component in the vSphere with Tanzu VMware ecosystem [32].

Chapter 3

Install and Configure Tanzu

After reading chapter 2, the reader should have a basic understanding of the various technologies used in this thesis. This chapter explains how the project group configured vSphere with Tanzu on NSX-T during the project period. Section 3.1 gives an overall description of the vSphere and NSX-T environments set up by HDO. This chapter also covers the setup of each prerequisite component that must be installed and configured before installing vSphere with Tanzu on NSX-T. Several figures captured during the installation and configuration process are used to provide a better understanding of the process.

3.1 System architecture

The environment used to install and configure vSphere with Tanzu was set up by HDO. The environment consists of three VMware Horizon clients used for accessing the vSphere environment client and the NSX-T environment, as shown in Figure 3.1. For this thesis, HDO has set up a vSphere environment with three ESXi hosts. It is important to note that the environment HDO has set up is only accessible through the horizon clients and is not accessible from the outside world. HDO has also provided the project with two datastores that each has a capacity of 16 terabytes. These datastores will be used as storage for the different components that will be created during the installation and configuration process of vSphere with Tanzu.

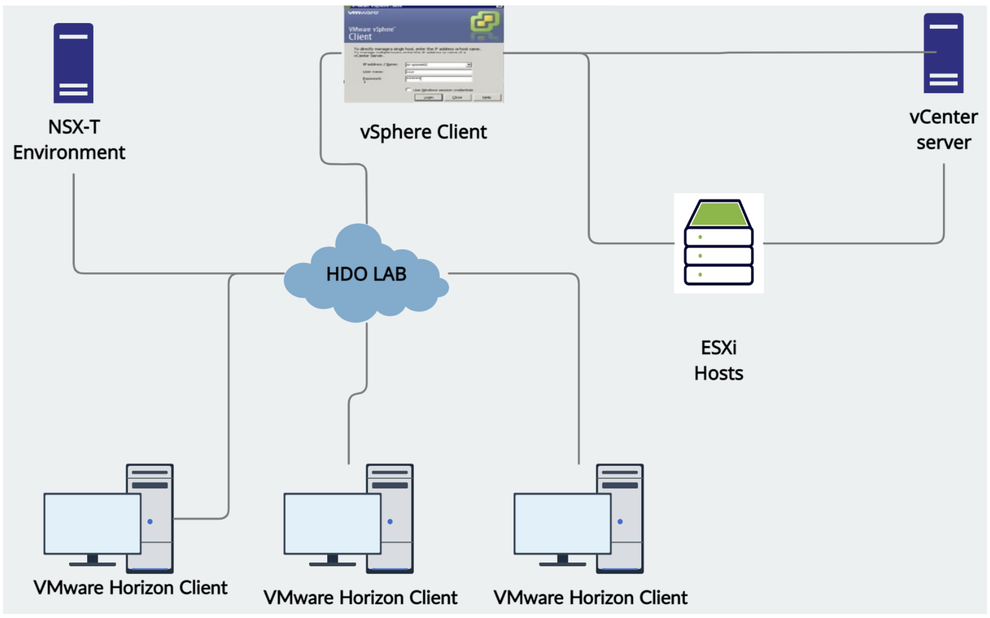


Figure 3.1: Setup with VWware horizon client, vSphere, vCenter and NSX-T

3.2 NSX Advanced Load Balancer

As mentioned in Section 3.1, the vSphere environment provided by HDO includes a vSphere cluster with three hosts that will be configured as a Supervisor cluster by the group. The vSphere cluster that will be configured as a Supervisor Cluster throughout this thesis uses the NSX-T as a network solution. An NSX advanced load balancer (ALB) is required to use the NSX-T for vSphere with Tanzu.

The NSX-T advanced load balancer, also called the Avi controller, provides virtual IP addresses for the Supervisor Control Plane, the Tanzu Kubernetes cluster, and applications that are deployed using the Tanzu Kubernetes cluster and require a load balancer service. The Avi controller and the service engine group are the two main components of ALB. The Avi controller is a VM that provides networking management center for cloud infrastructures such as VMware. As shown in Figure 3.2, Avi controller is configured to communicate with the vCenter and NSX-T Manager to manage resources in the VMware infrastructure. Service engines are the components of ALB that execute instructions from the Avi controller. Service engines run on VMs and are automatically created by default after the Avi controller has been deployed. The following subsections discuss how the project group has implemented the advanced load balancer.

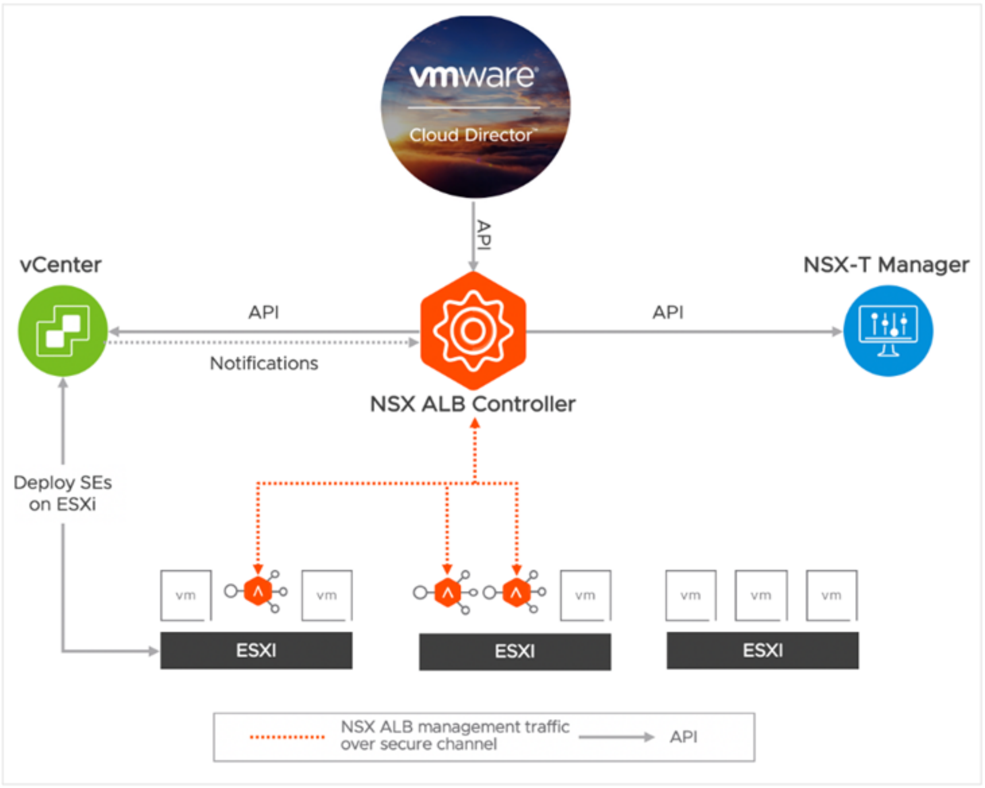


Figure 3.2: Avi controller communicates with vCenter and NSX-T [33]

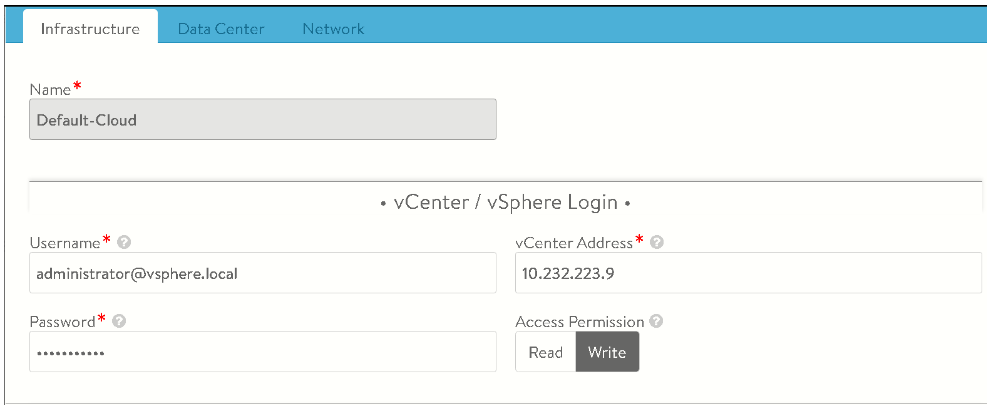
3.2.1 Deploying and configuring Avi Controller

The Avi controller was deployed in the vSphere environment using an open virtualization appliance (OVA) downloaded from the VMware customer connect portal[1] website. An OVA is a template that includes pre-installed software and is used to deploy virtual machines. After deploying the controller, it can be accessed using an IP address automatically assigned to it. Avi controller needs to be configured for Tanzu. The first step of the configuration process includes creating an Administrator account, setting a password, and other system settings such as DNS Resolver and DNS Search domain.

The Avi controller needs to connect to the vCenter. This is done by configuring the infrastructure settings of the Avi controller by selecting VMware as the infrastructure type and filling in the vCenter login credentials used to access the vSphere environment, as shown in Figure 3.3. After connecting the Avi controller with the vCenter, the controller automatically fetches all the available data centers associated with the vCenter. HDO has provided the project group with only one

[1]VMware Customer Connect: portal for VMware products, documentation, customer support and learning about VMware products [34].

data center that will be used throughout this thesis. It is only possible to select one data center, and the data center selected is where the workload management will be enabled. Enabling workload management means configuring vSphere for Tanzu, and how it is enabled will be discussed in the upcoming sections.



Name* Default-Cloud

• vCenter / vSphere Login •

Username* administrator@vsphere.local

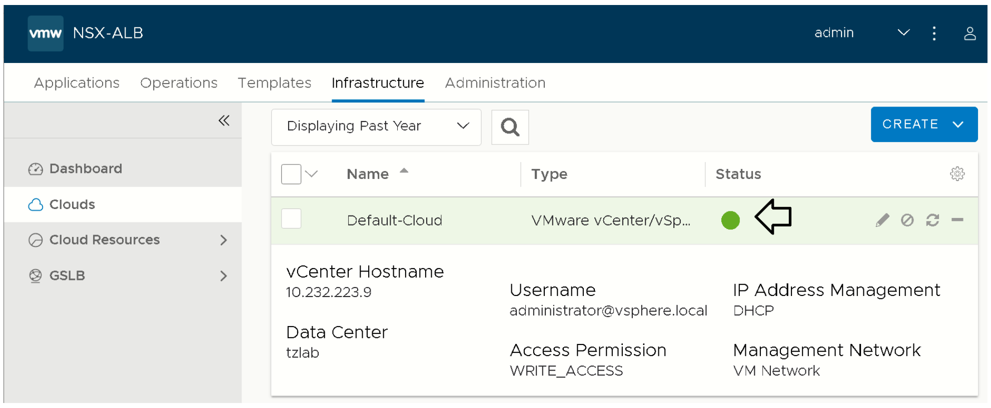
vCenter Address* 10.232.223.9

Password* •••••••••••

Access Permission Read [✓ Write]

Figure 3.3: Connecting Avi controller with vCenter

After selecting the infrastructure type and successfully connecting the controller to the vCenter, the Management Network needs to be configured. This management network has an interface that is used by service engines to connect with the controller and is connected to port groups that were created when setting up the NSX-T. When configuring the network settings for the Avi controller, it is optional to use DHCP with the Management Network. The project group has chosen to select "DHCP Enabled". The reason for this is that the group has deployed DHCP Server for use on the vSphere port groups. After the infrastructure, Data Center and network settings for the Avi controller are configured, it will reconnect again to the vCenter and pool more data as needed. After this has been done, the "Status" of the Avi controller should be green, as shown in Figure 3.4.



Name	Type	Status
Default-Cloud	VMware vCenter/vSp...	●

vCenter Hostname 10.232.223.9

Data Center tzlab

Username administrator@vsphere.local

Access Permission WRITE_ACCESS

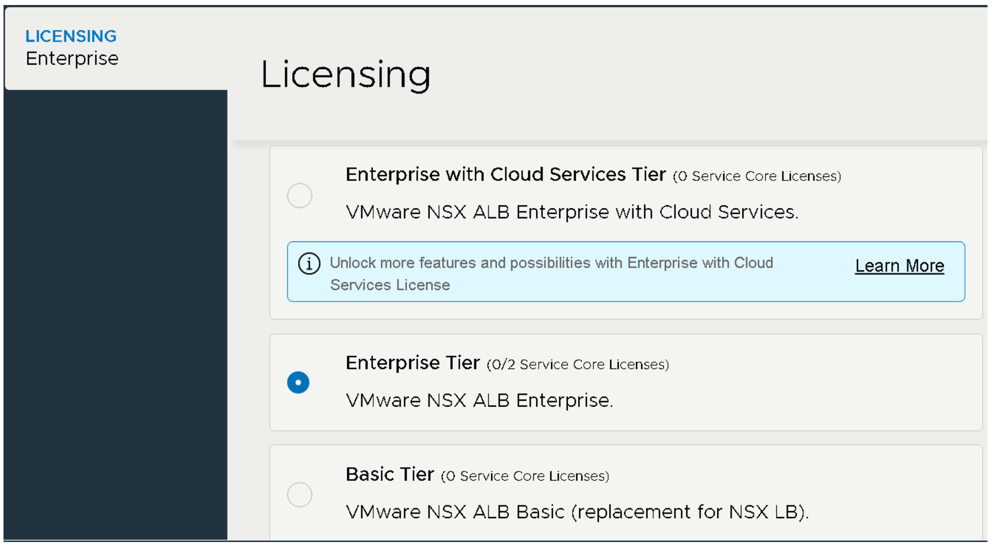
IP Address Management DHCP

Management Network VM Network

Figure 3.4: Status of the Avi controller

3.2.2 Add license to the NSX Advanced Load Balancer

A license needs to be added to the Advanced Load Balancer previously deployed because the Avi controller deployed is in evaluation mode, and it is required to have an Enterprise license to be able to use the Avi controller after the evaluation period has expired. Figure 3.5 shows Enterprise licence added for Avi.


LICENSING
Enterprise

Licensing

Enterprise with Cloud Services Tier (0 Service Core Licenses)
VMware NSX ALB Enterprise with Cloud Services.
Unlock more features and possibilities with Enterprise with Cloud Services License
Learn More

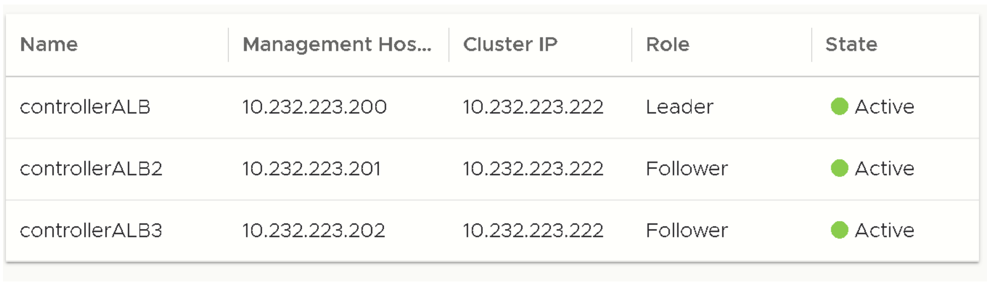
[✓ Enterprise Tier (0/2 Service Core Licenses)
VMware NSX ALB Enterprise.]

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

Figure 3.5: License for Avi

3.2.3 Deploy Avi Controller Cluster

The project group has chosen to deploy a cluster of three controller nodes, as shown in Figure 3.6. Deploying a controller cluster ensures redundancy and high availability. To form a controller cluster of three Avi controllers, two more Avi controller VMs need to be deployed and powered on. The implementation of the two Avi controller VMs is similar to that of the first Avi controller. This includes the same administrator password for the three Avi controllers. The two Avi Controller VMs are deployed using the same OVA template file used to deploy the first controller. After the controller VMs have been deployed and configured, a cluster of three advanced load balancers is created. This is done by adding each Avi controller VM as a cluster node to form a cluster. Figure 3.6 shows "controllerALB", which is the first Avi controller deployed and becomes the leader of the cluster. It synchronizes its configuration with the two Avi controllers: "controllerALB2" and "controllerALB3" that become followers.



Name	Management Hos...	Cluster IP	Role	State
controllerALB	10.232.223.200	10.232.223.222	Leader	● Active
controllerALB2	10.232.223.201	10.232.223.222	Follower	● Active
controllerALB3	10.232.223.202	10.232.223.222	Follower	● Active

Figure 3.6: Avi controller cluster

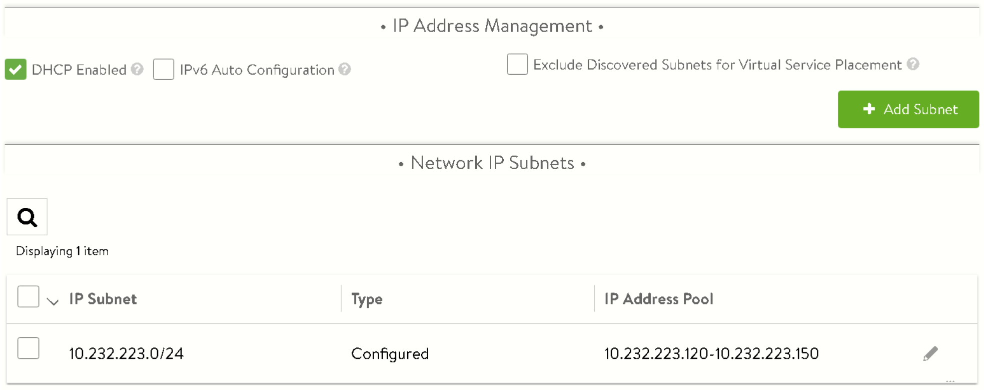
3.2.4 Add a Certificate to the Controller

The controller deployed requires a valid certificate that matches its IP address and the Avi controller IP address. The purpose of adding Certificate is to secure the communication between the controller and its clients. The certificate selected is a self-signed certificate that was created by the project group. Using a self-signed certificate means creating your own certificate instead of using other trusted certificates. It is also possible to use a default self-signed certificate automatically created when the controller was deployed.

3.2.5 Configure Service Engine Group

A service engine is a distributed load balancer that is created within a group and shares resources [35]. The advanced load balancer requires at least one service engine group that is associated with the infrastructure type selected. The group has chosen to use the default service engine group set-up that was created when deploying the first Avi controller.

An IP subnet with an IP address pool needs to be configured for the service engines and the services that are going to use the advanced load balancer. Each application service that requires a load balancer service will automatically be assigned an IP address to use from this IP address pool. As shown in Figure 3.7, the project group has configured a subnet with an IP address range within the network 10.232.223.0/24, which is a network address provided by HDO for this thesis. Figure 3.7 shows the IP subnet with IP addresses that range from 10.232.223.120 to 10.232.223.150. This subnet will be used as a virtual IP network for the service engines and services that use the load balancer service.



• IP Address Management •

✓ DHCP Enabled ☐ IPv6 Auto Configuration ☐ Exclude Discovered Subnets for Virtual Service Placement

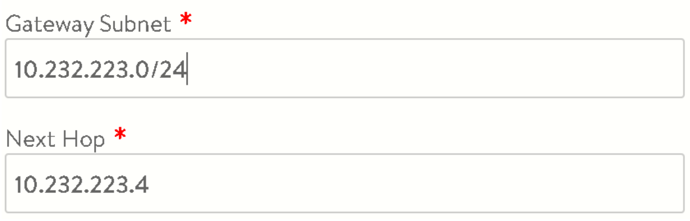
• Network IP Subnets •

Displaying 1 item

IP Subnet	Type	IP Address Pool
10.232.223.0/24	Configured	10.232.223.120-10.232.223.150

Figure 3.7: Virtual IP network configuration for the service engines

A default gateway is required after configuring the virtual IP for the service engine. This is because virtual IP addresses are only available within the service group and need to connect to the physical network. To achieve this, the project group has added a static route that connects the virtual IP network with the actual IP network, as shown in Figure 3.8 .



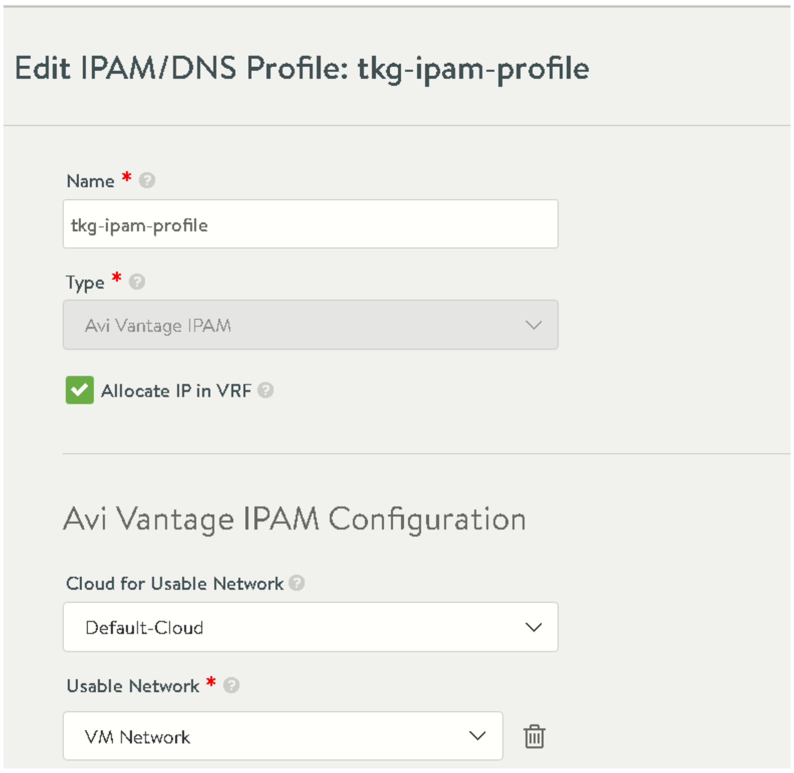
Gateway Subnet *
10.232.223.0/24

Next Hop *
10.232.223.4

Figure 3.8: Gateway for the 10.232.223.0/24 network

3.2.6 Configure IPAM profile

IP Address Management (IPAM) is needed to tell service engines which IP pool to use. The IPAM profile administers IP addresses by tracking, and managing IP addresses that are in use on the provided network. IPAM does this by working closely with the DNS and DHCP services that are responsible for resolving and assigning IP addresses. An IPAM profile is used when the vCenter is provisioning its resources. Figure 3.9 shows an IPAM profile created and connected to the vCenter environment


Edit IPAM/DNS Profile: tkg-ipam-profile

Name *
tkg-ipam-profile

Type *
Avi Vantage IPAM

[✓] Allocate IP in VRF

Avi Vantage IPAM Configuration

Cloud for Usable Network
Default-Cloud

Usable Network *
VM Network

Figure 3.9: IPAM profile created

3.3 Workload management and its Configuration

Workload management in a vSphere cluster is the deployment and operational environment for different computed infrastructures like network, storage devices, memory, CPU, or other resources. These resources are utilized by vSphere Pods and Tanzu Kubernetes clusters in namespace.

After enabling the workload management, three Supervisor Clusters are automatically created in the vSphere cluster. As it has been discussed in Section 2.4.4, enabling workload management can be done in two ways. The first option is to use NSX-T as the networking solution, and the second option is to use the vSphere networking stack. Since this thesis includes Tanzu configuration with NSX-T and will have both vSphere pods and Tanzu Kubernetes clusters, it will have an NSX-T as its networking solution. The configurations will be done using the graphical user interface (GUI) of vSphere client version 7.0.3.00300.

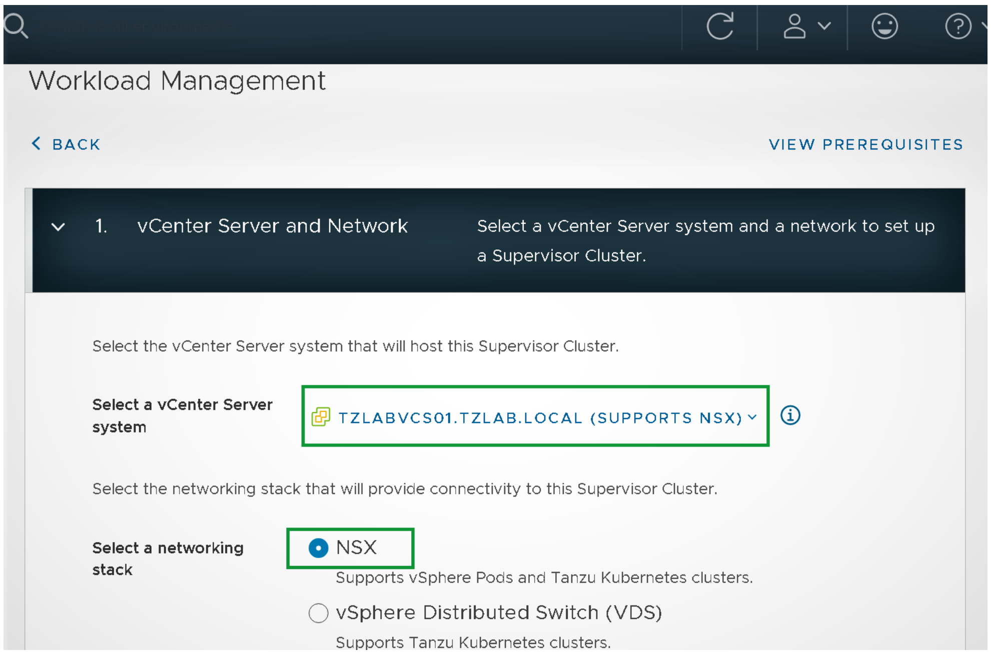
3.3.1 Enabling Workload Management

Enabling workload management is done through the workload management interface. This workload management interface gives administrators different approaches and detailed information about components and requirements that they need. The following are the main steps to enabling the workload management:

vCenter Server and Networking stack

The vCenter Server and Network pop-up window is a place where administrators specify which vCenter Server and network will be used to enable workload management. As shown in Figure 3.10:

- The only available vCenter Server in the system in which the workload management will be enabled is tzlabvcs01.tzlab.local.
- Setting up network with NSX solution



Select a vCenter Server system and a network to set up a Supervisor Cluster.

Select the vCenter Server system that will host this Supervisor Cluster.

Select a vCenter Server system: TZLABVCS01.TZLAB.LOCAL (SUPPORTS NSX)

Select the networking stack that will provide connectivity to this Supervisor Cluster.

Select a networking stack:
- [✓ NSX] Supports vSphere Pods and Tanzu Kubernetes clusters.
- vSphere Distributed Switch (VDS) Supports Tanzu Kubernetes clusters.

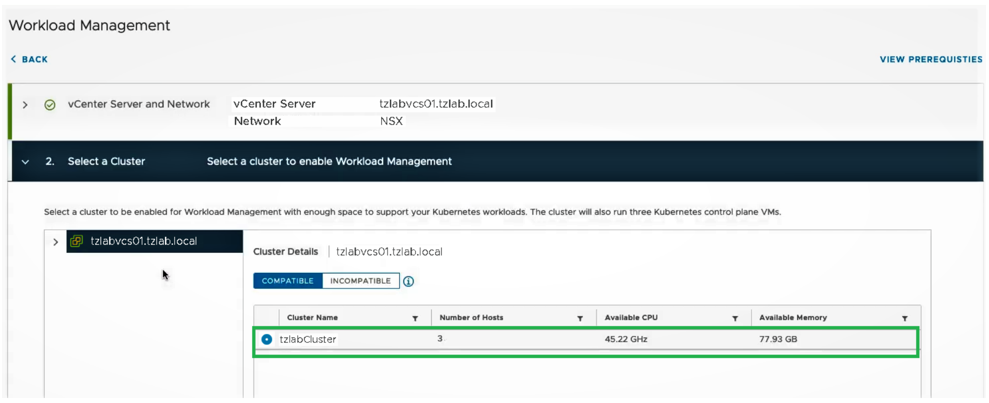
Figure 3.10: vCenter Server and Networking

Note that: If one chooses to use a "vSphere distributed switch" then the configuration will be a bit different from this setup. One needs to configure an Advanced Load Balancer with HAProxy or Avi. Then, that will make the cluster only applicable to the Tanzu Kubernetes cluster but not the vSphere Pods.

Compatible vSphere clusters for Supervisor Cluster

The compatible vSphere clusters that are available in the vCenter server are displayed in Figure 3.11.

- "tzlabCluster" is the only vSphere cluster that is configured by HDO, and is used to set up as the supervisor cluster.


Workload Management
BACK
VIEW PREREQUISTIES

vCenter Server	tzlabvcs01.tzlab.local
Network	NSX

vCenter Server and Network
2. Select a Cluster
Select a cluster to enable Workload Management

Select a cluster to be enabled for Workload Management with enough space to support your Kubernetes workloads. The cluster will also run three Kubernetes control plane VMs.

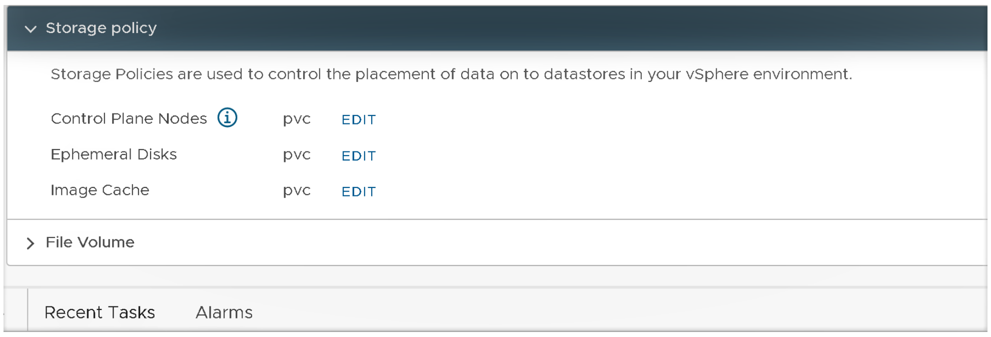
tzlabvcs01.tzlab.local
Cluster Details | tzlabvcs01.tzlab.local
COMPATIBLE
INCOMPATIBLE

Cluster Name	Number of Hosts	Available CPU	Available Memory
tzlabCluster	3	45.22 GHz	77.93 GB

Figure 3.11: vSphere cluster wizard

Storage policy

Since there are two storages available in this infrastructure, it is important to have a policy specified for the correct placement of the Supervisor Cluster and other objects. This storage policy will make sure that everything goes to the corresponding storage device associated with the policy. The group has created a storage policy named "pvc", as shown in Figure 3.12.



Storage Policies are used to control the placement of data on to datastores in your vSphere environment.

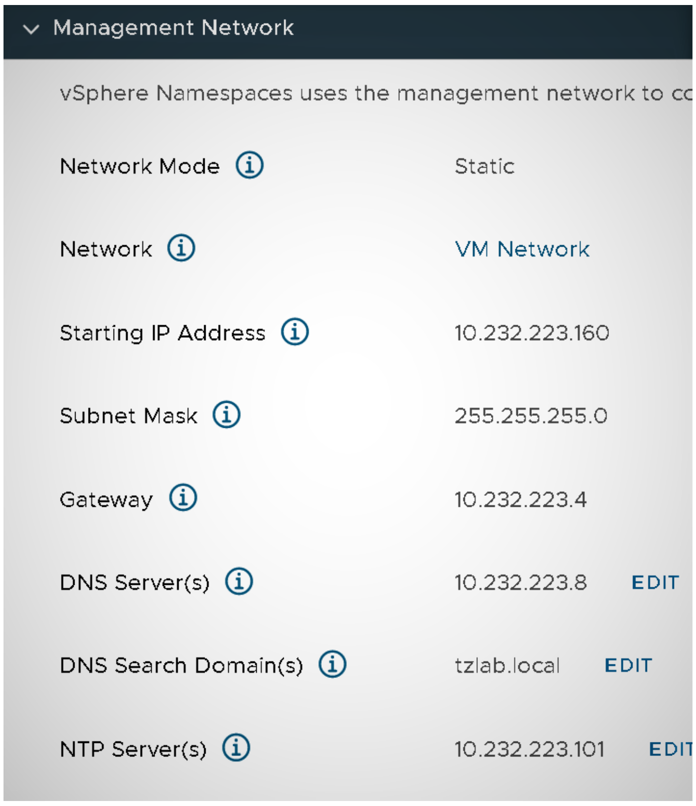
Control Plane Nodes	pvc	EDIT
Ephemeral Disks	pvc	EDIT
Image Cache	pvc	EDIT

Figure 3.12: Storage policies for the different objects on the Supervisor Cluster

Management network

Network information for the Supervisor Cluster is specified using the management network window, as shown in Figure 3.13.

- The Network Mode: The control plane VMs need five IP addresses, three for the Supervisor Cluster, one floating and one reserved for upgrading. Therefore, network mode has been set to "static".
- Network: In this case VM network, it is the management network in which it was created by the contractor during the configuration of the environment.
- Starting IP address: Five IP addresses, including three for the Control plane VMs, one floating and one reserved for upgrading, are allocated starting from this IP address: 10.232.223.160.
- Subnet mask – the subnet mask of the management network is 255.255.255.0.
- The gateway of the vCenter server is 10.232.223.4
- DNS search Domain(s) – the domain of the lab "tzlab.local" which is configured by the contractor.
- NTP Server(s) of the system: The NTP server, which was created by the group, has been used to synchronize time between the ESXi hosts for optimal functionality.


Management Network

vSphere Namespaces uses the management network to cc

Network Mode	Static	
Network	VM Network	
Starting IP Address	10.232.223.160	
Subnet Mask	255.255.255.0	
Gateway	10.232.223.4	
DNS Server(s)	10.232.223.8	EDIT
DNS Search Domain(s)	tzlab.local	EDIT
NTP Server(s)	10.232.223.101	EDIT

Figure 3.13: Supervisor Cluster management network configuration

Workload Network

The Workload Network is where the administrator configures networking to support traffic to the Kubernetes API, vSphere Pods, workloads and services, and egress and ingress CIDRs[2].

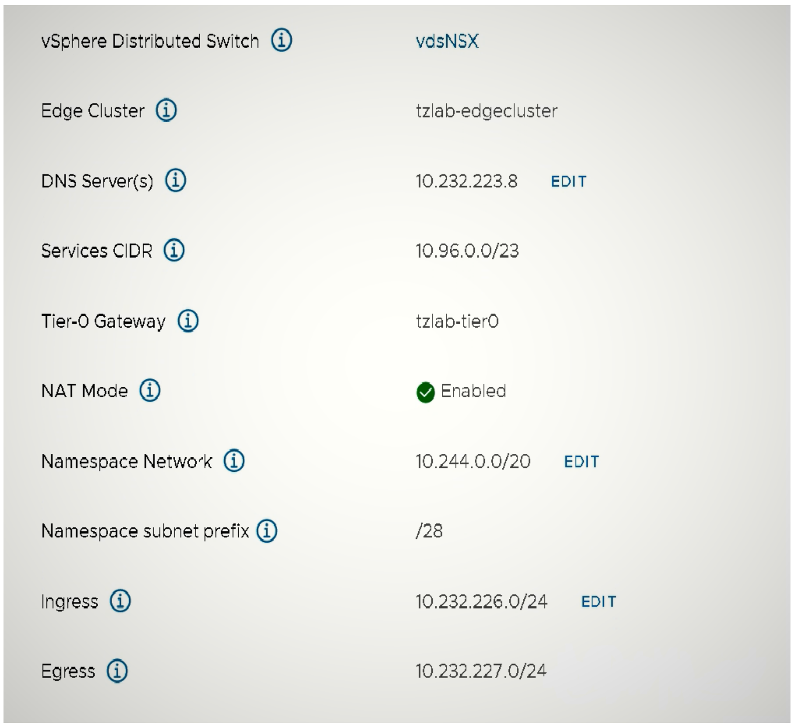
- A vSphere Distributed Switch allows for centralized management and monitoring of all VMs, pods and Kubernetes clusters connected to the switch's networking configuration. There is a VDS (vdsNSX) that has been configured by HDO.
- Edge cluster enables north-south routing[3] and network service in a workload. There is an edge cluster in NSX-T configured by contractor. The edge cluster "tzlab-cluster" is used.
- DNS Server - The IP address of the DNS server is set to be 10.232.223.8.
- Tier-0 gateway – is NSX-T logical router that gives north-south network for

[2]**CIDRs:** stands for Classless Inter-Domain Routing and it is an IP address arrangement that helps in improving IP address allocation in a system. It has two components for example, in the following network address 10.232.233.0/24. These components are prefix IP address and the network suffix.

[3]**north-south routing** is a routing traffic between two or more data center or traffic coming in and going out of the domain[36]

NSX-T logical networks to the physical network device in the infrastructure. Tier-0 gateway is configured by contractor and it is used in workload networking.

- NAT(network address translation) is enabled. It is used to access external network.
- Namespace Subnet prefix - /28 (255.255.255.240) is used.
- Namespace network – 10.244.0.0/20
- Service CIDR – IP address for the service that can be assigned to namespace to expose application in pods and Tanzu Kubernetes cluster i.e. 10.93.0.0/23
- Ingress CIDR – IP address for incoming service to the namespace i.e. 10.232.226.0/24
- Egress CIDR – IP address used for source netting from namespace to the external i.e. 10.232.227.0/24



vSphere Distributed Switch	vdsNSX	
Edge Cluster	tzlab-edgecluster	
DNS Server(s)	10.232.223.8	EDIT
Services CIDR	10.96.0.0/23	
Tier-0 Gateway	tzlab-tier0	
NAT Mode	✓ Enabled	
Namespace Network	10.244.0.0/20	EDIT
Namespace subnet prefix	/28	
Ingress	10.232.226.0/24	EDIT
Egress	10.232.227.0/24	

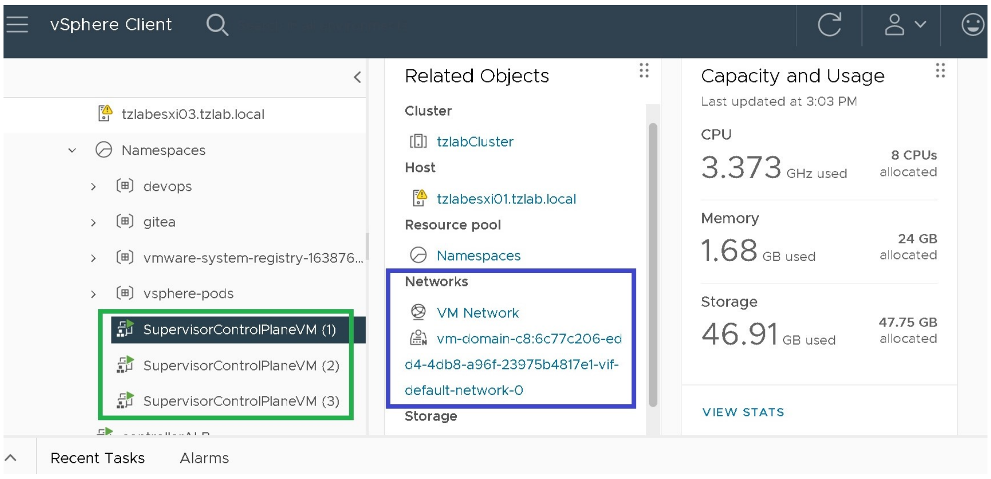
Figure 3.14: workload management network configuration

Tanzu Kubernetes Grid Service

Setting up the Tanzu Kubernetes Grid Service helps to enable self-service of Tanzu Kubernetes Service for the developers. To enable that, the content library subscription “tanzucontent” is added to Tanzu Kubernetes Grid Service to enable the developers to use an OVF templates when they deploy Tanzu Kubernetes cluster in the workload.

Review and Confirm

Finally, reviewed and confirmed the settings that has been set in and started the process of building the Supervisor Cluster. A cluster of three Supervisor Control planes is created as shown in Figure 3.15. This Supervisor Cluster is connected to the management network (VM Network).


vSphere Client
tzlabesxi03.tzlab.local
Namespaces
devops
gitea
vmware-system-registry-163876...
vsphere-pods
SupervisorControlPlaneVM (1)
SupervisorControlPlaneVM (2)
SupervisorControlPlaneVM (3)
Recent Tasks
Alarms
VIEW STATS

Related Objects

Cluster
tzlabCluster
Host
tzlabesxi01.tzlab.local
Resource pool
Namespaces
Networks
VM Network
vm-domain-c8:6c77c206-edd4-4db8-a96f-23975b4817e1-vif-default-network-0
Storage

Capacity and Usage
Last updated at 3:03 PM

CPU
3.373 GHz used 8 CPUs allocated

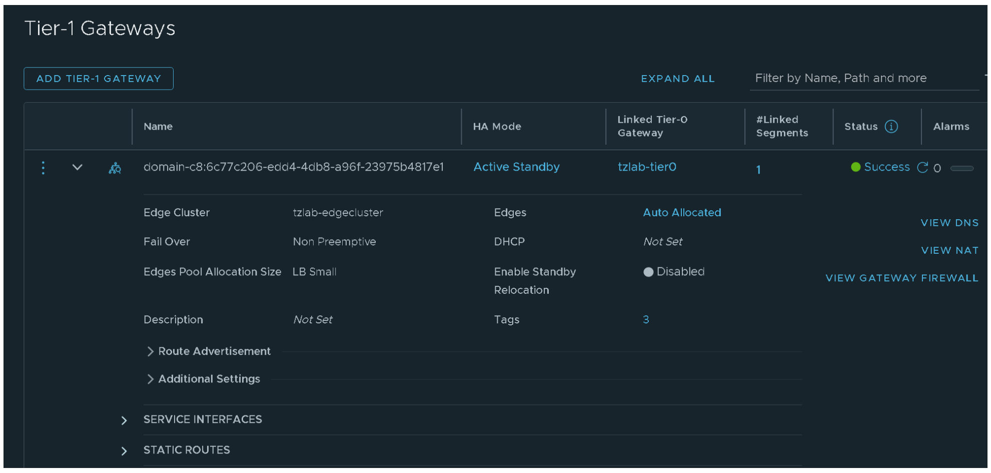
Memory
1.68 GB used 24 GB allocated

Storage
46.91 GB used 47.75 GB allocated

Figure 3.15: Successful creation of the Supervisor Control planes and logical switch

There is also another switch that automatically created in the set up in the “vd-sNSX” vSphere Distributed Switch stack. That virtual switch is labeled with a name “*vm-domain-c8:6c77c206-edd4-4db8-a96f-23975b4817e1-vif-default-network-0*” and is a Teir-1 gateway. It will be reserved for the Supervisor Cluster and all that related with it in connection with NSX-T networking.

This tier-1 gateway will run only one segment that links with the three Supervisor Control planes and as shown in Figure 3.16 that segment is also created during the setup of the Supervisor Cluster.


Tier-1 Gateways

ADD TIER-1 GATEWAY   EXPAND ALL   Filter by Name, Path and more

Name	HA Mode	Linked Tier-0 Gateway	#Linked Segments	Status	Alarms
domain-c8:6c77c206-edd4-4db8-a96f-23975b4817e1	Active Standby	tzlab-tier0	1	Success	0

Edge Cluster	tzlab-edgecluster	Edges	Auto Allocated
Fail Over	Non Preemptive	DHCP	*Not Set*
Edges Pool Allocation Size	LB Small	Enable Standby Relocation	Disabled
Description	*Not Set*	Tags	3

VIEW DNS
VIEW NAT
VIEW GATEWAY FIREWALL

> Route Advertisement
> Additional Settings
> SERVICE INTERFACES
> STATIC ROUTES

Figure 3.16: successful creation of the segment in NSX-T

3.4 Configuring and Managing vSphere namespaces

The purpose of creating and configuring a vSphere namespace in this thesis is to enable the Tanzu Kubernetes cluster and vSphere Pods provisioning. The vSphere namespace should be defined and configured after creating and managing a Supervisor Cluster. The vSphere namespace is required because the Tanzu Kubernetes clusters and vSphere Pods are deployed in it. As mentioned in Chapter 2, the vSphere administrator creates and configures the vSphere namespace on the Supervisor Cluster with the resource limitations and user permissions. After the namespace has been created and configured, the DevOps team will be able to access the namespace they have permission to via the URL of the Kubernetes control plane that the vSphere administrator has provided.

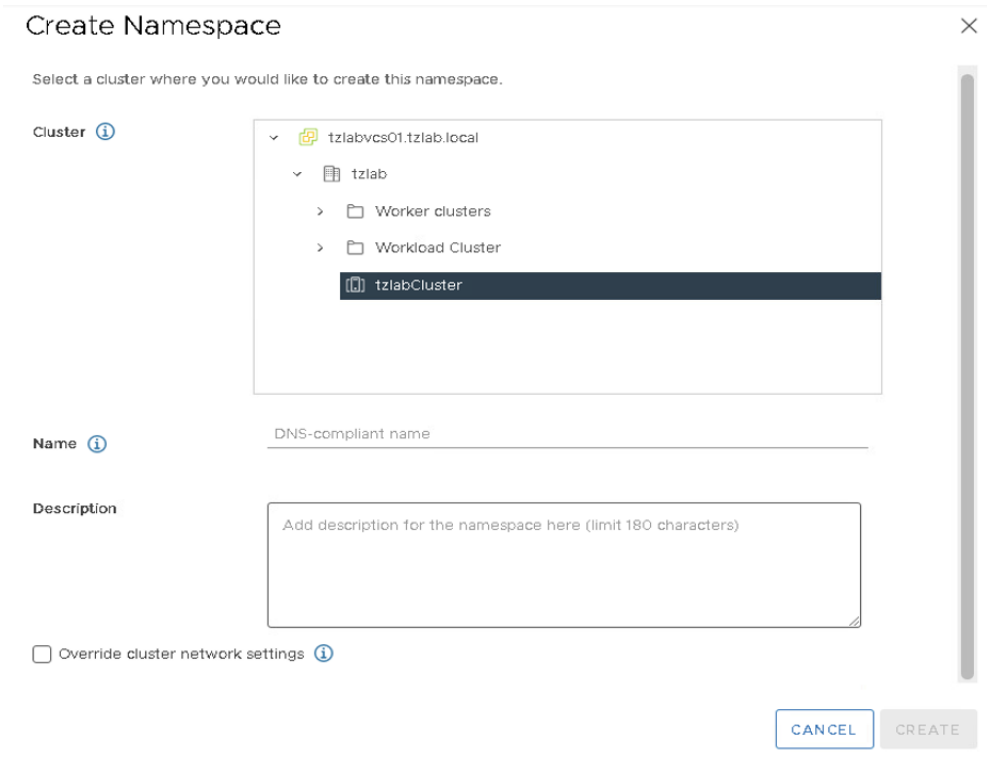
What is required prior to the creation of a namespace?

1. The Supervisor Cluster has already been configured.
2. The storage policies have already been defined.
3. DevOps engineers' user accounts have already been created.
4. The Tanzu Kubernetes content library has already been created.

Creating and configuring a vSphere namespace

To create a new namespace, one needs to choose the desired cluster on which the namespace will be created. Only the clusters with workload management enabled will be displayed automatically to choose from. As mentioned before, the cluster with workload management enabled is called the Supervisor Cluster. Figure 3.17 illustrates that the "tzlabCluster" is the only available cluster for the selection. It's also worth mentioning that the namespace name must be unique across all

Supervisor Clusters and must be DNS compliant (a-z, 0-9, and "-" characters up to 63 characters).


Create Namespace

Select a cluster where you would like to create this namespace.

Cluster
- tzlabvcs01.tzlab.local
  - tzlab
    - Worker clusters
    - Workload Cluster
    - tzlabCluster

Name: DNS-compliant name

Description: Add description for the namespace here (limit 180 characters)

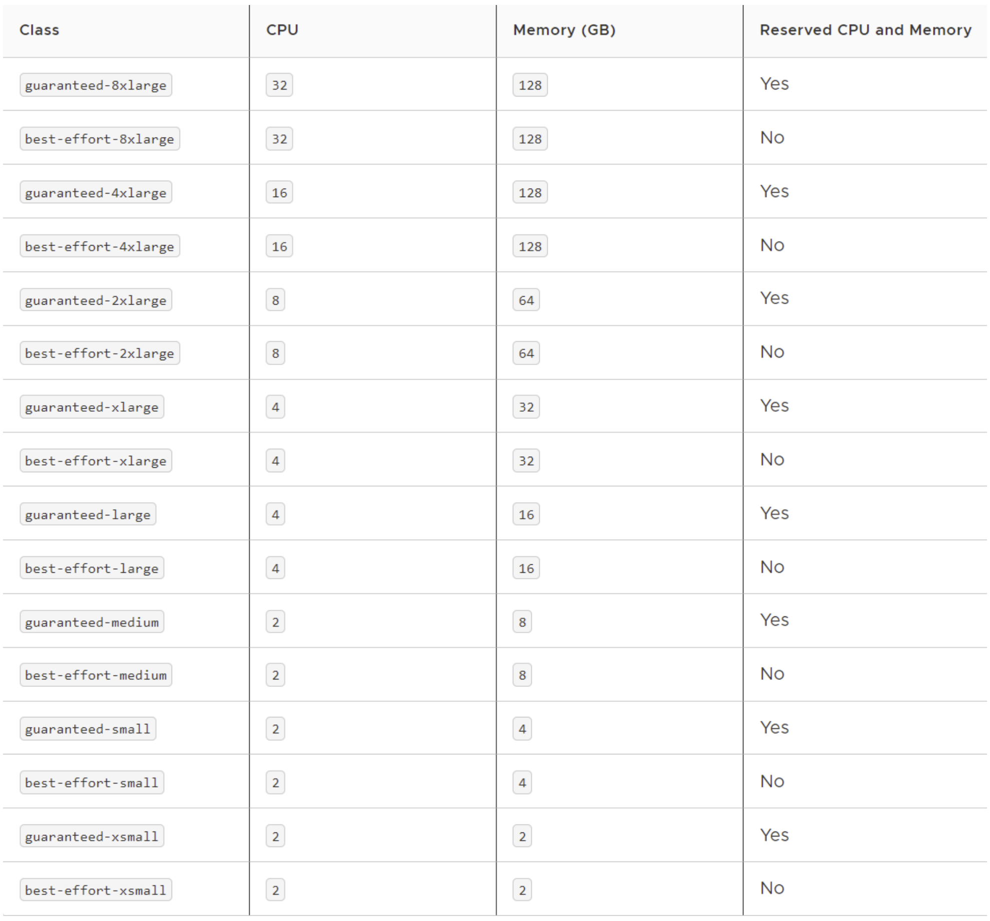
☐ Override cluster network settings

Figure 3.17: Creating namespace on tzlabCluster

After the namespace is created, the vSphere administrator must grant permission to the DevOps engineers so that they can access it. It is also essential to set up a storage policy for the namespace to allow the DevOps team to access persistent storage. Once the storage policy has been assigned, vSphere with Tanzu creates a matching Kubernetes storage class in the namespace. It is worth mentioning that the Tanzu Kubernetes Grid Service is employed in this thesis, which enables the storage class to be automatically copied from the namespace to the Kubernetes cluster.

Setting up resource limitations is yet another important aspect of namespace configuration. It is best practice to limit CPU, memory, and storage consumption as appropriate. The overall amount of resources allocated to the namespace will be controlled by this resource limitation.
Another component that should be configured is the VM service. The VM service is a feature offered by vSphere with Tanzu to enable DevOps engineers to deploy and manage virtual machines in a vSphere namespace using Kubernetes standard APIs. The VM service also enables vSphere administrators to deliver resources and supply VM class templates to Kubernetes. The VM class is a VM specification

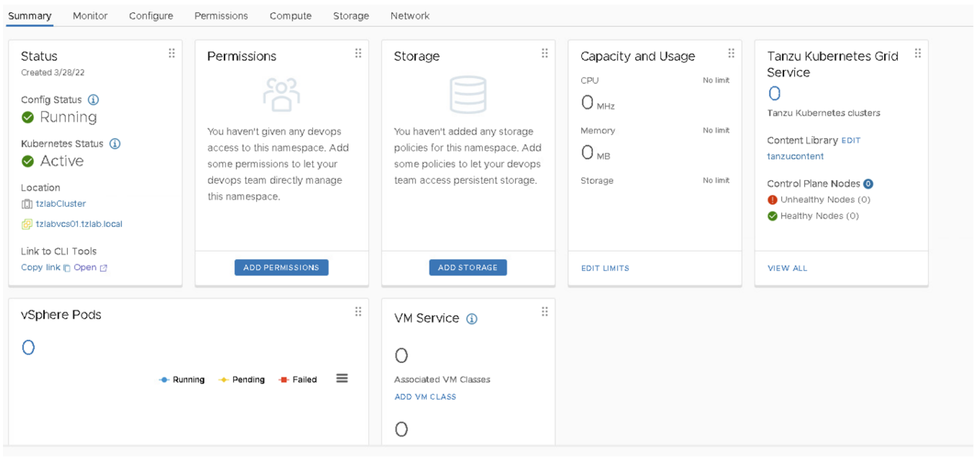
that defines the number of CPUs and quantity of memory in gigabytes (GB) [37]. Although vSphere administrator can design custom VM classes, the default VM classes offered by vSphere with Tanzu, as shown in Figure 3.18 will be used in this thesis.



Class	CPU	Memory (GB)	Reserved CPU and Memory
guaranteed-8xlarge	32	128	Yes
best-effort-8xlarge	32	128	No
guaranteed-4xlarge	16	128	Yes
best-effort-4xlarge	16	128	No
guaranteed-2xlarge	8	64	Yes
best-effort-2xlarge	8	64	No
guaranteed-xlarge	4	32	Yes
best-effort-xlarge	4	32	No
guaranteed-large	4	16	Yes
best-effort-large	4	16	No
guaranteed-medium	2	8	Yes
best-effort-medium	2	8	No
guaranteed-small	2	4	Yes
best-effort-small	2	4	No
guaranteed-xsmall	2	2	Yes
best-effort-xsmall	2	2	No

Figure 3.18: Default Virtual Machine Classes
[38]

Before providing the DevOps engineers with the URL of the Kubernetes control plane, the vSphere namespace needs to be associated with the content library for Tanzu Kubernetes releases. Associating the namespace with the content library is a required step if the Tanzu Kubernetes clusters are planned to be provisioned, which is the case in this thesis. Figure 3.19 shows the namespace configuration panel before being configured, whereas Figure 3.20 shows the same panel but after being configured. Figure 3.20 shows the URL of the Kubernetes control plane (Link to CLI tools), the permissions granted to the "devops" user with the edit role as well as the "administrator" user with the owner role, the resources allocated to the namespace, the Tanzu content library selected, and the VM services with 16 VM classes.


Summary Monitor Configure Permissions Compute Storage Network

Status
Created 3/28/22
Config Status
Running
Kubernetes Status
Active
Location
tzlabCluster
tzlabvcs01.tzlab.local
Link to CLI Tools
Copy link Open

Permissions
You haven't given any devops access to this namespace. Add some permissions to let your devops team directly manage this namespace.
ADD PERMISSIONS

Storage
You haven't added any storage policies for this namespace. Add some policies to let your devops team access persistent storage.
ADD STORAGE

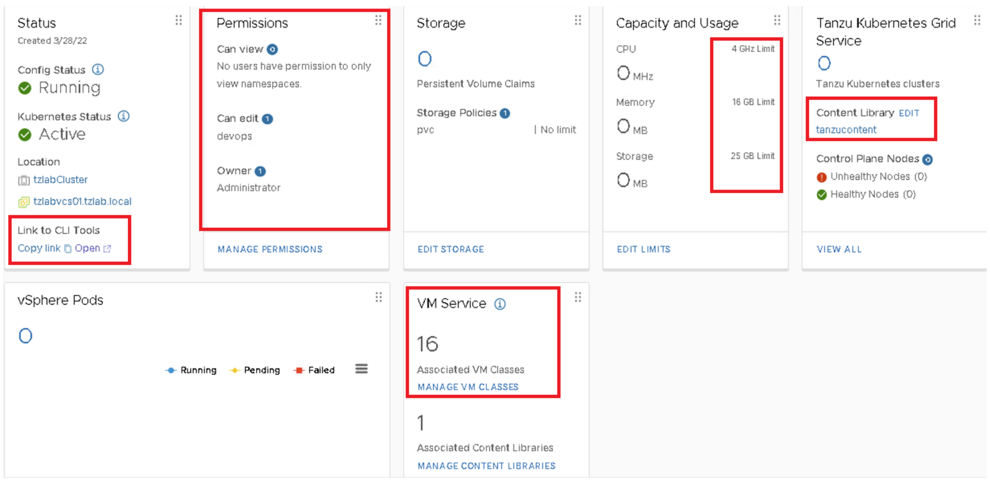
Capacity and Usage
CPU No limit
0 MHz
Memory No limit
0 MB
Storage No limit
EDIT LIMITS

Tanzu Kubernetes Grid Service
0
Tanzu Kubernetes clusters
Content Library EDIT
tanzucontent
Control Plane Nodes
Unhealthy Nodes (0)
Healthy Nodes (0)
VIEW ALL

vSphere Pods
0
Running Pending Failed

VM Service
0
Associated VM Classes
ADD VM CLASS
0

Figure 3.19: The namespace configuration Panel (before configuring the namespace)


Status
Created 3/28/22
Config Status
Running
Kubernetes Status
Active
Location
tzlabCluster
tzlabvcs01.tzlab.local
Link to CLI Tools
Copy link Open

Permissions
Can view 0
No users have permission to only view namespaces.
Can edit 1
devops
Owner 1
Administrator
MANAGE PERMISSIONS

Storage
0
Persistent Volume Claims
Storage Policies 1
pvc | No limit
EDIT STORAGE

Capacity and Usage
CPU 4 GHz Limit
0 MHz
Memory 16 GB Limit
0 MB
Storage 25 GB Limit
0 MB
EDIT LIMITS

Tanzu Kubernetes Grid Service
0
Tanzu Kubernetes clusters
Content Library EDIT
tanzucontent
Control Plane Nodes 0
Unhealthy Nodes (0)
Healthy Nodes (0)
VIEW ALL

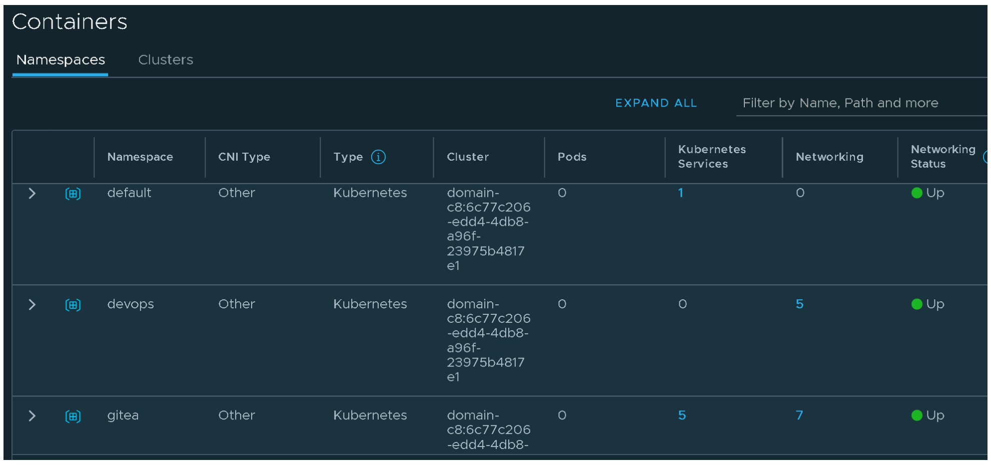
vSphere Pods
0
Running Pending Failed

VM Service
16
Associated VM Classes
MANAGE VM CLASSES
1
Associated Content Libraries
MANAGE CONTENT LIBRARIES

Figure 3.20: The namespace configuration Panel (after configuring the namespace)

vSphere Namespace on NSX-T

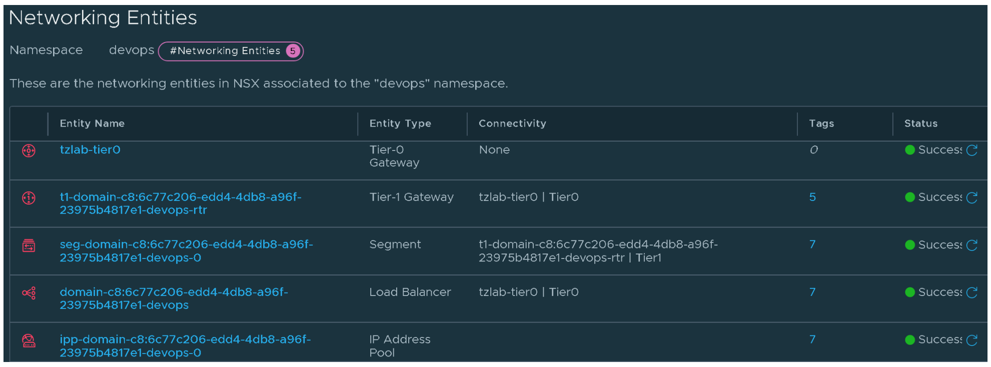
As previously mentioned, NSX-T provides connectivity for each namespace that is created using the vSphere client. Figure 3.21 illustrates the inventory overview, which shows all related namespaces that were created as well as network configuration details for each namespace.



Namespace	CNI Type	Type	Cluster	Pods	Kubernetes Services	Networking	Networking Status
default	Other	Kubernetes	domain-c8:6c77c206-edd4-4db8-a96f-23975b4817e1	0	1	0	● Up
devops	Other	Kubernetes	domain-c8:6c77c206-edd4-4db8-a96f-23975b4817e1	0	0	5	● Up
gitea	Other	Kubernetes	domain-c8:6c77c206-edd4-4db8-	0	5	7	● Up

Figure 3.21: Namespaces on NSX-T

As shown in Figure 3.22, networking objects are created for the "devops" namespace along with a dedicated tier-1 gateway. This tier-1 gateway is a logical segment for the namespace "devops" and has a dedicated load balancer instantiated on it. Other networking objects created for the namespace include a unique logical NSX-T segment.

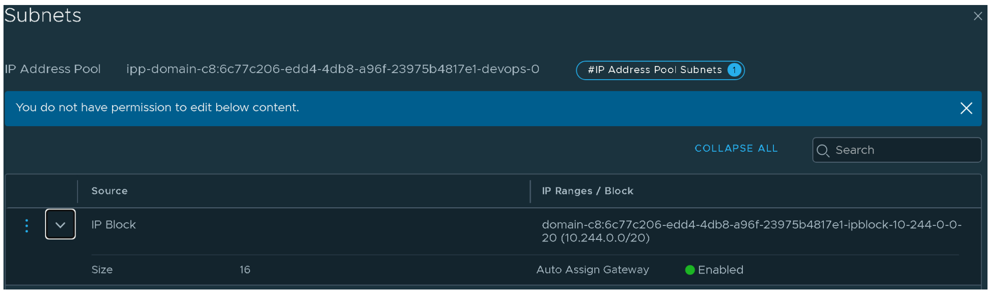


These are the networking entities in NSX associated to the "devops" namespace.

Entity Name	Entity Type	Connectivity	Tags	Status
tzlab-tier0	Tier-0 Gateway	None	0	● Succes
t1-domain-c8:6c77c206-edd4-4db8-a96f-23975b4817e1-devops-rtr	Tier-1 Gateway	tzlab-tier0 | Tier0	5	● Succes
seg-domain-c8:6c77c206-edd4-4db8-a96f-23975b4817e1-devops-0	Segment	t1-domain-c8:6c77c206-edd4-4db8-a96f-23975b4817e1-devops-rtr | Tier1	7	● Succes
domain-c8:6c77c206-edd4-4db8-a96f-23975b4817e1-devops	Load Balancer	tzlab-tier0 | Tier0	7	● Succes
ipp-domain-c8:6c77c206-edd4-4db8-a96f-23975b4817e1-devops-0	IP Address Pool		7	● Succes

Figure 3.22: Networking details for the “devops” namespace

The segment being created is also linked to the IP address pool which means that a subnet will be associated automatically with that segment as shown in Figure 3.23.



Subnets

IP Address Pool ipp-domain-c8:6c77c206-edd4-4db8-a96f-23975b4817e1-devops-0 #IP Address Pool Subnets 1

You do not have permission to edit below content.

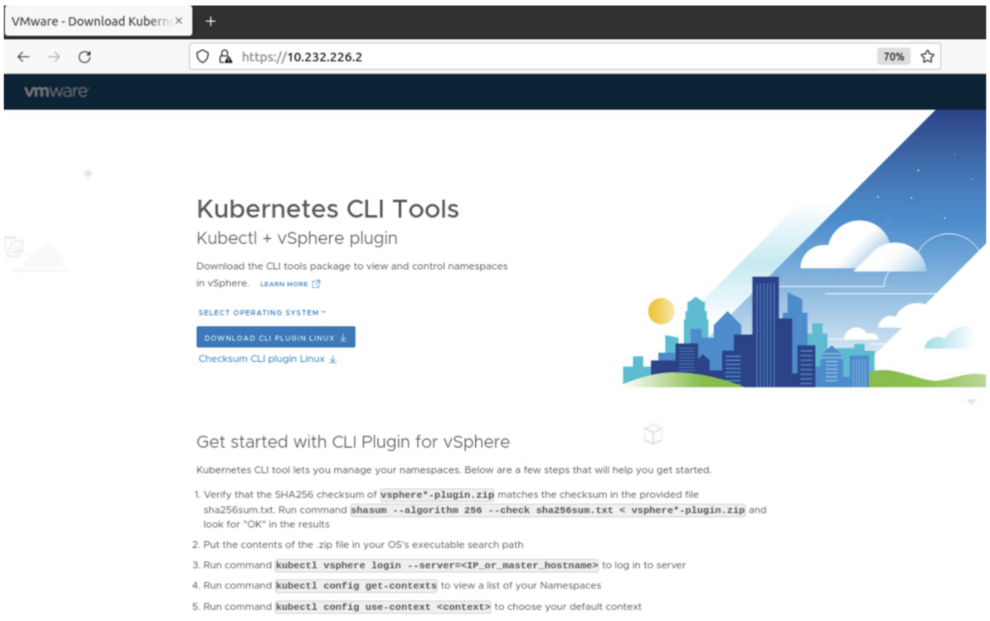
COLLAPSE ALL Search

Source		IP Ranges / Block	
IP Block		domain-c8:6c77c206-edd4-4db8-a96f-23975b4817e1-ipblock-10-244-0-0-20 (10.244.0.0/20)	
Size	16	Auto Assign Gateway	● Enabled

Figure 3.23: IP Address Pool created for the “devops” namespace

3.5 Provisioning Tanzu Kubernetes Clusters

The vSphere administrator will deliver the URL link to the DevOps team after creating and configuring the namespace. The DevOps team will then utilize this URL link to connect to the Supervisor Cluster for provisioning Tanzu Kubernetes clusters. The DevOps team will begin by downloading the Kubernetes CLI Tools package following the installation guide provided by the URL link as shown in Figure 3.24.



Kubernetes CLI Tools

Kubectl + vSphere plugin

Download the CLI tools package to view and control namespaces in vSphere. LEARN MORE

SELECT OPERATING SYSTEM

DOWNLOAD CLI PLUGIN LINUX

Checksum CLI plugin Linux

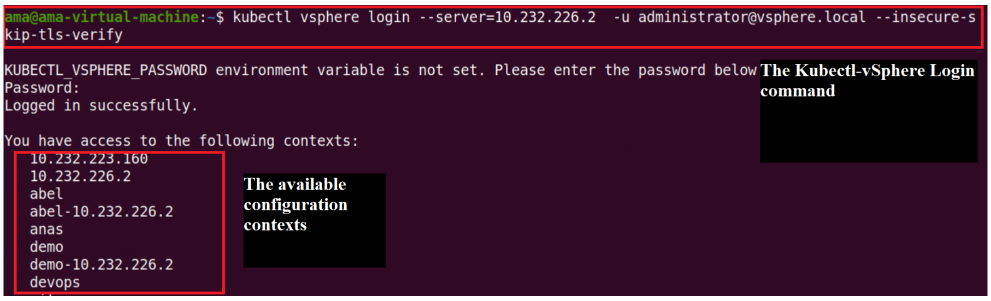
Get started with CLI Plugin for vSphere

Kubernetes CLI tool lets you manage your namespaces. Below are a few steps that will help you get started.

1. Verify that the SHA256 checksum of `vsphere*-plugin.zip` matches the checksum in the provided file sha256sum.txt. Run command `shasum --algorithm 256 --check sha256sum.txt < vsphere*-plugin.zip` and look for "OK" in the results
2. Put the contents of the .zip file in your OS's executable search path
3. Run command `kubectl vsphere login --server=<IP_or_master_hostname>` to log in to server
4. Run command `kubectl config get-contexts` to view a list of your Namespaces
5. Run command `kubectl config use-context <context>` to choose your default context

Figure 3.24: The installation guide for the Kubernetes CLI Tools via the URL of the Kubernetes control plane

There are two variants of the Kubernetes CLI Tools: the vSphere Plugin for kubectl and the standard open-source kubectl. The vSphere Plugin for kubectl will be utilized in this project. The vSphere Plugin expands kubectl’s command set, enabling the DevOps team to connect to the Supervisor Cluster and authenticate with vCenter Single Sign-On credentials as shown in Figure 3.25.

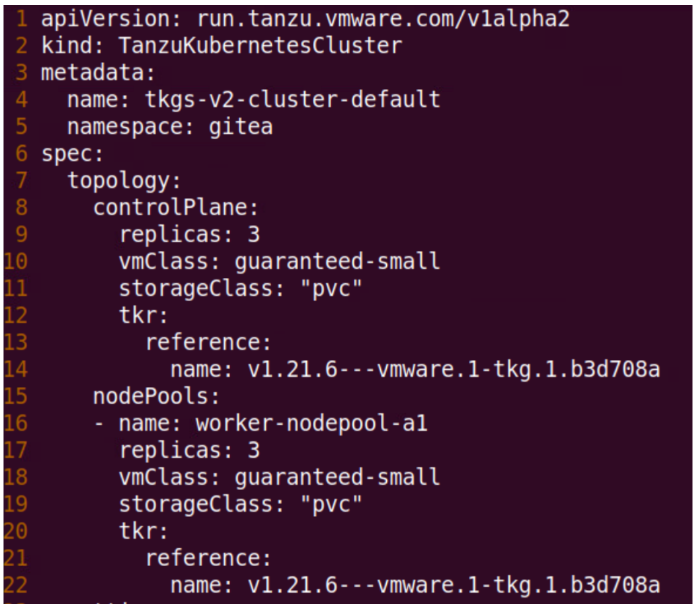


```
ama@ama-virtual-machine:~$ kubectl vsphere login --server=10.232.226.2  -u administrator@vsphere.local --insecure-s
kip-tls-verify

KUBECTL_VSPHERE_PASSWORD environment variable is not set. Please enter the password below
Password:
Logged in successfully.

You have access to the following contexts:
   10.232.223.160
   10.232.226.2
   abel
   abel-10.232.226.2
   anas
   demo
   demo-10.232.226.2
   devops
```

The Kubectl-vSphere Login command
The available configuration contexts

Figure 3.25: Connecting to the Supervisor Cluster using with vCenter Single Sign-On credentials

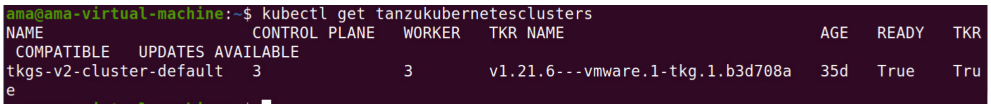
After the DevOps engineers have logged into the Supervisor Cluster using the Kubectl-vSphere CLI plugin, they need to switch to the working Kubernetes context they have access to using the kubectl command. Once they have switched to the context for the vSphere namespace where they want to provision the Tanzu Kubernetes Cluster, they can create the YAML file for provisioning it. As shown in Figure 3.26, this YAML file should specify at least the name of the Tanzu Kubernetes cluster, the namespace, the number of control planes, the number of worker nodes, the VM class for both control planes and worker nodes, the storage policy, and the Tanzu Kubernetes release version.



```yaml
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-v2-cluster-default
  namespace: gitea
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-small
      storageClass: "pvc"
      tkr:
        reference:
          name: v1.21.6---vmware.1-tkg.1.b3d708a
    nodePools:
    - name: worker-nodepool-a1
      replicas: 3
      vmClass: guaranteed-small
      storageClass: "pvc"
      tkr:
        reference:
          name: v1.21.6---vmware.1-tkg.1.b3d708a
```

Figure 3.26: Example YAML for Provisioning a Default Tanzu Kubernetes Cluster with the minimum configuration

After customizing the YAML file as needed, the DevOps team can apply it using the kubectl command to provision the Tanzu Kubernetes cluster. Once it is done, the DevOps team can monitor the deployment of this cluster as shown in Figure 3.27.



```
ama@ama-virtual-machine:~$ kubectl get tanzukubernetesclusters
NAME                       CONTROL PLANE   WORKER   TKR NAME                          AGE   READY   TKR
 COMPATIBLE   UPDATES AVAILABLE
tkgs-v2-cluster-default   3               3        v1.21.6---vmware.1-tkg.1.b3d708a   35d   True    Tru
e
```

Figure 3.27: Monitor the deployment of cluster nodes using kubectl

The Tanzu Kubernetes cluster will be visible to the vSphere administrator via vSphere client, as shown in Figure 3.28. In this figure, there are the Tanzu Kubernetes cluster which was deployed to the “gitea” namespace. This cluster contains the three control plane and three worker VMs that were deployed using the previous YAML file as indicated by the two arrows in Figure 3.28.

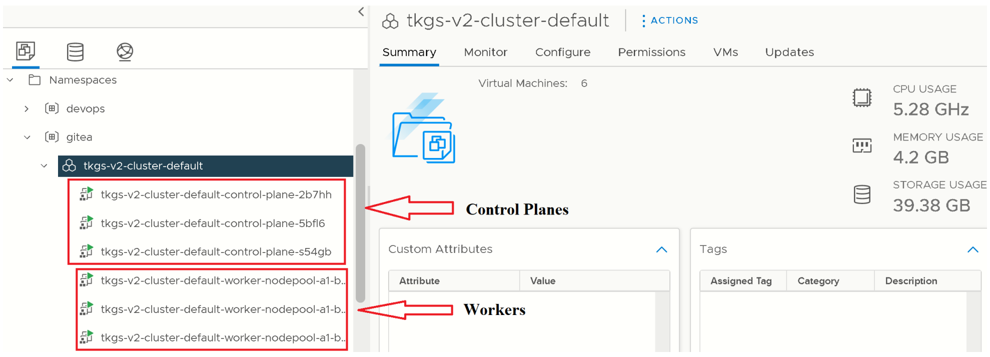
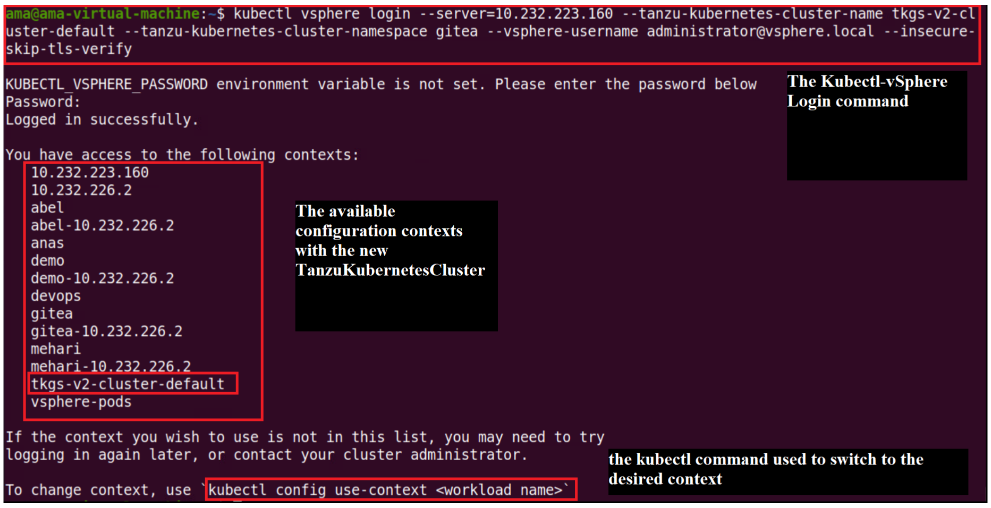


Figure 3.28: vSphere Client with the provisioned Tanzu Kubernetes Cluster

Login to the Tanzu Kubernetes Cluster

To deploy a workload on the Tanzu Kubernetes Cluster, the DevOps need to log in to this cluster using the Kubectl-vSphere CLI command as shown in Figure 3.29. This command appears complicated, but it can be easily scripted using Bash script.



```
ama@ama-virtual-machine:~$ kubectl vsphere login --server=10.232.223.160 --tanzu-kubernetes-cluster-name tkgs-v2-cl
uster-default --tanzu-kubernetes-cluster-namespace gitea --vsphere-username administrator@vsphere.local --insecure-
skip-tls-verify

KUBECTL_VSPHERE_PASSWORD environment variable is not set. Please enter the password below
Password:
Logged in successfully.

You have access to the following contexts:
   10.232.223.160
   10.232.226.2
   abel
   abel-10.232.226.2
   anas
   demo
   demo-10.232.226.2
   devops
   gitea
   gitea-10.232.226.2
   mehari
   mehari-10.232.226.2
   tkgs-v2-cluster-default
   vsphere-pods

If the context you wish to use is not in this list, you may need to try
logging in again later, or contact your cluster administrator.

To change context, use `kubectl config use-context <workload name>`
```

The Kubectl-vSphere Login command
The available configuration contexts with the new TanzuKubernetesCluster
the kubectl command used to switch to the desired context

Figure 3.29: Log in to the Tanzu Kubernetes cluster using Kubectl-vSphere CLI

Figure 3.29 shows a new configuration context named tkg-v2-cluster-default, which is the Tanzu Kubernetes cluster context that the DevOps team will switch to when they want to deploy a workload to this cluster. However, prior to deploying a workload, the DevOps team should define an adequate pod security policy as will be described in chapter 6.

Chapter 4

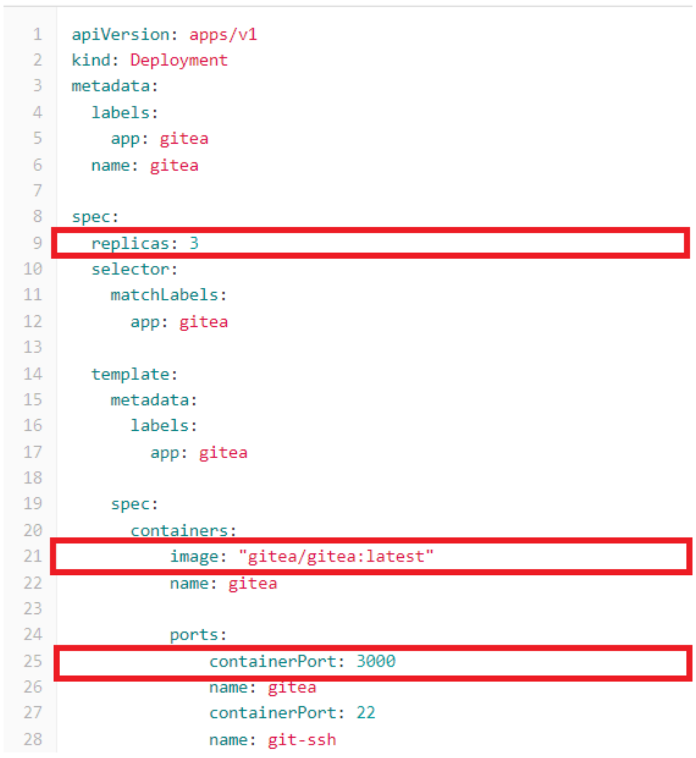
Application Management

To evaluate the use of Tanzu for running Kubernetes clusters, the project group has deployed an open-source application called Gitea that will be orchestrated using Tanzu, which was installed and configured in the previous chapter. Section 4.1 explains how the application is deployed to the Tanzu Kubernetes cluster, and Section 4.2 goes through the creation of persistent volume and persistent volume claim for the application. Section 4.3 is about the deployment of the MySQL database that will be used by the Gitea application to save and request data. Section 4.4 explains how the application can be scaled both manually and automatically. 4.5 briefly explains how Gitea can be updated and patched.

4.1 Gitea Application Deployment

Gitea is an open-source application used for software version control, similar to Gitlab [39]. This section deals with the deployment of the application using the Tanzu Kubernetes cluster that was provisioned in section 3.5. As mentioned in section 3.5, it is required to log in to the Supervisor Cluster using vCenter Single Sign-On credentials and then switch to the working Kubernetes context using the kubectl command, as shown in Figure 3.25. Gitea is deployed using the YAML configuration file displayed in Figure 4.1 after logging in to the Supervisor Cluster and switching to the working Kubernetes context.

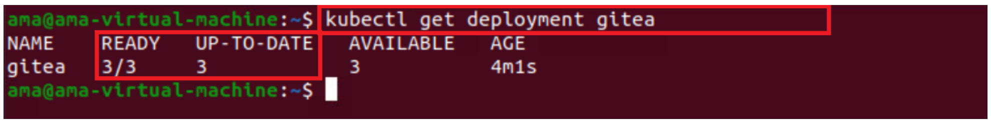
A deployment is abstraction of pods that automates the manual process of deploying, scaling and updating applications. Figure 4.1 shows the deployment YAML file that pulls the latest Gitea image for the Gitea container to use (line 21). This container listens on port 3000, which is the port that will be used by Gitea (line 25) and port 22 is what SSH listens to (line 27). The deployment file shown in Figure 4.1 was applied to the Tanzu Kubernetes cluster using the kubectl command "kubectl apply -f frontend-gitea.yaml".



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: gitea
  name: gitea

spec:
  replicas: 3
  selector:
    matchLabels:
      app: gitea

  template:
    metadata:
      labels:
        app: gitea

    spec:
      containers:
          image: "gitea/gitea:latest"
          name: gitea

          ports:
              containerPort: 3000
              name: gitea
              containerPort: 22
              name: git-ssh
```

Figure 4.1: YAML configuration file used to create a deployment for Gitea

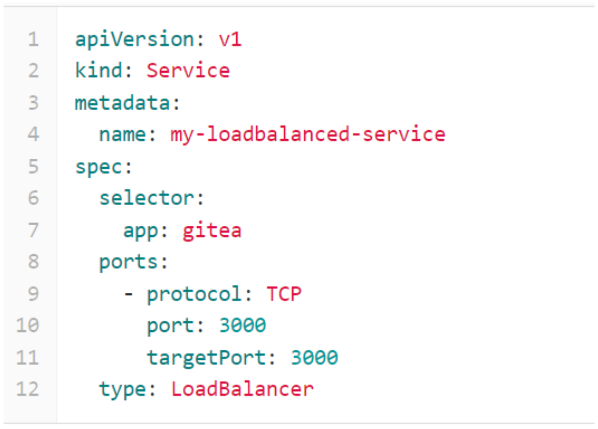
After successfully applying the configuration file, As shown in Figure 4.2, a deployment of three pods should be listed as "READY" and "UP-TO-DATE," which means that the pods are up and running.



```
ama@ama-virtual-machine:~$ kubectl get deployment gitea
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
gitea   3/3     3            3           4m1s
ama@ama-virtual-machine:~$ 
```

Figure 4.2: Deployment created for Gitea

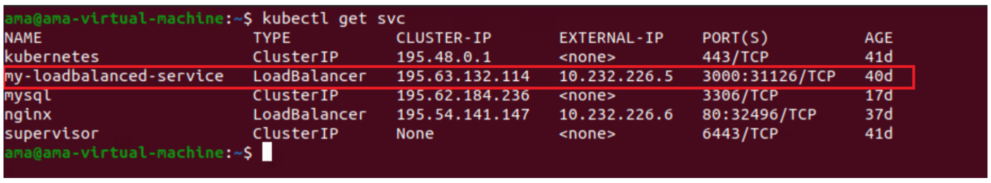
After creating a deployment by applying the YAML configuration file, the next step is to create a service for the deployment. A service in Kubernetes was introduced in section 2.3.3. As shown in Figure 4.3, a service of type Loadbalancer is created for the Gitea application using a YAML configuration file that was constructed by the project group. A Loadbalancer as a service type was chosen to make the service accessible externally through the advanced load balancer that was deployed in 3.2.1. As shown in figure Figure 4.3, the loadbalancer service exposes port 3000 which is the same port used by the ClusterIP service (line 10).



```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-loadbalanced-service
spec:
  selector:
    app: gitea
  ports:
    - protocol: TCP
      port: 3000
      targetPort: 3000
  type: LoadBalancer
```

Figure 4.3: LoadBalancer service for the application

After successfully creating a loadbalancer service, the application deployed is accessible using the external IP address automatically assigned by the advanced load balancer. This service created has a name and an external IP address that can be shown using the command "kubectl get svc" as shown in Figure 4.4. The IP address 10.232.226.5 along with port 3000 is used to access the Gitea application using a web browser.



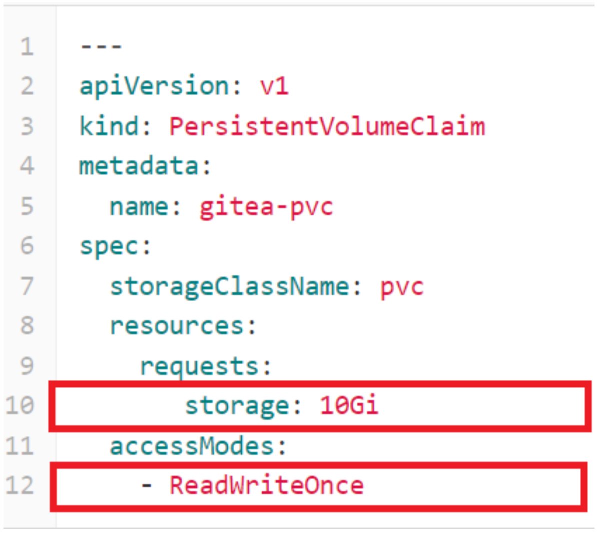
```
ama@ama-virtual-machine:~$ kubectl get svc
NAME                      TYPE           CLUSTER-IP       EXTERNAL-IP     PORT(S)          AGE
kubernetes                ClusterIP      195.48.0.1       <none>          443/TCP          41d
my-loadbalanced-service   LoadBalancer   195.63.132.114   10.232.226.5    3000:31126/TCP   40d
mysql                     ClusterIP      195.62.184.236   <none>          3306/TCP         17d
nginx                     LoadBalancer   195.54.141.147   10.232.226.6    80:32496/TCP     37d
supervisor                ClusterIP      None             <none>          6443/TCP         41d
ama@ama-virtual-machine:~$
```

Figure 4.4: Loadbalancer service

4.2 Creating Persistent Volume and Persistent Volume Claim

Gitea can be deployed in a lot of different ways. The project group has chosen to deploy MySQL database that the Gitea application can use to save and request data. It is also possible to deploy Gitea without a database application, but this solution requires a persistent volume that the application can use as a storge. Instead of having a Gitea application solution that directly uses persistent volume, the project group has deployed MySQL application that uses persistent volume as storage. Before deploying a MySQL application on the Tanzu Kubernetes cluster, it is required to provision a persistent volume for the MySQL application that will be deployed on the Tanzu Kubernetes cluster.

The project group has provisioned a persistent volume by creating a storage policy in the vSphere client. This storage policy is configured to use one of HDO's datastores for persistent storage and attached to the Tanzu Kubernetes cluster. Applications deployed in this cluster where a persistent volume is attached, can use this persistent volume storage. Before using the persistent volume created, it is required to create a persistent volume claim that requests storage resources from the persistent volume. A persistent volume claim is used to mount the persistent volume by requesting a specific size and access mode from the persistent volume. As shown in Figure 4.5, the project group has provisioned a persistent volume claim that requests 10 gigabytes of storage resources from the persistent volume for the MySQL application to use (line 10). The access mode "ReadWriteOnce" (in line 12) is used, which means that only one pod is allowed to perform the read-write operation [40].



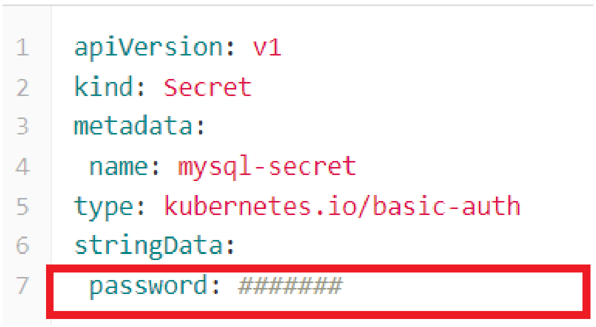
```
1   ---
2   apiVersion: v1
3   kind: PersistentVolumeClaim
4   metadata:
5     name: gitea-pvc
6   spec:
7     storageClassName: pvc
8     resources:
9       requests:
10        storage: 10Gi
11    accessModes:
12      - ReadWriteOnce
```

Figure 4.5: Persistent volume claim

After successfully creating a persistent volume and a persistent volume claim, the next step is to deploy the MySQL application on the Tanzu Kubernetes cluster. This application will be using the persistent volume claim created as a storage.

4.3 MySQL Database Deployment for Gitea

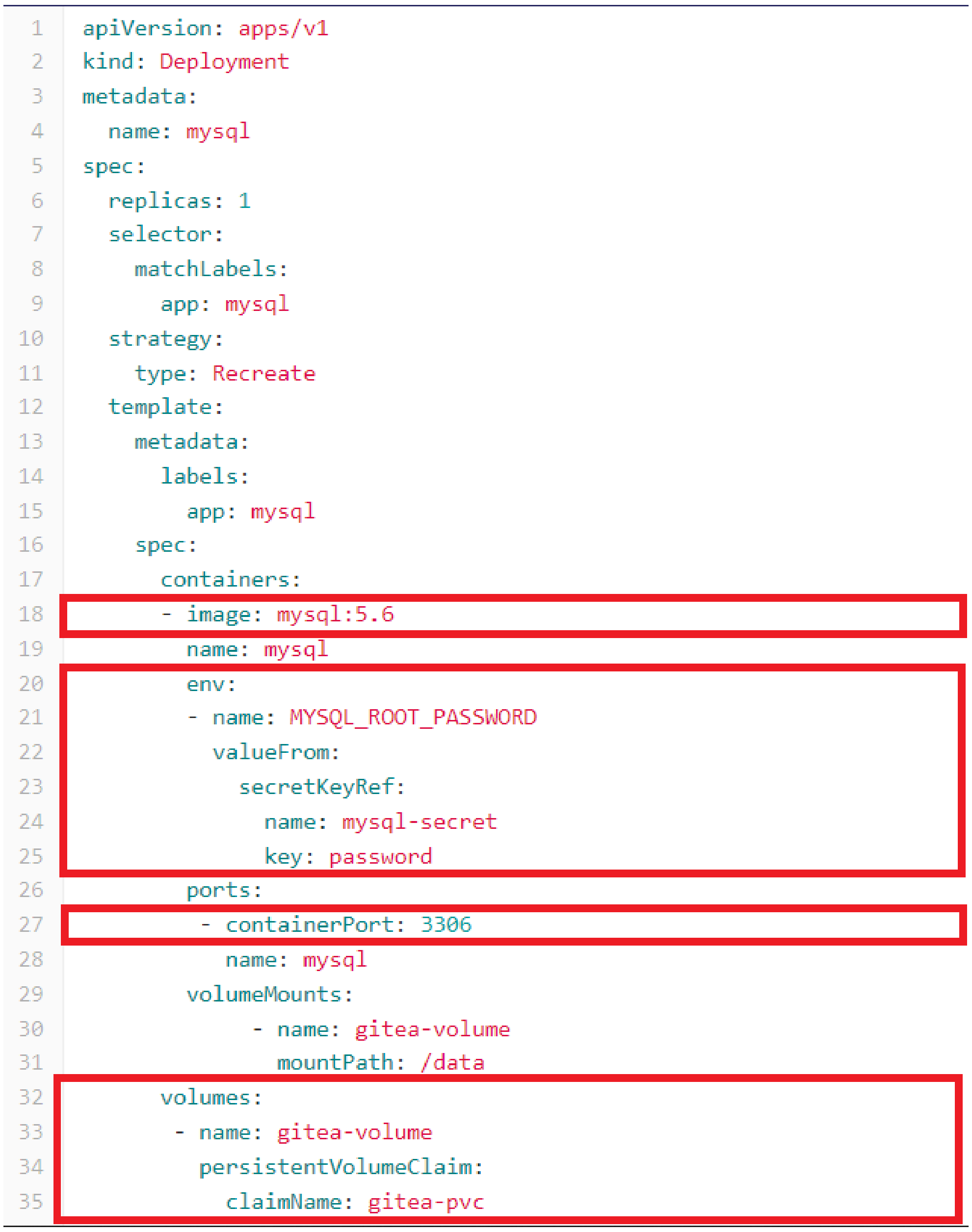
It was mentioned that the project group has deployed MySQL for the Gitea application to be used when requesting data. This MySQL deployment consists of one pod that performs read-write operations on the persistent volume created. Root is the default user for MySQL and has no password. It is best practise to set password for the username Root in MySQL. Passwords are sensitive information that should not be visible to others who have access to YAML configuration file used to deploy the application. Kubernetes has a functionality called secrets used to store sensitive data like passwords. Secret in Kubernetes was introduced in section 2.3.3. As shown in Figure 4.6, the project group has created a secret that stores the password for the MySQL application and is mapped to the YAML configuration file used to deploy MySQL (line 7).



```
1  apiVersion: v1
2  kind: Secret
3  metadata:
4    name: mysql-secret
5  type: kubernetes.io/basic-auth
6  stringData:
7    password: #######
```

Figure 4.6: Secret created

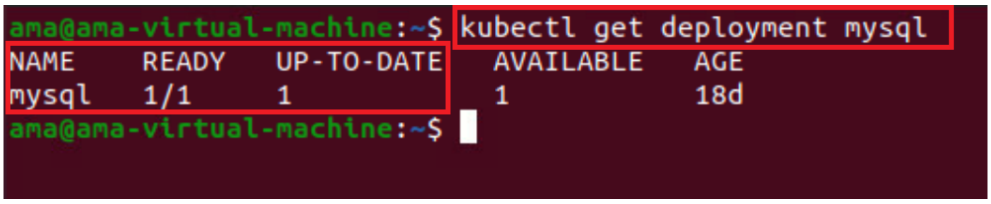
Figure 4.7 shows the deployment file that pulls the MySQL container that uses the MySQL image version 5.6 from the Docker Hub registry (line 18). This container exposes port 3306, which will be used by the Gitea application when communicating with the MySQL database (line 27). The "persistentVolumeClaim" section of the YAML configuration file declares the "gitea-pvc" persistent volume claim created in the previous section to be used (line 35). The "env" section is used to map to the secret that stores the password for the MySQL Root user (line 21 and 24). The password for the MySQL Root user is referenced to the secret created using "secretKeyRef" (line 23).



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - image: mysql:5.6
        name: mysql
        env:
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-secret
              key: password
        ports:
         - containerPort: 3306
           name: mysql
        volumeMounts:
             - name: gitea-volume
               mountPath: /data
      volumes:
       - name: gitea-volume
         persistentVolumeClaim:
           claimName: gitea-pvc
```

Figure 4.7: YAML configuration file for MySQL deployment

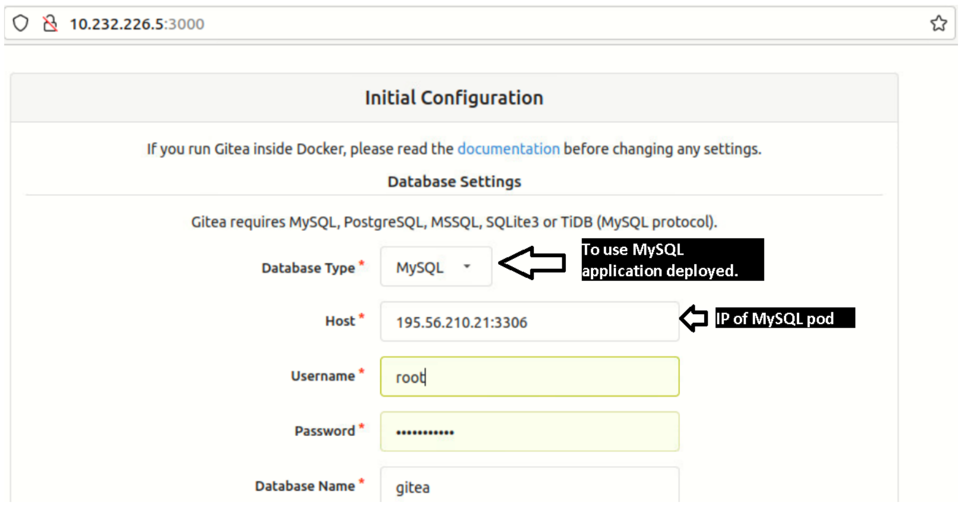
A service for the MySQL application is created in the same YAML configuration file. This service exposes port 3306 and uses ClusterIP as a service instead of using a loadbalancer service that provides an external IP address. This is because it is best practice to not expose database applications to the outside world. The MySQL application only needs to be available for the Gitea deployment. After successfully applying the YAML configuration file, the MySQL deployment should be visible using “kubectl get deployment mysql”, as shown in Figure 4.8.



```
ama@ama-virtual-machine:~$ kubectl get deployment mysql
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
mysql    1/1     1            1           18d
ama@ama-virtual-machine:~$ 
```

Figure 4.8: MySQl deployment

After successfully deploying a MySQL database, Gitea needs to be configured to use the MySQL application. This is done through the initial configuration of the Gitea application's web page, which one is redirected to when accessing the Gitea application deployed, as shown in Figure 4.9. This initial configuration page is used to connect the Gitea application with the MySQL application using the root username and password of the MySQL database.



10.232.226.5:3000

Initial Configuration

If you run Gitea inside Docker, please read the documentation before changing any settings.

Database Settings

Gitea requires MySQL, PostgreSQL, MSSQL, SQLite3 or TiDB (MySQL protocol).

Database Type *	MySQL
Host *	195.56.210.21:3306
Username *	root
Password *	•••••••••••
Database Name *	gitea

Figure 4.9: Gitea initial configuration page

After successfully configuring Gitea to be connected to the MySQL application, Gitea is now ready to be used as shown in the Figure 4.10. An account with a username and password is required to be created before one can start using the application. After creating an account, Gitea can be used as a software version control that functions like GitLab. For testing purposes, the project group has created a test repository using Gitea, as shown in Figure 4.10.

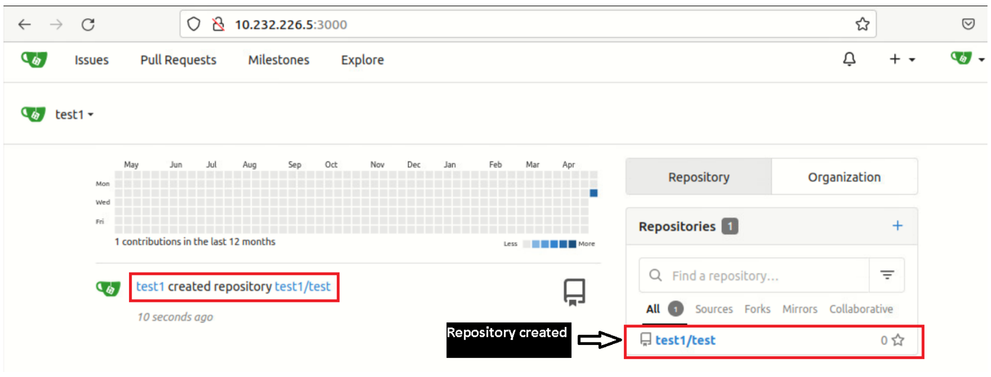
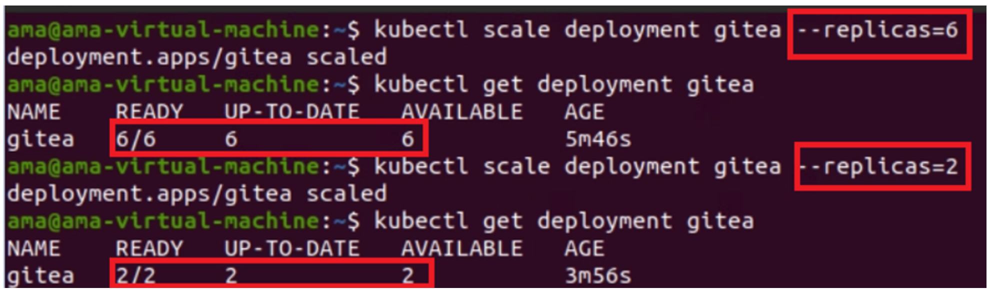


Figure 4.10: Repository created using Gitea

4.4 Scaling the application

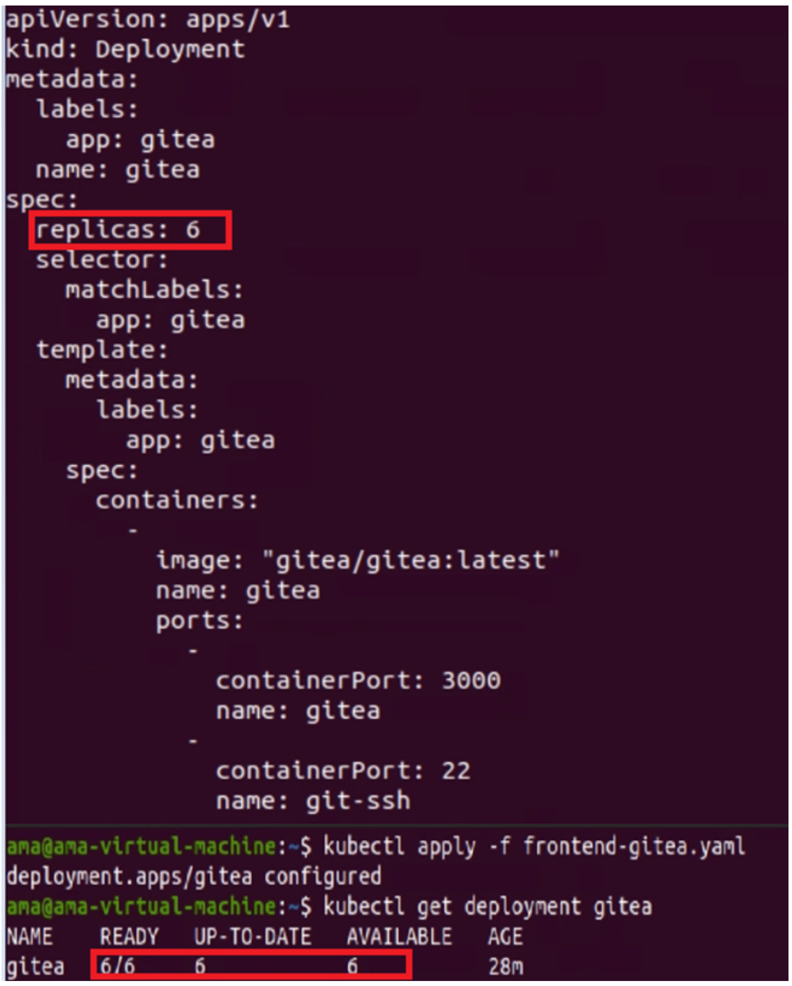
Scaling up or down the gitea pod within the resource limitations of a namespace is quite simple. There are two methods to accomplish this: the first is using the kubectl scale command, and the second is by modifying the YAML configuration file.
As previously shown in Figure 4.2, the gitea deployment currently consists of three running pods. The DevOps team can scale up the deployment by increasing the number of gitea application replicas. Figure 4.11 shows the kubectl command being used to scale up and down the deployment. It demonstrates that when scaling up, the number of replicas increased to 6 and thereafter decreased to 2 when scaling down.



```
ama@ama-virtual-machine:~$ kubectl scale deployment gitea --replicas=6
deployment.apps/gitea scaled
ama@ama-virtual-machine:~$ kubectl get deployment gitea
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
gitea   6/6     6            6           5m46s
ama@ama-virtual-machine:~$ kubectl scale deployment gitea --replicas=2
deployment.apps/gitea scaled
ama@ama-virtual-machine:~$ kubectl get deployment gitea
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
gitea   2/2     2            2           3m56s
```

Figure 4.11: Scaling up and down the Gitea application using kubectl scale command

The same thing can be achieved by modifying the YAML file and then running "kubectl apply -f name-file.yaml" to update the deployment with the new scaling settings, as illustrated in Figure 4.12 .



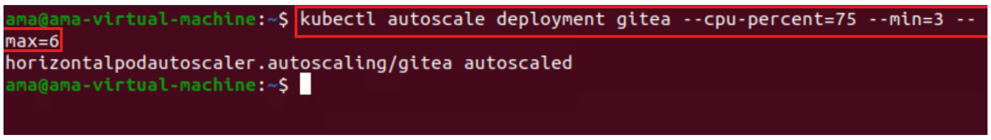
```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: gitea
  name: gitea
spec:
  replicas: 6
  selector:
    matchLabels:
      app: gitea
  template:
    metadata:
      labels:
        app: gitea
    spec:
      containers:
        -
          image: "gitea/gitea:latest"
          name: gitea
          ports:
            -
              containerPort: 3000
              name: gitea
            -
              containerPort: 22
              name: git-ssh
```

```
ama@ama-virtual-machine:~$ kubectl apply -f frontend-gitea.yaml
deployment.apps/gitea configured
ama@ama-virtual-machine:~$ kubectl get deployment gitea
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
gitea   6/6     6            6           28m
```

Figure 4.12: Scaling up the Gitea application by modifying the YAML configuration file

Scaling in Kubernetes can also be automated to scale up or down depending on demand. This type of scaling in Kubernetes is called Horizontal Pod Autoscaling and is based on increasing or decreasing the number of pods that are already running a workload based on metrics such as CPU utilization or average memory utilization [41]. A HorizontalPodAutoscaler[1] accesses Kubernetes objects that support scaling, such as deployment, and automatically scales pods based on metrics that are observed or customized by DevOps engineers. A HorizontalPodAutoscaler is created using the "kubectl autoscale deployment" command, as shown in Figure 4.13. This command maintains between 3 and 6 replicas of pods created by the Gitea deployment to maintain an average CPU utilization across all pods of 75 percent [42].

[1]Object in Kubernetes that automatically scales pods in response to CPU or memory utilization.



```
ama@ama-virtual-machine:~$ kubectl autoscale deployment gitea --cpu-percent=75 --min=3 --
max=6
horizontalpodautoscaler.autoscaling/gitea autoscaled
ama@ama-virtual-machine:~$ 
```

Figure 4.13: Horizontal Pod Autoscaling

Automatically scaling applications in Kubernetes is more effective and less time-consuming than manually scaling because it makes an application more responsive all the time, independent of the load. The downside of automatically scaling Kubernetes applications is that it may lead to unnecessary scaling and inefficient use of resources.

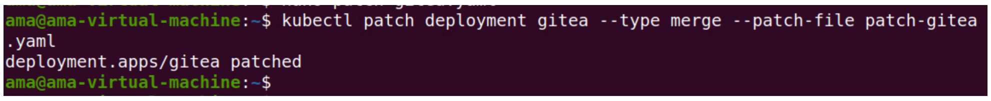
4.5 Updating and Patching

The DevOps team may need to patch or upgrade the Gitea application to a newer or more secure version. This may be accomplished by utilizing the “kubectl patch” command. This command allows the DevOps team to make changes to a running application by modifying only the fields that they want to change. They can change the number of replicas, the application name, the application version, and so on. The best practice for using the patch command is to create a YAML file and indicate just the necessary changes in it. As illustrated in Figure 4.14, when the patch command is applied, this YAML file will replace the older version of Gitea with the newer version v2 (Line 6).

```
1  spec:
2    template:
3     spec:
4        containers:
5          -
6            image: "gitea/gitea:v2"
7            name: gitea
8
```

Figure 4.14: YAML file to patch the Gitea application

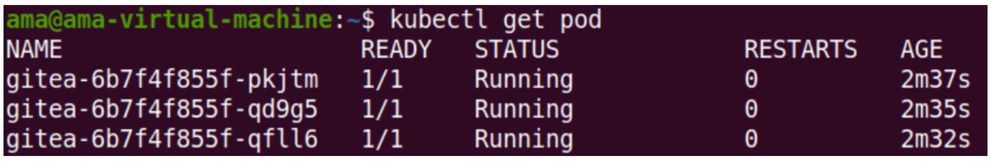
This file can be applied using the patch command, as shown in Figure 4.15; the patching method used is "strategic merge patch." This method is used to either merge or replace the object, however it will replace the version of Gitea in this case.



```
ama@ama-virtual-machine:~$ kubectl patch deployment gitea --type merge --patch-file patch-gitea
.yaml
deployment.apps/gitea patched
ama@ama-virtual-machine:~$
```

Figure 4.15: Patching the Gitea application using patch command

After applying the previous command, the older Gitea pods will be terminated and newer ones will be deployed. Figure 4.16 shows the newly deployed Gitea pods as a consequence of the patch command.



```
ama@ama-virtual-machine:~$ kubectl get pod
NAME                     READY   STATUS             RESTARTS   AGE
gitea-6b7f4f855f-pkjtm   1/1     Running            0          2m37s
gitea-6b7f4f855f-qd9g5   1/1     Running            0          2m35s
gitea-6b7f4f855f-qfll6   1/1     Running            0          2m32s
```

Figure 4.16: The newly deployed Gitea pods

Chapter 5

Monitoring

Monitoring is essential for the Tanzu Kubernetes cluster, vSphere Pods, and applications deployed there to be operational at all times. Monitoring Kubernetes environments involves monitoring the condition and resource usage of the clusters, nodes, containers, and vSphere Pods. This helps to detect abnormal and unexpected conditions, which are then notified to the administrator. By storing information about what is going on in the Kubernetes environment, problems can be identified faster, and troubleshooting can go faster. This chapter deals with the implementation of Tanzu Observability by Wavefront, which is a monitoring and visualization platform.

5.1 Tanzu Observability by Wavefont

Tanzu Observability is a platform used to monitor and observe Kubernetes environments [43]. Tanzu Observability monitors metrics such as CPU utilization and memory usage across nodes, clusters, pods, containers, and namespaces. It offers dashboards to give users real-time visibility of the Kubernetes environment by displaying metrics, histograms, and span logs through a web browser to help them identify and troubleshoot problems faster.

5.1.1 Capabilities

Tanzu Observability by Wavefront is an enterprise-level monitoring platform that provides capabilities such as charts and dashboards, alerts, and queries. It includes integration and services that allow the DevOps team to send raw data to the Wavefront service[1] through a Wavefront proxy[2]. It is important to note that there are different ways to get data into Tanzu Observability depending on the user's use case. For example, it is possible to create a customized metrics pipeline to collect metrics and send them to the Wavefront proxy. Data sent to

[1]Wavefront service: is a service that pulls data from the Wavefront proxy and runs metrics [44].
[2]Wavefront proxy: application that forwards metrics to the Wavefront service.

the Wavefront service is visualized using a preconfigured integration dashboard or customized dashboards. Tanzu Observability provides the DevOps teams with Wavefront Query Language (WQL) used to express queries that describe data at a particular time [45].

Creating alerts is one of the main capabilities of Tanzu Observability that allows the DevOps team to create alerts to detect problems and targets to be notified. For instance, alerts can be created to notify administrators when the memory usage of a Tanzu Kubernetes cluster reaches a predetermined limit.

5.1.2 Tanzu Observability Implementation

To get started with implementing Tanzu Observability, establishing a Tanzu Observability account is required. The trial account was created and will be used in this thesis. This account is required to obtain login credentials and thus begin visualizing and analyzing data metrics on the Tanzu Observability URL "longboard.wavefront.com."
After acquiring access credentials and logging into the Tanzu Observability URL, the DevOps engineer will start connecting Tanzu Observability with the Tanzu Kubernetes clusters. Figure 5.1 shows some of the integrations offered by Tanzu Observability. In this thesis, the "vSphere with Tanzu" integration will be implemented.

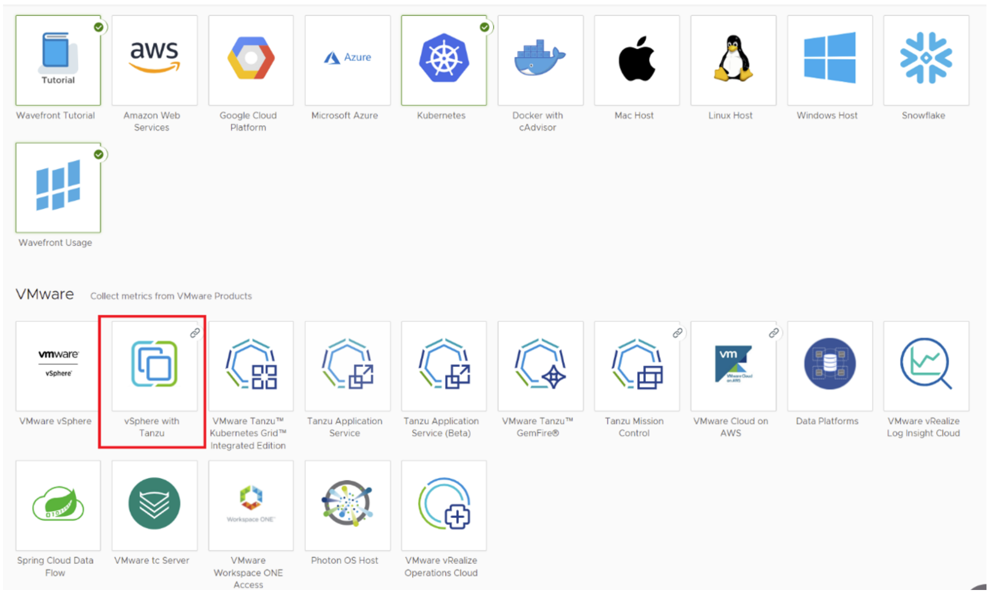
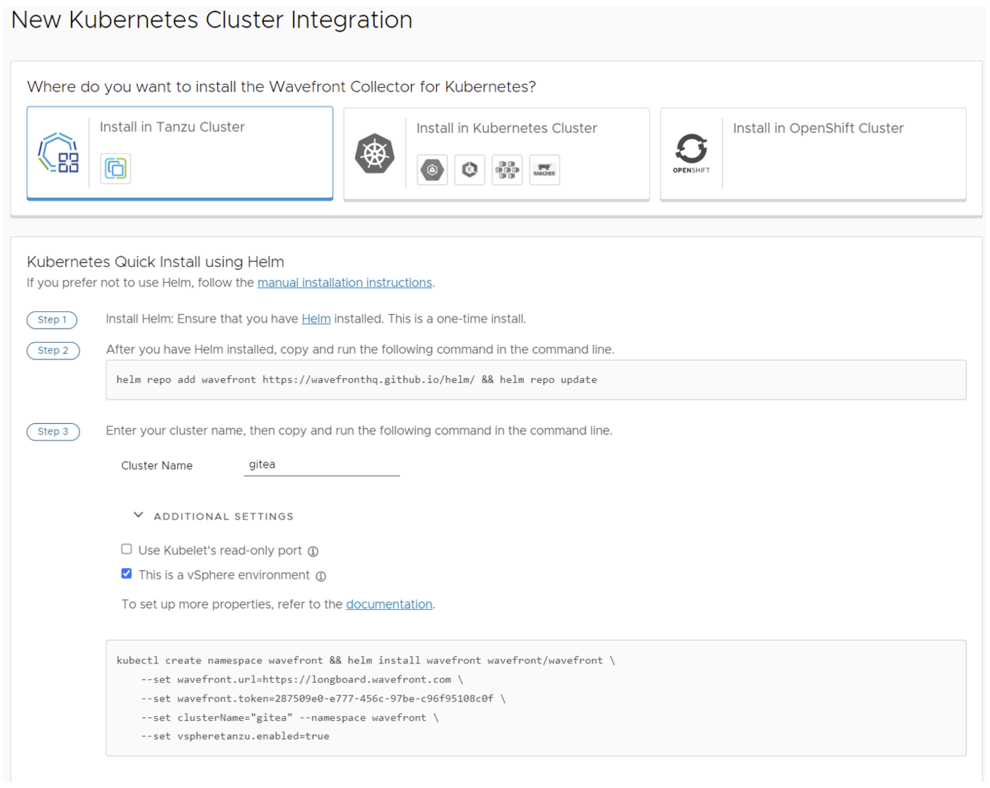


Figure 5.1: Tanzu Observability integration dashboards

Following the procedures shown in Figure 5.2, the Gitea cluster will be connected to Tanzu Observability, and the DevOps engineer will be able to create customized dashboards or select from the preconfigured integration dashboards, as shown in Figure 5.3.


New Kubernetes Cluster Integration

Where do you want to install the Wavefront Collector for Kubernetes?

Kubernetes Quick Install using Helm

If you prefer not to use Helm, follow the manual installation instructions.

Step 1 Install Helm: Ensure that you have Helm installed. This is a one-time install.

Step 2 After you have Helm installed, copy and run the following command in the command line.

```
helm repo add wavefront https://wavefronthq.github.io/helm/ && helm repo update
```

Step 3 Enter your cluster name, then copy and run the following command in the command line.

Cluster Name gitea

ADDITIONAL SETTINGS

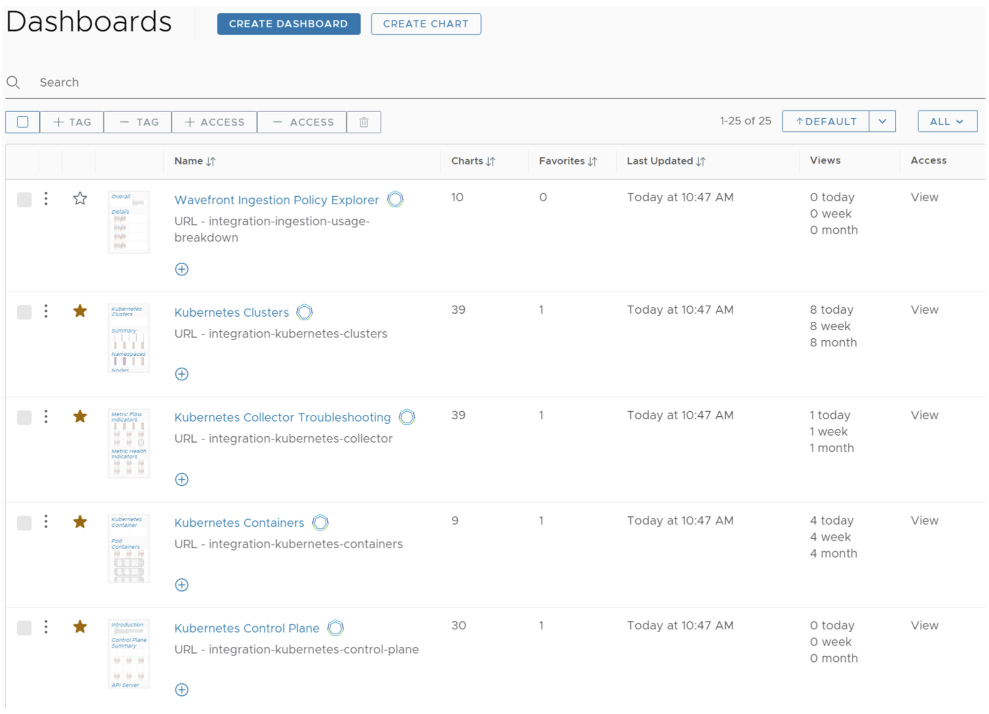
☐ Use Kubelet's read-only port

☑ This is a vSphere environment

To set up more properties, refer to the documentation.

```
kubectl create namespace wavefront && helm install wavefront wavefront/wavefront \
    --set wavefront.url=https://longboard.wavefront.com \
    --set wavefront.token=287509e0-e777-456c-97be-c96f95108c0f \
    --set clusterName="gitea" --namespace wavefront \
    --set vspheretanzu.enabled=true
```

Figure 5.2: The installation guide to integrate Tanzu Kubernetes clusters

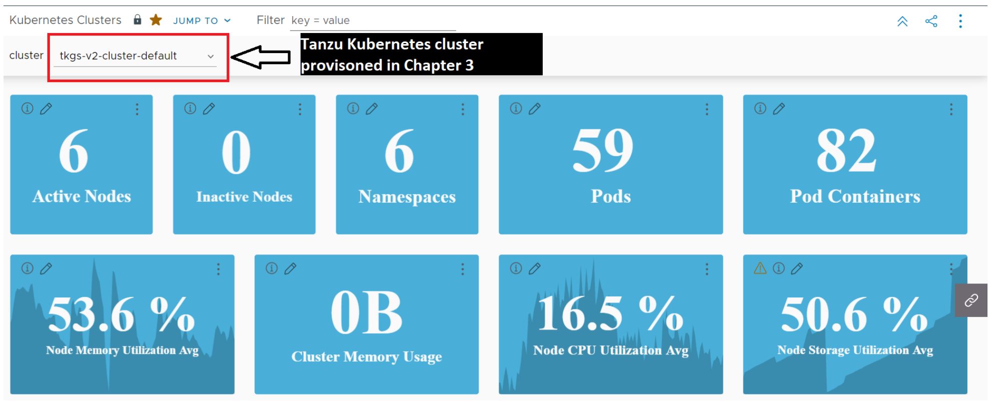


Name	Charts	Favorites	Last Updated	Views	Access
Wavefront Ingestion Policy Explorer URL - integration-ingestion-usage-breakdown	10	0	Today at 10:47 AM	0 today 0 week 0 month	View
Kubernetes Clusters URL - integration-kubernetes-clusters	39	1	Today at 10:47 AM	8 today 8 week 8 month	View
Kubernetes Collector Troubleshooting URL - integration-kubernetes-collector	39	1	Today at 10:47 AM	1 today 1 week 1 month	View
Kubernetes Containers URL - integration-kubernetes-containers	9	1	Today at 10:47 AM	4 today 4 week 4 month	View
Kubernetes Control Plane URL - integration-kubernetes-control-plane	30	1	Today at 10:47 AM	0 today 0 week 0 month	View

Figure 5.3: Tanzu Observability Dashboards

5.2 Monitoring with Tanzu Observability

Tanzu Observability by Wavefront can monitor the infrastructure associated with Kubernetes. This includes nodes, pods, and services. It can also monitor applications contained within the Kubernetes clusters deployed. Tanzu Observability does this by using Wavefront Collector for Kubernetes, which is an agent that runs on each node of the Kubernetes cluster. The Wavefront Collector for Kubernetes collects Kubernetes infrastructure metrics such as the state of the Kubernetes cluster and sends them to Wavefront. As shown in Figure 5.4, Tanzu Observability by Wavefront provides the DevOps team with complete visibility of a Kubernetes cluster, including pods, containers running, deployments, and namespaces. The figure shows the summary visualization nodes, pods, and namespaces within the Tanzu Kubernetes cluster that was provisioned in Chapter 3. It visualizes the average utilization of CPU, memory, and storage in the cluster.



cluster tkgs-v2-cluster-default

6 Active Nodes

0 Inactive Nodes

6 Namespaces

59 Pods

82 Pod Containers

53.6 % Node Memory Utilization Avg

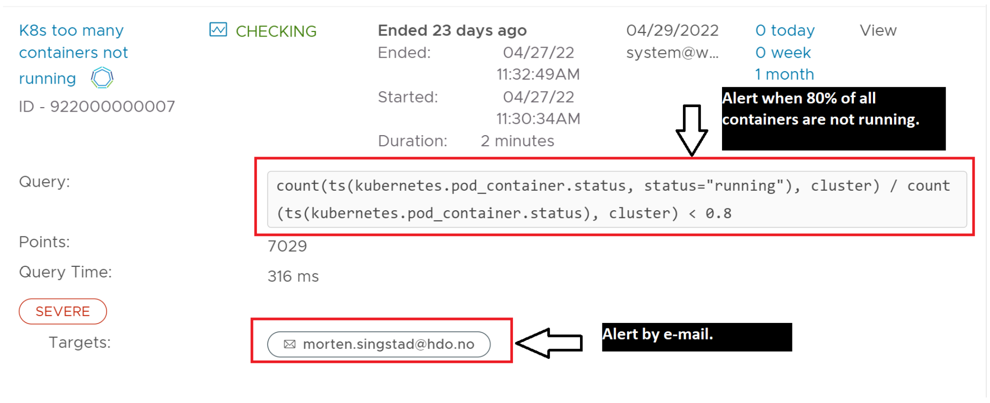
0B Cluster Memory Usage

16.5 % Node CPU Utilization Avg

50.6 % Node Storage Utilization Avg

Figure 5.4: Kubernetes cluster summary dashboard

Tanzu Observability offers alert functionality used to monitor metrics such as CPU and memory utilization averages that indicate system crashes or problems. For example, typical use cases for creating alerts are to notify an administrator when the CPU usage of a Kubernetes pod is too high, when too many containers are not running, or when too many pods have crashed. Figure 5.5 depicts an alert that sends an email to HDO's administrator when more than 20 percent of the pods are not running. This means at least 80 percent of pods should have the status "running" all the time.



K8s too many containers not running
ID - 922000000007

CHECKING

Ended 23 days ago
Ended: 04/27/22 11:32:49AM
Started: 04/27/22 11:30:34AM
Duration: 2 minutes

04/29/2022
system@w...

0 today
0 week
1 month

Query:

```
count(ts(kubernetes.pod_container.status, status="running"), cluster) / count
(ts(kubernetes.pod_container.status), cluster) < 0.8
```

Points: 7029

Query Time: 316 ms

SEVERE

Targets: morten.singstad@hdo.no

Figure 5.5: Alert in Tanzu Observability

Chapter 6

Security

Security has been an important part of this bachelor's thesis. The group has followed best practices for installing and configuring vSphere with Tanzu on NSX-T. This chapter provides an overview of relevant security features used to secure vSphere with Tanzu on NSX-T. This includes vSphere user management, vSphere encryption, Tanzu Kubernetes cluster security, and the NSX-T distributed firewall.

6.1 vSphere Security

The different components in the vSphere environment have security features like authentication, authorization, permission restriction, etc. that can be applied and modified immediately after they are exposed in use [46]. In the vSphere environment, objects are set up in a hierarchical order. This hierarchy inventory enables vSphere administrators to organize and track activities on objects. This helps in such a way that if an object is compromised then it can be contained from the environment. Securing such objects can be done using roles, permission, and global permissions. Therefore, each user in vSphere environment can be restricted to their necessary environment only.

6.1.1 vSphere Permissions and Management Task

Users in vSphere can use vCenter Single Sign-On[1] domain login to log into the vSphere environment [47]. This type of logging system allows vSphere administrators to assign other users or groups with different roles with the corresponding privileges and permissions to objects. These permissions are very critical since they not only limit access to resources in the vSphere environment but also, because the vSphere administrators have full access, these permissions can help the administrators identify failures in vCenter objects and track their sources from the user's perspective. vSphere has built-in roles with their privileges, for example, vSphere Administrator, ReadOnly, NSX Administrator, etc., or one can create

[1]**Single Sign-On:**is an authentication method that provide users the ability to securely log to their respective system, application, or other by using one set of credentials.

user-customized role.

Permissions provide limitations to users based on the predefined or customized roles they have in the system. For example, if a user is a content library administrator, then that means the user has full access to the content library inventory but not to other objects on the vCenter server. In this thesis, the administrator role, which gives full access to all objects in the environment, is used. This is due to the fact that installing vSphere with Tanzu requires administrator privileges.

6.1.2 Best Practices for Roles and Permissions

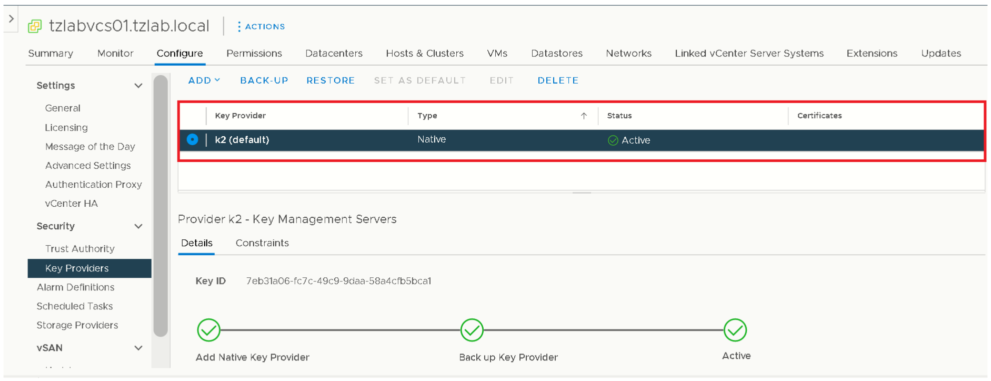
VMware has best practice of setting roles and permissions for customers to help them acquire the intended level of security. Following these best practices can help administrators in maximizing the security of the vSphere environment and its effectiveness [48]. Because of the fact that users or groups have access limited to the intended activities or assigned workflow, it helps to avoid, to a certain degree, human errors. This means if a user, for example, has "a virtual machine user" role, this role provides the user with the permission to perform virtual machine interactions like powering on and off VMs, installing VMware tools, etc. Therefore, this user has no way of making changes unintentionally or by other means, for example, to the hosts in the environment that can affect the operation of the environment. Hence, in this thesis, these best practices have been widely used to implement security solutions in the system.

6.1.3 vSphere VM Encryption and Key Management

In vSphere, VMs can be encrypted using the vSphere virtual machine encryption feature. Encrypting virtual machines also includes encrypting disk files and other files that are associated with the VMs. vSphere utilizes this encryption feature to provision secure Tanzu Kubernetes clusters and vSphere Pods. vSphere encryption works by establishing a trusted relationship between a Standard Key Provider and the vCenter. A Standard Key Provider, also known as a Key Management Server, is an external server that manages the creation, usage, and deletion of encryption keys [49]. The vCenter contacts a Standard Key Provider, which is external to the vCenter, and fetches encryption keys from the Standard Key Provider. The encryption keys are then distributed to the ESXi hosts, which handles the encryption of the virtual machines.

As an alternative to having an external Key Krovider, VMware offers a Standard Native Key Provider, a Key Provider integrated into vCenter but not configured by default. This removes the need to have an external Key Provider that generates encryption keys. As shown in Figure 6.1, a Standard Native Key Provider has been configured in this thesis. vSphere uses this Key Provider to provision secure Tanzu Kubernetes clusters and vSphere Pods. Standard Native Key Provider works by fetching a primary key called Key Derivation Key (KDK)[2] generated by the vCenter Server [50]. Every time a new VM is created, a Data Encryption Key[3] , is generated to encrypt the VM. This encryption key is encrypted by the KDK and stored in the VM.

The difference between Standard Key Provider and a Standard Native Key Provider is that ESXi host is the one that fetches key from the Key Provider. It is important to note that a Standard Native Key Provider is limited to VMware infrastructure unlike a Standard Key Provider that also works with other infrastructure types such as Amazon Web Services.



Key Provider	Type	Status	Certificates
k2 (default)	Native	Active	

Provider k2 - Key Management Servers

Key ID 7eb31a06-fc7c-49c9-9daa-58a4cfb5bca1

Figure 6.1: Standard Native Key Provider

6.2 Tanzu Kubernetes Security

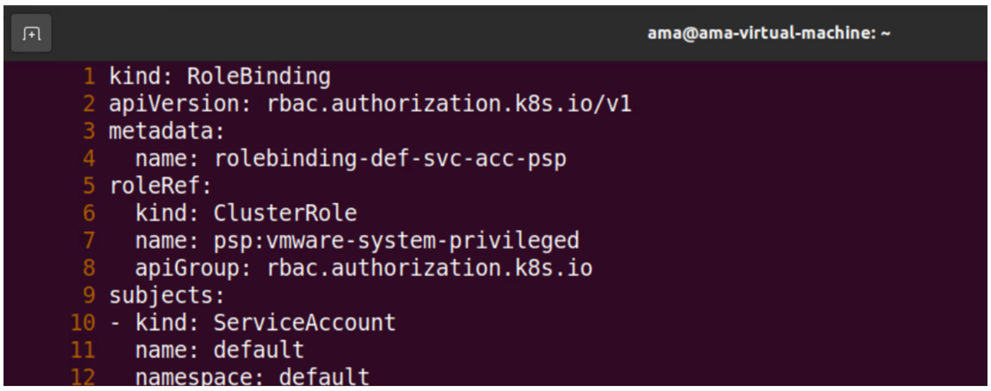
6.2.1 Tanzu Observability Implementation

Tanzu Kubernetes clusters (TKCs) are secured by default as a result of leveraging the Tanzu Kubernetes Grid Service (TKGS). The TKGS makes use of pre-packaged components to give Tanzu clusters solid security. By default, all TKGS-provisioned clusters will have a restricted PodSecurityPolicy (PSB) [51]. This means that the Tanzu Kubernetes Grid Service follows the principle of least privilege, which states that "*a subject should be given only those privileges needed for it to complete its task*" [52].

As a result, if a DevOps engineer attempts to deploy a workload (privileged or unprivileged pods), it will fail with a notice stating that the PSP prohibits the creation of pods. Therefore, role bindings to PSP should be created to grant permission for workload deployment. Role bindings, also known as role-based access control (RBAC), are classified into two types: ClusterRoleBinding, which is used to provide permissions across the whole cluster, and RoleBinding, which is used to

[2]A Key Derivation Key is a primary key used to encrypt Data Encryption Key.
[3]Data Encryption Key is a key used to encrypt and decrypt data.

provide permissions inside a certain namespace [53]. In this thesis, RoleBinding will be applied as shown in Figure 6.2.
Figure 6.2 shows the RoleBinding that will provide the default namespace (Line 12) permission to execute a privileged set of workloads. This is accomplished by binding the vmware-system-privileged PSP (Line 7) to the default ServiceAccount (Line 10). The default ServiceAccount is used by developers when they want to indirectly deploy workloads to clusters instead of using their own user accounts. Once this RoleBinding is applied, the developers can create and deploy pods in the default namespace.



```
 1 kind: RoleBinding
 2 apiVersion: rbac.authorization.k8s.io/v1
 3 metadata:
 4   name: rolebinding-def-svc-acc-psp
 5 roleRef:
 6   kind: ClusterRole
 7   name: psp:vmware-system-privileged
 8   apiGroup: rbac.authorization.k8s.io
 9 subjects:
10 - kind: ServiceAccount
11   name: default
12   namespace: default
```

Figure 6.2: RoleBinding that grants permissions within the default namespace

6.2.2 Authenticating with Tanzu Kubernetes Clusters

vSphere with Tanzu provides each different role, such as DevOps engineer and cluster administrator, with a different authentication mechanism [54]. These different mechanisms correspond to the two types of roles (Owner and Can Edit) in the vSphere namespace.
Owner role will be provided to DevOps engineers, allowing them to connect and authenticate to the **Supervisor Cluster** using the Kubectl-vSphere CLI plugin with vCenter Single Sign-On credentials.
Cluster administrators will be granted the Edit role, which allows them to connect and authenticate to the **Tanzu Kubernetes cluster** in one of two ways:

- Using the Kubectl-vSphere CLI plugin with vCenter Single Sign-On credentials generated for each administrator.

- If vCenter Single Sign-On authentication method failed, cluster administrators can get access to the Tanzu Kubernetes cluster using the Kubernetes-admin account.

6.2.3 Namespace Security

As mentioned in chapter 2, the vSphere administrator creates and configures the vSphere namespace with the resource limitations and user permissions. It is best practice to use various namespaces for each workload or team. As a result, different teams' resources will be isolated from one another [55].
Limiting CPU, memory, and storage consumption as necessary can also help to improve security. This is implemented in chapter 3, as shown in Figure 3.20. This resource constraint will limit the total number of resources allocated to the namespace, preventing resource starvation and boosting performance.

6.2.4 Network Policy

There are no restrictions on communication between pods inside a namespace in Kubernetes by default [56]. Implementing NetworkPolicy increases security by allowing the DevOps team to regulate which pods can communicate with each other. This is implemented in this thesis by utilizing the NSX-T distributed firewall, which will be explained in 6.3.

6.2.5 Container Image Security

It is best practice to proactively check container images for known vulnerabilities before deploying them to the production environment. Container image scanning may scan images for vulnerabilities found in the CVE database and help the DevOps team prevent them from being exploited [57]. vSphere with Tanzu provides an Embedded Harbor Registry that can store the images that the DevOps team deploys to Tanzu Kubernetes clusters [58]. The Embedded Harbor Registry can also be used as a vulnerability scanning tool to scan and detect container images with vulnerabilities. The Embedded Harbor Registry is not enabled and configured by default. As shown in Figure 6.3, this registry has been enabled and configured in this thesis.

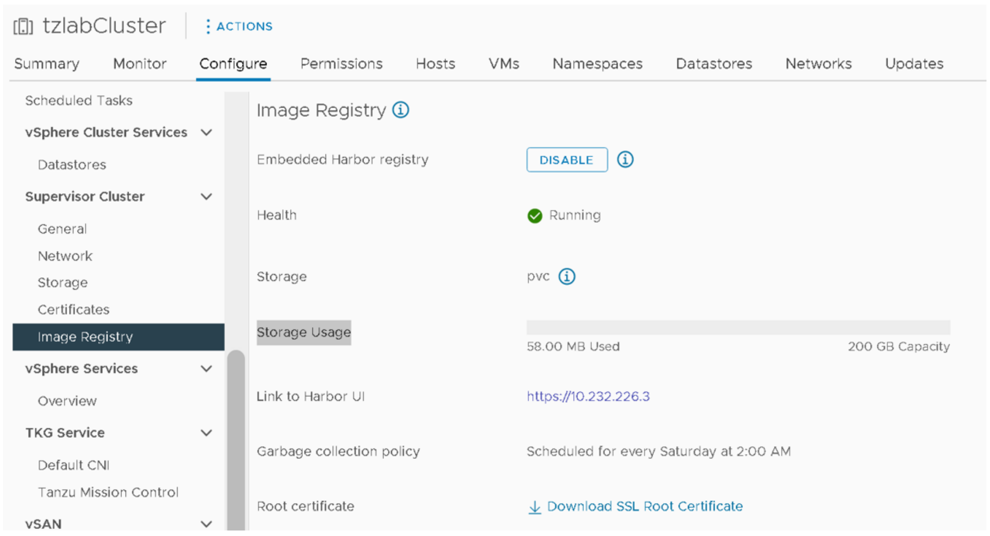


Image Registry

Embedded Harbor registry	DISABLE
Health	Running
Storage	pvc
Storage Usage	58.00 MB Used 200 GB Capacity
Link to Harbor UI	https://10.232.226.3
Garbage collection policy	Scheduled for every Saturday at 2:00 AM
Root certificate	Download SSL Root Certificate

Figure 6.3: The Embedded Harbor Registry

6.2.6 Kubernetes Runtime Security

Runtime security is the scanning activity that prevents threats from arising after containers have been deployed and run. Even though the images inside the container have been scanned and approved, they are still exposed to any vulnerabilities discovered inside the container. Runtime security tools are used to detect and prevent any malicious activity within the container. Aqua Security, Falco, and Sysdig are among the tools that can be used for runtime security scanning [56].

6.2.7 Monitoring and Alerts

Monitoring and alarms are not provided by default in Kubernetes. As a result, the DevOps team should install and configure tools that can support monitoring with automatically notify them when abnormal and unexpected events on the cluster are detected. As stated in the chapter 5, this security approach is implemented and explained in this thesis.

6.3 NSX-T security

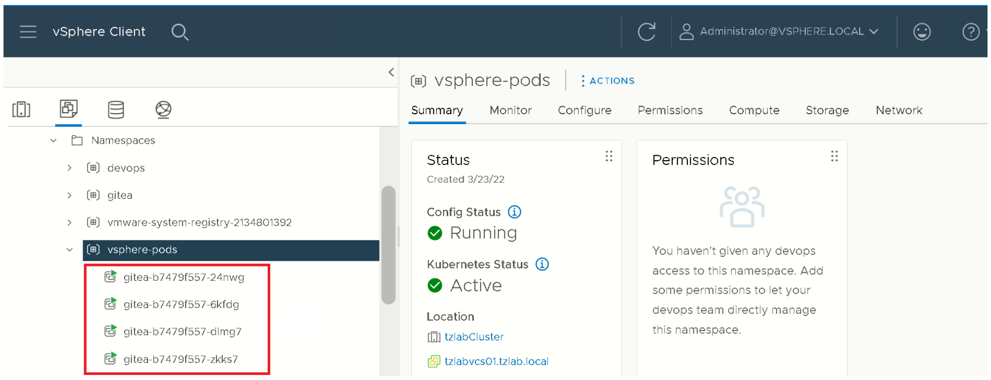
6.3.1 NSX-T distributed firewall

NSX-T offers a distributed firewall (DFW) to secure traffic across workloads. Once NSX-T is installed, all traffic between applications is blocked, and a DFW needs to be configured to allow traffic. The NSX-T distributed firewall offers Micro-segmentation to segment all the components in the network into logical units and provides a security perimeter around them to prevent lateral movement of attacks. This is achieved by DFW, which distributes firewalling to hosts in the vSphere environment to monitor traffic on VMs. Micro-segmentation is used by DFW to apply security policies and rules to VMs, IP addresses, port groups, clusters, and logical switches. In this thesis, a distributed firewall is created using a YAML configuration file for defining network policy.

6.3.2 Configuring NSX-T Distributed Firewall

By default, micro-segmentation in the NSX-T DFW is turned off. DFW uses blacklist as a connectivity strategy by default, meaning all traffic is allowed unless any rule that denies a specific traffic is specified [59]. To configure a DFW, micro-segmentation needs to be turned on and a security group to add security rules needs to be created. Security groups are objects in the vSphere environment such as VMs, IP addresses, and MAC addresses that can be created within a group and have security policies applied to them. Security groups can be created statically or dynamically and used as the source and destination of a firewall rule [60].

Attempts were first made to configure DFW for the Gitea application deployed in the Tanzu Kubernetes cluster. A network policy was first created for the Gitea application, but it was not visible as a DFW in NSX-T. An alternative solution that the group came up with was to deploy Gitea using vSphere Pods instead of a Tanzu Kubernetes cluster to test how DFW works. The background for the choice of this option is that vSphere Pods provide strong security isolation and can run containers without the need to customize a Tanzu Kubernetes cluster [61]. Figure 6.4 shows the Gitea application deployed using vSphere Pods. The same YAML configuration file in Figure 4.1 is used to deploy Gitea vSphere Pods.

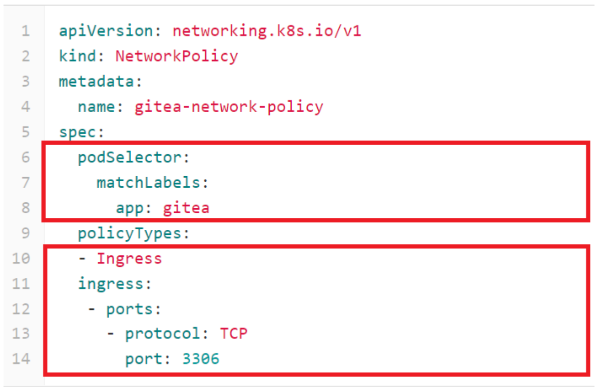


Status
Created 3/23/22
Config Status: Running
Kubernetes Status: Active
Location: tzlabCluster, tzlabvcs01.tzlab.local

You haven't given any devops access to this namespace. Add some permissions to let your devops team directly manage this namespace.

Figure 6.4: vSphere Pods

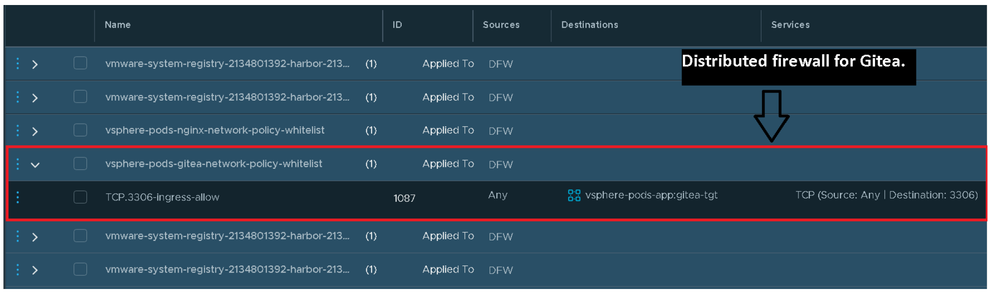
Distributed firewall rules have been created in NSX-T for the Gitea application deployed using vSphere Pods to isolate and control what traffic is allowed. Figure 6.5 shows YAML configuration file used to created network policy for the Gitea application. This network policy is automatically deployed as a DFW rule by NSX-T. The "podSelector" section in line 7 uses matching labels to select which pods to apply the network policy rule to. The network policy rule is only applied to vSphere Pods that are labelled with the name "gitea" (line 8). This network policy opens TCP service on port 3306 to allow ingress, meaning incoming traffic on port 3306, which is the port used by the Gitea application (line 13 and 14).



```
 1  apiVersion: networking.k8s.io/v1
 2  kind: NetworkPolicy
 3  metadata:
 4    name: gitea-network-policy
 5  spec:
 6    podSelector:
 7      matchLabels:
 8        app: gitea
 9    policyTypes:
10    - Ingress
11    ingress:
12     - ports:
13       - protocol: TCP
14         port: 3306
```

Figure 6.5: Network policy

After applying the YAML configuration file, a new distributed firewall rule gets created in NSX-T as shown in Figure 6.6. This new distributed firewall includes a security group that is dynamically created after applying the YAML file. The IP addresses of all the Gitea pods are added as members to the security group. This distributed firewall defines the source (any) and destination (port 3306) of the traffic and which service to open for the application.



Name	ID	Sources	Destinations	Services
vmware-system-registry-2134801392-harbor-213... (1)	Applied To	DFW		
vmware-system-registry-2134801392-harbor-213... (1)	Applied To	DFW		
vsphere-pods-nginx-network-policy-whitelist (1)	Applied To	DFW		
vsphere-pods-gitea-network-policy-whitelist (1)	Applied To	DFW		
TCP.3306-ingress-allow	1087	Any	vsphere-pods-app:gitea-tgt	TCP (Source: Any | Destination: 3306)
vmware-system-registry-2134801392-harbor-213... (1)	Applied To	DFW		
vmware-system-registry-2134801392-harbor-213... (1)	Applied To	DFW		

Figure 6.6: Distributed firewall

Chapter 7

Discussion and Conclusion

7.1 Result

The project group has achieved all the goals for this thesis. Tanzu has been successfully installed and configured for vSphere and NSX-T. It has been demonstrated how an application deployed using the Tanzu Kubernetes cluster can be patched, scaled(manually and automatically), and monitored. Following HDO's wishes, firewall rules for the self-elected application have been configured and demonstrated in this thesis. The findings of this thesis have been that Tanzu is a great solution for running and managing containerized applications without the need to change the underlying infrastructure. Using Tanzu to manage Kubernetes clusters automates the security and delivery of containerized applications. It also improves the performance of containerized applications since Kubernetes workloads can run directly on ESXi hosts.

Running Kubernetes workloads using vSphere with Tanzu can be done using vSphere Pods or Tanzu Kubernetes clusters. Another finding in this thesis was that using vSphere Pods provides better security isolation since vSphere Pods are directly deployed as VMs on ESXi. The group discovered that vSphere Pods provide better visibility since vSphere Pods are vSphere objects and are visible in NSX-T, allowing vSphere administrators to control these objects by creating NSX-T distributed firewall rules for each one. There is also no need to manage the lifecycle of a Kubernetes cluster when using vSphere Pods. The drawback with vSphere Pods is that there is no root access, as one has with a Tanzu Kubernetes cluster. An advantage of Tanzu Kubernetes clusters is that it allows HDO to have customized Kubernetes clusters and have root-level access control. Tanzu Kubernetes cluster also fully complies with upstream Kubernetes. For these reasons, the group recommends HDO use vSphere Pods to run workloads when strong security isolation is required and use the Tanzu Kubernetes cluster when it is desired to use a customized Kubernetes cluster.

7.2 Evaluation of technology selection and further work

HDO has restricted the technologies or solutions that could be used in this thesis. Despite the fact that the project group was restricted to using vSphere as a cloud computing virtualization platform, NSX-T as a network virtualization and security platform, and the Tanzu Basic edition, there was some flexibility in picking some of the technological solutions, or even as suggestions/recommendations for future work.

Tanzu Observability vs Prometheus

HDO gave the project group freedom to conduct their own research to determine the optimal monitoring solution. After researching various monitoring solutions [62–65] , the choice was between Prometheus and Tanzu Observability and the project group chose the Tanzu Observability based on the following factors:

- Prometheus is an open-source system that provides monitoring and alerting for small and medium-sized enterprises, whereas Tanzu Observability by Wavefront can provide monitoring and alerting for large businesses.
- Wiring up the first cluster in Prometheus is straightforward, but it gets difficult when the DevOps team wants to wire up several clusters since each one requires its own version of Prometheus as well as its own user management, dashboards, alarms, and so on. Tanzu Observability, on the other hand, ensures that even if the DevOps team wires up many clusters, the metrics from all of them simply end up in the same place, making monitoring administration much easier for the DevOps team.
- Because Prometheus operates on the same cluster that is being monitored, the Prometheus pods will use part of the cluster memory, which might result in out-of-memory events as Prometheus pods grow. Tanzu Observability, on the other hand, runs outside the network of the cluster and the metrics are streamed out, by the Wavefront proxy, to high–availability servers, where they are gathered and shown on dashboards. Consequently, Tanzu Observability does not cause cluster load by consuming large amounts of memory.
- Tanzu Observability's alerts are closely integrated, allowing the DevOps team to set up and integrate with other alert notification systems such as PagerDuty, VictorOps, and Slack without the need to write YAML rule files. On the other hand, creating alerting rules in Prometheus requires using the Prometheus expression language to define alert conditions and writing these rules in a YAML file, which will take some time to learn.
- Tanzu Observability offers more than 220 pre-packaged integrations.

Tanzu Mission Control

Before beginning with application management in chapter 4, the group got advice from HDO's contact person to investigate installing and using Tanzu Mission Control. The group requested a trial version of Tanzu Mission Control to learn and discover whatever features it provides, but they were unable to install or use it because there is no trial version available at this time.
"VMware Tanzu® Mission Control™ is a centralized management hub with a unified policy engine that simplifies multi-cloud and multi-cluster Kubernetes management across stakeholder teams within the enterprise" [66].
Based on some research [66–68], the group discovered that using Tanzu Mission Control is an excellent option for HDO for the following reasons:

- Tanzu Mission Control is a Kubernetes management solution that allows the DevOps team to effortlessly control and manage the Kubernetes clusters/-namespaces and their objects.
- It can be integrated with Tanzu Observability and other third-party services. By integrating Tanzu Mission Control with Tanzu Observability, the DevOps team will be able to easily discover and resolve issues by seeing the health of Kubernetes clusters and their objects, as supplied by Tanzu Mission Control.
- It enables the DevOps team to easily create and apply policies to a single cluster or groups of clusters, allowing for consistent management of clusters and workloads.
- Tanzu Mission Control has a built-in security policy that increases control and security of Kubernetes clusters.
- It also has a built-in data-protection features that simplifies the backup and recovery of cluster/namespace resources.

Tanzu Ops Manager

Before beginning manually patching and updating the Gitea application in chapter 4, the project group performed some research to see if there were any better automated solutions. Based on research [69], the group found that using Tanzu Ops Manager is a suitable automated solution, but it could not be installed or used in this thesis because it is incompatible with the standard version of Tanzu Grid Service that the group has worked with since the beginning; instead, it requires a license for VMware Tanzu Kubernetes Grid Integrated Edition (TKGI) [70].
The group believe that HDO can benefit from utilizing the Tanzu Ops Manager for the following reasons:

- The Tanzu Ops Manager helps the DevOps team to manage the Kubernetes application deployment and upgrade with just a few clicks.
- It also enables the DevOps team to deploy security patches and the Kubernetes application automatically, rapidly, effortlessly, and with zero downtime.

7.3 Group Work

The group has had good collaboration throughout the project work. In the beginning, most of the work was configuration and setup of the required components. This phase of the work required all the group members to sit down together and configure the vSphere with Tanzu on NSX-T. Moreover, the group had a weekly status meeting where they divided the load of work that should be done and discussed the things that had already been done the previous week. Before the meeting, each group member goes through the work done by the others and gives feedback. During the meeting, they discussed the feedback that had been given by each member and what it mean. Eventually, settle any disagreement based on what they mean and what is best for the report and divide the next assignment. The project group has also had regular meetings every Thursday with the supervisor. They present the status and get feedback and recommendations from the supervisor, which has been of great help to the group.

7.4 Learning Process

Working on this thesis has been inspiring, with both ups and downs. Through the project, new knowledge has been gained in several fields, such as virtualization and networking. In addition to gaining new knowledge, the group has put what they have learnt in the study program, such as virtualization, containers, networking, and security, to good use. The group has gained a good insight into various technologies such as Kubernetes and several VMware products such as vSphere, Tanzu, and NSX-T. The project itself has been demanding with a sharp learning curve, and there has been a lot of joy in succeeding with all the foreseen and unforeseen challenges that arose during the work. The group has also gained more experience in working systematically on a larger project.

7.5 Conclusion

This thesis aimed to investigate the possibility of using vSphere with Tanzu on NSX-T to run, orchestrate, and administer Kubernetes clusters. HDO's request was to install and configure vSphere with Tanzu on NSX-T. To meet this requirement, the group has followed best practices to install and configure Tanzu. All the different steps taken to install and configure are broadly discussed in this thesis. To evaluate the use of Tanzu, a self-selected interactive application has been deployed after configuring and installing vSphere with Tanzu on NSX-T. Setting up a monitoring system was another requirement by HDO. Tanzu Observability, which is a monitoring and visualization platform, has been implemented. Because security was an important part of this thesis, best practices for protecting and securing vSphere with Tanzu on NSX-T were also followed.

Bibliography

[1] VMware. "What is micro-segmentation?" (2022), [Online]. Available: https://www.vmware.com/topics/glossary/content/micro-segmentation.html. (accessed: 13.05.2022).

[2] HDO. "Om oss." (2021), [Online]. Available: https://www.hdo.no/om-oss. (accessed: 22.02.2022).

[3] VMware. "Vmware tanzu." (2022), [Online]. Available: https://tanzu.vmware.com/tanzu. (accessed: 14.05.2022).

[4] C. Richardson. "What are microservices?" (2021), [Online]. Available: https://microservices.io/. (accessed: 22.02.2022).

[5] Opensource. "What is virtualization." (2021), [Online]. Available: https://opensource.com/resources/virtualization. (accessed: 07.05.2022).

[6] IBM. "Containerization." (2021), [Online]. Available: https://www.ibm.com/cloud/learn/containerization. (accessed: 07.05.2022).

[7] VMware. "A new chapter in vmware: Spin-off from dell technologies completed." (2021), [Online]. Available: https://news.vmware.com/releases/vmware-announces-completion-of-spin-off-from-dell-technologies. (accessed: 17.02.2022).

[8] VMware. "Vmware vsphere documentation." (2021), [Online]. Available: https://docs.vmware.com/en/VMware-vSphere/index.html. (accessed: 25.02.2022).

[9] VMware. "Virtual machine storage policies." (2019), [Online]. Available: https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.storage.doc/GUID-A8BA9141-31F1-4555-A554-4B5B04D75E54.html. (accessed: 08.04.2022).

[10] VMware. "Storage classes." (2019), [Online]. Available: https://kubernetes.io/docs/concepts/storage/storage-classes/. (accessed: 08.04.2022).

[11] VMware. "Using content libraries." (2021), [Online]. Available: https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.vm_admin.doc/GUID-254B2CE8-20A8-43F0-90E8-3F6776C2C896.html. (accessed: 08.04.2022).

[12] Gigamon. “What’s all the fuss about vmware nsx-t, and why does gigamon care?” (2019), [Online]. Available: https://blog.gigamon.com/2019/08/21/whats-all-the-fuss-about-vmware-nsx-t-and-why-does-gigamon-care/. (accessed: 18.04.2022).

[13] VMware. “Add a distributed logical router.” (2019), [Online]. Available: https://docs.vmware.com/en/VMware-NSX-Data-Center-for-vSphere/6.4/com.vmware.nsx.install.doc/GUID-E825C0C7-F4CC-4B26-90AF-A2167AC519DB.html. (accessed: 18.04.2022).

[14] VMware. “Install the nsx manager virtual appliance.” (2019), [Online]. Available: https://docs.vmware.com/en/VMware-NSX-Data-Center-for-vSphere/6.4/com.vmware.nsx.install.doc/GUID-CFB0DC96-C329-490E-B2A9-D92C5704E853.html. (accessed: 25.03.2022).

[15] VMware. “Key concepts.” (2020), [Online]. Available: https://docs.vmware.com/en/VMware-NSX-T-Data-Center/3.1/installation/GUID-A1BBC650-CCE3-4AC3-A774-92B195183492.html. (accessed: 20.04.2022).

[16] VMware. “Add a compute manager.” (2019), [Online]. Available: https://docs.vmware.com/en/VMware-NSX-T-Data-Center/2.5/administration/GUID-D225CAFC-04D4-44A7-9A09-7C365AAFCA0E.html?hWord=N4IghgNiBc4CZwARkQYwPYFsAOBXAL
QA. (accessed: 25.03.2022).

[17] VMware. “What is a kubernetes cluster?” (2022), [Online]. Available: https://www.vmware.com/topics/glossary/content/kubernetes-cluster.html. (accessed: 23.02.2022).

[18] J. SPALETA. “How kubernetes works.” (2019), [Online]. Available: https://sensu.io/blog/how-kubernetes-works. (accessed: 16.05.2022).

[19] Kubernetes. “Pods?” (2021), [Online]. Available: https://kubernetes.io/docs/concepts/workloads/pods/. (accessed: 23.02.2022).

[20] VMware. “What are kubernetes services?” (2022), [Online]. Available: https://www.vmware.com/topics/glossary/content/kubernetes-services.html. (accessed: 23.02.2022).

[21] VMware. “What is a kubernetes deployment?” (2021), [Online]. Available: https://www.vmware.com/topics/glossary/content/kubernetes-deployment.html. (accessed: 23.02.2022).

[22] Kubernetes. “Replicaset.” (2022), [Online]. Available: https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/. (accessed: 23.02.2022).

[23] Kubernetes. “Ingress.” (2022), [Online]. Available: https://kubernetes.io/docs/concepts/services-networking/ingress/. (accessed: 23.02.2022).

[24] VMware. “Vmware tanzu documentation.” (2022), [Online]. Available: https://docs.vmware.com/en/VMware-Tanzu/index.html. (accessed: 23.02.2022).

[25] VMware. “What is vsphere with tanzu?” (2022), [Online]. Available: `https://docs.vmware.com/en/VMware-vSphere/7.0/vmware-vsphere-with-tanzu/GUID-70CAF0BB-1722-4526-9CE7-D5C92C15D7D0.html`. (accessed: 23.02.2022).

[26] VMware. “Vsphere with tanzu architecture.” (2021), [Online]. Available: `https://docs.vmware.com/en/VMware-vSphere/7.0/vmware-vsphere-with-tanzu/GUID-3E4E6039-BD24-4C40-8575-5AA0EECBBBEC.html`. (accessed: 23.02.2022).

[27] VMware. “What is a vsphere pod?” (2021), [Online]. Available: `https://docs.vmware.com/en/VMware-vSphere/7.0/vmware-vsphere-with-tanzu/GUID-276F809D-2015-4FC6-92D8-8539D491815E.html`. (accessed: 27.02.2022).

[28] M. Foley. “Vsphere 7 – introduction to the vsphere pod service.” (2020), [Online]. Available: `https://blogs.vmware.com/vsphere/2020/04/vsphere-7-vsphere-pod-service.html`. (accessed: 23.02.2022).

[29] VMware. “Create and configure a vsphere namespace.” (2022), [Online]. Available: `https://docs.vmware.com/en/VMware-vSphere/7.0/vmware-vsphere-with-tanzu/GUID-177C23C4-ED81-4ADD-89A2-61654C18201B.html`. (accessed: 19.02.2022).

[30] VMware. “What is tanzu kubernetes cluster?” (2021), [Online]. Available: `https://docs.vmware.com/en/VMware-vSphere/7.0/vmware-vsphere-with-tanzu/GUID-DC22EA6A-E086-4CFE-A7DA-2654891F5A12.html`. (accessed: 19.02.2022).

[31] VMware. “Vmware tanzu kubernetes grid documentation.” (2022), [Online]. Available: `https://docs.vmware.com/en/VMware-Tanzu-Kubernetes-Grid/index.html`. (accessed: 23.02.2022).

[32] VMware. “Install and configure nsx-t data center for vsphere with tanzu.” (2021), [Online]. Available: `https://docs.vmware.com/en/VMware-vSphere/7.0/vmware-vsphere-with-tanzu/GUID-8D0E905F-9ABB-4CFB-A206-C027F847FAAC.html`. (accessed: 19.02.2022).

[33] A. Tripathi. “What is a vsphere pod?” (2020), [Online]. Available: `https://vmtechie.blog/2020/10/19/load-balancer-as-a-service-with-cloud-director/`. (accessed: 17.05.2022).

[34] VMware. “Customer connect.” (2022), [Online]. Available: `https://customerconnect.vmware.com/`. (accessed: 23.04.2022).

[35] A. networks. “Service engine.” (2022), [Online]. Available: `https://avinetworks.com/glossary/service-engine/`. (accessed: 23.04.2022).

[36] VMware. “Nsx-t routing for the management domain in a single-region sddc.” (2021), [Online]. Available: https://docs.vmware.com/en/VMware-Validated-Design/6.2/sddc-architecture-and-design-for-the-management-domain/GUID-1CE27536-E934-4B99-AA0A-4D7B48A55D72.html. (accessed: 17.05.2022).

[37] VMware. “Deploying and managing virtual machines in vsphere with tanzu.” (2021), [Online]. Available: https://docs.vmware.com/en/VMware-vSphere/7.0/vmware-vsphere-with-tanzu/GUID-F81E3535-C275-4DDE-B35F-CE759EA3B4A0.html#GUID-F81E3535-C275-4DDE-B35F-CE759EA3B4A0. (accessed: 07.04.2022).

[38] VMware. “Default virtual machine classes.” (2021), [Online]. Available: https://docs.vmware.com/en/VMware-vSphere/7.0/vmware-vsphere-with-tanzu/GUID-7351EEFF-4EF0-468F-A19B-6CEA40983D3D.html. (accessed: 18.04.2022).

[39] T. G. Authors. “Gitea - git with a cup of tea.” (2022), [Online]. Available: https://gitea.io/en-us/. (accessed: 23.04.2022).

[40] Kubernetes. “Persistent volumes.” (2022), [Online]. Available: https://kubernetes.io/docs/concepts/storage/persistent-volumes/#access-modes. (accessed: 07.04.2022).

[41] Kubernetes. “Horizontal pod autoscaling.” (2022), [Online]. Available: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/. (accessed: 26.04.2022).

[42] Kubernetes. “Horizontalpodautoscaler walkthrough.” (2022), [Online]. Available: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/. (accessed: 26.04.2022).

[43] VMware. “Monitoring kubernetes.” (2022), [Online]. Available: https://tanzu.vmware.com/developer/guides/monitoring-at-scale-wavefront/. (accessed: 27.04.2022).

[44] VMware. “What is tanzu observability by wavefront?” (2022), [Online]. Available: https://docs.wavefront.com/wavefront_introduction.html. (accessed: 27.04.2022).

[45] VMware. “Wavefront query language reference.” (2022), [Online]. Available: https://docs.wavefront.com/query_language_reference.html. (accessed: 27.04.2022).

[46] VMware. “Security in the vsphere environment.” (2019), [Online]. Available: https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.security.doc/GUID-9E10799B-0F6E-47D5-A59B-983E271EC8C0.html. (accessed: 05.05.2022).

[47] VMware. "Vsphere permissions and user management tasks." (2022), [Online]. Available: https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.security.doc/GUID-5372F580-5C23-4E9C-8A4E-EF1B4DD9033E.html. (accessed: 06.05.2022).

[48] VMware. "Best practices for roles and permissions." (2019), [Online]. Available: https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.security.doc/GUID-FAA074CC-E8C9-4F13-ABCF-6CF7F15F04EE.html. (accessed: 06.05.2022).

[49] T. Security. "What is vmware encryption?" (2022), [Online]. Available: https://info.townsendsecurity.com/vmware-encryption-key-management-definitive-guide#deploying-vmware-encryption-key-management. (accessed: 15.05.2022).

[50] VMware. "What is vsphere native key provider." (2022), [Online]. Available: https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.security.doc/GUID-54B9FBA2-FDB1-400B-A6AE-81BF3AC9DF97.html. (accessed: 15.05.2022).

[51] VMware. "Vsphere with tanzu security." (2021), [Online]. Available: https://docs.vmware.com/en/VMware-vSphere/7.0/vmware-vsphere-with-tanzu/GUID-A60A132D-F993-427C-B6BA-D7F60C87FD02.html. (accessed: 12.05.2022).

[52] M. G. S. Barnum. "Least privilege." (2013), [Online]. Available: https://www.cisa.gov/uscert/bsi/articles/knowledge/principles/least-privilege. (accessed: 12.05.2022).

[53] VMware. "Using pod security policies with tanzu kubernetes clusters." (2021), [Online]. Available: https://docs.vmware.com/en/VMware-vSphere/7.0/vmware-vsphere-with-tanzu/GUID-CD033D1D-BAD2-41C4-A46F-647A560BAEAB.html#GUID-CD033D1D-BAD2-41C4-A46F-647A560BAEAB. (accessed: 12.05.2022).

[54] VMware. "Authenticating with tanzu kubernetes clusters." (2021), [Online]. Available: https://docs.vmware.com/en/VMware-vSphere/7.0/vmware-vsphere-with-tanzu/GUID-F00BCB9F-1AE7-4D8E-A87F-076E6733E096.html. (accessed: 13.05.2022).

[55] Sysdig. "Kubernetes security 101: Fundamentals and best practices." (2022), [Online]. Available: https://sysdig.com/learn-cloud-native/kubernetes-security/kubernetes-security-101/. (accessed: 13.05.2022).

[56] VMware. "Platform security." (2022), [Online]. Available: https://tanzu.vmware.com/developer/guides/platform-security/#node-and-container-runtime-hardening. (accessed: 13.05.2022).

[57] VMware. "Managing and securing container images in a registry." (2022), [Online]. Available: https://tanzu.vmware.com/developer/guides/managing-container-images-registry/. (accessed: 13.05.2022).

[58] VMware. “Use the embedded harbor registry with tanzu kubernetes clusters.” (2021), [Online]. Available: https://docs.vmware.com/en/VMware-vSphere/7.0/vmware-vsphere-with-tanzu/GUID-EC79A6DE-477A-40C1-A90C-9DF96465CDA6.html. (accessed: 13.05.2022).

[59] VMware. “How to enable whitelist strategy in distributed firewall.” (2021), [Online]. Available: https://kb.vmware.com/s/article/83326. (accessed: 13.05.2022).

[60] VMware. “Add a group.” (2022), [Online]. Available: https://docs.vmware.com/en/VMware-NSX-T-Data-Center/3.2/administration/GUID-9DFF6EE2-2E00-4097-A412-B72472596E4D.html#GUID-9DFF6EE2-2E00-4097-A412-B72472596E4DD. (accessed: 13.05.2022).

[61] VMware. “What is a vsphere pod?” (2021), [Online]. Available: https://docs.vmware.com/en/VMware-vSphere/7.0/vmware-vsphere-with-tanzu/GUID-276F809D-2015-4FC6-92D8-8539D491815E.html. (accessed: 19.05.2022).

[62] Grafana. “Step-by-step guide to setting up prometheus alertmanager with slack, pagerduty, and gmail.” (2020), [Online]. Available: https://grafana.com/blog/2020/02/25/step-by-step-guide-to-setting-up-prometheus-alertmanager-with-slack-pagerduty-and-gmail/. (accessed: 15.05.2022).

[63] Wavefront. “Create and manage custom alert targets.” (2022), [Online]. Available: https://docs.wavefront.com/webhooks_alert_notification.html. (accessed: 15.05.2022).

[64] R. FISHER. “Prometheus or tanzu observability by wavefront for kubernetes? an sre’s point of view.” (2020), [Online]. Available: https://tanzu.vmware.com/content/blog/prometheus-or-tanzu-observability-by-wavefront-for-kubernetes-an-sre-s-point-of-view-2. (accessed: 15.05.2022).

[65] Wavefront. “Integrations overview.” (2022), [Online]. Available: https://docs.wavefront.com/integrations.html. (accessed: 15.05.2022).

[66] VMware. “Vmware tanzu mission control solution brief.” (2022), [Online]. Available: https://d1fto35gcfffzn.cloudfront.net/tanzu/TMC-Solution-Brief-1.5.22.pdf. (accessed: 15.05.2022).

[67] VMware. “Vmware tanzu mission control.” (2022), [Online]. Available: https://tanzu.vmware.com/mission-control. (accessed: 15.05.2022).

[68] Kunal. “Introduction to vmware tanzu mission control.” (2022), [Online]. Available: https://kb.vmtestdrive.com/hc/en-us/articles/360044177194-Introduction-to-VMware-Tanzu-Mission-Control. (accessed: 15.05.2022).

[69] VMware. “Vmware tanzu ops manager.” (2022), [Online]. Available: https://tanzu.vmware.com/components/ops-manager. (accessed: 15.05.2022).

[70] VMware. “Vmware tanzu kubernetes grid integrated edition documentation.” (2022), [Online]. Available: `https://docs.vmware.com/en/VMware-Tanzu-Kubernetes-Grid-Integrated-Edition/index.html`. (accessed: 15.05.2022).

Additional Material

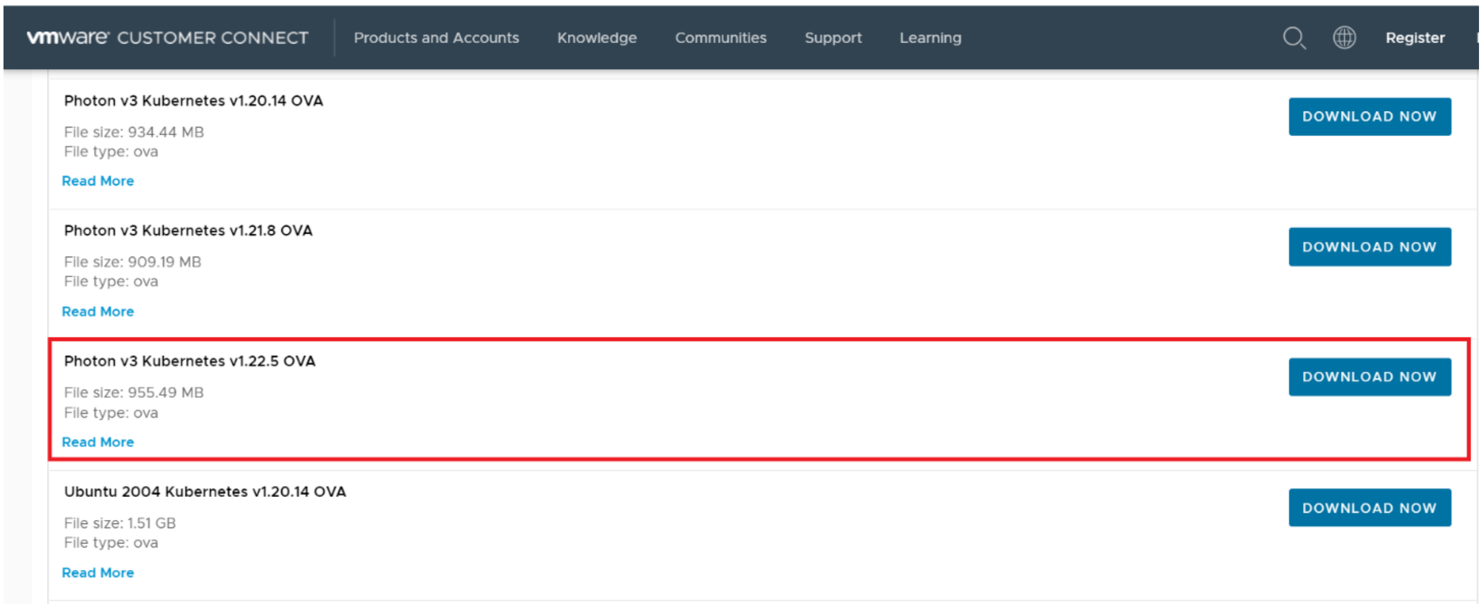
Documentation

Content

NSX-T Configuration

Configure NSX Advanced Load Balancer for vSphere with Tanzu

- Download OVA template for the Avi controller from https://customerconnect.vmware.com/en/downloads/details?downloadGroup=TKG-152&productId=988&rPId=86183



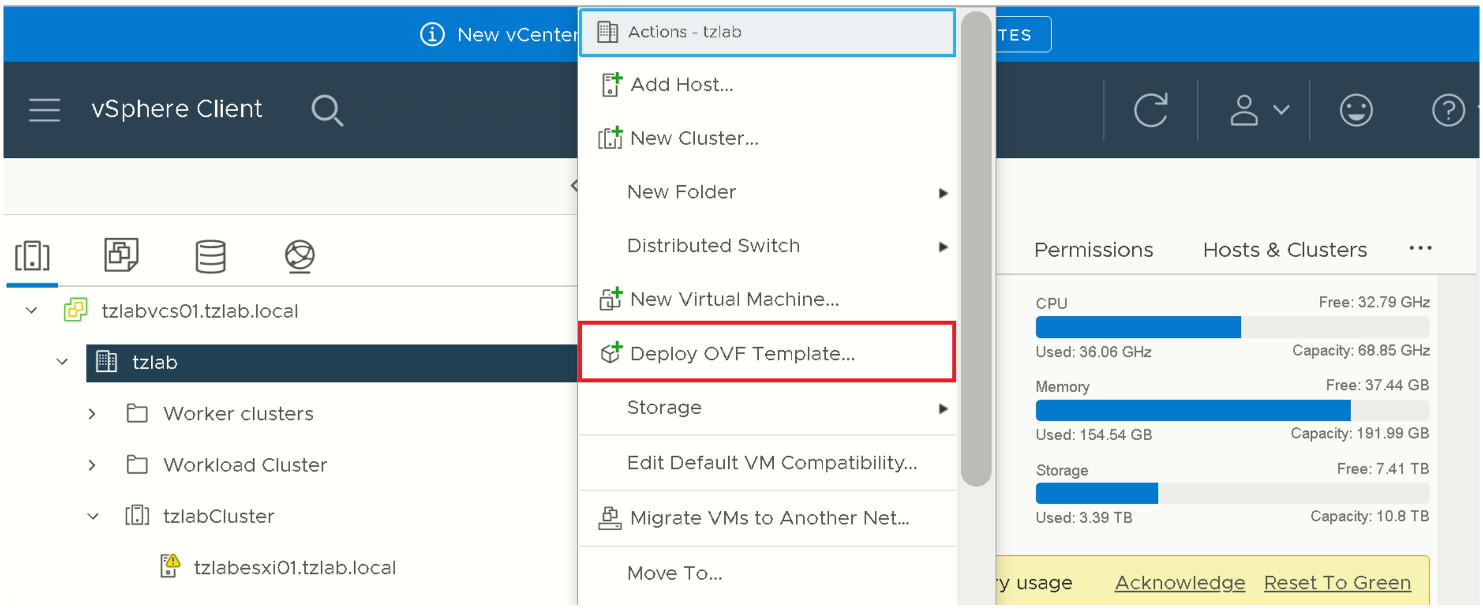
Photon v3 Kubernetes v1.20.14 OVA
File size: 934.44 MB
File type: ova
Read More
DOWNLOAD NOW

Photon v3 Kubernetes v1.21.8 OVA
File size: 909.19 MB
File type: ova
Read More
DOWNLOAD NOW

Photon v3 Kubernetes v1.22.5 OVA
File size: 955.49 MB
File type: ova
Read More
DOWNLOAD NOW

Ubuntu 2004 Kubernetes v1.20.14 OVA
File size: 1.51 GB
File type: ova
Read More
DOWNLOAD NOW

- Deploy the OVA template on the vSphere cluster: Right click the vSphere cluster → Upload the OVA template → select the **cluster** as resource pool → Select **VM policy** as Storage policy → Select the Management network → Enter **IP address for the controller** and set the **default Gateway**.

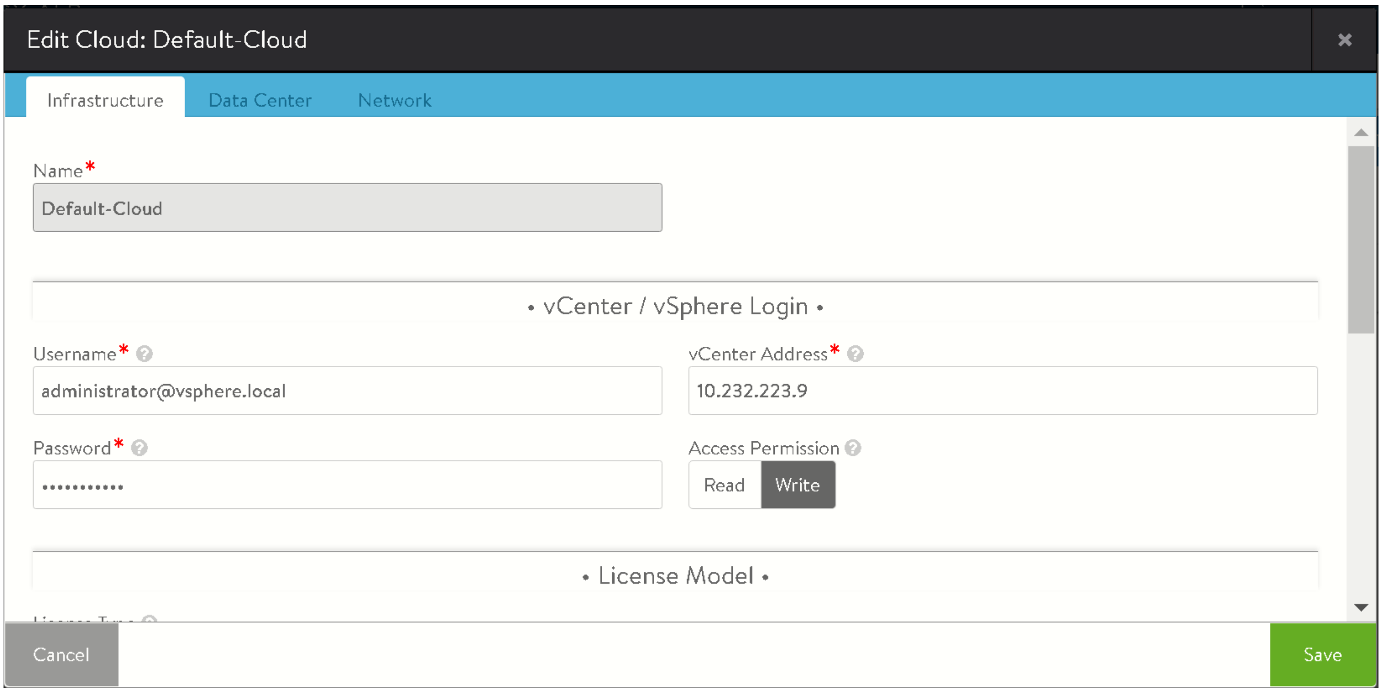


- Power on the controller

- Access the controller using the IP address assigned to the controll. Specify name and password for the administrator account.

- Enter the IP address of the DNS and NTP Server.

- Use the IP address the controller got assigned to access the dashboard and connect the controller to the vCenter by clicking **Infrastructure → Clouds → edit the Default-Cloud** and filling in the vCenter log in credentials.


Edit Cloud: Default-Cloud

Infrastructure | Data Center | Network

Name*: Default-Cloud

• vCenter / vSphere Login •

Username*: administrator@vsphere.local

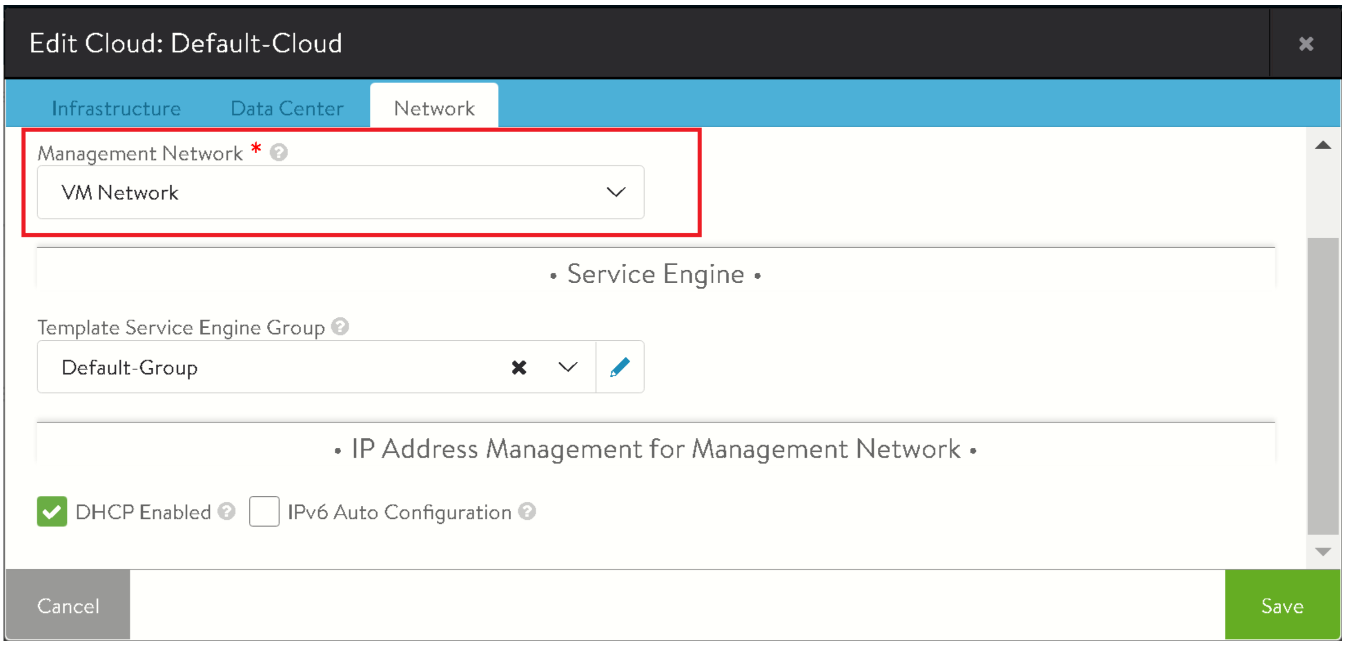
vCenter Address*: 10.232.223.9

Password*: •••••••••••

Access Permission: Read [✓ Write]

• License Model •

- Select the management network. **Infrastructure → Clouds → edit the Default-Cloud → Network → select "VM Network"** which is the management network.


Edit Cloud: Default-Cloud

Infrastructure | Data Center | Network

Management Network*: VM Network

• Service Engine •

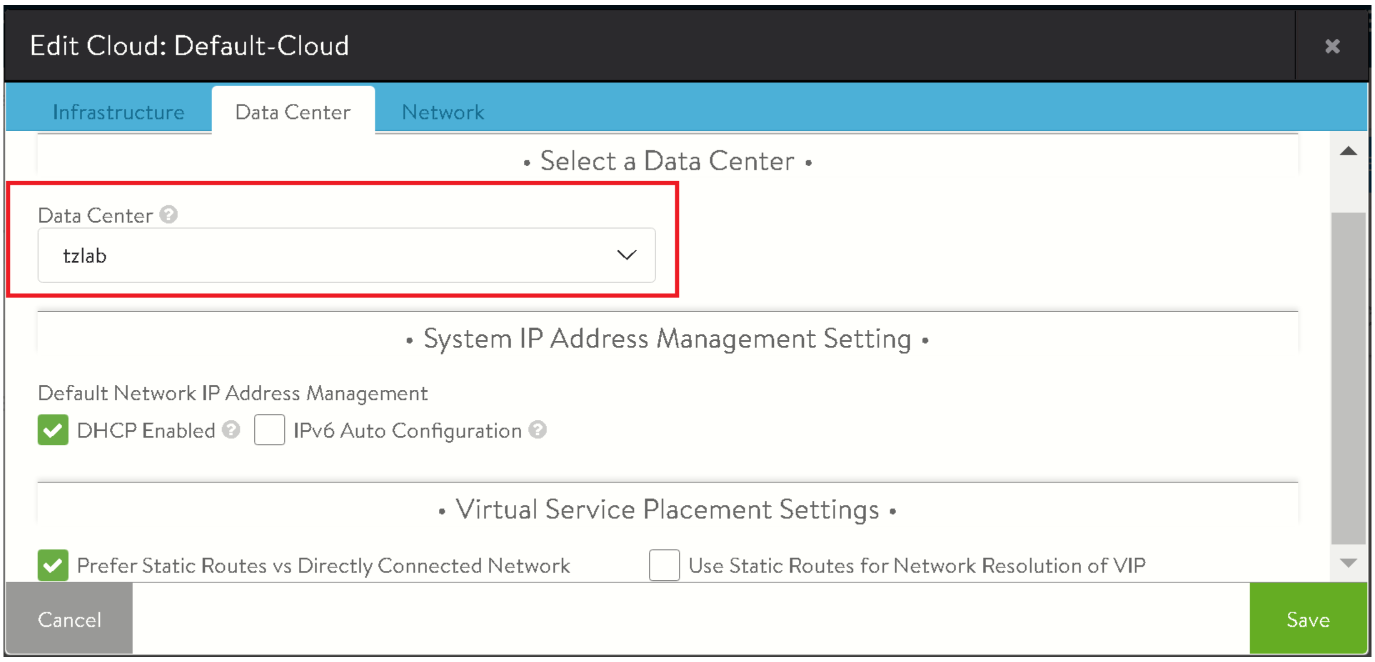
Template Service Engine Group: Default-Group

• IP Address Management for Management Network •

DHCP Enabled [✓]

IPv6 Auto Configuration [ ]

- Connect the controller to the desired Data Center. **Infrastructure → Clouds →edit the Default-Cloud → Data Center → select “tzlab”.**


Edit Cloud: Default-Cloud

Infrastructure | Data Center | Network

• Select a Data Center •

Data Center: tzlab

• System IP Address Management Setting •

Default Network IP Address Management

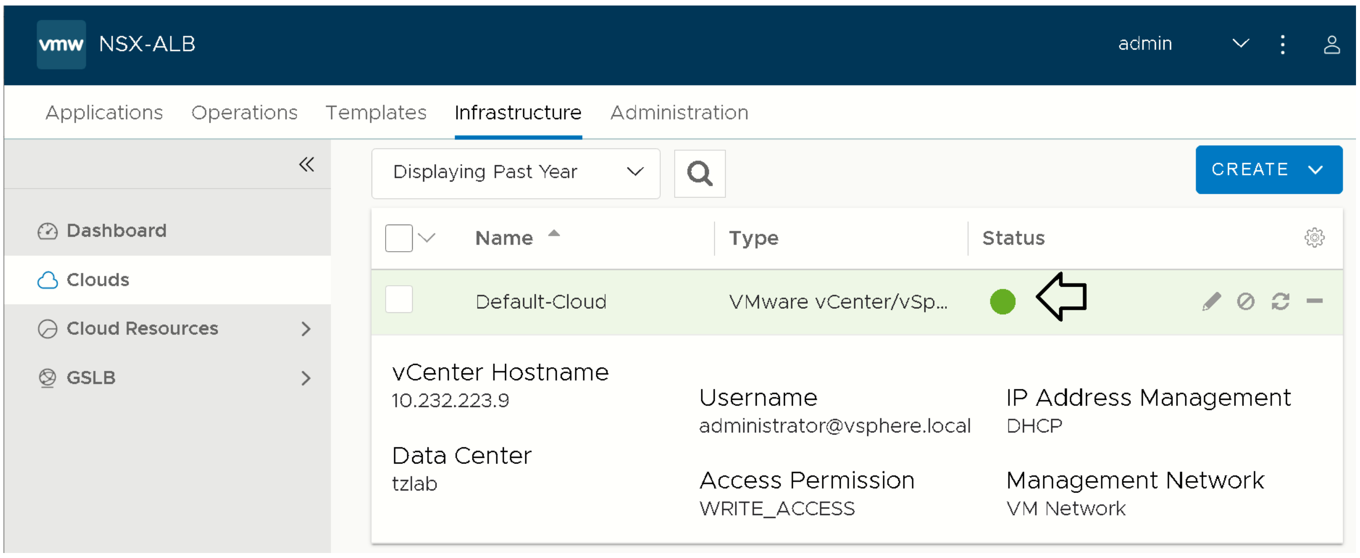
✓ DHCP Enabled ☐ IPv6 Auto Configuration

• Virtual Service Placement Settings •

✓ Prefer Static Routes vs Directly Connected Network ☐ Use Static Routes for Network Resolution of VIP

Cancel | Save

- Verify the status of the controller after the configuration is complete. **Dashboard → Infrastructure → Clouds.** Status should be **green**.



Name	Type	Status
Default-Cloud	VMware vCenter/vSp...	●

vCenter Hostname: 10.232.223.9

Username: administrator@vsphere.local

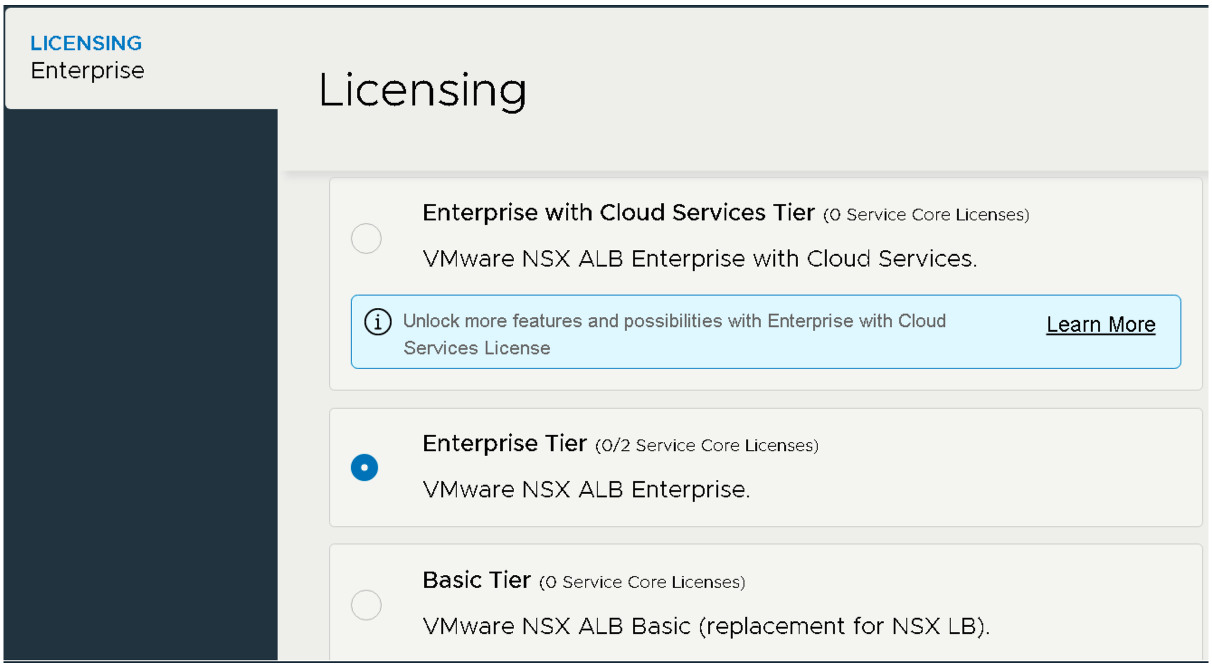
IP Address Management: DHCP

Data Center: tzlab

Access Permission: WRITE_ACCESS

Management Network: VM Network

- Add a License to the controller. **Administration → Settings → licencing**


LICENSING
Enterprise

Licensing

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

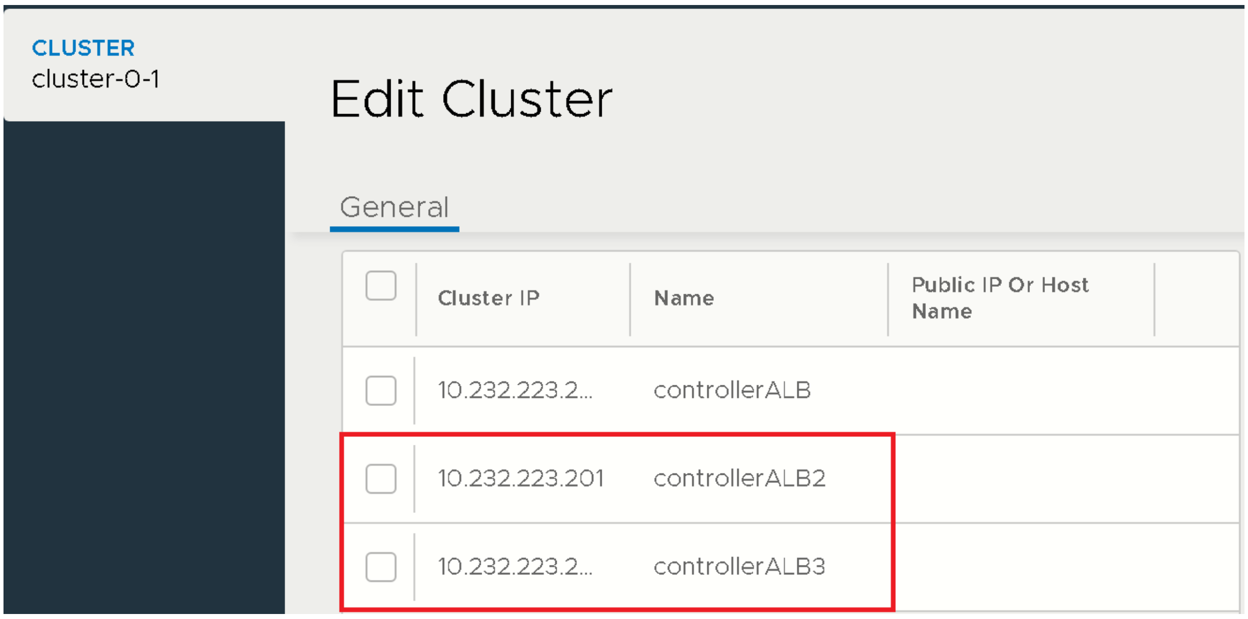
Unlock more features and possibilities with Enterprise with Cloud Services License

Learn More

[✓ Enterprise Tier (0/2 Service Core Licenses)]
VMware NSX ALB Enterprise.

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

- Deploy to more controllers to form a cluster of three controllers. This is to ensure redundancy and high availability. The two new controllers are deployed using the same OVA file used to deploy the first controller. The new controllers are added as node to form a cluster. This is done by clicking **Administration → Controller → Nodes → Edit → Add IP addresses of the two new controllers**



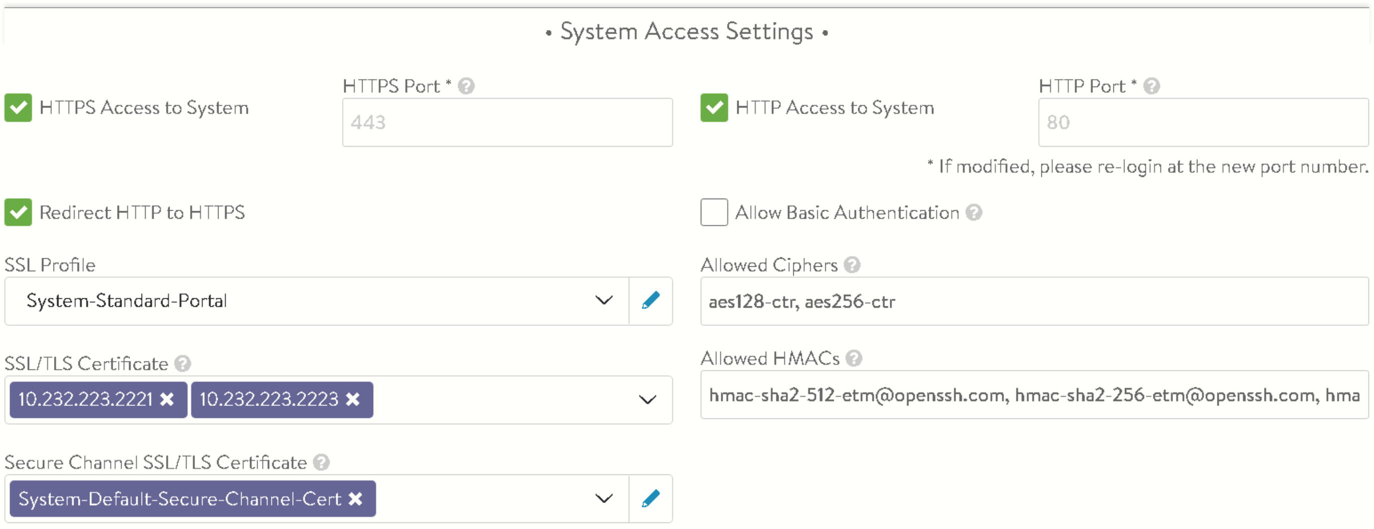
CLUSTER
cluster-0-1

Edit Cluster

General

	Cluster IP	Name	Public IP Or Host Name
☐	10.232.223.2...	controllerALB	
☐	10.232.223.201	controllerALB2	
☐	10.232.223.2...	controllerALB3	

- Add a Certificate to the controller, in the AVI dashboard go to **Templates → Security -> SSL/TLS Certificate → create → Controller Certificate.** A new certificate window appears, then enter name for the certificate and select " self-Signed".
  Source: https://docs.vmware.com/en/VMware-vSphere/7.0/vmware-vsphere-with-tanzu/GUID-9435390C-E04C-43E7-B87F-910453AED797.html



• System Access Settings •

✓ HTTPS Access to System
HTTPS Port *: 443
✓ HTTP Access to System
HTTP Port *: 80
* If modified, please re-login at the new port number.

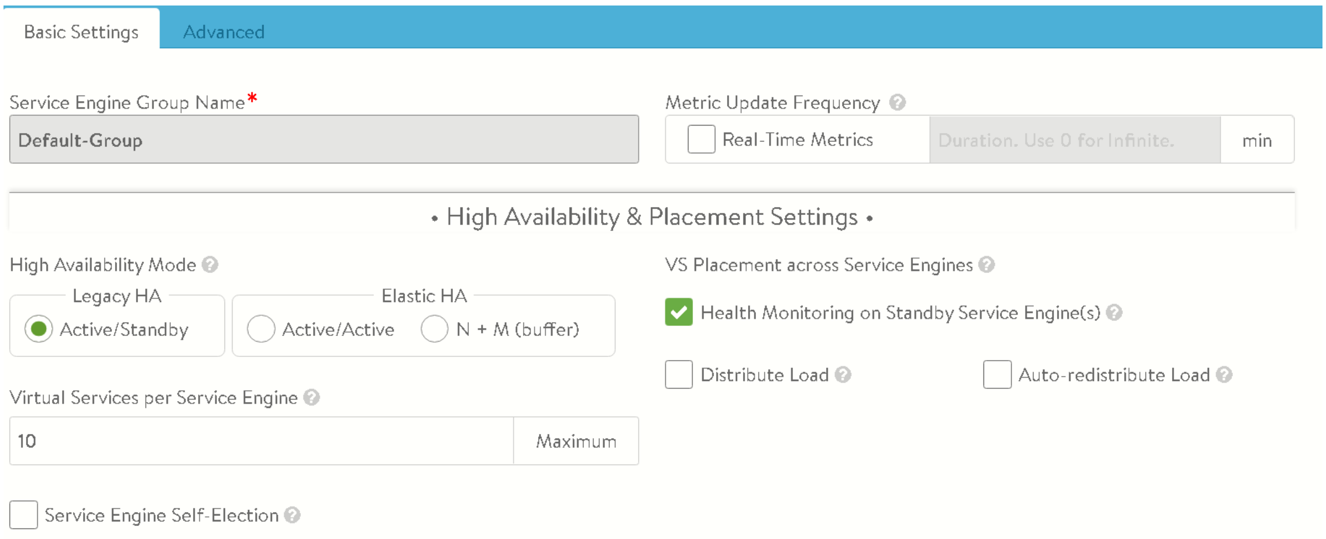
✓ Redirect HTTP to HTTPS
☐ Allow Basic Authentication

SSL Profile: System-Standard-Portal
Allowed Ciphers: aes128-ctr, aes256-ctr

SSL/TLS Certificate: 10.232.223.2221 ✖ 10.232.223.2223 ✖
Allowed HMACs: hmac-sha2-512-etm@openssh.com, hmac-sha2-256-etm@openssh.com, hma

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

- Configure Service Engine Group **Infrastructure → Cloud Resources → Service Engine**


Basic Settings | Advanced

Service Engine Group Name*: Default-Group

Metric Update Frequency: [ ] Real-Time Metrics — Duration. Use 0 for Infinite. min

• High Availability & Placement Settings •

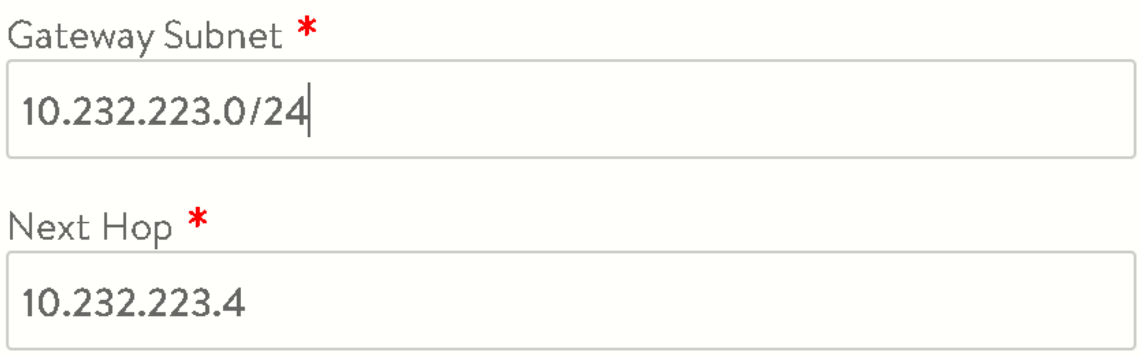
High Availability Mode
- Legacy HA: [● Active/Standby]
- Elastic HA: ○ Active/Active ○ N + M (buffer)

VS Placement across Service Engines
- [✓] Health Monitoring on Standby Service Engine(s)
- [ ] Distribute Load
- [ ] Auto-redistribute Load

Virtual Services per Service Engine: 10 Maximum

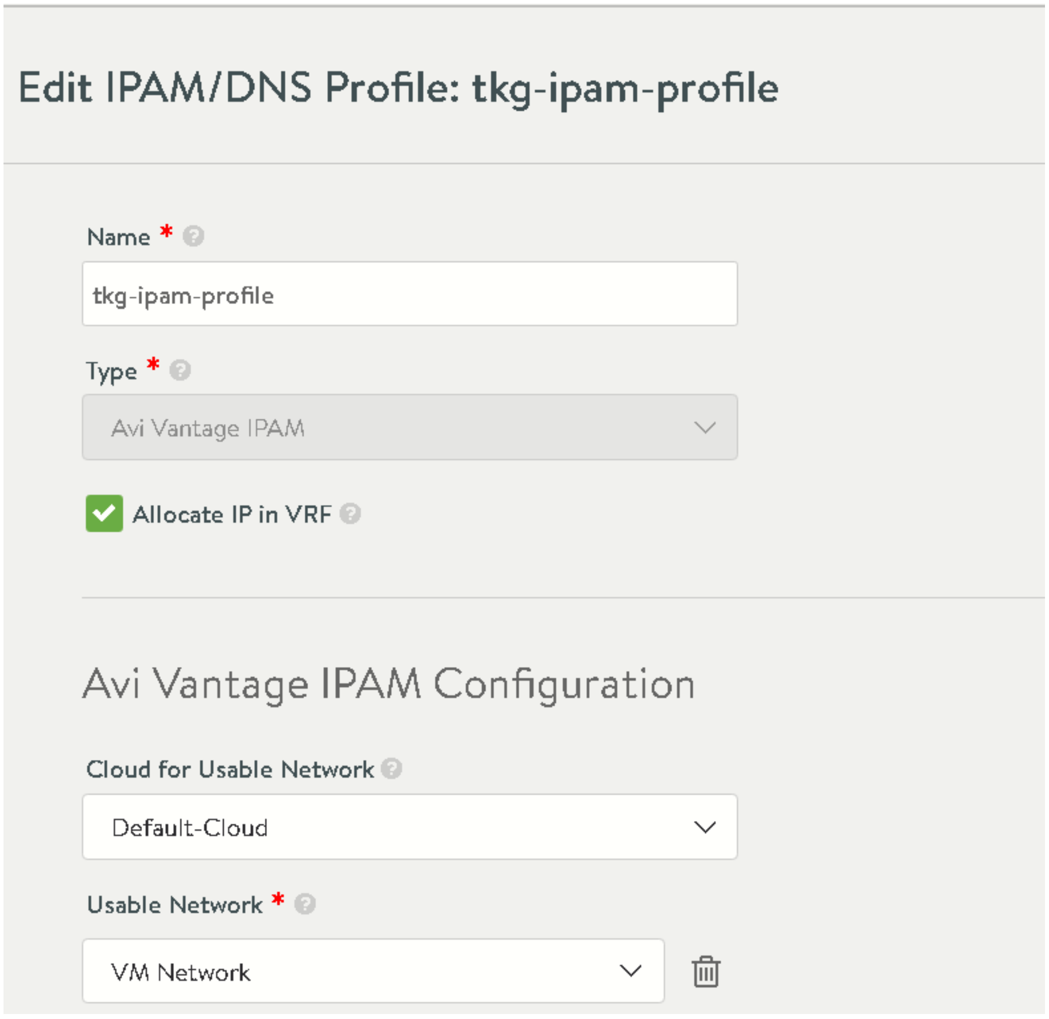
[ ] Service Engine Self-Election

- Configure default gateway:
  Add a static route by clicking **Infrastructure → Routing → Edit "Default-Cloud.**


Gateway Subnet *: 10.232.223.0/24

Next Hop *: 10.232.223.4

- Configure IPAM profile: **Templates → profiles → IPAM/DNS**. This is to tell which IP pool to use. IPAM profile is connected to the vCenter:



Edit IPAM/DNS Profile: tkg-ipam-profile

Name *: tkg-ipam-profile

Type *: Avi Vantage IPAM

✓ Allocate IP in VRF

Avi Vantage IPAM Configuration

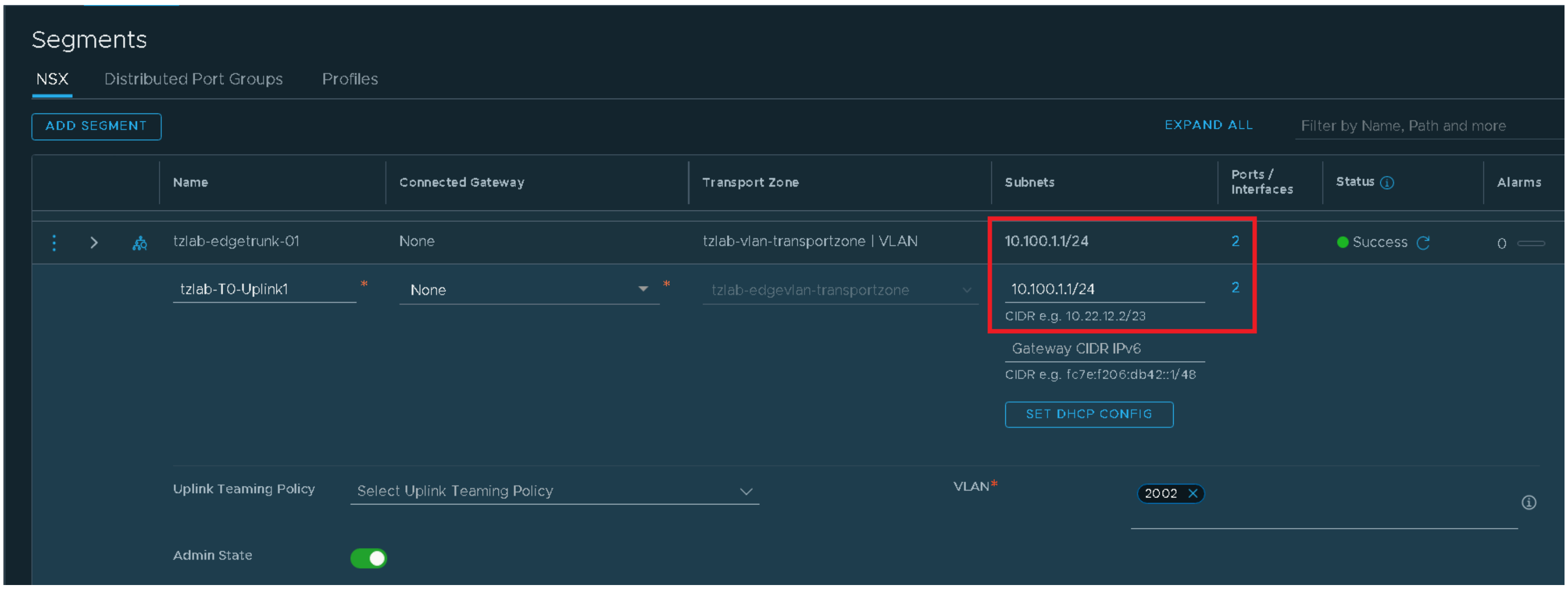
Cloud for Usable Network: Default-Cloud

Usable Network *: VM Network

Network Connection Configuration

- Add IP address pools and create subnet and set the subnet 10.100.1.1/24 for **the tzlab-TO-Uplink1 segment**:

In the NSX Manager: click **Networking → Segments → tzlab-edgetrunk-01 → edit**



Segments

NSX | Distributed Port Groups | Profiles

ADD SEGMENT | EXPAND ALL | Filter by Name, Path and more

Name	Connected Gateway	Transport Zone	Subnets	Ports / Interfaces	Status	Alarms
tzlab-edgetrunk-01	None	tzlab-vlan-transportzone | VLAN	10.100.1.1/24	2	● Success	0
tzlab-T0-Uplink1	None	tzlab-edgevlan-transportzone	10.100.1.1/24	2		

CIDR e.g. 10.22.12.2/23

Gateway CIDR IPv6

CIDR e.g. fc7e:f206:db42::1/48

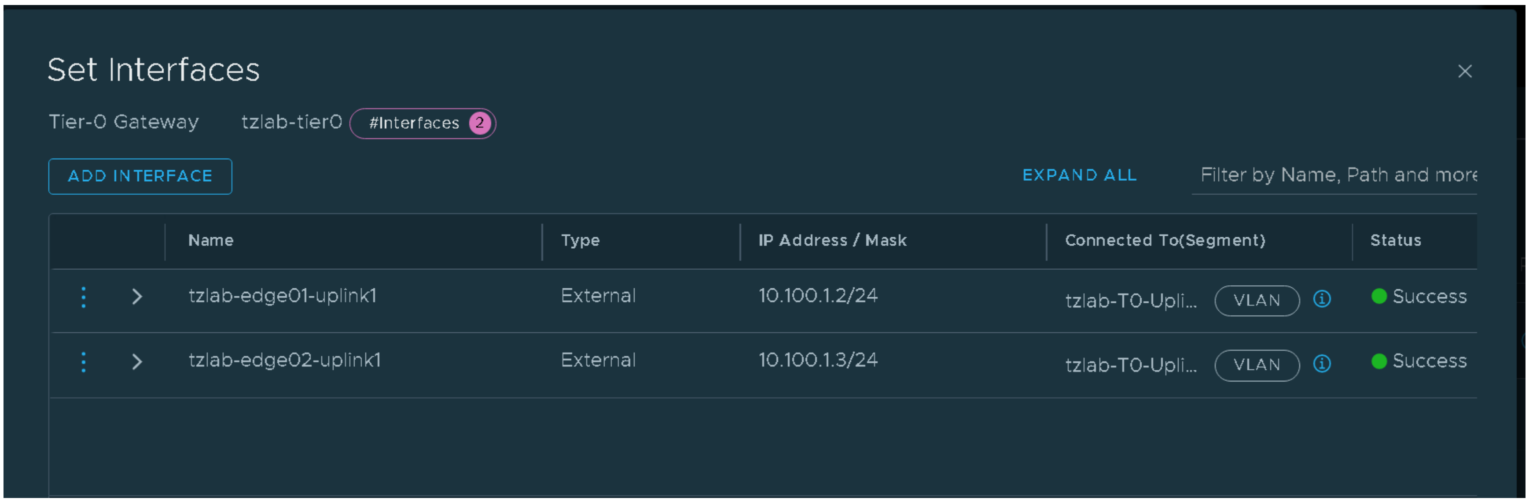
SET DHCP CONFIG

Uplink Teaming Policy: Select Uplink Teaming Policy

VLAN*: 2002

Admin State: on

- Set "Their-0 Gateways" interfaces. In the NSX Manager Click **Networking → Tzlab-tier → edit → click interfaces**



Set Interfaces

Tier-0 Gateway tzlab-tier0 #Interfaces 2

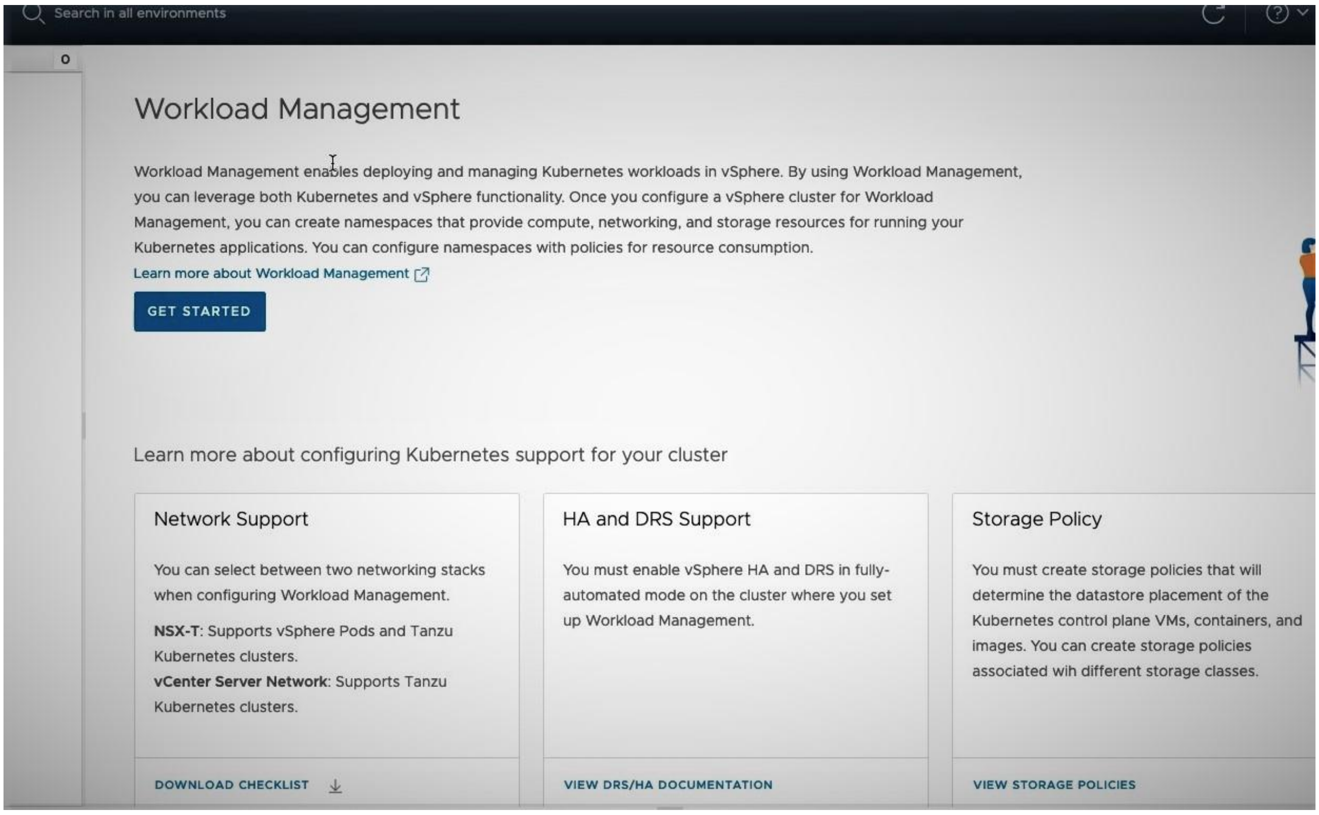
ADD INTERFACE | EXPAND ALL | Filter by Name, Path and more

Name	Type	IP Address / Mask	Connected To(Segment)	Status
tzlab-edge01-uplink1	External	10.100.1.2/24	tzlab-T0-Upli... VLAN	● Success
tzlab-edge02-uplink1	External	10.100.1.3/24	tzlab-T0-Upli... VLAN	● Success

Enable workload management with NSX-T Data Center

Enabling workload management creates a Supervisor of three nodes. To enable workload management:

- Go to **vSphere Client → home menu -> Workload Management → click Get Started**



Workload Management

Workload Management enables deploying and managing Kubernetes workloads in vSphere. By using Workload Management, you can leverage both Kubernetes and vSphere functionality. Once you configure a vSphere cluster for Workload Management, you can create namespaces that provide compute, networking, and storage resources for running your Kubernetes applications. You can configure namespaces with policies for resource consumption.

Learn more about Workload Management

GET STARTED

Learn more about configuring Kubernetes support for your cluster

Network Support

You can select between two networking stacks when configuring Workload Management.

NSX-T: Supports vSphere Pods and Tanzu Kubernetes clusters.
vCenter Server Network: Supports Tanzu Kubernetes clusters.

DOWNLOAD CHECKLIST

HA and DRS Support

You must enable vSphere HA and DRS in fully-automated mode on the cluster where you set up Workload Management.

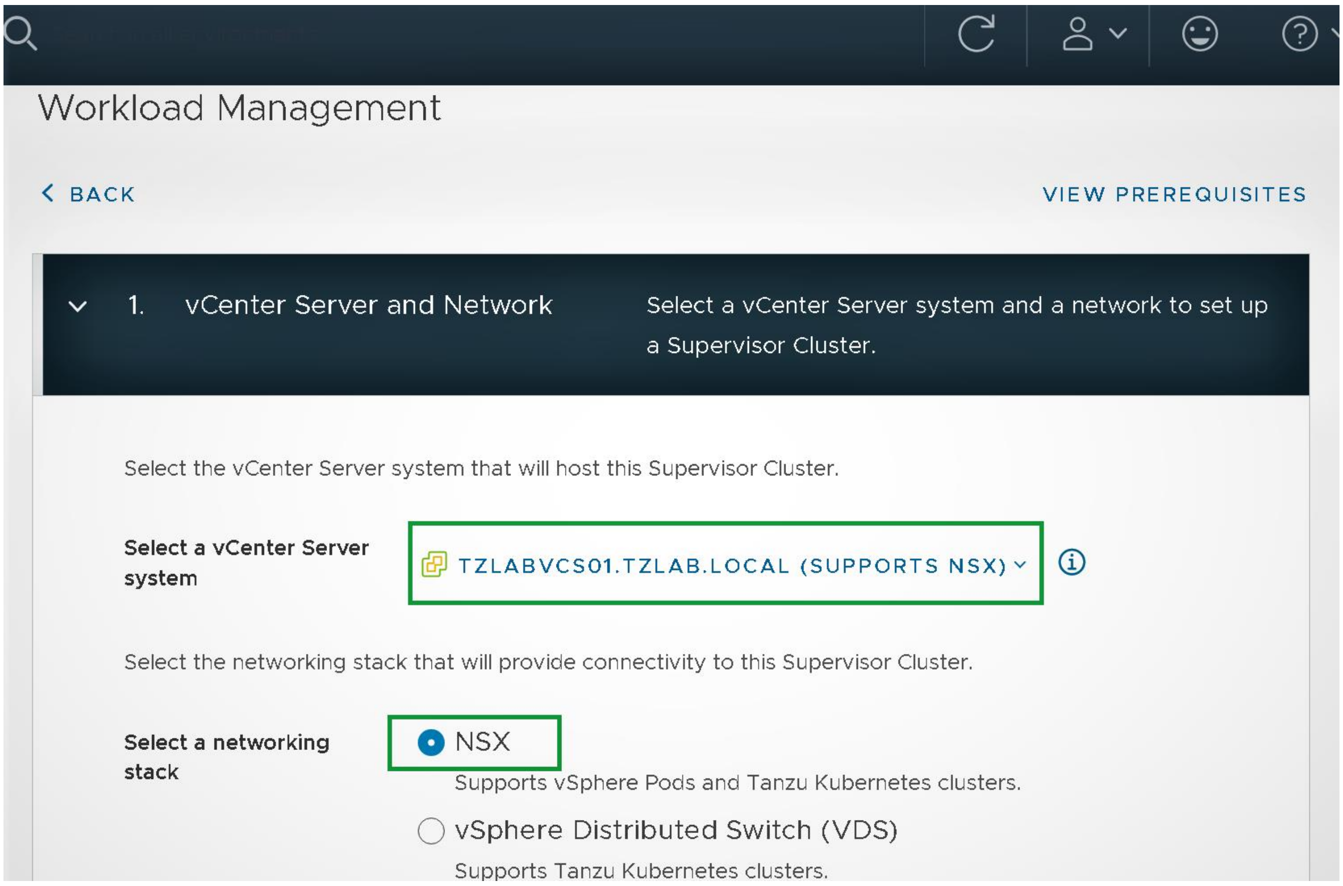
VIEW DRS/HA DOCUMENTATION

Storage Policy

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

VIEW STORAGE POLICIES

- Select **NSX-T** and then **select a Cluster → vCenter Server -> next**



Workload Management

BACK   VIEW PREREQUISITES

1. vCenter Server and Network — Select a vCenter Server system and a network to set up a Supervisor Cluster.

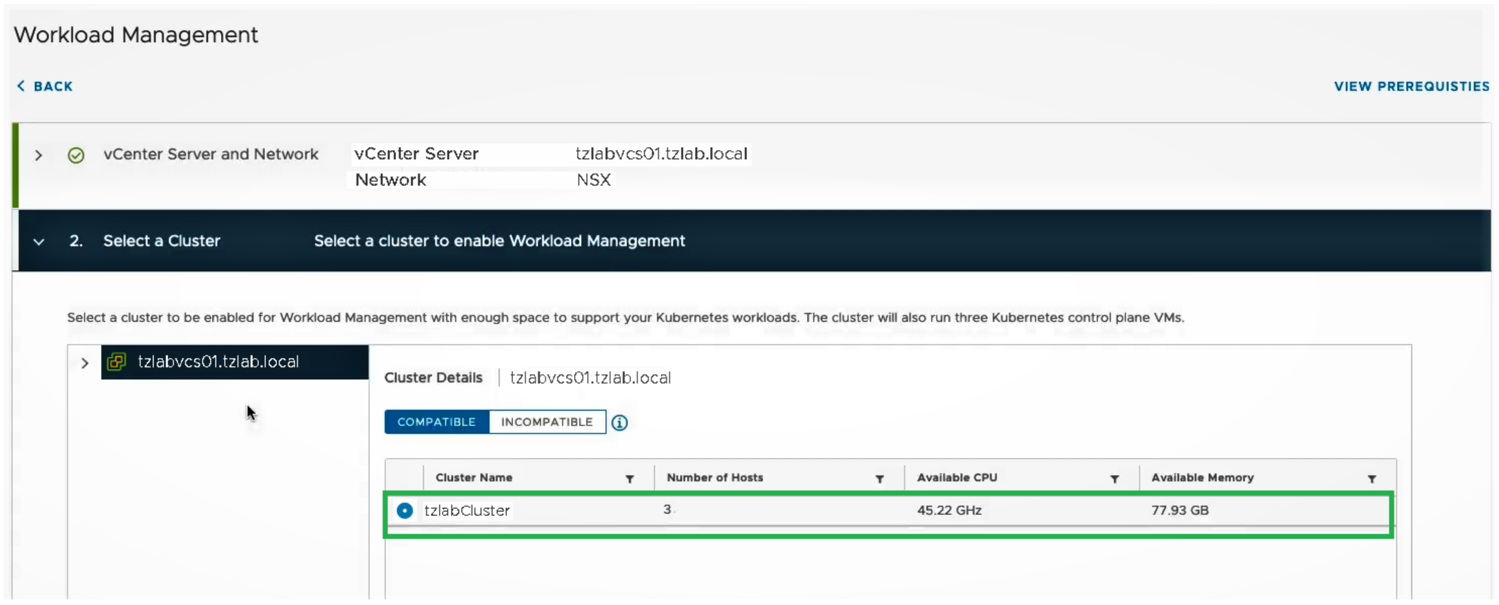
Select the vCenter Server system that will host this Supervisor Cluster.

Select a vCenter Server system: TZLABVCS01.TZLAB.LOCAL (SUPPORTS NSX)

Select the networking stack that will provide connectivity to this Supervisor Cluster.

Select a networking stack:
[✓ NSX] Supports vSphere Pods and Tanzu Kubernetes clusters.
vSphere Distributed Switch (VDS) Supports Tanzu Kubernetes clusters.

- Select a **Compatible Cluster → next**



Workload Management

BACK   VIEW PREREQUISTIES

vCenter Server and Network — vCenter Server: tzlabvcs01.tzlab.local; Network: NSX

2. Select a Cluster — Select a cluster to enable Workload Management

Select a cluster to be enabled for Workload Management with enough space to support your Kubernetes workloads. The cluster will also run three Kubernetes control plane VMs.

tzlabvcs01.tzlab.local

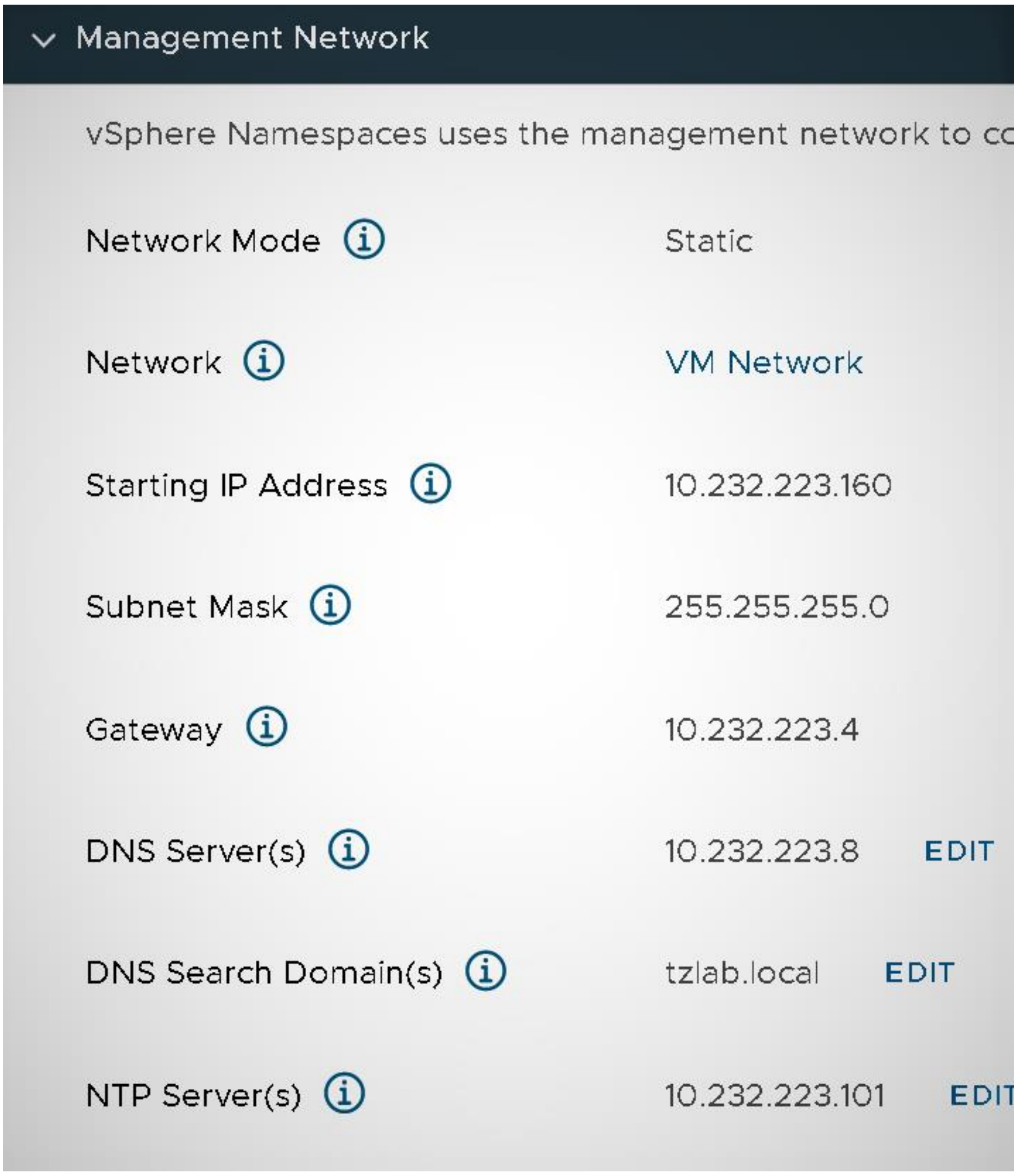
Cluster Details | tzlabvcs01.tzlab.local

COMPATIBLE | INCOMPATIBLE

Cluster Name	Number of Hosts	Available CPU	Available Memory
● tzlabCluster	3	45.22 GHz	77.93 GB

- Select a size for the control plane → next

- Configure the **Management Network: then next**

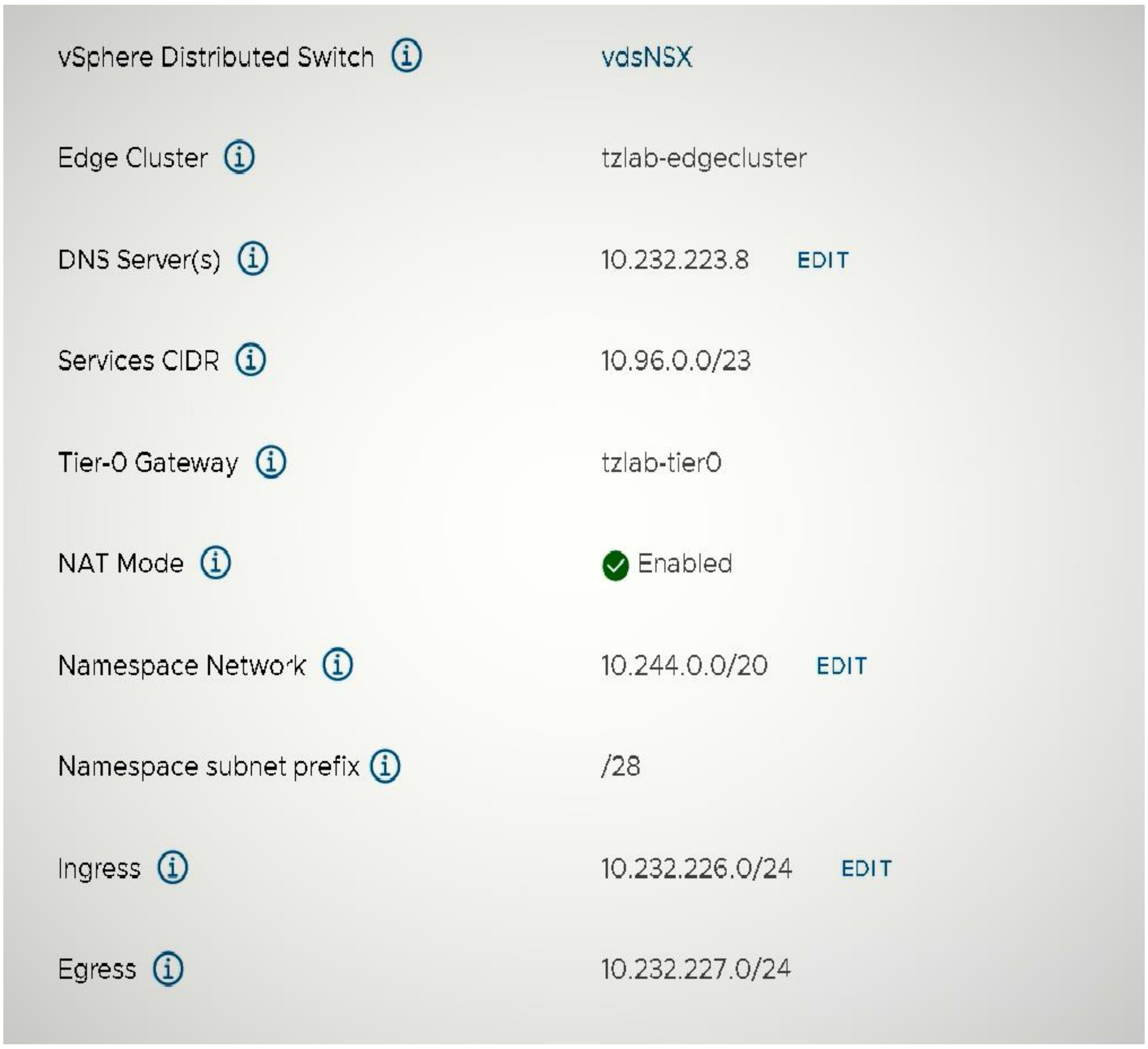


Management Network

vSphere Namespaces uses the management network to cc

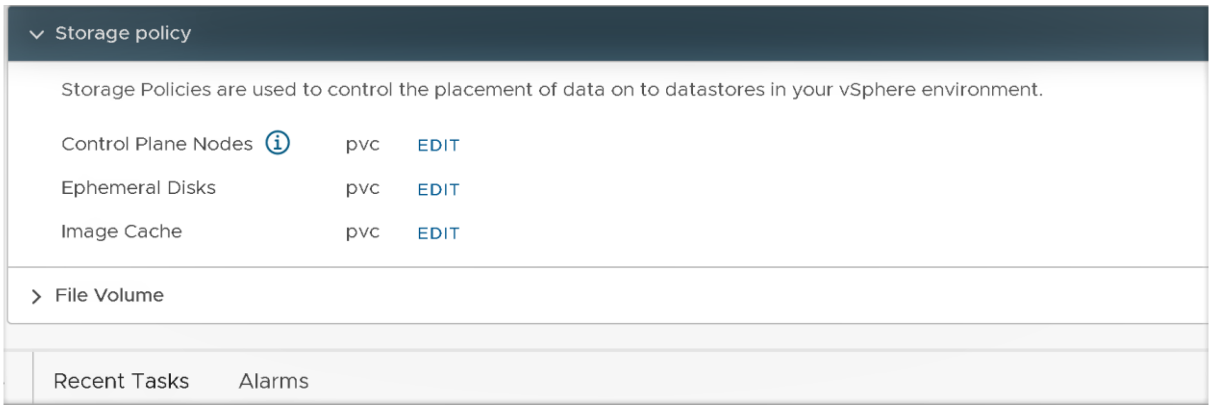
Network Mode	Static
Network	VM Network
Starting IP Address	10.232.223.160
Subnet Mask	255.255.255.0
Gateway	10.232.223.4
DNS Server(s)	10.232.223.8 EDIT
DNS Search Domain(s)	tzlab.local EDIT
NTP Server(s)	10.232.223.101 EDIT

- Configure the **network for namespaces: then next**



vSphere Distributed Switch	vdsNSX
Edge Cluster	tzlab-edgecluster
DNS Server(s)	10.232.223.8 EDIT
Services CIDR	10.96.0.0/23
Tier-0 Gateway	tzlab-tier0
NAT Mode	✓ Enabled
Namespace Network	10.244.0.0/20 EDIT
Namespace subnet prefix	/28
Ingress	10.232.226.0/24 EDIT
Egress	10.232.227.0/24

- Configure **storage policy: then next**

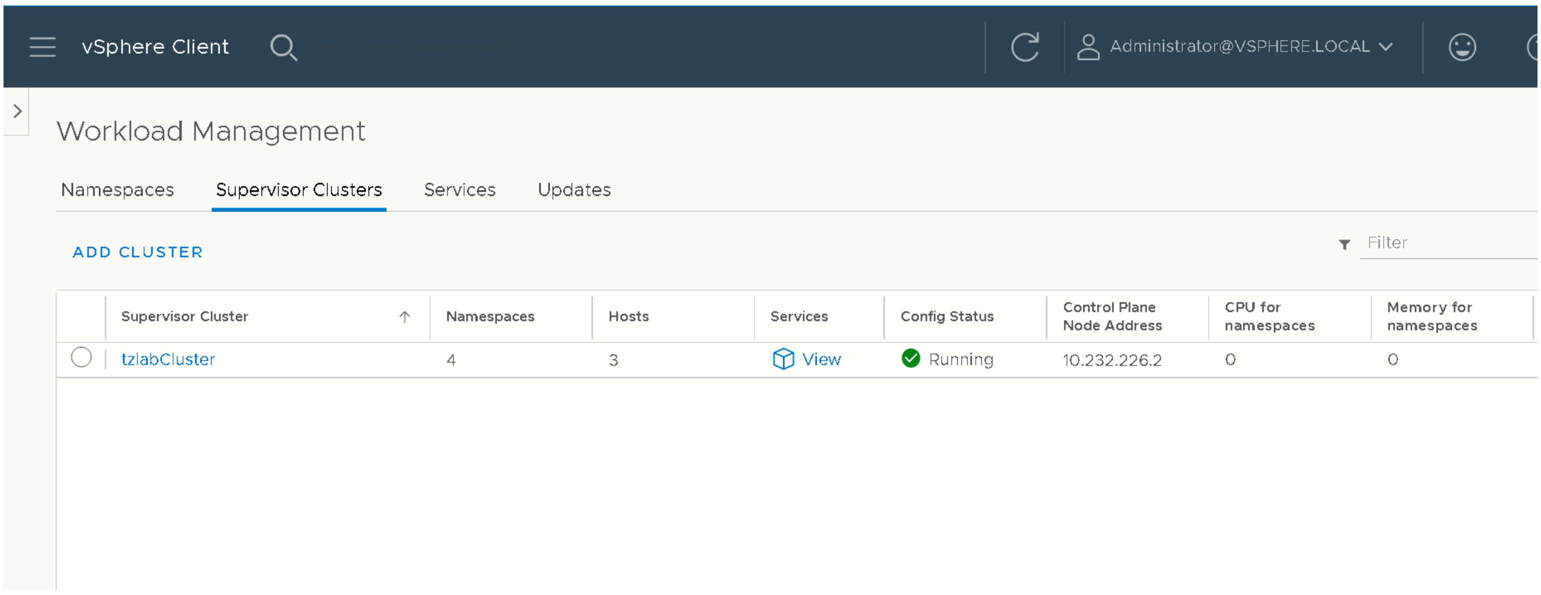


Storage policy

Storage Policies are used to control the placement of data on to datastores in your vSphere environment.

Control Plane Nodes	pvc	EDIT
Ephemeral Disks	pvc	EDIT
Image Cache	pvc	EDIT

- Review the configuration and then **click Finish.**
- Verify that there are not any errors, and the **config status** of the supervisor cluster is **“running”.**

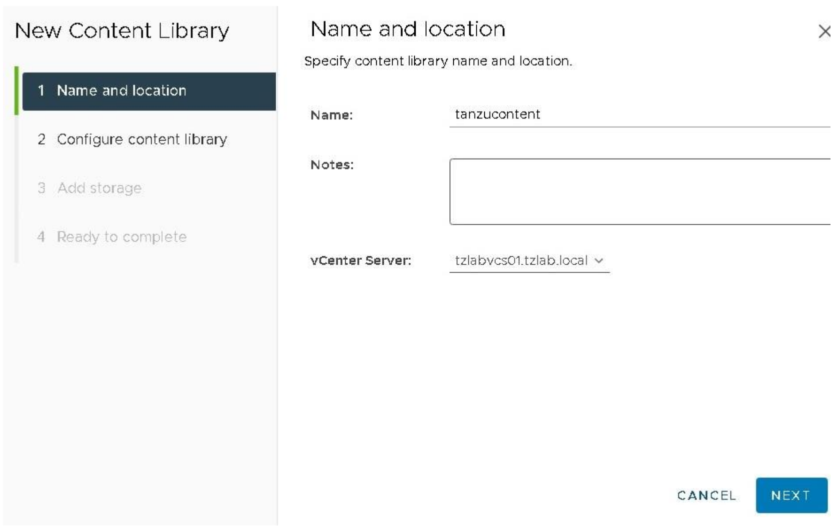


Supervisor Cluster	Namespaces	Hosts	Services	Config Status	Control Plane Node Address	CPU for namespaces	Memory for namespaces
tzlabCluster	4	3	View	✓ Running	10.232.226.2	0	0

Creating Content Library

From the vShpere Client menu **select content libraries** and **click create** :

- The following window will open



Name and location

Specify content library name and location.

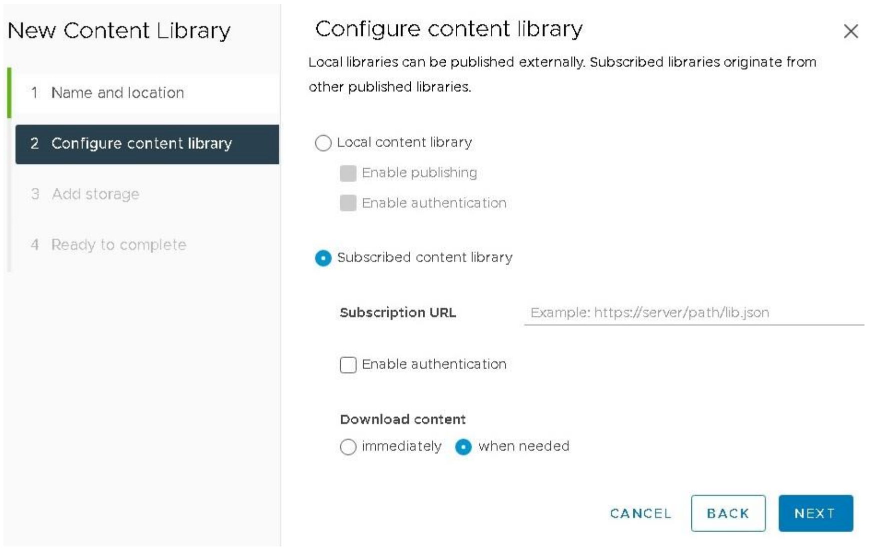
Name: tanzucontent

Notes:

vCenter Server: tzlabvcs01.tzlab.local

  - A descriptive name has been inserted.
  - The only available vCenter Server is “tzlabvcs01.tzlab.local” and it will be used.

- On the configure content library wizard window:



Configure content library

Local libraries can be published externally. Subscribed libraries originate from other published libraries.

○ Local content library

☐ Enable publishing

☐ Enable authentication

● Subscribed content library

Subscription URL: Example: https://server/path/lib.json

☐ Enable authentication

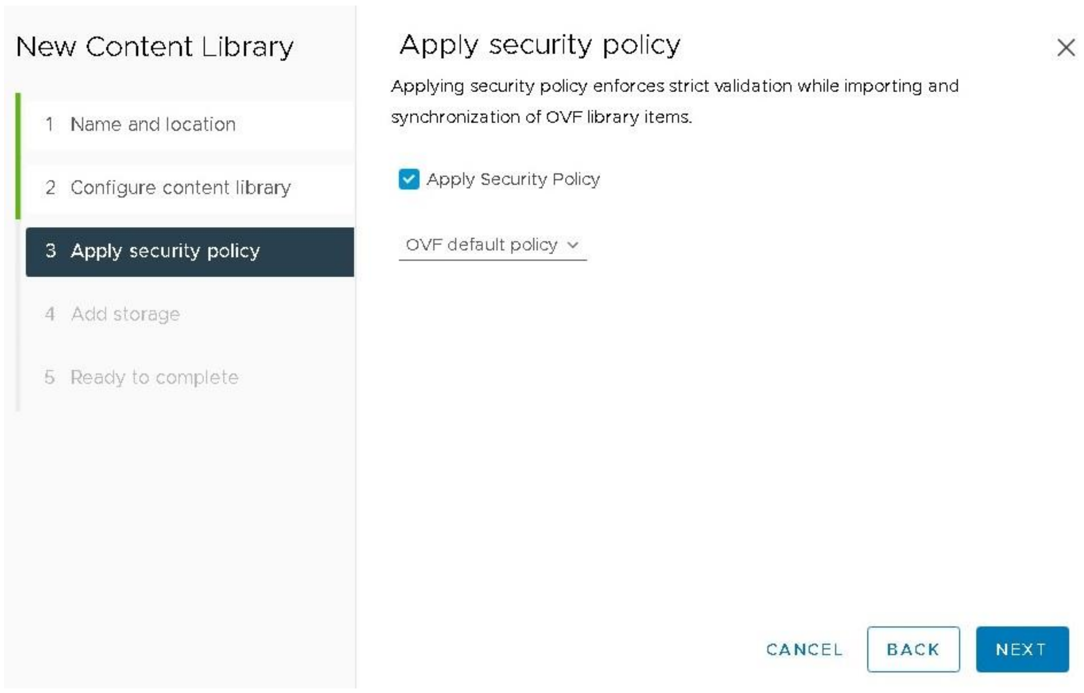
Download content

○ immediately ● when needed

On the configure content library wizard, specify whether one will use local content library in which an open virtualization format(OVF) template will be imported by administrator or will use the subscribed content library by subscribing to a publisher. The subscribed content library should be published by a publisher. The administrators of vSphere are only allowed to use them but have not the right to import or update an OVF template.

- A subscribed configuration with subscription URL https://wp-content.vmware.com/v2/latest/lib.json from VMware is used[5]. To save storage availability, it has been used download content as "when needed". This option enables Tanzu Kubernetes Grid service to download an image when it is published. Whenever the administrator feels there is no need of that content then it can be deleted.

- Applying security policy



Apply security policy

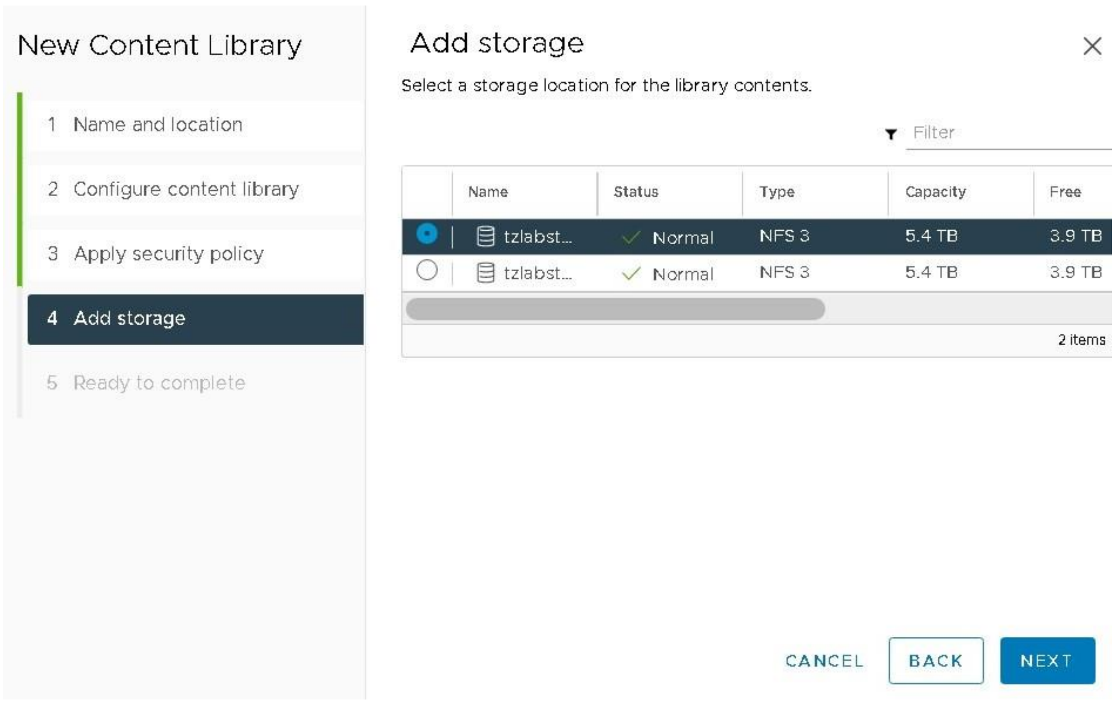
Applying security policy enforces strict validation while importing and synchronization of OVF library items.

☑ Apply Security Policy

OVF default policy

  - Here there is only the default OVF policy that impose a strict validation policy whenever a content is in synchronization. This mean, if the OVF template fails to pass the certification of that policy then it will not synchronize.

- Adding a storage from the available storage in the vCenter server



Add storage

Select a storage location for the library contents.

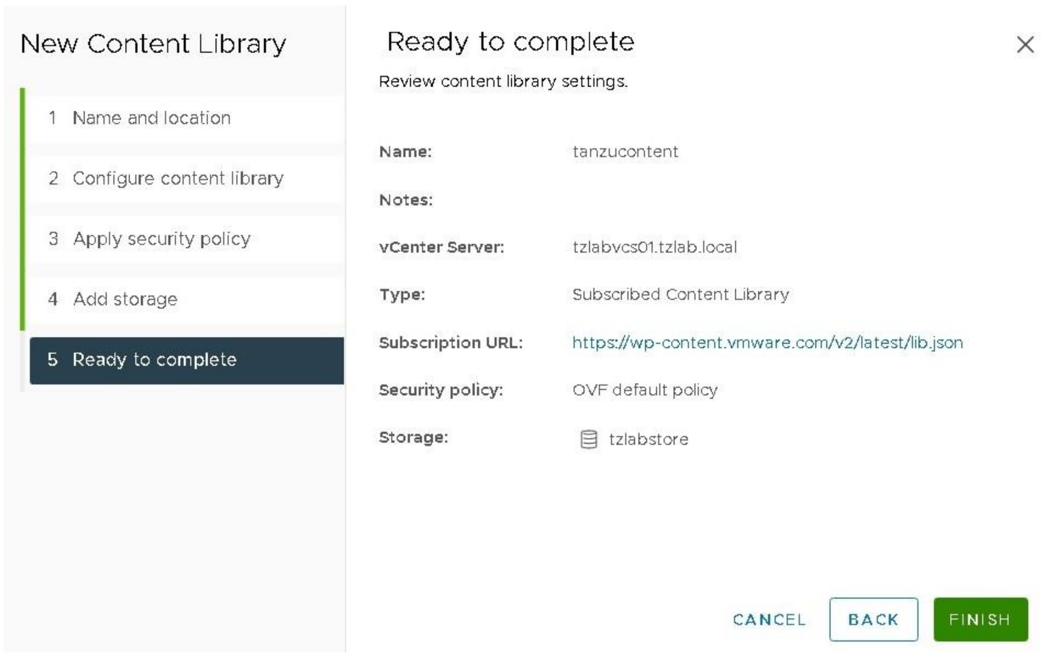
Filter

	Name	Status	Type	Capacity	Free
●	tzlabst...	✓ Normal	NFS 3	5.4 TB	3.9 TB
○	tzlabst...	✓ Normal	NFS 3	5.4 TB	3.9 TB

2 items

- ○ In the storage window wizard, one of the two storage device is associated with the content library.

- Finalizing the procedure by checking the summary window and **click Finish**



Ready to complete

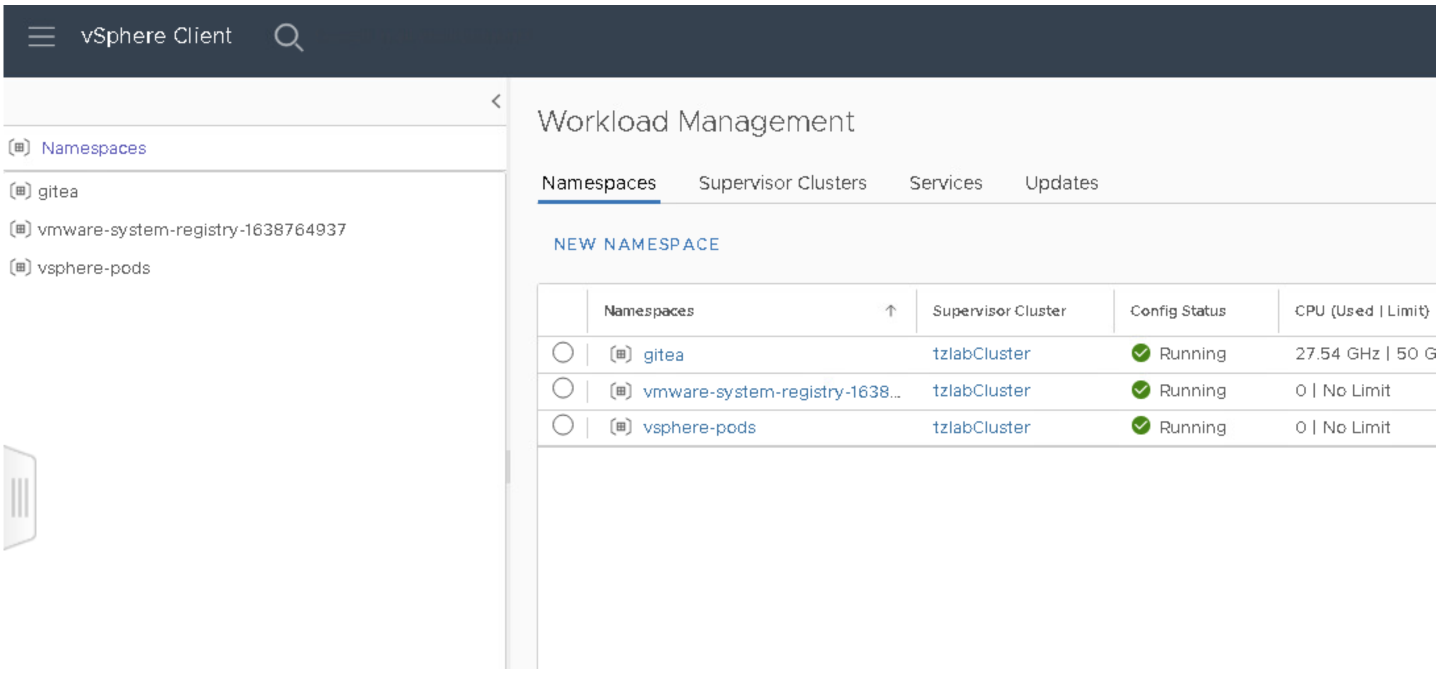
Review content library settings.

Name:	tanzucontent
Notes:	
vCenter Server:	tzlabvcs01.tzlab.local
Type:	Subscribed Content Library
Subscription URL:	https://wp-content.vmware.com/v2/latest/lib.json
Security policy:	OVF default policy
Storage:	tzlabstore

Creating and configuring vSphere Namespaces

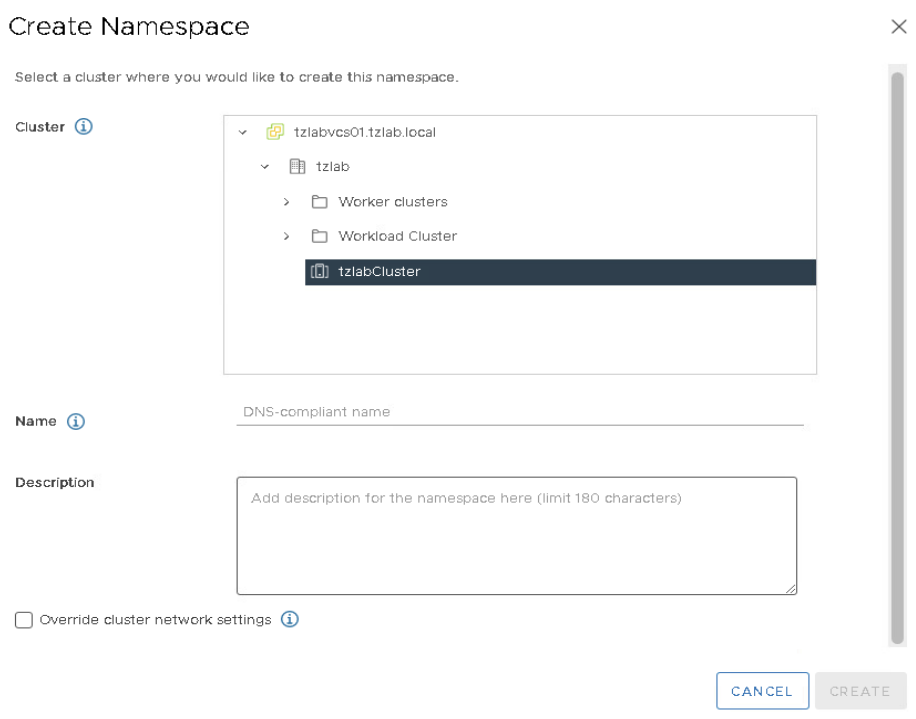
1. Go to vSphere Client
2. Select **Workload Management** from Home menu

   The following window will pop up



Namespaces	Supervisor Cluster	Config Status	CPU (Used | Limit)
gitea	tzlabCluster	Running	27.54 GHz | 50 G
vmware-system-registry-1638...	tzlabCluster	Running	0 | No Limit
vsphere-pods	tzlabCluster	Running	0 | No Limit

3. Click on **New Namespace**


Create Namespace

Select a cluster where you would like to create this namespace.

Cluster: tzlabvcs01.tzlab.local > tzlab > Worker clusters / Workload Cluster / tzlabCluster (selected)

Name: DNS-compliant name

Description: Add description for the namespace here (limit 180 characters)

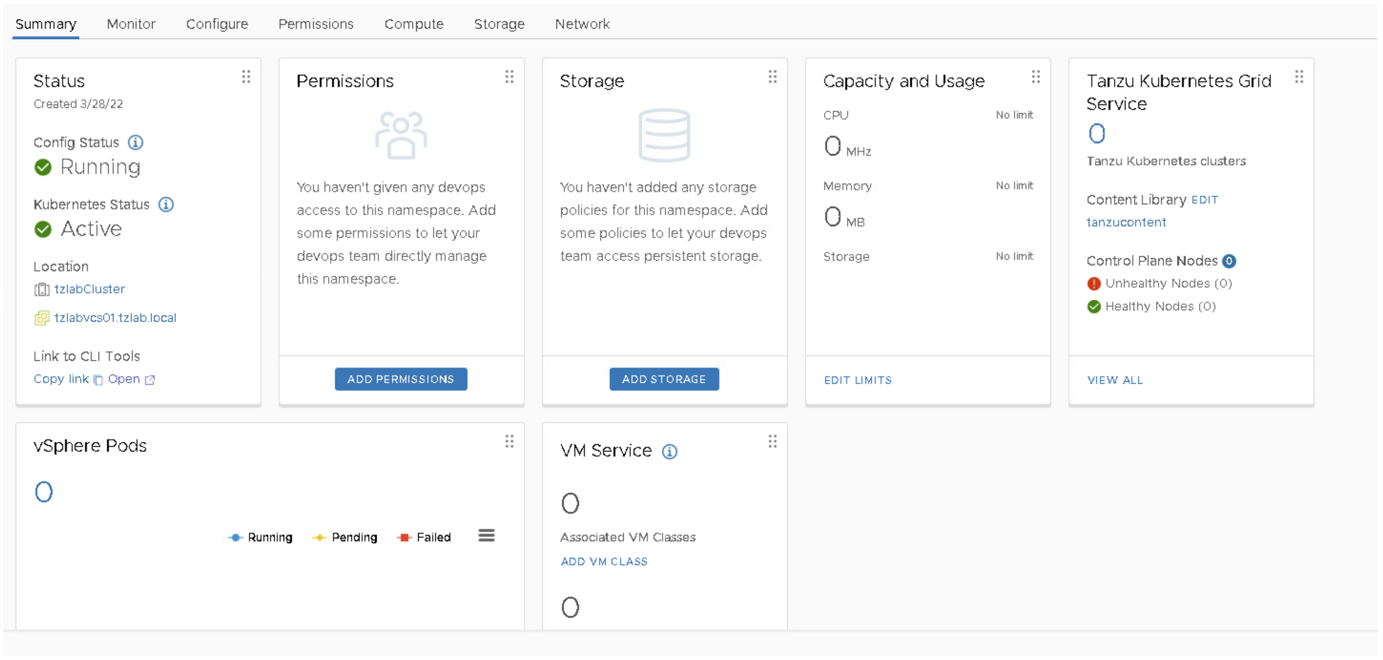
☐ Override cluster network settings

Cluster -> Choose the Supervisor cluster (only the cluster that have workload management enabled on it will be shown).

Name -> Supply a namespace name which must be unique across all supervisor clusters and must be DNS compliant (a-z, 0-9, and "-" characters up to 63 characters).

Description -> Enter the namespace description

4. Click **CREATE**
   After this step the namespace will be created on the supervisor cluster.



Status
Created 3/28/22
Config Status: Running
Kubernetes Status: Active
Location: tzlabCluster, tzlabvcs01.tzlab.local
Link to CLI Tools: Copy link, Open

Permissions
You haven't given any devops access to this namespace. Add some permissions to let your devops team directly manage this namespace.
ADD PERMISSIONS

Storage
You haven't added any storage policies for this namespace. Add some policies to let your devops team access persistent storage.
ADD STORAGE

Capacity and Usage
CPU: 0 MHz (No limit)
Memory: 0 MB (No limit)
Storage: No limit
EDIT LIMITS

Tanzu Kubernetes Grid Service
0 Tanzu Kubernetes clusters
Content Library EDIT: tanzucontent
Control Plane Nodes 0
Unhealthy Nodes (0)
Healthy Nodes (0)
VIEW ALL

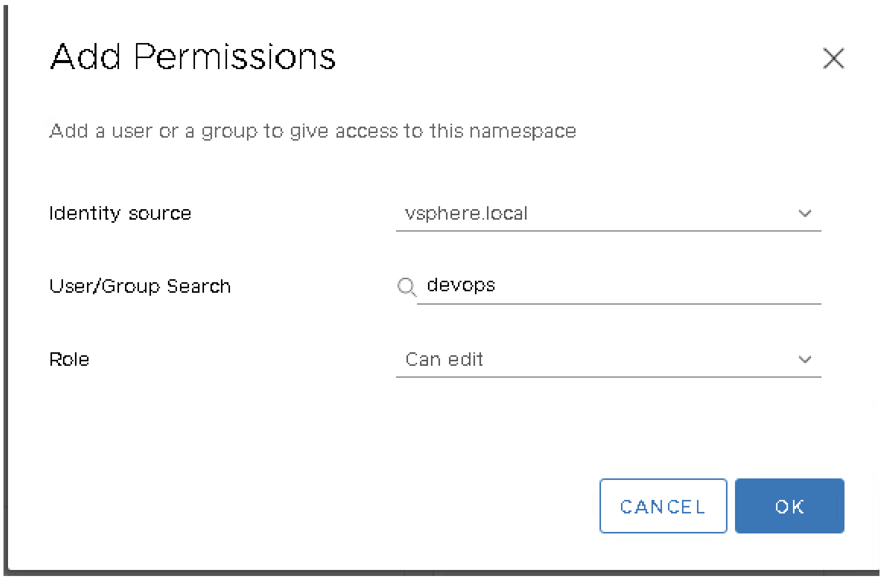
vSphere Pods
0
Running · Pending · Failed

VM Service
0 Associated VM Classes
ADD VM CLASS
0

5. The next step is to grant permission to the DevOps engineers so they can access the namespace.

   Permissions -> Select **ADD PERMISSIONS**

   The following window will pop up



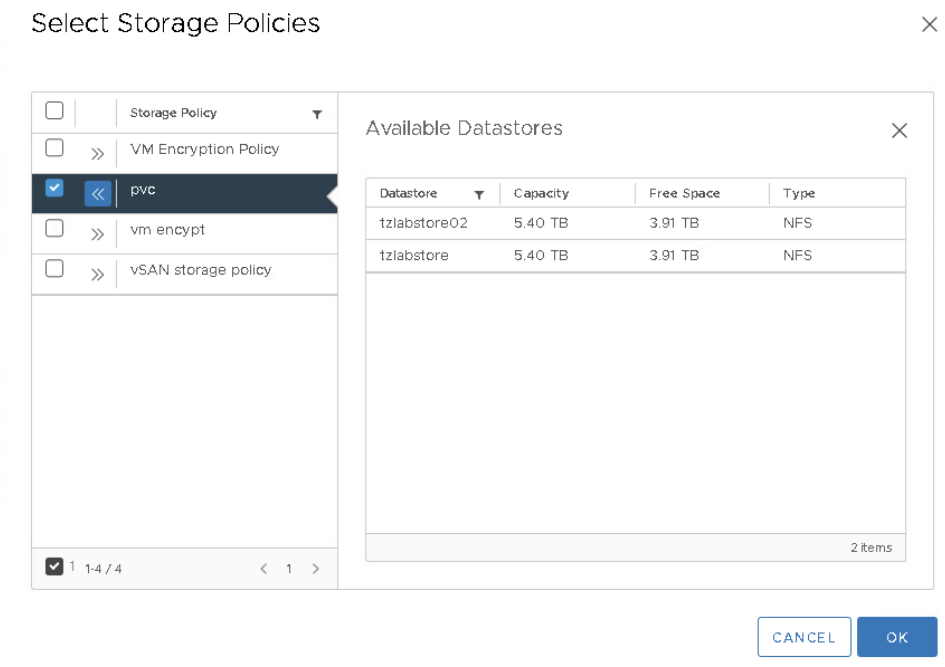
Add Permissions

Add a user or a group to give access to this namespace

Identity source	vsphere.local
User/Group Search	devops
Role	Can edit

   Choose an identity source and a user with the appropriate role. Then click **OK**.

6. Add the storage policy for the namespace.
   Storage -> Select **Add Storage.** Then, select a storage policy that has been configured to enable the DevOps team to access persistent storage. Click **OK**.



Select Storage Policies

Storage Policy
VM Encryption Policy
[✓ pvc]
vm encypt
vSAN storage policy

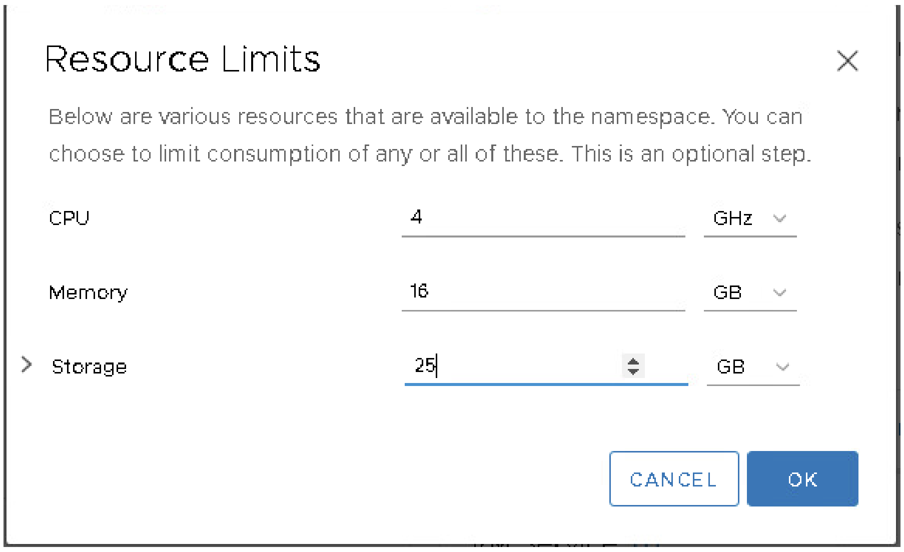
1 1-4 / 4

Available Datastores

Datastore	Capacity	Free Space	Type
tzlabstore02	5.40 TB	3.91 TB	NFS
tzlabstore	5.40 TB	3.91 TB	NFS

2 items

7. Setting up the resource limitations.
   Select **Edit Limits** from the **Capacity and Usage** menu. Configure the resource limits as needed.

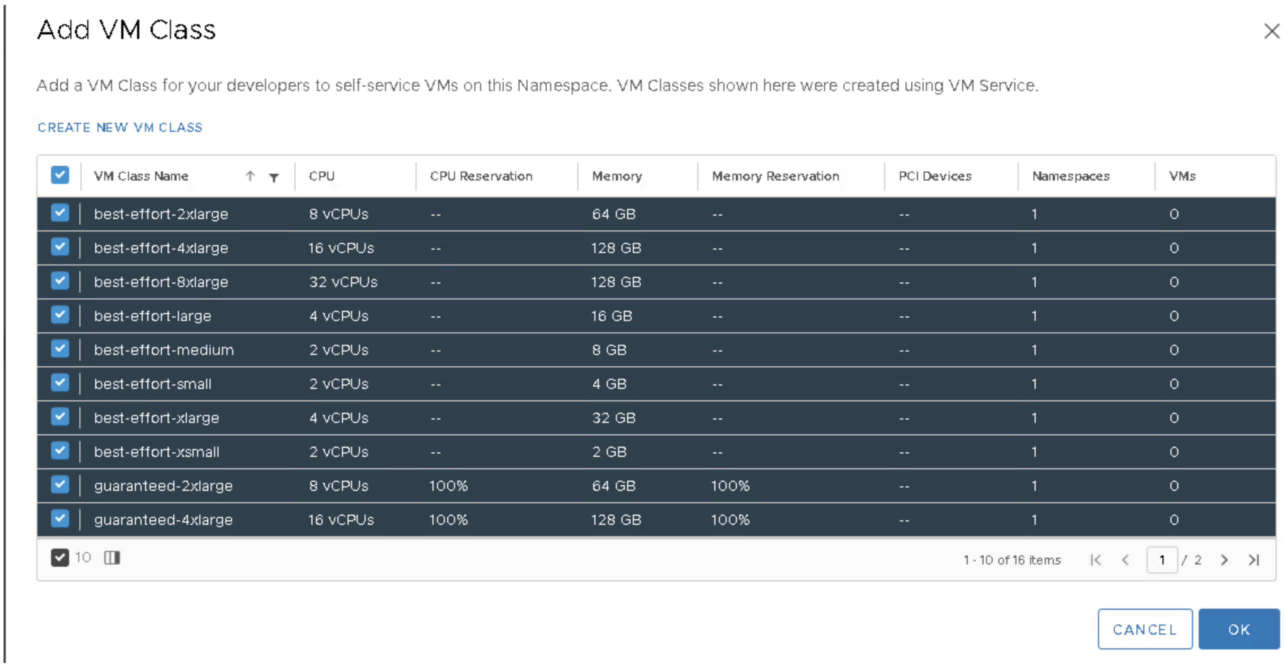


Resource Limits

Below are various resources that are available to the namespace. You can choose to limit consumption of any or all of these. This is an optional step.

CPU	4	GHz
Memory	16	GB
Storage	25	GB

8. Providing VM classes created in VM services and content libraries.
   Select **ADD VM CLASS** from **VM Service** menu. Then, choose the VM classes that are needed.



Add VM Class

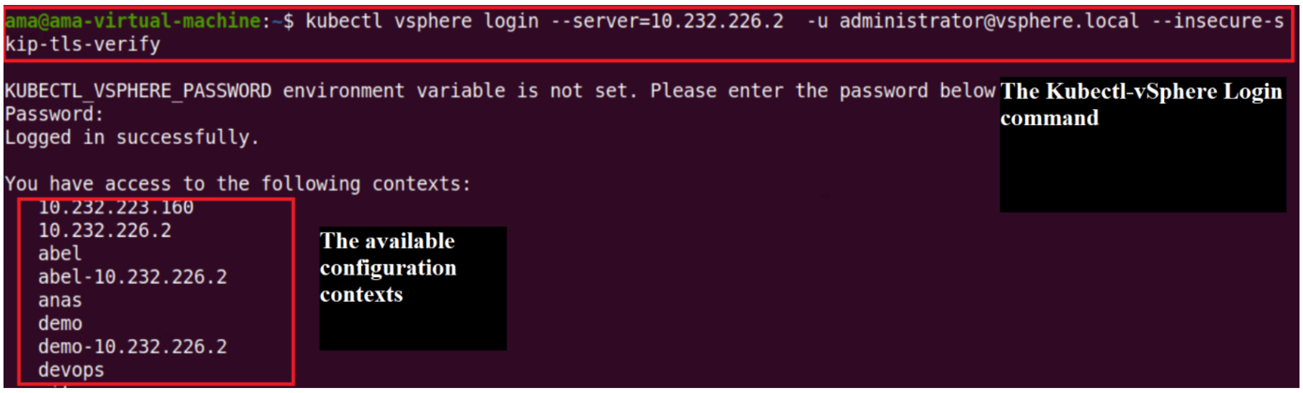
Add a VM Class for your developers to self-service VMs on this Namespace. VM Classes shown here were created using VM Service.

	VM Class Name	CPU	CPU Reservation	Memory	Memory Reservation	PCI Devices	Namespaces	VMs
✓	best-effort-2xlarge	8 vCPUs	--	64 GB	--	--	1	0
✓	best-effort-4xlarge	16 vCPUs	--	128 GB	--	--	1	0
✓	best-effort-8xlarge	32 vCPUs	--	128 GB	--	--	1	0
✓	best-effort-large	4 vCPUs	--	16 GB	--	--	1	0
✓	best-effort-medium	2 vCPUs	--	8 GB	--	--	1	0
✓	best-effort-small	2 vCPUs	--	4 GB	--	--	1	0
✓	best-effort-xlarge	4 vCPUs	--	32 GB	--	--	1	0
✓	best-effort-xsmall	2 vCPUs	--	2 GB	--	--	1	0
✓	guaranteed-2xlarge	8 vCPUs	100%	64 GB	100%	--	1	0
✓	guaranteed-4xlarge	16 vCPUs	100%	128 GB	100%	--	1	0

Provisioning Tanzu Kubernetes Clusters

- **Log in to Supervisor cluster**
  This requires that the users download Kubernetes CLI tools for vSphere.
  The command used to log in to the Supervisor cluster using the password provided when creating a user: kubectl vsphere login –server=<IP ADDRESS OF THE NAMESPACE> -u <USERNAME>@vsphere.local –insecure-skip-tls-verify

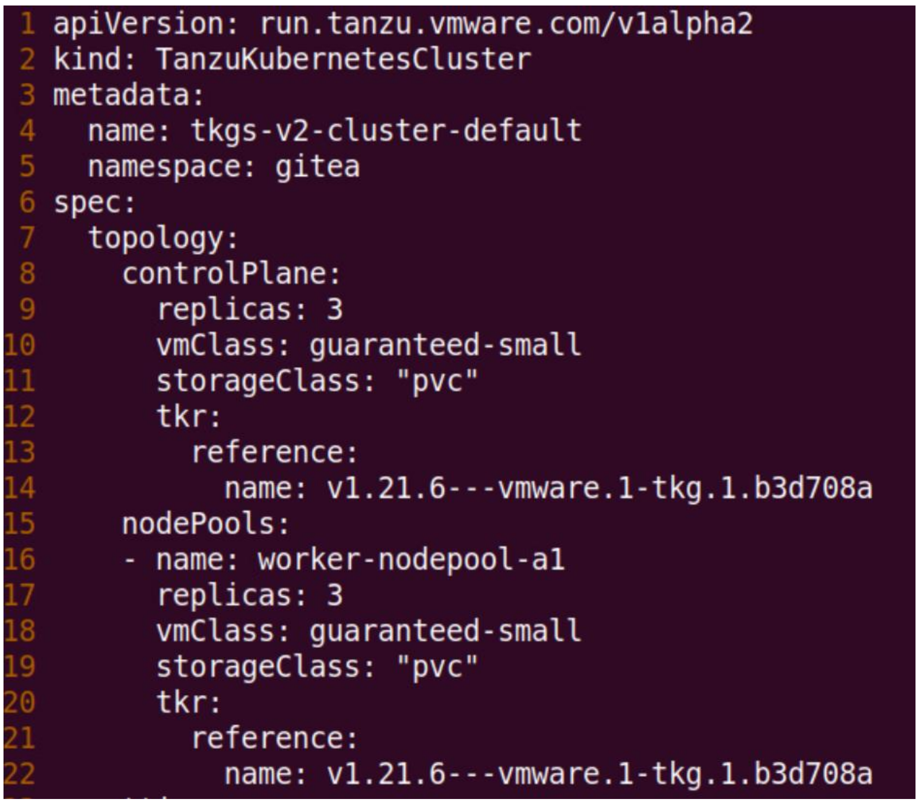
- **Then switch context to the desired namespace using** kubectl config use-context   <NAMESPACE>



```
ama@ama-virtual-machine:~$ kubectl vsphere login --server=10.232.226.2  -u administrator@vsphere.local --insecure-s
kip-tls-verify

KUBECTL_VSPHERE_PASSWORD environment variable is not set. Please enter the password below
Password:
Logged in successfully.

You have access to the following contexts:
   10.232.223.160
   10.232.226.2
   abel
   abel-10.232.226.2
   anas
   demo
   demo-10.232.226.2
   devops
```

The Kubectl-vSphere Login command
The available configuration contexts

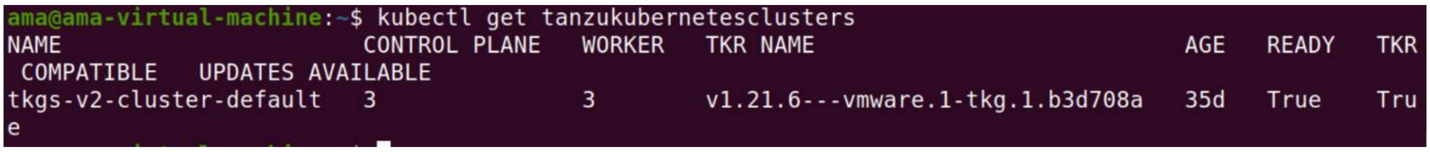
- **Create YAML file for provisioning the Tanzu Kubernetes cluster as needed**



```
apiVersion: run.tanzu.vmware.com/v1alpha2
kind: TanzuKubernetesCluster
metadata:
  name: tkgs-v2-cluster-default
  namespace: gitea
spec:
  topology:
    controlPlane:
      replicas: 3
      vmClass: guaranteed-small
      storageClass: "pvc"
      tkr:
        reference:
          name: v1.21.6---vmware.1-tkg.1.b3d708a
    nodePools:
    - name: worker-nodepool-a1
      replicas: 3
      vmClass: guaranteed-small
      storageClass: "pvc"
      tkr:
        reference:
          name: v1.21.6---vmware.1-tkg.1.b3d708a
```

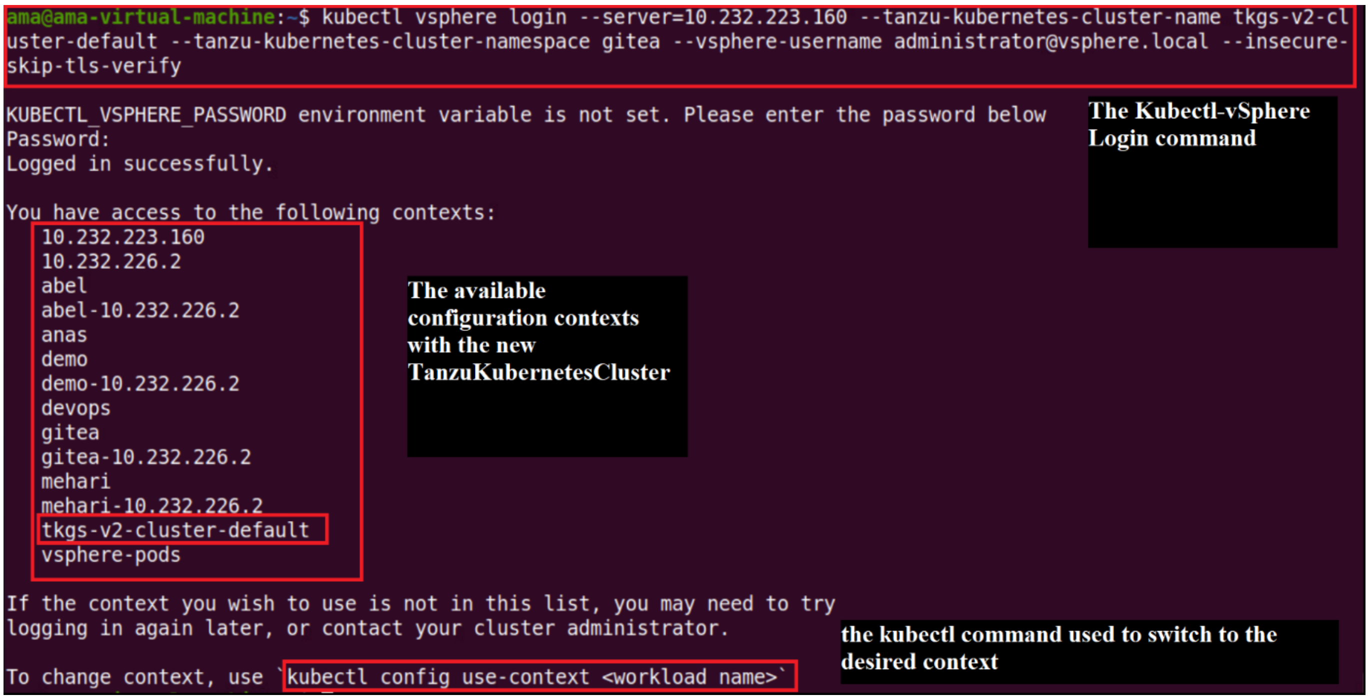
- **Provision the cluster by running the kubectl command**
  kubectl apply -f tkgs-cluster.yaml

- **Monitor the deployment of cluster nodes using kubectl**
  kubectl get tanzukubernetesclusters



```
ama@ama-virtual-machine:~$ kubectl get tanzukubernetesclusters
NAME                       CONTROL PLANE   WORKER   TKR NAME                           AGE   READY   TKR
 COMPATIBLE   UPDATES AVAILABLE
tkgs-v2-cluster-default   3               3        v1.21.6---vmware.1-tkg.1.b3d708a   35d   True    Tru
e
```

- **Login to the Tanzu Kubernetes Cluster**
  kubectl vsphere login --server=IP-ADDRESS --vsphere-username USERNAME \
  --tanzu-kubernetes-cluster-name CLUSTER-NAME \
  --tanzu-kubernetes-cluster-namespace NAMESPACE-NAME


The Kubectl-vSphere Login command
The available configuration contexts with the new TanzuKubernetesCluster
the kubectl command used to switch to the desired context

```
ama@ama-virtual-machine:~$ kubectl vsphere login --server=10.232.223.160 --tanzu-kubernetes-cluster-name tkgs-v2-cl
uster-default --tanzu-kubernetes-cluster-namespace gitea --vsphere-username administrator@vsphere.local --insecure-
skip-tls-verify

KUBECTL_VSPHERE_PASSWORD environment variable is not set. Please enter the password below
Password:
Logged in successfully.

You have access to the following contexts:
   10.232.223.160
   10.232.226.2
   abel
   abel-10.232.226.2
   anas
   demo
   demo-10.232.226.2
   devops
   gitea
   gitea-10.232.226.2
   mehari
   mehari-10.232.226.2
   tkgs-v2-cluster-default
   vsphere-pods

If the context you wish to use is not in this list, you may need to try
logging in again later, or contact your cluster administrator.

To change context, use `kubectl config use-context <workload name>`
```

Projectplan

Abel Weldu Abraha
Mohammed Anas Rihan
Mehari Berhe Zerai

Contents

1 Mål og rammer

1.1 Bakgrunn

Helsetjenestens driftsorganisasjon for nødnett HF (HDO HF) sørger for driften av nødmeldetjenester løsninger som brukes av helsetjenesten. HDO ble stiftet 29. april 2013, og er organisert som en del av spesialisthelsetjenesten, og er eid av de fire helseregionene. Selskapet skal bidra til sikker og trygg kommunikasjon til en samlet akuttmedisinsk kjede. Hovedmålet deres er å skape merverdi for kunder, eiere, helseforetak og kommuner.

HDO skal bidra til å nå de overordnede målene til den nasjonale medisinske nødmeldetjenesten. Selskapets primære og prioriterte er å levere stabile, kostnadseffektive og landsdekkende tjenester som skal ivareta spesialist og primærhelsetjenestens behov. Selskapet skal bidra til utvikling av helsesektorens nødmeldetjeneste, og samt yte døgnåpen drift og kundesupport for helseforetakets brukere.

1.2 Prosjektmål

1.2.1 Resultatmål

Hovedmålet med prosjektet er å dra inn kontainermetodikken inn i vmware for å øke kapasiteten til HDO. En slik virtualiseringsplattform vil bidra til mer effektiv bruk av ressurser, samt en driftsmodell som vil være med på å kunne bidra til stabile, skalerbare og sikre tjenester for kunder.

1.2.2 Effektmål

Det skal produseres en rapport som skal gi HDO grunnlag for å forbedre driften med fremtidsrettet, gjennomført, sentralisert, skalerbar og automatisert metode for å administrere tjenester. Det skal skje ved bruk av moderne virtualiseringsplattform basert på hyper-converged infrastrukturløsning (HCI) fra VMware og Dell.

1.3 Rammer

1.3.1 Tidsramme

Prosjektplanen skal sammen med den signerte prosjektavtalen leveres den 31 januar. Selve prosjektarbeidet skal foregå fra 11.01.2022 i uke 2 til 20.05.2022 (uke 20). Hovedrapporten skal leveres inn i Inspera den 20.05.2022. Den muntlige fremførings delen skal skje i juni.

1.3.2 Kravspesifikke rammer

Tanzu skal installeres og konfigureres for i vSphere og NSX-T, og det skal tenkes best practice konfigurasjon av de nevnte.

1.3.3 Utstyr

Tanzu skal konfigureres i vSphere og NSX-T som allerede er ferdig satt opp i NTNUs skyplattform SkyHigh. Det vil si at NTNUs Openstack vil være vårt arbeidsmiljø gjennom prosjektperioden. Teams er kommunikasjonskanal verktøyet vil skal bruke for å gjennomføre møter og dele informasjonsressurser. Gruppen har bestemt at alt som skal være med i hovedrapporten skal skrives i overleaf (LaTeX). For timeregistrering, har vi blitt enige om å bruke nettbaserte verktøyet toggl.

2 Omfang

2.1 Problemområde

HDO har virtualisert, og må være forberedt til å være i stand for å benytte metodikker som kan brukes inn i virtualiseringen. For at dette skal være mulig er HDO nødt til å tenke mer på effektivisert bruk av ressursen deres. Hovedmålet med denne oppgaven er å utforske muligheten for å kjøre kontainere i vmware. Dette innebærer installasjon og konfigurasjon av Tanzu inn i vSphere og NSX-T. Det skal også gjøres evaluering av Tanzu for å kjøre, bygge og administrerer kontainere.

ESXi, vSphere, vSAN, og NSX-T er teknologier fra vmware og Dell, og er basert på hyperconverged infrastrukturløsning, og er teknologier som vi vil kunne utforske under oppgaven. Fagområder som administrering/orkestrering av kontainere og sikkerhet er også andre relevante fagområdene som vi vil komme inne på under oppgaven.

2.2 Problemavgrensing

Dette prosjektet har følgende avgrensninger:

- Det skal installeres og konfigureres for Tanzu i v Sphere og NSX-T.
- Vi er ansvarlige for å sette opp overvåkning for applikasjonen, men HDO vil selv være ansvarlige for å sjekke om kontainere er oppe og kjører som de skal.
- Vi er avgrenset for å bruke SkyHigh som virtualiseringsmiljø, altså ikke andre skyplattform som Amazon cloud eller Microsoft Azure.

2.2.1 Oppgavebeskrivelse

Oppgavens formål er å dra inn kontainer metodikken inn i Vmware. Dette innebærer å installere og konfigurere for Tanzu inn i NSX-T og vSphere. Det skal gjøres en evaluering av Tanzu, for å bygge, kjøre, administrering og orkestrere kontainere. Videre skal det settes opp overvåking for applikasjon/tjeneste. For å oppnå dette, så skal vi bruke selvvalgt Open Source applikasjon eller tjeneste som vi skal sette opp overvåking for. Oppgaven omfatter hovedsakelig fem aspekter:

- Installere og konfigurere Tanzu i vSphere
- Installere og konfigurere for Tanzu i NSX-T
- Administrering/orkestrering av kontainere

- Overvåking/Monitorering av tjeneste
- Sikkerhet i løsningen.

3 Prosjektorganisering

3.1 Gruppemedlemmer

Gruppemedlemmene består av Abel Weldu, Mohammed Anas og Mehari Berhe. Alle tre studerer bachelor i Digital infrastruktur og Cybersikkerhet med oppstart i 2019. Alle medlemmene har hatt emnene Robuste skalerbare tjenester og Infrastructure as a code, begge disse to emnene er relevante for bacheloroppgaven. Gruppen består av følgende medlemmer:

- Abel Weldu, e-post: abelwa@stud.ntnu.no
- Mohammed Anas Rihan, e-post: mohammri@stud.ntnu.no
- Mehari Berhe Zerai, e-post: meharibz@stud.ntnu.no

3.2 Roller og ansvar

Hensikten med dette under kapittel er for å understrekke rollene til alle tre gruppemedlemmene har, og tydeliggjøre hvilke ansvar disse rollene innebærer.

Navn	**Rolle**	**Ansvar**
Abel Weldu	Leder	Koordinerer arbeidsoppgaver, og er ansvarlig for å sørge for sprinter gjennomføres som de skal.
Mohammed Anas	Drifter og produkt eier	Ansvarlig for å sørge for driften av oppsettet og samt støtte leder ved behov.
Mehari Berhe	Møteleder	Ansvarlig for alt relatert til møter med gruppen, veileder og oppdragsgiver.

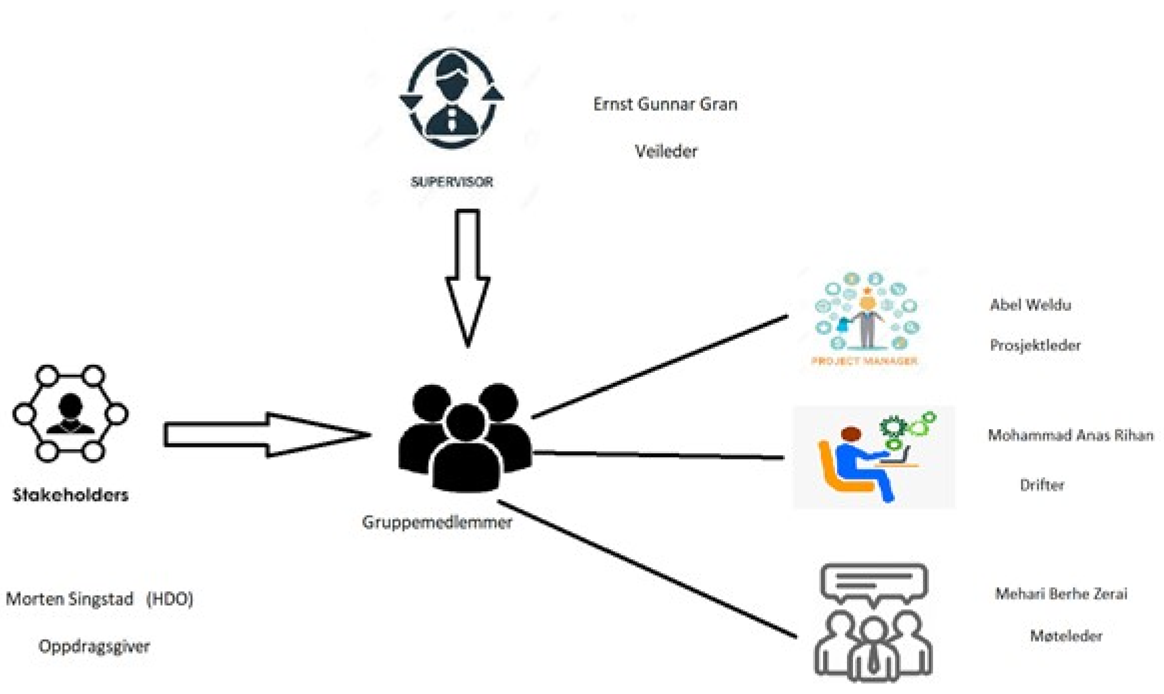


Figure 2: Roller og ansvar

3.3 Regler og rutiner

Generelle gruppereglar

- Det er forventes at gruppemedlemene setter av ca. 30 timer hver uke for prosjektarbeidet.
- Gruppemedlemmene bestemmer selv hvordan de ukentlige 30 timene som er satt av for prosjektarbeidet skal fordeles utover uken.
- Det er forventet at arbeid skal utføres til den avtalte tiden, ved sykdom eller andre gyldige årsaker kan det gjøres unntak. Ved slike tilfeller skal gruppemedlemmene kontakte resten av gruppen i god tid slik at gruppen finner en løsning.
- Gruppemedlemmene er pliktige til å gjennomføre tildelte arbeidsoppgaver innen avtalt tidspunkt.

Møteplikt

- Alle gruppemedlemmene er forpliktet til å delta på avtale møter med gruppen, med oppdragsgiver og med veileder.
- Ved fravær eller sykdom, skal det gis beskjed til resten av gruppen så tidlig som mulig slik at møter kan flyttes.
- Det forventes at hvert gruppemedlem deltar aktiv i møte med veileder, oppdragsgiver og i interne møter med gruppen.
- Gruppen har avtalt for å ha faste ukentlige statusmøter torsdager klokken 10:00.
- Statusmøter skal arrangeres for å diskutere arbeidet som ble gjennomført og videre arbeid, og alt relatert til prosjektarbeidet. Det kan også arrangeres ekstra møter ved behov.
- Det er avtalt sprint review møter hver torsdag kl: 09:00 annenhver uke.
- Det skal tas møtereferat av alle de ukentlige møtene med veilederen.
- Det er avtalt ukentlig møter med veileder hver torsdag kl: 11:00.

Uenigheter/Konflikter
Ved uenigheter skal gruppemedlemmene komme til enighet ved å fatte en flertallsavgjørelse. Veilederen kan eventuelt varsles/kontaktes.

4 Planlegging,oppfølging og rapportering

4.1 Hovedinndeling av prosjektet

Oppdragsgiver har ikke kommet med noe faste krav på hvordan oppgaven skal utføres, men det skal installeres og konfigureres for Tanzu i vSpherer og NSX-T. Det skal også gjøres evaluering av Tanzu for å bygge, kjøre og administrerer kontainere. Selve oppgaven har vi valgt å dele i fire faser, første fasen går ut på å installasjon og konfigurasjon av Tanzu i NSX-T og vSphere. I den tredje fasen kommer vi til å ha fokus på orkestrering av kontainere, og i de to siste fasene skal vi se nærmere på sikkerheten i lønsningen og sette opp overvåking.

4.1.1 Valg av SU-modell/processrammeverk med argumentasjon

Etter møte med oppdragsgiver og interne diskusjoner, har gruppen kommet fram til at Agile scrum er utviklingsmodellen som er best egnet for oss. Gruppemedlemmene ønsker korte iterasjoner og løpende kontroll av resultatene, og samtidig ønsker vi å ha muligheten til gradvis tilpassing av den overordnede planen. Scrum passer for et team fra tre til ni medlemmer som bryter ned oppgaven i mindre deler som kan utførers innen gitte tidsrom. Dette gir oss muligheten til hyppig utvikling og gjør alle ansvarlige, noe som holder motivasjonen oppe. Med hyppig utvikling kan gruppen i stand til å fullføre prosjektarbeidet raskere. Agile scrum gir også oss muligheten til å få kontinuerlig tilbakemelding fra oppdragsgiveren, noe som vil hjelpe oss med å forstå behovet bedre og justere deretter.

4.1.2 Gruppens måte å følge modellen på

Vi har blitt enige om å ha sprinter som varer i to uker. Etter slutten av hver sprint, har vi sprint review møter sammen med oppdragsgiveren hvor vi diskuterer hva som vi har oppnådd i løpet av sprinten og samt diskutere hva som forventes nådd i neste sprint. Gruppen har blitt enige om å bruke issue board på gitlab som en slags scrum board for å holde oversikt over hvilke oppgaver som gjøres av hvem. I tillegg til sprint review møtene annenhver uke, har vi interne status møter hvor vi oppdaterer hverandre om statusen på oppgavene og eventuelle utfordringer tilknyttet oppgavene.

4.2 Plan for statusmøter og beslutningspunkter

Gruppen har bestemt for å ha faste og ukentlige statusmøter hver torsdag i etterkant av veileder møter som også hver torsdag. Hensikten med statusmøtene er tett oppfølging av progresjonen gjennom prosjektet. Statusmøtene skal også brukes til å diskutere utfordringer tilknyttet oppgavene, og for å komme frem til beslutninger. Ukentlige status møter vil skje hver torsdag kl: 10:00, og kan skje enten fysisk eller gjennom den digitale kommunikasjons kanalen Teams.I satusmøtene er gruppemedlemmene pålagt til å besvare følgende spørsmål:

- Hva har jeg bidratt med så langt i denne sprint perioden?
- Hvilke utfordringer har jeg møt så langt, og hvordan har jeg løst disse?
- Hvilke valg har jeg tatt? Har jeg husket å dokumentere ting underveis?
- Hva er det neste jeg skal gjøre/løse?

Møte med oppdragsgiver vil som regel skje etter slutten av en sprint, men kan også avtales etter behov. Møtene vil typisk være på teams, men kan også gjennomføres fysisk hvis det er ønskelig. Ukentlige møtene med veileder vil skje via teams(digitalt), men møtene kan også gjennomføres fysisk hvis det er ønskelig.

5 Organisering og kvalitetssikring

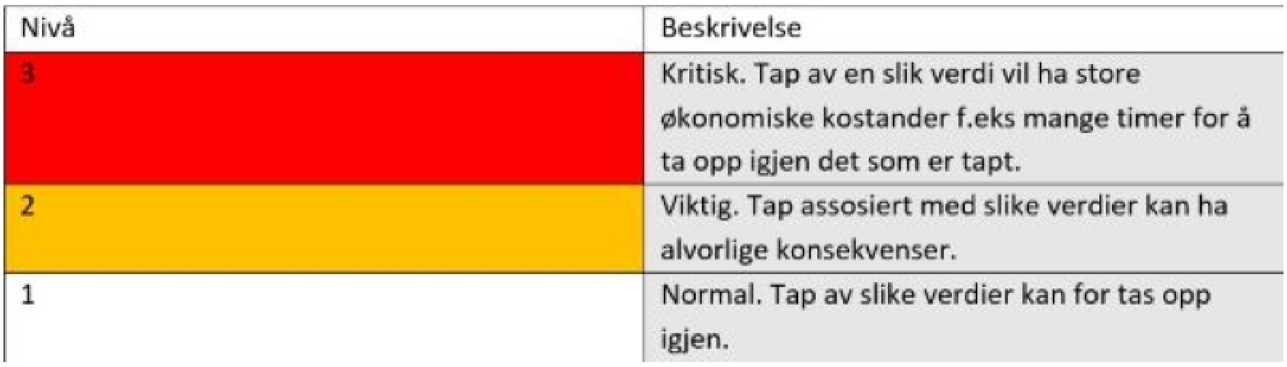
5.1 Dokumentasjon og beste praksis

Alle stegene en tar for å installere, konfigurere og implementeres skal dokumenteres fortløpende slik at gruppen har god kontroll på prosjektets progresjon. God dokumentasjon vil også være nyttig når gruppen skal produsere hovedrapporten som skal leveres på slutten av semesteret. Gruppen skal benytte Microsoft Word å dokumentere alle konfigurasjoner og installasjoner. Alt tekst og skjermdump som skal være med i hovedrapporten skal skrives i overleaf (LaTeX).

Teknologier som skal installeres skal følge beste praksis konfigurasjon av verktøyene, og sikkerhet skal være ett sentral tema ved oppsettet av disse verktøyene. Alle stegene og valg en tar for å installere og konfigurerer skal dokumenteres underveis.

5.2 Risiko analyse

Analysen starter med identifisering av verdier og rangering av hvor kritiske disse verdiene er. Videre har vi utarbeidet en risiko tabell med identifiserte verdier, trusseller, og sannsynlighet og konsekvenser. Til slutt har vi kommet med tiltak for å redusere risikoen. Verdiene som er identifisert er rangert fra 1 til 3 hvor 1 er lav, 2 er viktig og 3 er kritisk. Tabellen under beskriver hva som de forskjellige nivåene.

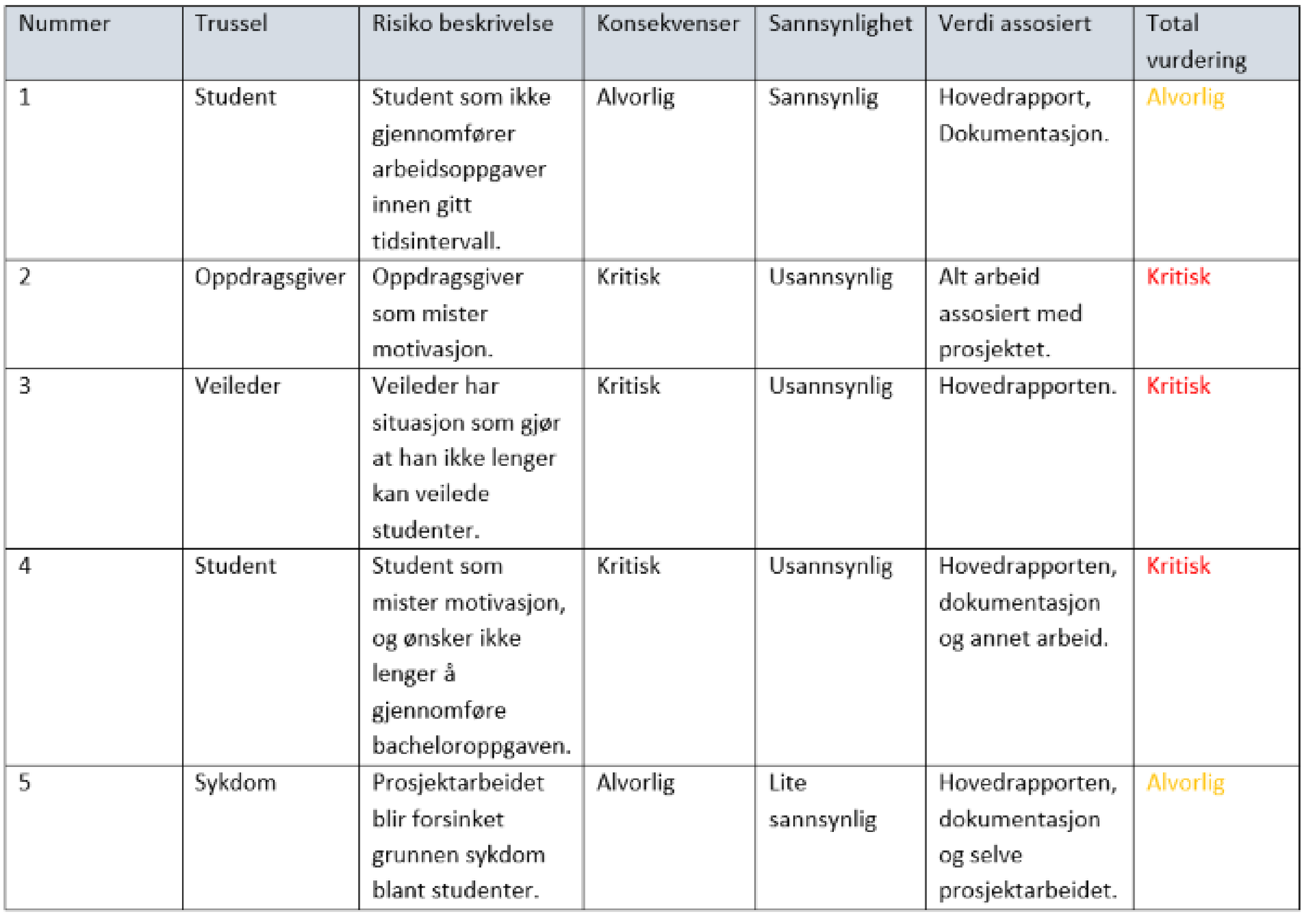


Nivå	Beskrivelse
3	Kritisk. Tap av en slik verdi vil ha store økonomiske kostander f.eks mange timer for å ta opp igjen det som er tapt.
2	Viktig. Tap assosiert med slike verdier kan ha alvorlige konsekvenser.
1	Normal. Tap av slike verdier kan for tas opp igjen.

Verdivurdering av verdier som er assosiert med prosjektet vårt:

Verdi	Nivå 2
Hovedrapport	Kritisk
Dokumentasjon	Viktig
Møtereferater	Normal
Installasjon/konfigurasjon	Viktig

Her er en risiko tabell med trusler som er assosiert med prosjektarbeidet, og beskrivelse av konsekvenser og sannsynlighet assosiert med den aktuelle trusselen. Tabellen er basert på NTNUs ROS veiledning, og inneholder også hvilke verdier som er utsatt av den aktuelle trusselen. Det er gitt en totalvurdering på selve risikoen ut fra sannsynligheten og konsekvensen av trusselen.

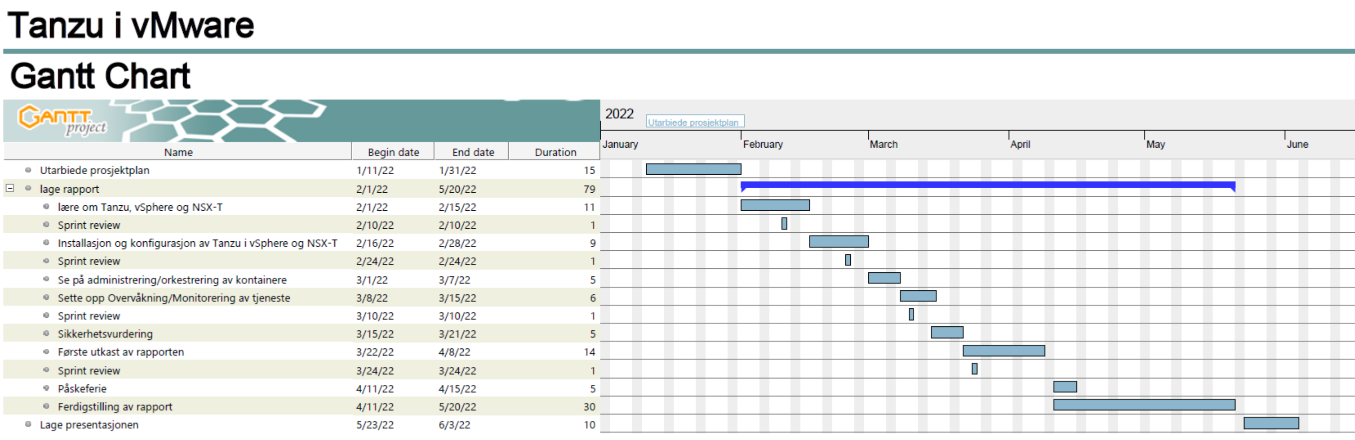


Nummer	Trussel	Risiko beskrivelse	Konsekvenser	Sannsynlighet	Verdi assosiert	Total vurdering
1	Student	Student som ikke gjennomfører arbeidsoppgaver innen gitt tidsintervall.	Alvorlig	Sannsynlig	Hovedrapport, Dokumentasjon.	Alvorlig
2	Oppdragsgiver	Oppdragsgiver som mister motivasjon.	Kritisk	Usannsynlig	Alt arbeid assosiert med prosjektet.	Kritisk
3	Veileder	Veileder har situasjon som gjør at han ikke lenger kan veilede studenter.	Kritisk	Usannsynlig	Hovedrapporten.	Kritisk
4	Student	Student som mister motivasjon, og ønsker ikke lenger å gjennomføre bacheloroppgaven.	Kritisk	Usannsynlig	Hovedrapporten, dokumentasjon og annet arbeid.	Kritisk
5	Sykdom	Prosjektarbeidet blir forsinket grunnen sykdom blant studenter.	Alvorlig	Lite sannsynlig	Hovedrapporten, dokumentasjon og selve prosjektarbeidet.	Alvorlig

Figure 3: Risiko tabell

6 Plan for gjennomføring

Gantskjema er brukt for å illustrere de ulike fasene og aktivitetene i prosjektet. Det er også inkludert estimerte start og slutt datoer for de ulike aktivitetene i prosjektet.



Tanzu i vMware

Gantt Chart

Name	Begin date	End date	Duration
Utarbiede prosjektplan	1/11/22	1/31/22	15
lage rapport	2/1/22	5/20/22	79
lære om Tanzu, vSphere og NSX-T	2/1/22	2/15/22	11
Sprint review	2/10/22	2/10/22	1
Installasjon og konfigurasjon av Tanzu i vSphere og NSX-T	2/16/22	2/28/22	9
Sprint review	2/24/22	2/24/22	1
Se på administrering/orkestrering av kontainere	3/1/22	3/7/22	5
Sette opp Overvåkning/Monitorering av tjeneste	3/8/22	3/15/22	6
Sprint review	3/10/22	3/10/22	1
Sikkerhetsvurdering	3/15/22	3/21/22	5
Første utkast av rapporten	3/22/22	4/8/22	14
Sprint review	3/24/22	3/24/22	1
Påskeferie	4/11/22	4/15/22	5
Ferdigstilling av rapport	4/11/22	5/20/22	30
Lage presentasjonen	5/23/22	6/3/22	10

Figure 4: Gantt-diagram

NTNU

Norges teknisk-naturvitenskapelige universitet

Fastsatt av prorektor for utdanning 10.12.2020

STANDARDAVTALE

om utføring av studentoppgave i samarbeid med ekstern virksomhet

Avtalen er ufravikelig for studentoppgaver (heretter oppgave) ved NTNU som utføres i samarbeid med ekstern virksomhet.

Forklaring av begrep

Opphavsrett
Er den rett som den som skaper et åndsverk har til å fremstille eksemplar av åndsverket og gjøre det tilgjengelig for allmennheten. Et åndsverk kan være et litterært, vitenskapelig eller kunstnerisk verk. En studentoppgave vil være et åndsverk.

Eiendomsrett til resultater
Betyr at den som eier resultatene bestemmer over disse. Utgangspunktet er at studenten eier resultatene fra sitt studentarbeid. Studenten kan også overføre eiendomsretten til den eksterne virksomheten.

Bruksrett til resultater
Den som eier resultatene kan gi andre en rett til å bruke resultatene, f.eks. at studenten gir NTNU og den eksterne virksomheten rett til å bruke resultatene fra studentoppgaven i deres virksomhet.

Prosjektbakgrunn
Det partene i avtalen har med seg inn i prosjektet, dvs. som vedkommende eier eller har rettigheter til fra før og som brukes i det videre arbeidet med studentoppgaven. Dette kan også være materiale som tredjepersoner (som ikke er part i avtalen) har rettigheter til.

Utsatt offentliggjøring
Betyr at oppgaven ikke blir tilgjengelig for allmennheten før etter en viss tid, f.eks. før etter tre år. Da vil det kun være veileder ved NTNU, sensorene og den eksterne virksomheten som har tilgang til studentarbeidet de tre første årene etter at studentarbeidet er innlevert.

1. Avtaleparter

Norges teknisk-naturvitenskapelige universitet (NTNU) Institutt: Informasjonssikkerhet og Kommunikasjonsteknologi
Veileder ved NTNU: Ernst Gunnar Gran e-post og tlf. ernst.g.gran@ntnu.no tlf: +4799644916
Ekstern virksomhet: ***Helsetjenestens Driftsorganisasjon for Nødnett (HDO)*** Ekstern virksomhet sin kontaktperson, e-post og tlf.: ***Morten Singstad*** ***morten.singstad@hdo.no*** ***99102096***
Student: Abel Weldu Abraha Fødselsdato: 15.02.2000
Student: Mohammad Anas Rihan Fødselsdato: 10.11.1991
Student: Mehari Berhe Zerai Fødselsdato: 15.11.1984

Partene har ansvar for å klarere eventuelle immaterielle rettigheter som studenten, NTNU, den eksterne eller tredjeperson (som ikke er part i avtalen) har til prosjektbakgrunn før bruk i forbindelse med utførelse av oppgaven. Eierskap til prosjektbakgrunn skal fremgå av eget vedlegg til avtalen der dette kan ha betydning for utførelse av oppgaven.

2. Utførelse av oppgave

Studenten skal utføre: (sett kryss)

Masteroppgave	
Bacheloroppgave	X
Prosjektoppgave	
Annen oppgave	

Startdato: 11.01.2022
Sluttdato: 31.08.2022

Oppgavens arbeidstittel er: Kubernets i Vmware og NSX-T.

Ansvarlig veileder ved NTNU har det overordnede faglige ansvaret for utforming og godkjenning av prosjektbeskrivelse og studentens læring.

3. Ekstern virksomhet sine plikter

Ekstern virksomhet skal stille med en kontaktperson som har nødvendig faglig kompetanse til å gi studenten tilstrekkelig veiledning i samarbeid med veileder ved NTNU. Ekstern kontaktperson fremgår i punkt 1.

Formålet med oppgaven er studentarbeid. Oppgaven utføres som ledd i studiet. Studenten skal ikke motta lønn eller lignende godtgjørelse fra den eksterne for studentarbeidet. Utgifter knyttet til gjennomføring av oppgaven skal dekkes av den eksterne. Aktuelle utgifter kan for eksempel være reiser, materialer for bygging av prototyp, innkjøp av prøver, tester på lab, kjemikalier. Studenten skal klarere dekning av utgifter med ekstern virksomhet på forhånd.

Ekstern virksomhet skal dekke følgende utgifter til utførelse av oppgaven:

Dekning av utgifter til annet enn det som er oppført her avgjøres av den eksterne underveis i arbeidet.

4. Studentens rettigheter

Studenten har opphavsrett til oppgaven[1]. Alle resultater av oppgaven, skapt av studenten alene gjennom arbeidet med oppgaven, eies av studenten med de begrensninger som følger av punkt 5, 6 og 7 nedenfor. Eiendomsretten til resultatene overføres til ekstern virksomhet hvis punkt 5 b er avkrysset eller for tilfelle som i punkt 6 (overføring ved patenterbare oppfinnelser).

I henhold til lov om opphavsrett til åndsverk beholder alltid studenten de ideelle rettigheter til eget åndsverk, dvs. retten til navngivelse og vern mot krenkende bruk.

Studenten har rett til å inngå egen avtale med NTNU om publisering av sin oppgave i NTNUs institusjonelle arkiv på Internett (NTNU Open). Studenten har også rett til å publisere oppgaven eller deler av den i andre sammenhenger dersom det ikke i denne avtalen er avtalt begrensninger i adgangen til å publisere, jf. punkt 8.

5. Den eksterne virksomheten sine rettigheter

Der oppgaven bygger på, eller videreutvikler materiale og/eller metoder (prosjektbakgrunn) som eies av den eksterne, eies prosjektbakgrunnen fortsatt av den eksterne. Hvis studenten

[1] Jf. Lov om opphavsrett til åndsverk mv. av 15.06.2018 § 1

skal utnytte resultater som inkluderer den eksterne sin prosjektbakgrunn, forutsetter dette at det er inngått egen avtale om dette mellom studenten og den eksterne virksomheten.

Alternativ a) (sett kryss) Hovedregel

x	Ekstern virksomhet skal ha bruksrett til resultatene av oppgaven

Dette innebærer at ekstern virksomhet skal ha rett til å benytte resultatene av oppgaven i egen virksomhet. Retten er ikke-eksklusiv.

Alternativ b) (sett kryss) Unntak

	Ekstern virksomhet skal ha eiendomsretten til resultatene av oppgaven og studentens bidrag i ekstern virksomhet sitt prosjekt

Begrunnelse for at ekstern virksomhet har behov for å få overført eiendomsrett til resultatene:

6. Godtgjøring ved patenterbare oppfinnelser

Dersom studenten i forbindelse med utførelsen av oppgaven har nådd frem til en patenterbar oppfinnelse, enten alene eller sammen med andre, kan den eksterne kreve retten til oppfinnelsen overført til seg. Dette forutsetter at utnyttelsen av oppfinnelsen faller inn under den eksterne sitt virksomhetsområde. I så fall har studenten krav på rimelig godtgjøring. Godtgjøringen skal fastsettes i samsvar med arbeidstakeroppfinnelsesloven § 7. Fristbestemmelsene i § 7 gis tilsvarende anvendelse.

7. NTNU sine rettigheter

De innleverte filer av oppgaven med vedlegg, som er nødvendig for sensur og arkivering ved NTNU, tilhører NTNU. NTNU får en vederlagsfri bruksrett til resultatene av oppgaven, inkludert vedlegg til denne, og kan benytte dette til undervisnings- og forskningsformål med de eventuelle begrensninger som fremgår i punkt 8.

8. Utsatt offentliggjøring

Hovedregelen er at studentoppgaver skal være offentlige.

Sett kryss

x	Oppgaven skal være offentlig

I særlige tilfeller kan partene bli enige om at hele eller deler av oppgaven skal være undergitt utsatt offentliggjøring i maksimalt tre år. Hvis oppgaven unntas fra offentliggjøring, vil den kun være tilgjengelig for student, ekstern virksomhet og veileder i denne perioden. Sensurkomiteen vil ha tilgang til oppgaven i forbindelse med sensur. Student, veileder og sensorer har taushetsplikt om innhold som er unntatt offentliggjøring.

Oppgaven skal være underlagt utsatt offentliggjøring i (sett kryss hvis dette er aktuelt):

Sett kryss		Sett dato
	ett år	
	to år	
	tre år	

Behovet for utsatt offentliggjøring er begrunnet ut fra følgende:

Dersom partene, etter at oppgaven er ferdig, blir enig om at det ikke er behov for utsatt offentliggjøring, kan dette endres. I så fall skal dette avtales skriftlig.

Vedlegg til oppgaven kan unntas ut over tre år etter forespørsel fra ekstern virksomhet. NTNU (ved instituttet) og student skal godta dette hvis den eksterne har saklig grunn for å be om at et eller flere vedlegg unntas. Ekstern virksomhet må sende forespørsel før oppgaven leveres.

De delene av oppgaven som ikke er undergitt utsatt offentliggjøring, kan publiseres i NTNUs institusjonelle arkiv, jf. punkt 4, siste avsnitt. Selv om oppgaven er undergitt utsatt offentliggjøring, skal ekstern virksomhet legge til rette for at studenten kan benytte hele eller deler av oppgaven i forbindelse med jobbsøknader samt videreføring i et master- eller doktorgradsarbeid.

9. Generelt

Denne avtalen skal ha gyldighet foran andre avtaler som er eller blir opprettet mellom to av partene som er nevnt ovenfor. Dersom student og ekstern virksomhet skal inngå avtale om konfidensialitet om det som studenten får kjennskap til i eller gjennom den eksterne virksomheten, kan NTNUs standardmal for konfidensialitetsavtale benyttes.

Den eksterne sin egen konfidensialitetsavtale, eventuell konfidensialitetsavtale den eksterne har inngått i samarbeidprosjekter, kan også brukes forutsatt at den ikke inneholder punkter i motstrid med denne avtalen (om rettigheter, offentliggjøring mm). Dersom det likevel viser seg at det er motstrid, skal NTNUs standardavtale om utføring av studentoppgave gå foran. Eventuell avtale om konfidensialitet skal vedlegges denne avtalen.

Eventuell uenighet som følge av denne avtalen skal søkes løst ved forhandlinger. Hvis dette ikke fører frem, er partene enige om at tvisten avgjøres ved voldgift i henhold til norsk lov. Tvisten avgjøres av sorenskriveren ved Sør-Trøndelag tingrett eller den han/hun oppnevner.

Denne avtale er signert i fire eksemplarer hvor partene skal ha hvert sitt eksemplar. Avtalen er gyldig når den er underskrevet av NTNU v/instituttleder.

Signaturer:

Instituttleder: Dato:
Veileder ved NTNU: Dato:
Ekstern virksomhet: [illegible] Dato: 25.01.2022
Student: Abel Weldu Abraha [illegible] Dato: 20.01.2022
Student: Mohammad Anas Rihan [illegible] Dato: 20.01.2022
Student: Mehari Berhe Zerai [illegible] Dato: 20.01.2022

NTNU

Kunnskap for en bedre verden