Containers and DevOps

NetApp HCI Solutions

NetApp
November 09, 2025

Table of Contents

Containers and DevOps

NVA-1141: NetApp HCI with Anthos, design and deployment

Alan Cowles

The program solutions described in this document are designed and thoroughly tested to minimize deployment risks and accelerate time to market.

This document is for NetApp and partner solutions engineers and customer strategic decision makers. It describes the architecture design considerations that were used to determine the specific equipment, cabling, and configurations required to support the validated workload.

NetApp HCI with Anthos is a verified, best-practice hybrid cloud architecture for the deployment of an on-premises Google Kubernetes Engine (GKE) environment in a reliable and dependable manner. This NetApp Verified Architecture reference document serves as both a design guide and a deployment validation of the Anthos solution on NetApp HCI. The architecture described in this document has been validated by subject matter experts at NetApp and Google to provide the advantage of running Anthos on NetApp HCI within your own enterprise data-center environment.

NetApp HCI, is the industry's first and leading disaggregated hybrid cloud infrastructure, providing the widely recognized benefits of hyperconverged solutions. Benefits include lower TCO and ease of acquisition, deployment, and management for virtualized workloads, while also allowing enterprise customers to independently scale compute and storage resources as needed. NetApp HCI with Anthos provides an on-premises, cloud-like experience for the deployment of containerized workloads managed by Anthos GKE on-premises. This solution provides simplified management, detailed metrics, and a range of additional functionalities that enable the easy movement of workloads deployed both on-site and in the cloud.

Features

With NetApp HCI for Anthos, you can deploy a fully integrated, production-grade Anthos GKE environment in your on-premises data center, which allows you to take advantage of the following features:

- NetApp HCI compute and storage nodes
  - Enterprise-grade hyperconverged infrastructure designed for hybrid cloud workloads
  - NetApp Element storage software
  - Intel-based server compute nodes, including options for Nvidia GPUs
- VMware vSphere 6.7U3
  - Enterprise hypervisor solution for deployment and management of virtual infrastructures
- Anthos GKE in Google Cloud and On-Prem
  - Deploy Anthos GKE instances in Google Cloud or on NetApp HCI

The NetApp Verified Architecture program gives customers reference configurations and sizing guidance for specific workloads and use cases.

Next: Solution Components

Solution components

The solution described in this document builds on the solid foundation of NetApp HCI, VMware vSphere, and the Anthos hybrid-cloud Kubernetes data center solution.

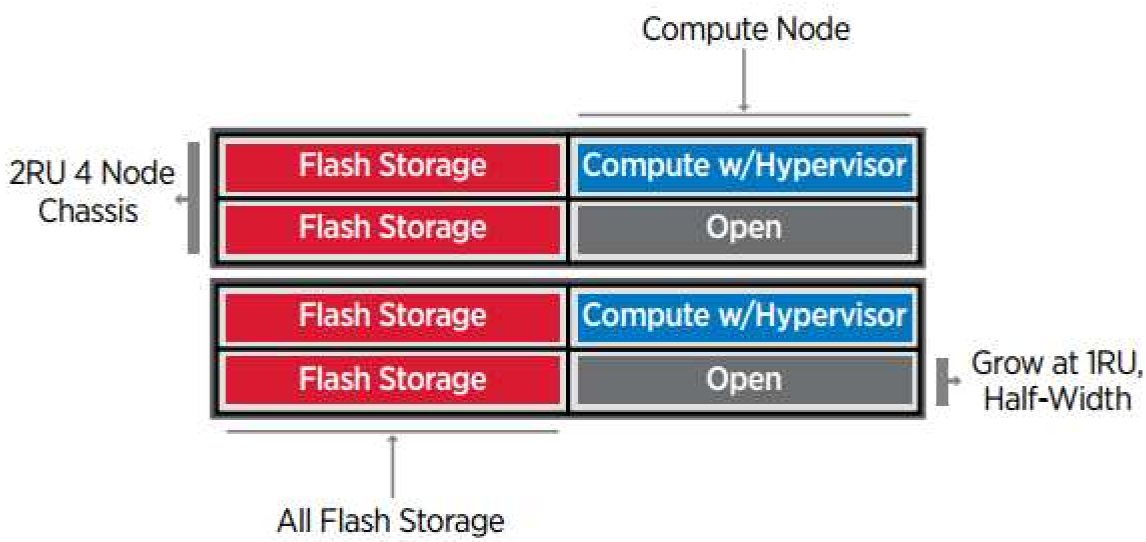
NetApp HCI

By providing an agile turnkey infrastructure platform, NetApp HCI enables you to run enterprise-class virtualized and containerized workloads in an accelerated manner. At its core, NetApp HCI is designed to provide predictable performance, linear scalability of both compute and storage resources, and a simple deployment and management experience.

- **Predictable.** One of the biggest challenges in a multitenant environment is delivering consistent, predictable performance for all your workloads. Running multiple enterprise-grade workloads can result in resource contention, in which one workload might interfere with the performance of another. NetApp HCI alleviates this concern with storage quality-of-service (QoS) limits that are available natively with NetApp Element software. Element enables the granular control of every application and volume, helps to eliminate noisy neighbors, and satisfies enterprise performance SLAs. NetApp HCI multitenancy capabilities can help eliminate many traditional performance-related problems.
- **Flexible.** Previous generations of hyperconverged infrastructures often required fixed resource ratios, limiting deployments to four-node and eight-node configurations. NetApp HCI is a disaggregated hyper-converged infrastructure that can scale compute and storage resources independently. Independent scaling prevents costly and inefficient overprovisioning, eliminates the 10% to 30% HCI tax from controller VM overhead, and simplifies capacity and performance planning. NetApp HCI is available in mix-and-match small, medium, and large storage and compute configurations.
  The architectural design choices offered enable you to confidently scale on your terms, making HCI viable for core Tier 1 data center applications and platforms. NetApp HCI is architected in building blocks at either the chassis or the node level. Each chassis can hold four nodes in a mixed configuration of storage or compute nodes.
- **Simple.** A driving imperative within the IT community is to simplify deployment and automate routine tasks, eliminating the risk of user error while freeing up resources to focus on more interesting, higher-value projects. NetApp HCI can help your IT department become more agile and responsive by both simplifying deployment and ongoing management. The NetApp Deployment Engine (NDE) tool eases the configuration and deployment of physical infrastructure, including the installation of the VMware vSphere environment and the integration of the NetApp Element Plug-in for vCenter Server. With NDE, future scaling operations can be performed without difficulty.

NetApp HCI configuration

NetApp HCI is an enterprise-scale disaggregated hybrid cloud infrastructure (HCI) solution that delivers compute and storage resources in an agile, scalable, and easy-to-manage two-rack unit (2RU) four-node building block. It can also be configured with 1RU compute and server nodes. The NetApp HCI deployment referenced in this guide consists of four NetApp HCI storage nodes and two NetApp HCI compute nodes. The compute nodes are installed as VMware ESXi hypervisors in an HA cluster without the enforcement of VMware DRS anti-affinity rules. This minimum deployment can be easily scaled to fit customer enterprise workload demands by adding additional NetApp HCI storage or compute nodes to expand available storage. The following figure depicts the minimum configuration for NetApp HCI.



The design for NetApp HCI for Anthos consists of the following components in a minimum starting configuration:

- NetApp H-Series all-flash storage nodes running NetApp Element software
- NetApp H-Series compute nodes running VMware vSphere 6.7U3

For more information about compute and storage nodes in NetApp HCI, see the NetApp HCI Datasheet.

NetApp Element software

NetApp Element software provides modular, scalable performance, with each storage node delivering guaranteed capacity and throughput to the environment. You can also specify per-volume storage QoS policies to support dedicated performance levels for even the most demanding workloads.

iSCSI login redirection and self-healing capabilities

NetApp Element software uses the iSCSI storage protocol, a standard way to encapsulate SCSI commands on a traditional TCP/IP network. When SCSI standards change or when Ethernet network performance improves, the iSCSI storage protocol benefits without the need for any changes.

Although all storage nodes have a management IP and a storage IP, NetApp Element software advertises a single storage virtual IP address (SVIP address) for all storage traffic in the cluster. As a part of the iSCSI login process, storage can respond that the target volume has been moved to a different address, and therefore it cannot proceed with the negotiation process. The host then reissues the login request to the new address in a process that requires no host-side reconfiguration. This process is known as iSCSI login redirection.

iSCSI login redirection is a key part of the NetApp Element software cluster. When a host login request is received, the node decides which member of the cluster should handle the traffic based on IOPS and the capacity requirements for the volume. Volumes are distributed across the NetApp Element software cluster and are redistributed if a single node is handling too much traffic for its volumes or if a new node is added. Multiple copies of a given volume are allocated across the array. In this manner, if a node failure is followed by volume redistribution, there is no effect on host connectivity beyond a logout and login with redirection to the new location. With iSCSI login redirection, a NetApp Element software cluster is a self-healing, scale-out architecture that is capable of nondisruptive upgrades and operations.

NetApp Element software cluster QoS

A NetApp Element software cluster allows QoS to be dynamically configured on a per-volume basis. You can use per-volume QoS settings to control storage performance based on SLAs that you define. The following three configurable parameters define the QoS:

- **Minimum IOPS.** The minimum number of sustained IOPS that the NetApp Element software cluster provides to a volume. The minimum IOPS configured for a volume is the guaranteed level of performance for a volume. Per-volume performance does not drop below this level.
- **Maximum IOPS.** The maximum number of sustained IOPS that the NetApp Element software cluster provides to a specific volume.
- **Burst IOPS.** The maximum number of IOPS allowed in a short burst scenario. The burst duration setting is configurable, with a default of 1 minute. If a volume has been running below the maximum IOPS level, burst credits are accumulated. When performance levels become very high and are pushed, short bursts of IOPS beyond the maximum IOPS are allowed on the volume.

Multitenancy

Secure multitenancy is achieved with the following features:

- **Secure authentication.** The Challenge-Handshake Authentication Protocol (CHAP) is used for secure volume access. The Lightweight Directory Access Protocol (LDAP) is used for secure access to the cluster for management and reporting.
- **Volume access groups (VAGs).** Optionally, VAGs can be used in lieu of authentication, mapping any number of iSCSI initiator-specific iSCSI Qualified Names (IQNs) to one or more volumes. To access a volume in a VAG, the initiator's IQN must be in the allowed IQN list for the group of volumes.
- **Tenant virtual LANs (VLANs).** At the network level, end-to-end network security between iSCSI initiators and the NetApp Element software cluster is facilitated by using VLANs. For any VLAN that is created to isolate a workload or a tenant, NetApp Element Software creates a separate iSCSI target SVIP address that is accessible only through the specific VLAN.
- **VPN routing/forwarding (VFR)-enabled VLANs.** To further support security and scalability in the data center, NetApp Element software allows you to enable any tenant VLAN for VRF-like functionality. This feature adds these two key capabilities:
  - **L3 routing to a tenant SVIP address.** This feature allows you to situate iSCSI initiators on a separate network or VLAN from that of the NetApp Element software cluster.
  - **Overlapping or duplicate IP subnets.** This feature enables you to add a template to tenant environments, allowing each respective tenant VLAN to be assigned IP addresses from the same IP subnet. This capability can be useful for service provider environments where scale and preservation of IP-space are important.

Enterprise storage efficiencies

The NetApp Element software cluster increases overall storage efficiency and performance. The following features are performed inline, are always on, and require no manual configuration by the user:

- **Deduplication.** The system only stores unique 4K blocks. Any duplicate 4K blocks are automatically associated to an already stored version of the data. Data is on block drives and is mirrored by using Element Helix data protection. This system significantly reduces capacity consumption and write operations within the system.
- **Compression.** Compression is performed inline before data is written to NVRAM. Data is compressed, stored in 4K blocks, and remains compressed in the system. This compression significantly reduces

capacity consumption, write operations, and bandwidth consumption across the cluster.

- **Thin provisioning.** This capability provides the right amount of storage at the time that you need it, eliminating capacity consumption that caused by overprovisioned volumes or underutilized volumes.
- **Helix.** The metadata for an individual volume is stored on a metadata drive and is replicated to a secondary metadata drive for redundancy.

Note: Element was designed for automation. All the storage features mentioned above can be managed with APIs. These APIs are the only method that the UI uses to control the system whether actions are performed directly through Element or through the vSphere plug-in for Element.

VMware vSphere

VMware vSphere is the industry leading virtualization solution built on VMware ESXi hypervisors and managed by vCenter Server, which provides advanced functionality often required for enterprise datacenters. When using the NDE with NetApp HCI, a VMware vSphere environment is configured and installed. The following features are available after the environment is deployed:

- **Centralized Management.** Through vSphere, individual hypervisors can be grouped into data centers and combined into clusters, allowing for advanced organization to ease the overall management of resources.
- **VMware HA.** This feature allows virtual guests to restart automatically if their host becomes unavailable. By enabling this feature, virtual guests become fault tolerant, and virtual infrastructures experience minimal disruption when there are physical failures in the environment.
- **VMware Distributed Resource Scheduler (DRS).** VMware vMotion allows for the movement of guests between hosts nondisruptively when certain user-defined thresholds are met. This capability makes the virtual guests in an environment highly available.
- **vSphere Distributed Switch (vDS).** A virtual switch is controlled by the vCenter server, enabling centralized configuration and management of connectivity for each host by creating port groups that map to the physical interfaces on each host.

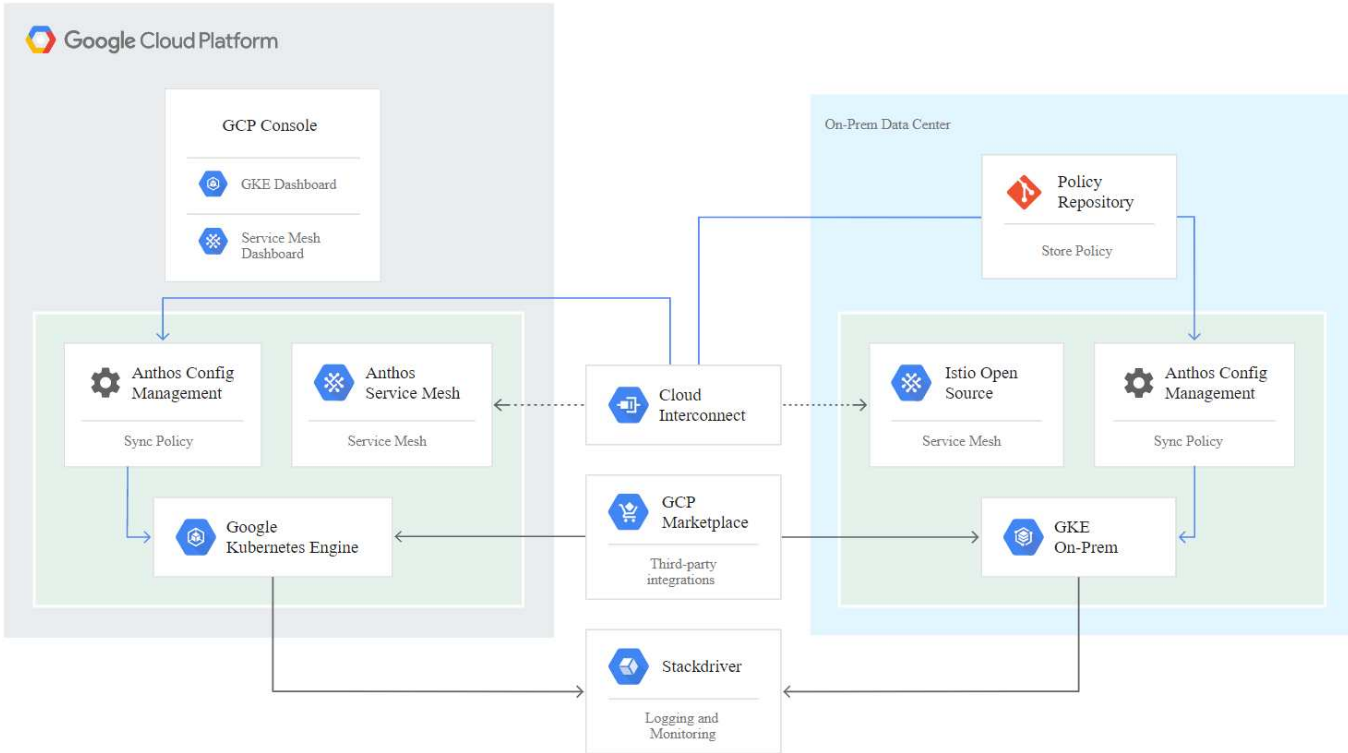
Anthos

Anthos is a hybrid-cloud Kubernetes data center solution that enables organizations to construct and manage modern hybrid-cloud infrastructures, while adopting agile workflows focused on application development. Anthos on VMware, a solution built on open-source technologies, runs on-premises in a VMware vSphere-based infrastructure, which can connect and interoperate with Anthos GKE in Google Cloud.
Adopting containers, service mesh, and other transformational technologies enables organizations to experience consistent application development cycles and production-ready workloads in local and cloud-based environments. The following figure depicts the Anthos solution and how a deployment in an on-premises data center interconnects with infrastructure in the cloud.
For more information about Anthos, see the Anthos website located here.

Anthos provides the following features:

- **Anthos configuration management.** Automates the policy and security of hybrid Kubernetes deployments.
- **Anthos Service Mesh.** Enhances application observability, security, and control with an Istio-powered service mesh.
- **Google Cloud Marketplace for Kubernetes Applications.** A catalog of curated container applications available for easy deployment.
- **Migrate for Anthos.** Automatic migration of physical services and VMs from on-premises to the cloud.

- **Stackdriver.** Management service offered by Google for logging and monitoring cloud instances.



Containers and Kubernetes orchestration

Container technology has been available to developers for a long time. However, it has only recently become a core concept in data center architecture and design as more enterprises have adopted application-specific workload requirements.

A traditional development environment requires a dedicated development host deployed on either a bare-metal or virtual server. Such environments require each application to have its own dedicated machine, complete with operating system (OS) and networking connectivity. These machines often must be managed by the enterprise system administration team, who must account for the application versions installed as well as host OS patches. In contrast, containers by design require less overhead to deploy. All that is needed is the packaging of application code and supporting libraries together, because all other services depend on the host OS. Rather than managing a complete virtual machine (VM) environment, developers can instead focus on the application development process.

As container technology began to find appeal in the enterprise landscape, many enterprise features, such as fault tolerance and application scaling, were both requested and expected. In response, Google partnered with the Linux Foundation to form the Cloud Native Computing Foundation (CNCF). Together, they introduced Kubernetes (K8s), an open-source platform for orchestrating and managing containers. Kubernetes was designed by Google to be a successor to both the Omega and Borg container management platforms that had been used in their data centers in the previous decade.

Anthos GKE

Anthos GKE is a certified distribution of Kubernetes in the Google Cloud. It allows end users to easily deploy managed, production-ready Kubernetes clusters, enabling developers to focus primarily on application development rather than on the management of their environment. Deploying Kubernetes clusters in Anthos GKE offers the following benefits:

- **Simplifying deployment of applications.** Anthos GKE allows for rapid development, deployment, and updates of applications and services. By providing simple descriptions of the expected system resources (compute, memory, and storage) required by the application containers, the Kubernetes Engine automatically provisions and manages the lifecycle of the cluster environment.
- **Ensuring availability of clusters.** The environment is made extremely accessible and easy to manage by using the dashboard built into the Google Cloud console. Anthos GKE clusters are continually monitored by Google Site Reliability Engineers (SREs) to make sure that clusters behave as expected by collecting regular metrics and observing the use of assigned system resources. A user can also leverage available health checks to make sure that their deployed applications are highly available and that they can recover easily should something go awry.
- **Securing clusters in Google Cloud.** An end user can ensure that clusters are secure and accessible by customizing network policies available from Google Cloud's Global Virtual Private Cloud. Public services can be placed behind a single global IP address for load balancing purposes. A single IP can help provide high availability for applications and protect against distributed denial of service (DDOS) and other forms of attacks that might hinder service performance.
- **Easily scaling to meet requirements.** An end user can enable auto-scaling on their cluster to easily counter both planned and unexpected increases in application demands. Auto-scaling helps make sure that system resources are always available by increasing capacity during high-demand windows. It also allows the cluster to return to its previous state and size after peak demand wanes.

Anthos on VMware

Anthos on VMware is an extension of the Google Kubernetes Engine that is deployed in an end user's private data center. An organization can deploy the same applications designed to run in containers in Google Cloud in Kubernetes clusters on premises. Anthos on VMware offers the following benefits:

- **Cost savings.** End users can realize significant cost savings by utilizing their own physical resources for their application deployments instead of provisioning resources in their Google Cloud environment.
- **Develop, then publish.** On-premises deployments can be used while applications are in development, which allows for testing of applications in the privacy of a local data center before being made publicly available in the cloud.
- **Security requirements.** Customers with increased security concerns or sensitive data sets that cannot be stored in the public cloud are able to run their applications from the security of their own data centers, thereby meeting organizational requirements.

Next: Design Considerations

Hardware and software requirements

This section describes the hardware and software requirements for the NetApp HCI and Anthos solution.

Hardware requirements

The following table lists the minimum number of hardware components that are required to implement the solution. The hardware components that are used in specific implementations of the solution might vary based on customer requirements.

Hardware	Model	Quantity
NetApp HCI compute nodes	NetApp H410C	2
NetApp HCI storage nodes	NetApp H410S	2

Hardware	Model	Quantity
Data switches	Cisco Nexus 3048	2
Management switches	Mellanox NS2010	2

Software requirements

The following table lists the software components that are required to implement the solution. The software components that are used in any implementation of the solution might vary based on customer requirements.

Software	Purpose	Version
NetApp HCI	Infrastructure (compute/storage)	1.8P1
VMware vSphere	Virtualization	6.7U3
Anthos on VMware	Container orchestration	1.7
F5 Big-IP Virtual Edition	Load balancing	15.0.1
NetApp Trident	Storage management	21.04

Next: Deployment steps.

Deployment Steps

This section provides detailed protocols for implementing the NetApp HCI solution for Anthos.

This deployment is divided into the following high-level tasks:
1. Configure management switches
2. Configure data switches
3. Deploy NetApp HCI with the NetApp Deployment Engine
4. Configure the vCenter Server
5. Deploy and configure the F5 Big-IP Virtual Edition Appliance
6. Complete Anthos prerequisites
7. Deploy the Anthos admin workstation
8. Deploy the admin cluster
9. Deploy user clusters
10. Enable access to cluster with the GKE console
11. Install and configure NetApp Trident storage provisioner

Next: Configure management switches.

1. Configure management switches

Cisco Nexus 3048 switches are used in this deployment procedure to provide 1Gbps connectivity for in- and out-of-band management of the compute and storage nodes. These steps begin after the switches have been racked, powered, and put through the initial setup process. To configure the switches to provide management connectivity to the infrastructure, complete the following steps:

Enable advanced features for Cisco Nexus

Run the following commands on each Cisco Nexus 3048 switch to configure advanced features:

1. Enter configuration mode.

```
Switch-01# configure terminal
```

2. Enable VLAN functionality.

```
Switch-01(config)# feature interface-vlan
```

3. Enable LACP.

```
Switch-01(config)# feature lacp
```

4. Enable virtual port channels (vPCs).

```
Switch-01(config)# feature vpc
```

5. Set the global port-channel load-balancing configuration.

```
Switch-01(config)# port-channel load-balance src-dst ip-l4port
```

6. Perform the global spanning-tree configuration.

```
Switch-01(config)# spanning-tree port type network default
Switch-01(config)# spanning-tree port type edge bpduguard default
```

Configure ports on the switch for in-band management

1. Run the following commands to create VLANs for management purposes.

```
Switch-01(config)# vlan 2
Switch-01(config-vlan)# Name Native_VLAN
Switch-01(config-vlan)# vlan 16
Switch-01(config-vlan)# Name OOB_Network
Switch-01(config-vlan)# vlan 3480
Switch-01(config-vlan)# Name MGMT_Network
Switch-01(config-vlan)# exit
```

2. Configure the ports ETH1/29-32 as VLAN trunk ports that connect to management interfaces on each HCI storage node.

```
Switch-01(config)# int eth 1/29
Switch-01(config-if)# description HCI-STG-01 PortA
Switch-01(config-if)# switchport mode trunk
Switch-01(config-if)# switchport trunk native vlan 2
Switch-01(config-if)# switchport trunk allowed vlan 3480
Switch-01(config-if)# spanning tree port type edge trunk
Switch-01(config-if)# int eth 1/30
Switch-01(config-if)# description HCI-STG-02 PortA
Switch-01(config-if)# switchport mode trunk
Switch-01(config-if)# switchport trunk native vlan 2
Switch-01(config-if)# switchport trunk allowed vlan 3480
Switch-01(config-if)# spanning tree port type edge trunk
Switch-01(config-if)# int eth 1/31
Switch-01(config-if)# description HCI-STG-03 PortA
Switch-01(config-if)# switchport mode trunk
Switch-01(config-if)# switchport trunk native vlan 2
Switch-01(config-if)# switchport trunk allowed vlan 3480
Switch-01(config-if)# spanning tree port type edge trunk
Switch-01(config-if)# int eth 1/32
Switch-01(config-if)# description HCI-STG-04 PortA
Switch-01(config-if)# switchport mode trunk
Switch-01(config-if)# switchport trunk native vlan 2
Switch-01(config-if)# switchport trunk allowed vlan 3480
Switch-01(config-if)# spanning tree port type edge trunk
Switch-01(config-if)# exit
```

Configure ports on the switch for out-of-band management

1. Run the following commands to configure the ports for cabling the IPMI interfaces on each HCI node.

```
Switch-01(config)# int eth 1/13
Switch-01(config-if)# description HCI-CMP-01 IPMI
Switch-01(config-if)# switchport mode access
Switch-01(config-if)# switchport access vlan 16
Switch-01(config-if)# spanning-tree port type edge
Switch-01(config-if)# int eth 1/14
Switch-01(config-if)# description HCI-STG-01 IPMI
Switch-01(config-if)# switchport mode access
Switch-01(config-if)# switchport access vlan 16
Switch-01(config-if)# spanning-tree port type edge
Switch-01(config-if)# int eth 1/15
Switch-01(config-if)# description HCI-STG-03 IPMI
Switch-01(config-if)# switchport mode access
Switch-01(config-if)# switchport access vlan 16
Switch-01(config-if)# spanning-tree port type edge
Switch-01(config-if)# exit
```

In the validated configuration, we cabled odd-node IPMI interfaces to Switch-01, and even-node IPMI interfaces to Switch-02.

Create a vPC domain to ensure fault tolerance

1. Activate the ports used for the vPC peer-link between the two switches.

```
Switch-01(config)# int eth 1/1
Switch-01(config-if)# description vPC peer-link Switch-02 1/1
Switch-01(config-if)# int eth 1/2
Switch-01(config-if)# description vPC peer-link Switch-02 1/2
Switch-01(config-if)# exit
```

2. Perform the vPC global configuration.

```
Switch-01(config)# vpc domain 1
Switch-01(config-vpc-domain)# role priority 10
Switch-01(config-vpc-domain)# peer-keepalive destination <switch-
02_mgmt_address> source <switch-01_mgmt_address> vrf managment
Switch-01(config-vpc-domain)# peer-gateway
Switch-01(config-vpc-domain)# auto recovery
Switch-01(config-vpc-domain)# ip arp synchronize
Switch-01(config-vpc-domain)# int eth 1/1-2
Switch-01(config-vpc-domain)# channel-group 10 mode active
Switch-01(config-vpc-domain)# int Po10
Switch-01(config-if)# description vPC peer-link
Switch-01(config-if)# switchport mode trunk
Switch-01(config-if)# switchport trunk native vlan 2
Switch-01(config-if)# switchport trunk allowed vlan 16,3480
Switch-01(config-if)# spanning-tree port type network
Switch-01(config-if)# vpc peer-link
Switch-01(config-if)# exit
```

Next: Configure Data Switches

2. Configure Data Switches

Mellanox SN2010 switches provide 25Gbps connectivity for the data plane of the compute and storage nodes. To configure the switches to provide data connectivity to the infrastructure, complete the following steps:

Create MLAG cluster to provide fault tolerance

1. Run the following commands on each Mellanox SN210 switch for general configuration:
   a. Enter configuration mode.

   ```
   Switch-01 enable
   Switch-01 configure terminal
   ```

   b. Enable the LACP required for the Inter-Peer Link (IPL).

   ```
   Switch-01 (config) # lacp
   ```

   c. Enable the Link Layer Discovery Protocol (LLDP).

   ```
   Switch-01 (config) # lldp
   ```

   d. Enable IP routing.

```
Switch-01 (config) # ip routing
```

e. Enable the MLAG protocol.

```
Switch-01 (config) # protocol mlag
```

f. Enable global QoS.

```
Switch-01 (config) # dcb priority-flow-control enable force
```

2. For MLAG to function, the switches must be made peers to each other through an IPL. This should consist of two or more physical links for redundancy. The MTU for the IPL is set for jumbo frames (9216), and all VLANs are enabled by default. Run the following commands on each switch in the domain:

a. Create port channel 10 for the IPL.

```
Switch-01 (config) # interface port-channel 10
Switch-01 (config interface port-channel 10) # description IPL
Switch-01 (config interface port-channel 10) # exit
```

b. Add interfaces ETH 1/20 and 1/22 to the port channel.

```
Switch-01 (config) # interface ethernet 1/20 channel-group 10 mode
active
Switch-01 (config) # interface ethernet 1/20 description ISL-SWB_01
Switch-01 (config) # interface ethernet 1/22 channel-group 10 mode
active
Switch-01 (config) # interface ethernet 1/22 description ISL-SWB_02
```

c. Create a VLAN outside of the standard range dedicated to IPL traffic.

```
Switch-01 (config) # vlan 4000
Switch-01 (config vlan 4000) # name IPL VLAN
Switch-01 (config vlan 4000) # exit
```

d. Define the port channel as the IPL.

```
Switch-01 (config) # interface port-channel 10 ipl 1
Switch-01 (config) # interface port-channel 10 dcb priority-flow-
control mode on force
```

e. Set an IP for each IPL member (non-routable; it is not advertised outside of the switch).

```
Switch-01 (config) # interface vlan 4000
Switch-01 (config vlan 4000) # ip address 10.0.0.1 255.255.255.0
Switch-01 (config vlan 4000) # ipl 1 peer-address 10.0.0.2
Switch-01 (config vlan 4000) # exit
```

3. Create a unique MLAG domain name for the two switches and assign an MLAG virtual IP (VIP). This IP is used for keep-alive heartbeat messages between the two switches. Run these commands on each switch in the domain:

   a. Create the MLAG domain and set the IP address and subnet.

```
Switch-01 (config) # mlag-vip MLAG-VIP-DOM ip a.b.c.d /24 force
```

   b. Create a virtual MAC address for the system MLAG.

```
Switch-01 (config) # mlag system-mac AA:BB:CC:DD:EE:FF
```

   c. Configure the MLAG domain so that it is active globally.

```
Switch-01 (config) # no mlag shutdown
```

> The IP used for the MLAG VIP must be in the same subnet as the switch management network (mgmt0).

> The MAC address used can be any unicast MAC address and must be set to the same value on both switches in the MLAG domain.

Configure ports to connect to storage and compute hosts

1. Create each of the VLANs needed to support the services for NetApp HCI. Run these commands on each switch in the domain:

   a. Create VLANs.

```
Switch-01 (config) # vlan 1172
Switch-01 (config vlan 1172) exit
Switch-01 (config) # vlan 3480-3482
Switch-01 (config vlan 3480-3482) exit
```

   b. Create names for each VLAN for easier accounting.

```
Switch-01 (config) # vlan 1172 name “VM_Network”
Switch-01 (config) # vlan 3480 name “MGMT_Network”
Switch-01 (config) # vlan 3481 name “Storage_Network”
Switch-01 (config) # vlan 3482 name “vMotion_Network”
+
```

2. Create hybrid VLAN ports on ports ETH1/9-10 so that you can tag the appropriate VLANs for the NetApp HCI compute nodes.
   a. Select the ports you want to work with.

   ```
   Switch-01 (config) # interface ethernet 1/9-1/10
   ```

   b. Set the MTU for each port.

   ```
   Switch-01 (config interface ethernet 1/9-1/10) # mtu 9216 force
   ```

   c. Modify spanning-tree settings for each port.

   ```
   Switch-01 (config interface ethernet 1/9-1/10) # spanning-tree
   bpdufilter enable
   Switch-01 (config interface ethernet 1/9-1/10) # spanning-tree port
   type edge
   Switch-01 (config interface ethernet 1/9-1/10) # spanning-tree
   bpduguard enable
   ```

   d. Set the switchport mode to hybrid.

   ```
   Switch-01 (config interface ethernet 1/9-1/10 ) # switchport mode
   hybrid
   Switch-01 (config interface ethernet 1/9-1/10 ) # exit
   ```

   e. Create descriptions for each port being modified.

   ```
   Switch-01 (config) # interface ethernet 1/9 description HCI-CMP-01
   PortD
   Switch-01 (config) # interface ethernet 1/10 description HCI-CMP-02
   PortD
   ```

   f. Tag the appropriate VLANs for the NetApp HCI environment.

```
Switch-01 (config) # interface ethernet 1/9 switchport hybrid
allowed-vlan add 1172
Switch-01 (config) # interface ethernet 1/9 switchport hybrid
allowed-vlan add 3480-3482
Switch-01 (config) # interface ethernet 1/10 switchport hybrid
allowed-vlan add 1172
Switch-01 (config) # interface ethernet 1/10 switchport hybrid
allowed-vlan add 3480-3482
```

3. Create MLAG interfaces and hybrid VLAN ports on ports ETH1/5-8 so that you can distribute connectivity between the switches and tag the appropriate VLANs for the NetApp HCI storage nodes.
   a. Select the ports that you want to work with.

```
Switch-01 (config) # interface ethernet 1/5-1/8
```

   b. Set the MTU for each port.

```
Switch-01 (config interface ethernet 1/5-1/8) # mtu 9216 force
```

   c. Modify spanning tree settings for each port.

```
Switch-01 (config interface ethernet 1/5-1/8) # spanning-tree
bpdufilter enable
Switch-01 (config interface ethernet 1/5-1/8) # spanning-tree port
type edge
Switch-01 (config interface ethernet 1/5-1/8) # spanning-tree
bpduguard enable
```

   d. Set the switchport mode to hybrid.

```
Switch-01 (config interface ethernet 1/5-1/8 ) # switchport mode
hybrid
Switch-01 (config interface ethernet 1/5-1/8 ) # exit
```

   e. Create descriptions for each port being modified.

```
Switch-01 (config) # interface ethernet 1/5 description HCI-STG-01
PortD
Switch-01 (config) # interface ethernet 1/6 description HCI-STG-02
PortD
Switch-01 (config) # interface ethernet 1/7 description HCI-STG-03
PortD
Switch-01 (config) # interface ethernet 1/8 description HCI-STG-04
PortD
```

f. Create and configure the MLAG port channels.

```
Switch-01 (config) # interface mlag-port-channel 115-118
Switch-01 (config interface mlag-port-channel 115-118) # exit
Switch-01 (config) # interface mlag-port-channel 115-118 no shutdown
Switch-01 (config) # interface mlag-port-channel 115-118 mtu 9216
force
Switch-01 (config) # interface mlag-port-channel 115-118 lacp-
individual enable force
Switch-01 (config) # interface ethernet 1/5-1/8 lacp port-priority 10
Switch-01 (config) # interface ethernet 1/5-1/8 lacp rate fast
Switch-01 (config) # interface ethernet 1/5 mlag-channel-group 115
mode active
Switch-01 (config) # interface ethernet 1/6 mlag-channel-group 116
mode active
Switch-01 (config) # interface ethernet 1/7 mlag-channel-group 117
mode active
Switch-01 (config) # interface ethernet 1/8 mlag-channel-group 118
mode active
```

g. Tag the appropriate VLANs for the storage environment.

```
Switch-01 (config) # interface mlag-port-channel 115-118 switchport
mode hybrid
Switch-01 (config) # interface mlag-port-channel 115 switchport
hybrid allowed-vlan add 1172 Switch-01 (config) # interface mlag-
port-channel 116 switchport hybrid allowed-vlan add 1172
Switch-01 (config) # interface mlag-port-channel 117 switchport
hybrid allowed-vlan add 1172
Switch-01 (config) # interface mlag-port-channel 118 switchport
hybrid allowed-vlan add 1172
Switch-01 (config) # interface mlag-port-channel 115 switchport
hybrid allowed-vlan add 3481
Switch-01 (config) # interface mlag-port-channel 116 switchport
hybrid allowed-vlan add 3481
Switch-01 (config) # interface mlag-port-channel 117 switchport
hybrid allowed-vlan add 3481
Switch-01 (config) # interface mlag-port-channel 118 switchport
hybrid allowed-vlan add 3481
```

The configurations in this section must also be run on the second switch in the MLAG domain. NetApp recommends that the descriptions for each port are updated to reflect the device ports that are cabled and configured on the other switch.

Create uplink ports for the switches

1. Create an MLAG interface to provide uplinks to both Mellanox SN2010 switches from the core network.

```
Switch-01 (config) # interface mlag port-channel 101
Switch-01 (config interface mlag port-channel) # description Uplink
CORE-SWITCH port PORT
Switch-01 (config interface mlag port-channel) # exit
```

2. Configure the MLAG members.

```
Switch-01 (config) # interface ethernet 1/18 description Uplink to CORE-
SWITCH port PORT
Switch-01 (config) # interface ethernet 1/18 speed 10000 force
Switch-01 (config) # interface mlag-port-channel 101 mtu 9216 force
Switch-01 (config) # interface ethernet 1/18 mlag-channel-group 101 mode
active
```

3. Set the switchport mode to hybrid and allow all VLANs from the core uplink switches.

```
Switch-01 (config) # interface mlag-port-channel switchport mode hybrid
Switch-01 (config) # interface mlag-port-channel switchport hybrid
allowed-vlan all
```

4. Verify that the MLAG interface is up.

```
Switch-01 (config) # interface mlag-port-channel 101 no shutdown
Switch-01 (config) # exit
```

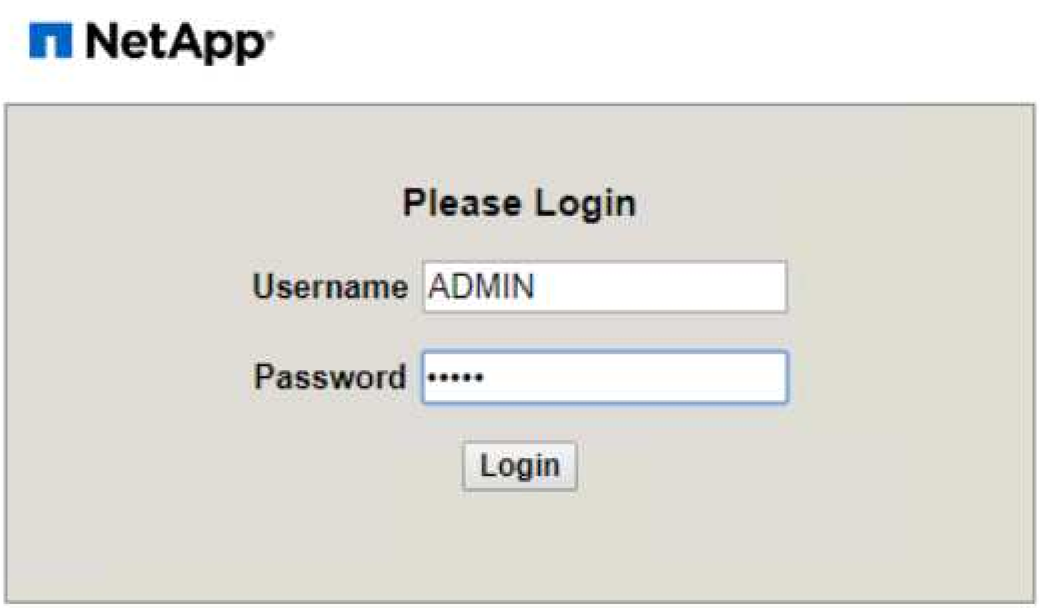
Next: Deploy NetApp HCI with the NetApp Deployment Engine

3. Deploy NetApp HCI with the NetApp Deployment Engine

NDE delivers a simple and streamlined deployment experience for the NetApp HCI solution. A detailed guide to using NDE 1.6 to deploy your NetApp HCI system can be found here.

These steps begin after the nodes have been racked, and cabled, and the IPMI port has been configured on each node using the console. To Deploy the NetApp HCI solution using NDE, complete the following steps:

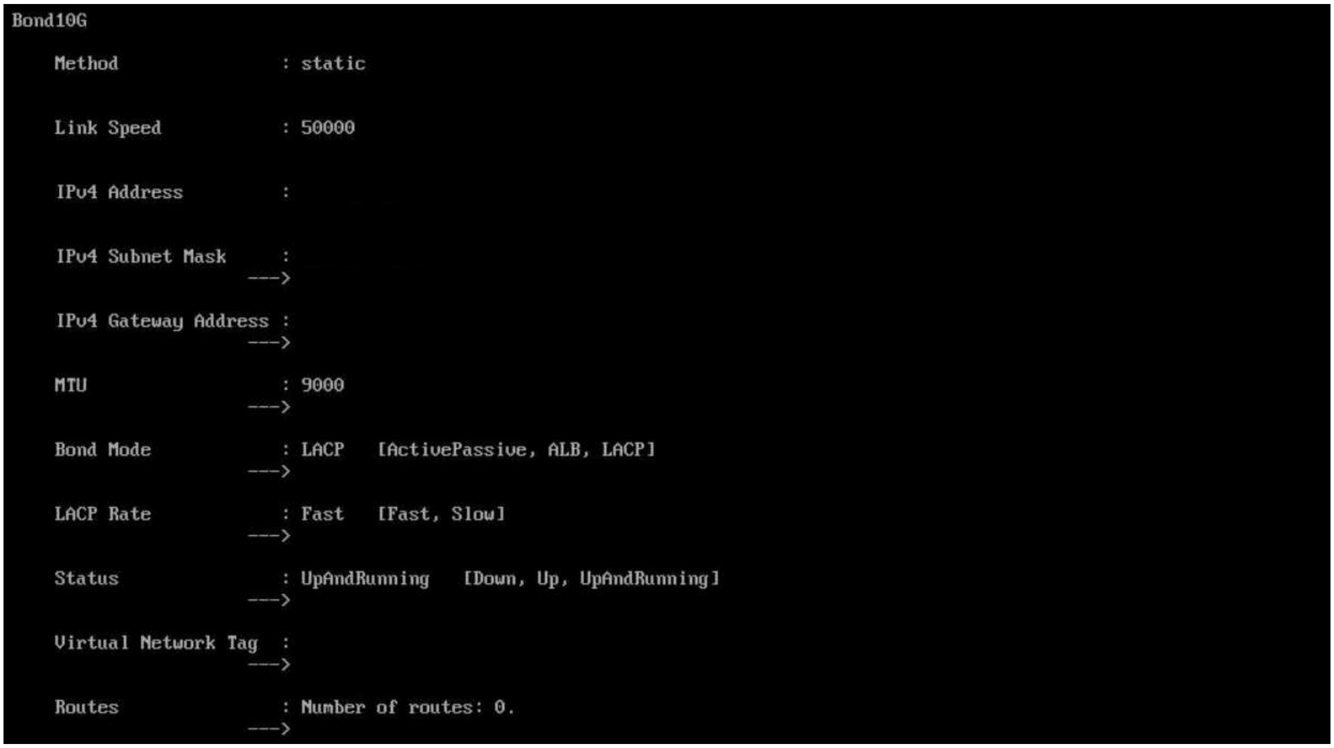
1. Access the out-of-band management console for one of the storage nodes in the cluster and log in with the default credentials ADMIN/ADMIN.



2. Click the Remote Console Preview image in the center of the screen to download a JNLP file launched by Java Web Start, which launches an interactive console to the system.
3. With the virtual console launched, a user can log in to the HCI storage node using the ADMIN/ADMIN

username and password combination.

4. The Bond1G interface must have an IP, a netmask, and a gateway set statically; its VLAN set to 3480; and DNS servers defined for the environment.

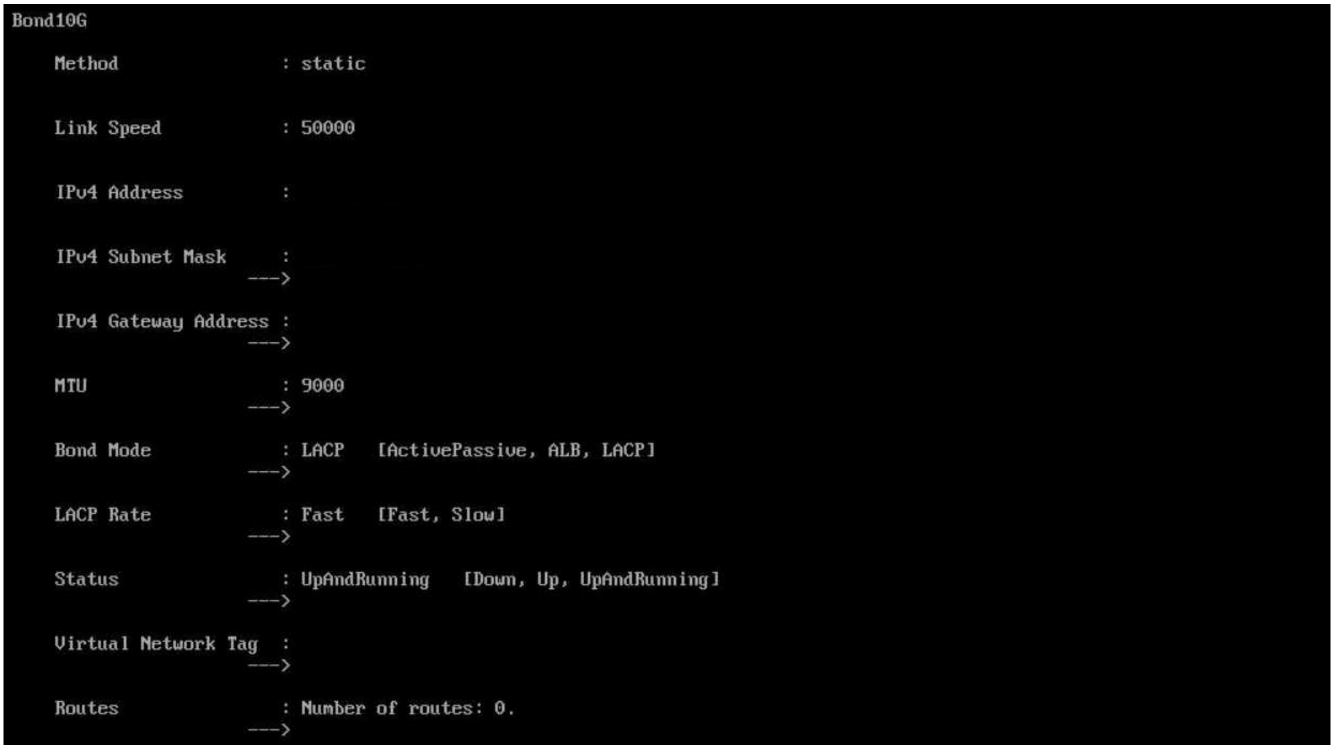


```
Bond10G
    Method                : static

    Link Speed            : 50000

    IPv4 Address          :

    IPv4 Subnet Mask      :
                       --->
    IPv4 Gateway Address :
                       --->
    MTU                   : 9000
                       --->
    Bond Mode             : LACP   [ActivePassive, ALB, LACP]
                       --->
    LACP Rate             : Fast   [Fast, Slow]
                       --->
    Status                : UpAndRunning   [Down, Up, UpAndRunning]
                       --->
    Virtual Network Tag   :
                       --->
    Routes                : Number of routes: 0.
                       --->
```

> Select an IP that is within the subnet you intend to use for in-band management but not an IP you would like to use in production. NDE reconfigures the node with a production IP after initial access.

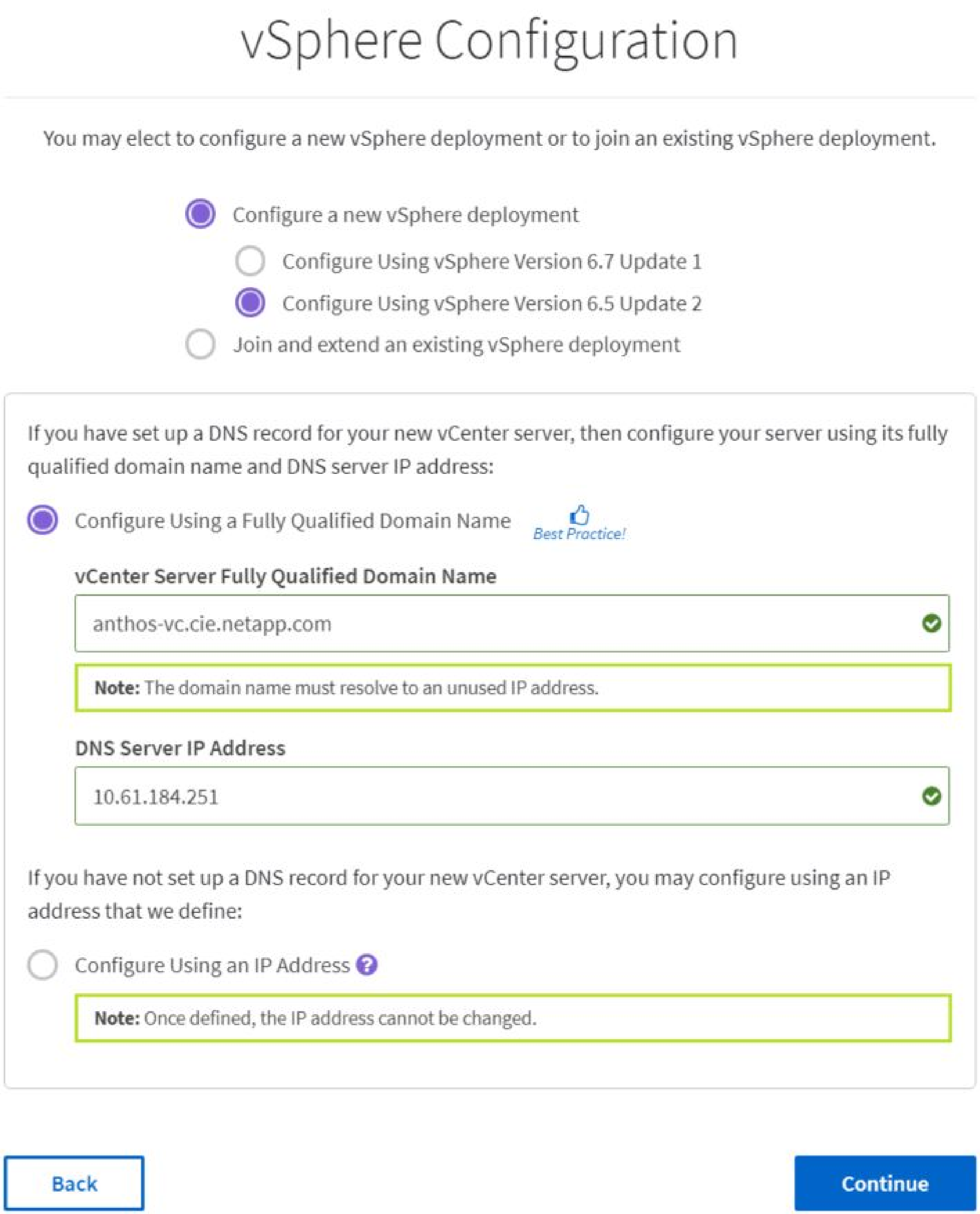
> This task must only be performed on the first storage node. Afterward, the other nodes in the infrastructure are discovered by the Automatic Private IP Address (APIPA) addresses assigned to each storage interface when left unconfigured.

5. The Bond 10G interface must have its MTU setting changed to enable jumbo frames and its bond mode changed to LACP.

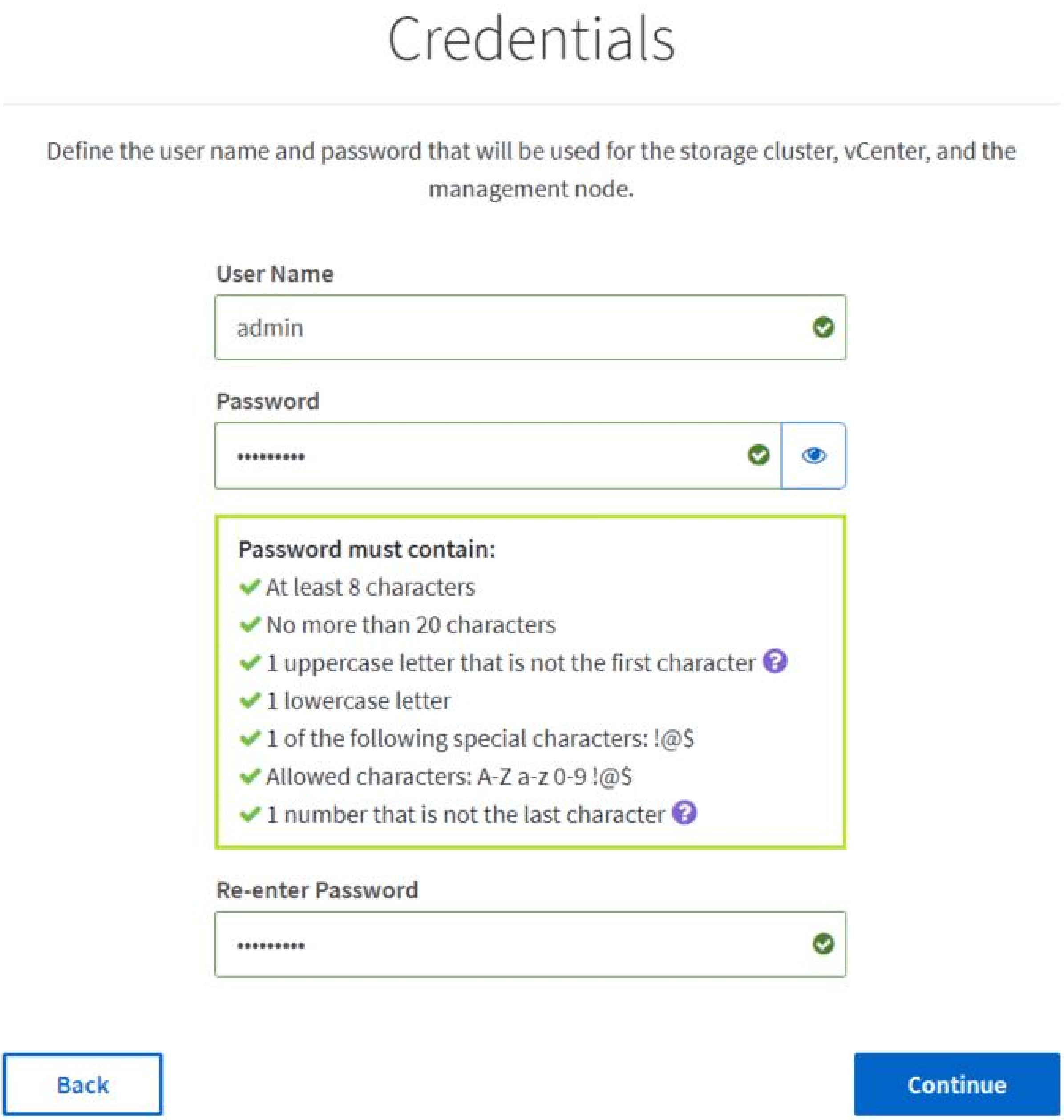


> ⓘ Configure each of the four storage nodes in the NetApp HCI solution this way. The NDE process is then able to discover all the nodes in the solution and configure them. You do not need to modify the Bond10g interfaces on the two compute nodes.

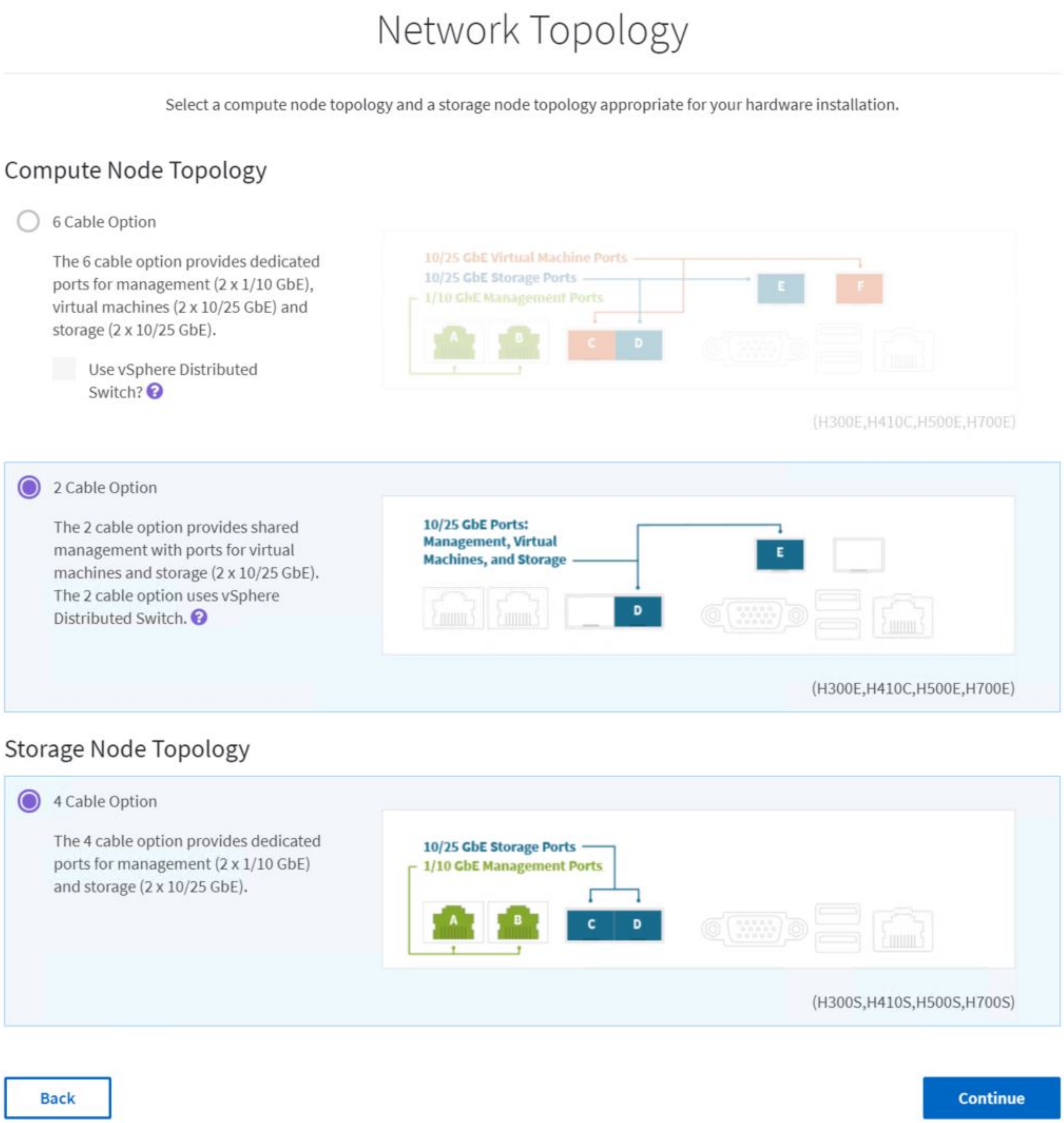
6. After completion, open a web browser and visit the IP address you configured for the management port to start NetApp HCI configuration with NDE.
7. On the Welcome to NetApp HCI page, click the Get Started button.
8. Check each associated box on the Prerequisites page and click Continue.
9. The next page presents End User Licenses for NetApp HCI and VMware vSphere. If you accept the terms, click I Accept at the end of each agreement and then click Continue.
10. Click Configure a New vSphere Deployment, select vSphere 6.5U2, and enter the Fully Qualified Domain Name (FQDN) of your vCenter Server. Then click Continue.



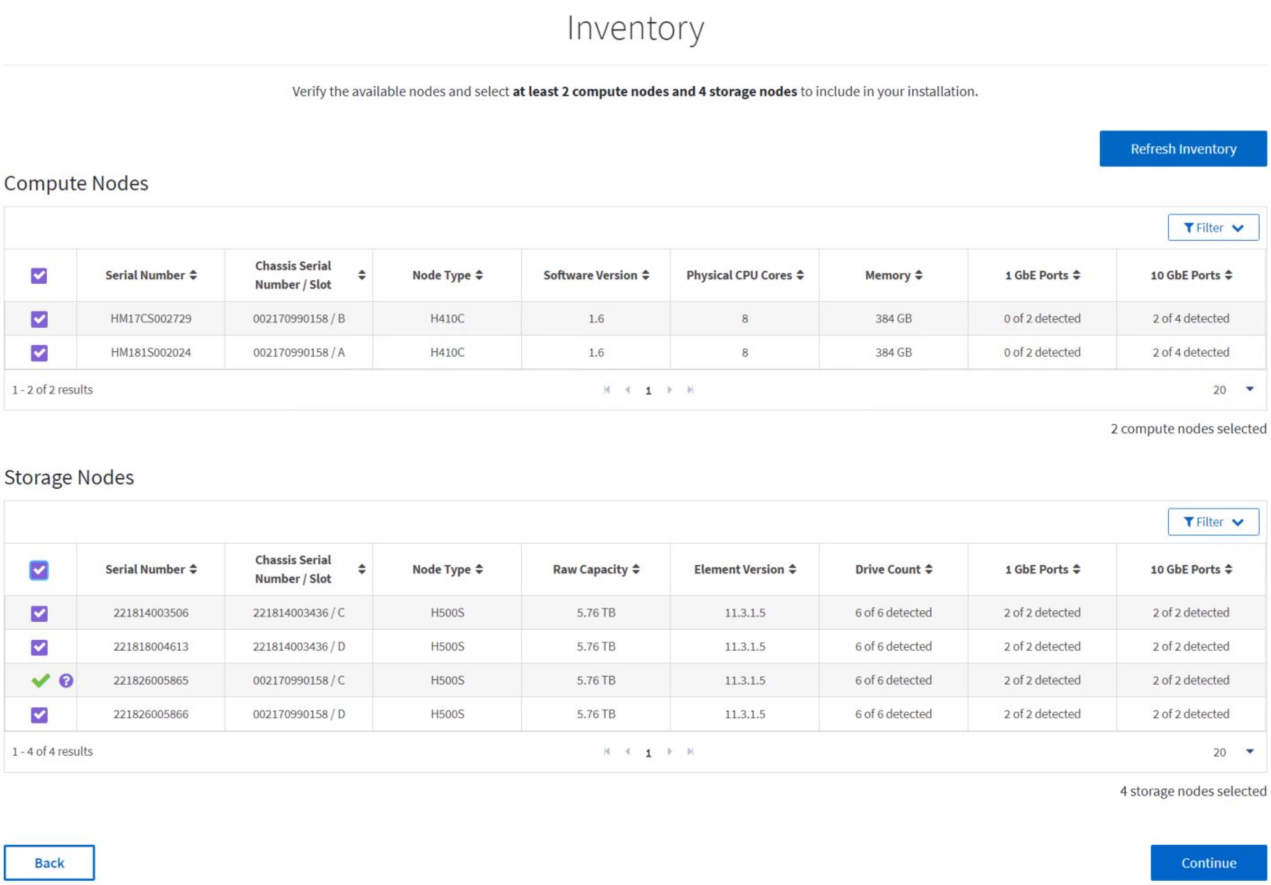
11. NDE asks for the credentials to be used in the environment. This is used for VMware vSphere, the NetApp Element storage cluster, and the NetApp Mnode, which provides management functionality for the cluster. When you are finished, click Continue.



12. NDE then prompts for the network topology used to cable the NetApp HCI environment. The validated solution in this document has been deployed using the two-cable option for the compute nodes, and the four-cable option for the storage nodes. Click Continue.

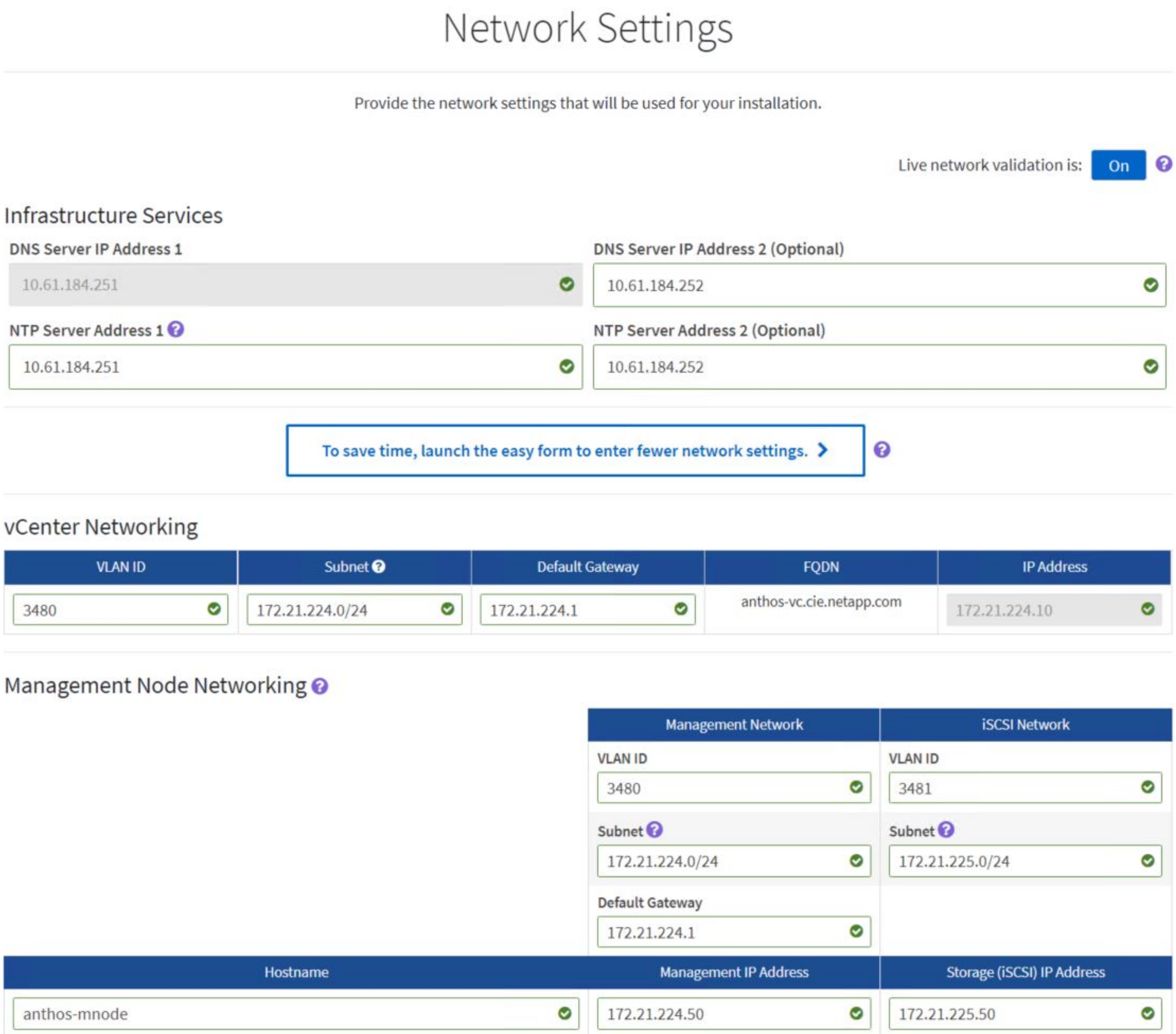


13. The next page presented by NDE is the inventory of the environment as discovered by the APIPA addressed on the storage network. The storage node that is currently running NDE is already selected with a green check mark. Select the corresponding boxes to add additional nodes to the NetApp HCI environment. Click Continue.

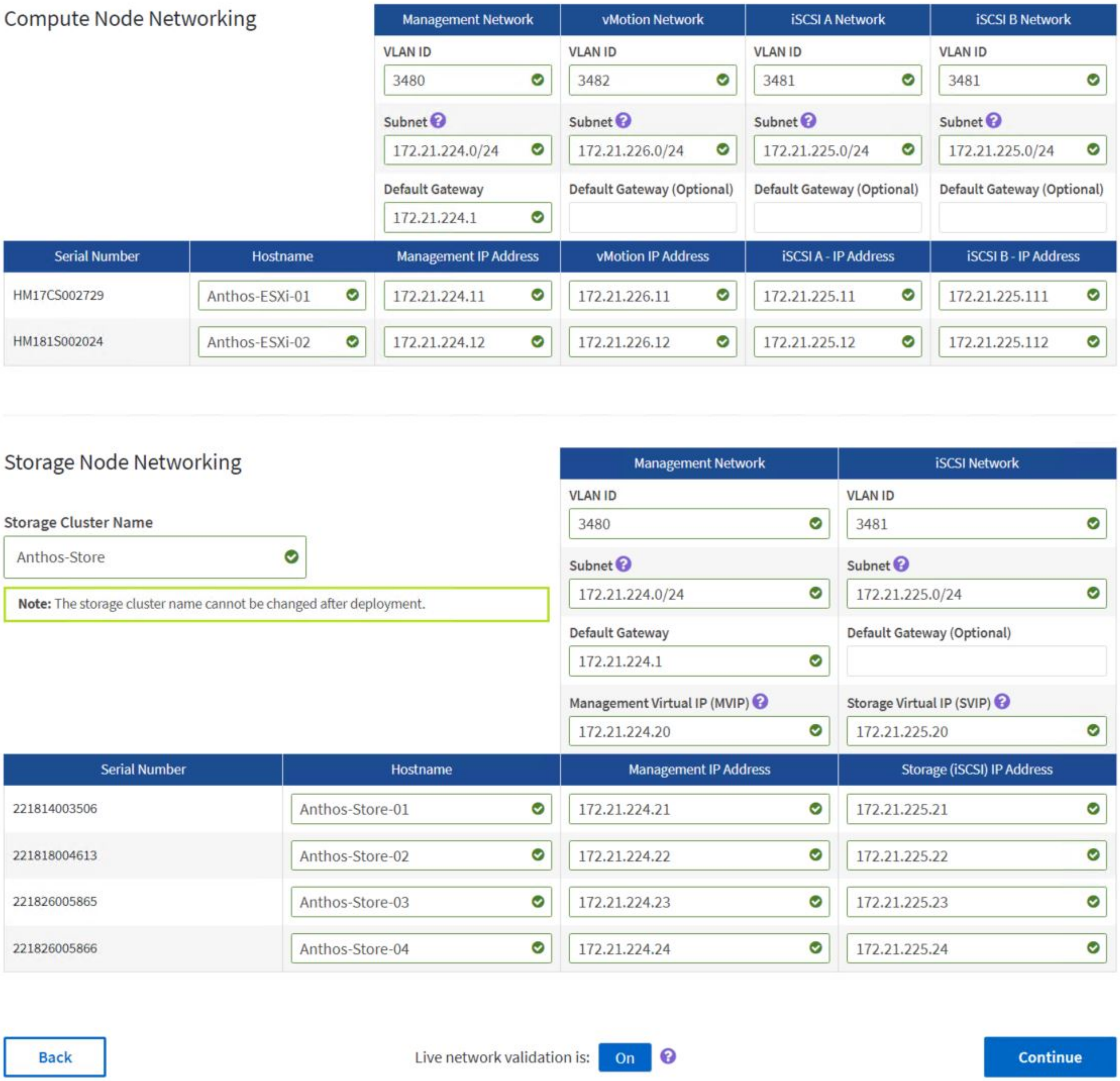


> If there are any nodes missing from the inventory screen, wait a few minutes and click Refresh Inventory. If the node still fails to appear, additional investigation of environment networking might be required.

14. You must next configure the permanent network settings for the NetApp HCI deployment. The first page configures infrastructure services (DNS and NTP), vCenter networking, and Mnode networking.



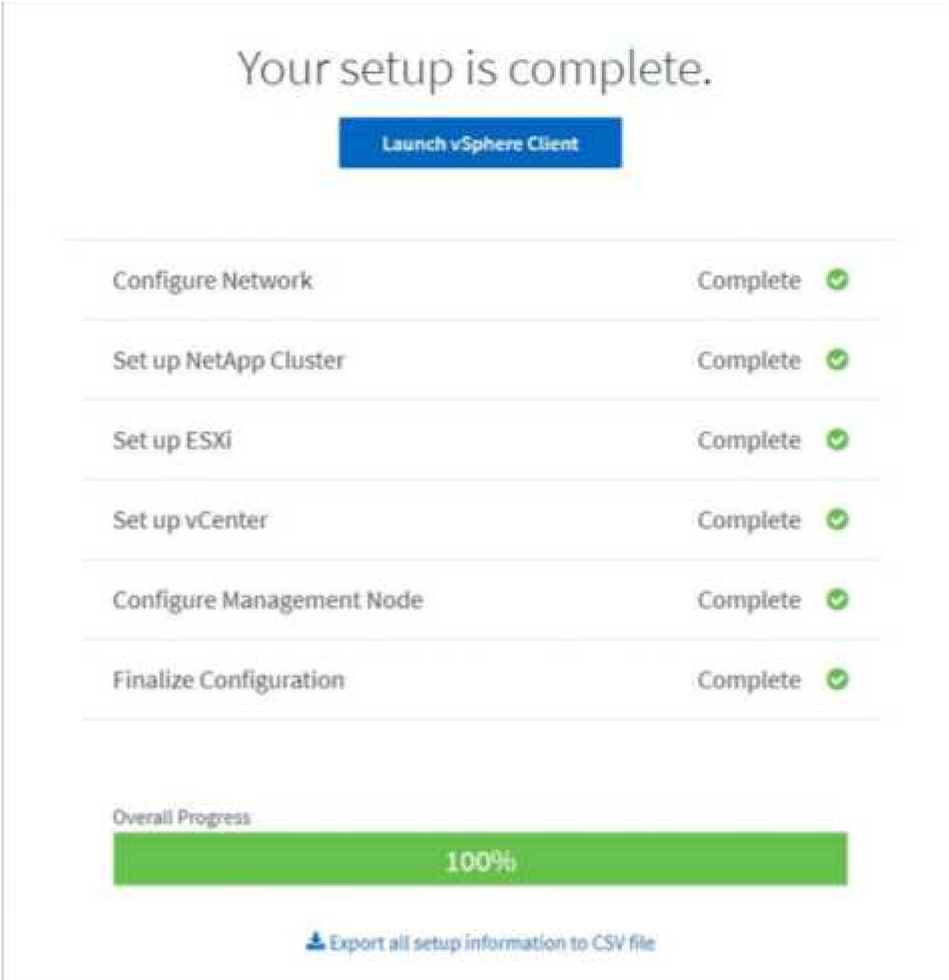
15. The next page allows you to configure each node in the environment. For the compute nodes, it allows you to configure the host name, management network, vMotion network, and storage network. For the storage nodes, name the storage cluster and configure the management and storage networks being used for each node. Click Continue.



16. On the next page, review all the settings that have been defined for the environment by expanding each section, and, if necessary, click Edit to make corrections. There is also a check box on this page that enables or disables the Mnode from sending real-time health and diagnostics information to NetApp Active IQ. If all the information is correct, click Start Deployment.

> If you want to enable Active IQ, verify that your management network can reach the internet. If NDE is unable to reach Active IQ, the deployment can fail.

17. A summary page appears along with a progress bar for each component of the NetApp HCI solution, as well as the overall solution. When complete, you are presented with an option to launch the vSphere client and begin working with your environment.



Next: Configure the vCenter Server

4. Configure the vCenter Server

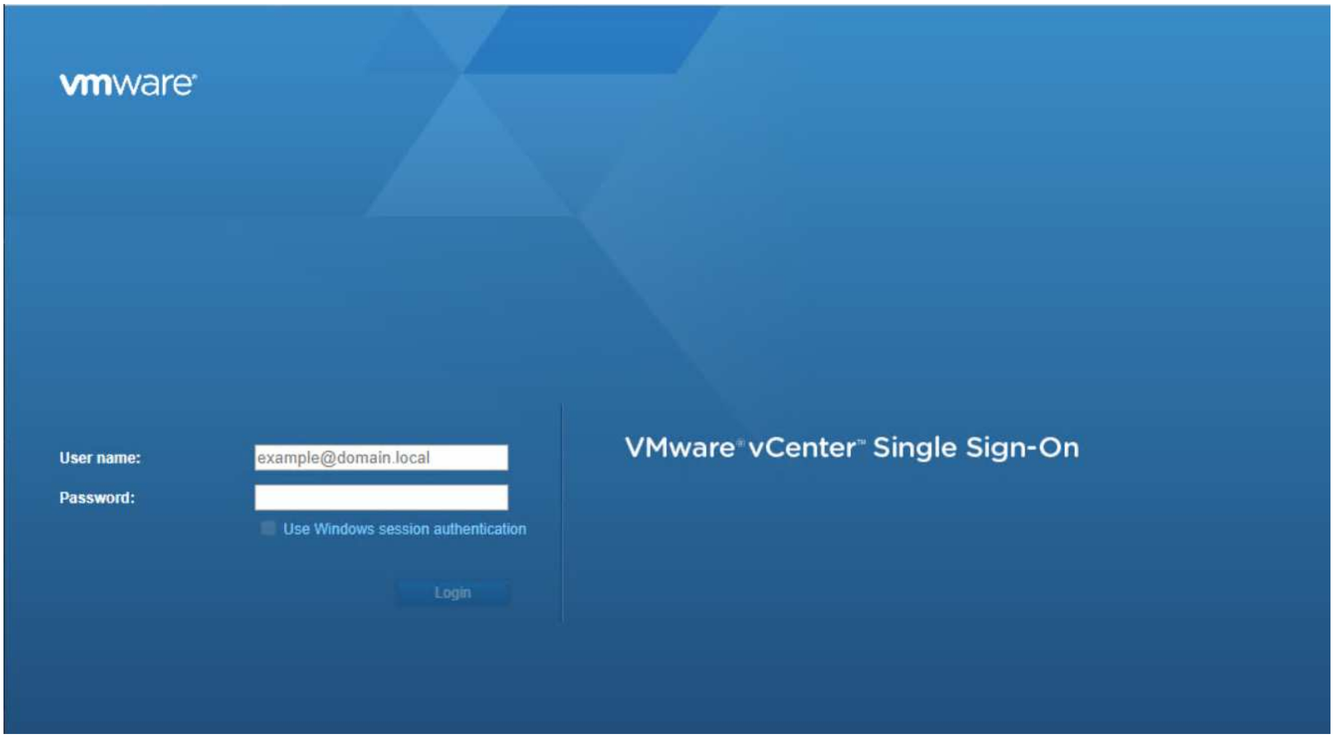
NDE deploys the solution with vCenter server and integrates the solution with the Element cluster by provisioning the Mnode VM and installing the NetApp Element Plug-in for vCenter.

> Note that NDE deploys vSphere 6.7U1. You can upgrade the Virtual Appliance and individual ESXi hosts by following the instructions from VMware here.

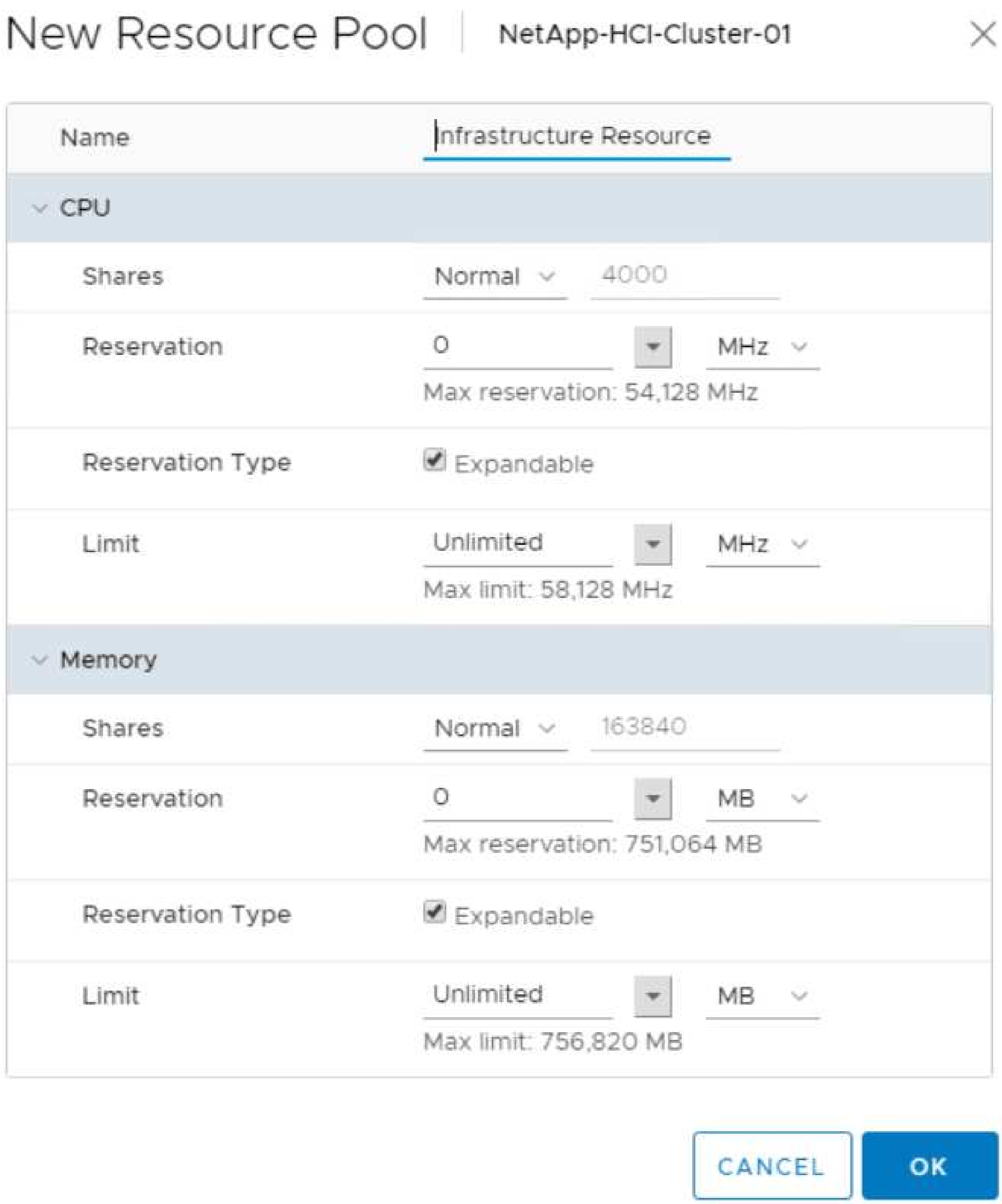
After deployment, you must make a few modifications to the environment, including the creation of additional vDS portgroups, datastores, and resource groups for the deployment of the Anthos on VMware solution.

Complete the following steps to configure your vCenter Server:

1. Log into the VMware vCenter server using the Administrator@vsphere.local account and the password chosen for the admin user during NDE configuration.



2. Right-click `NetApp-HCI-Cluster-01` created by NDE and select the option to create a new resource pool. Name this pool `Infrastructure-Resource-Pool` and accept the defaults by clicking OK. This resource pool is used in a later configuration step.



The reservations in this resource pool can be modified based on the resources available in the environment. NetApp HCI is deployed as an all-in-one solution. Therefore, NetApp recommends reserving the resources necessary to provide availability for the infrastructure services by placing them into this resource pool and adjusting the resources appropriately. Infrastructure services include vCenter Server, NetApp Mnode, and F5 Big-IP Load Balancer.

3. Repeat this step to create another resource pool for VMs deployed by Anthos. Name this pool Anthos-Resource-Pool, and click the OK button to accept the default values. Adjust the resource availability based on the specific environment in which you are deploying the solution. This resource pool is used in a later deployment step.
4. To configure Element volumes to be used as vSphere datastores, click the dropdown menu and select NetApp Element Management from the list.
5. A Getting Started screen appears with details about your Element cluster.

6. Click Management, and the vSphere client presents a list of datastores. Click Create Datastore to create one datastore to host VMs and another to host ISOs for future guest installs.
7. Next click the Network menu item in the left panel. This displays a screen with information about the vDS deployed by NDE.
8. Several virtual port groups are defined by the initial configuration. NetApp recommends leaving these alone to support the infrastructure, and additional port groups should be created for user-deployed virtual guests. Right-click the NetApp HCI VDS 01 vDS in the left panel, and then select Distributed Port Group followed by the New Distributed Port Group option from the expanded menu.
9. Create a new distributed port group called `Management_Network`. Then click Next.
10. On the next screen, select the VLAN type as VLAN, and set the VLAN ID to 3480 for management purposes. Click Next, and, after reviewing the options on the summary page, click Next again to complete the creation of the distributed port group.
11. Repeat these steps to create distributed port groups for the `VM_Network` (VLAN 1172) as well as any other networks that might be used in the NetApp HCI environment.

> Additional networks can be defined to segment any additional deployed VMs. Examples of this use could be for a dedicated HA network for additional F5 Big-IP appliances if provisioned. Such configurations are in addition to the environment deployed in this validated solution and are considered out of scope for this NVA document.

Next: Deploy and Configure the F5 Big-IP Virtual Edition Appliance

5. Deploy and Configure the F5 Big-IP Virtual Edition Appliance

Anthos enables native integration with F5 Big-IP load balancers to expose services from each pod to the world.

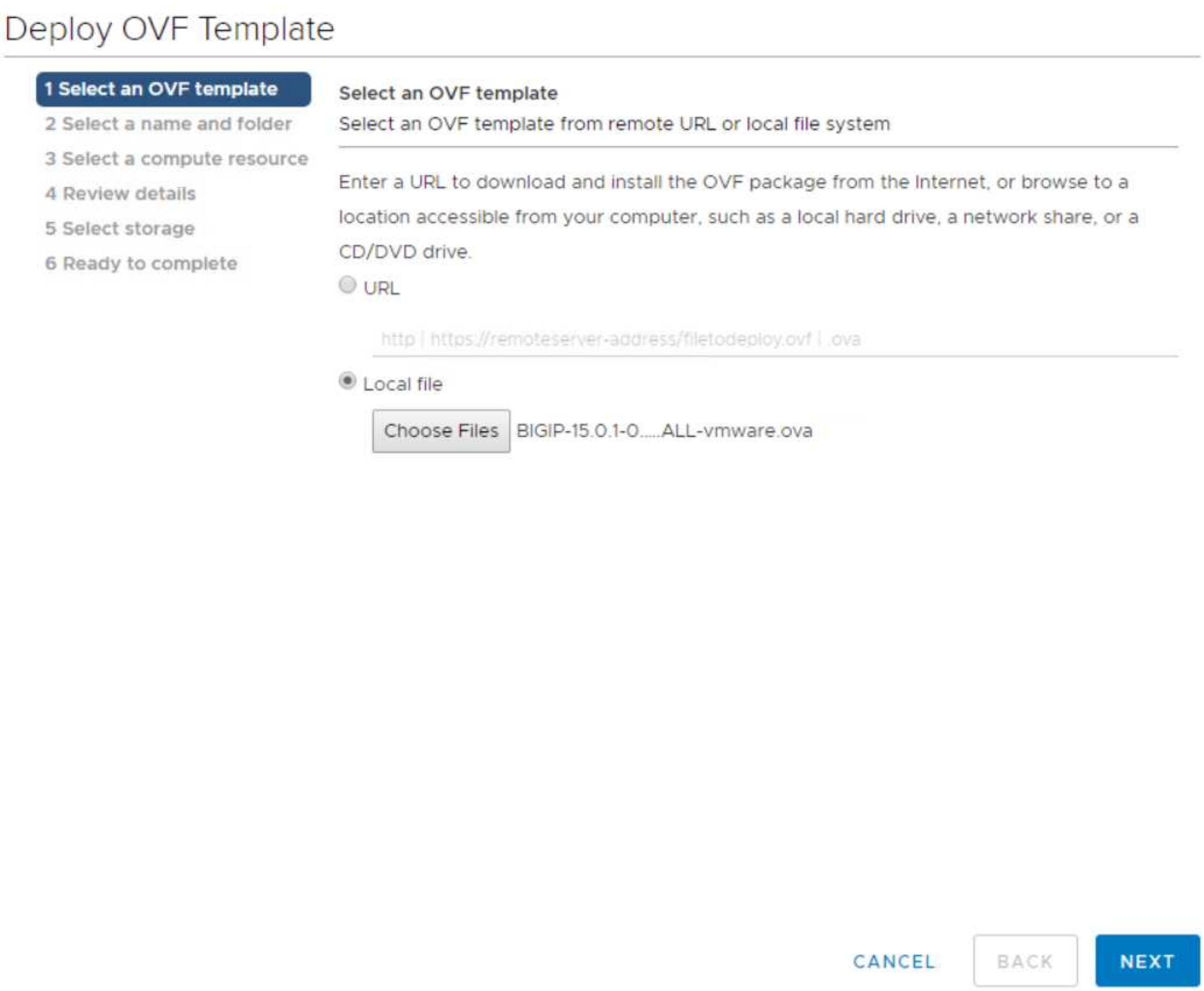
This solution makes use of the virtual appliance deployed in VMware vSphere as deployed by NDE. Networking for the F5 Big-IP virtual appliance can be configured in a two-armed or three-armed configuration based on your network environment. The deployment in this document is based on the two-armed configuration. Additional details for configuring the virtual appliance for use with Anthos can be found here.

To deploy the F5 Big-IP Virtual Edition appliance, complete the following steps:

1. Download the virtual application Open Virtual Appliance (OVA) file from F5 here.

> To download the appliance, a user must register with F5. They provide a 30-day demo license for the Big-IP Virtual Edition Load Balancer. NetApp recommends a permanent 10Gbps license for the production deployment of an appliance.

2. Right-click the infrastructure resource pool and select Deploy OVF Template. A wizard launches that allows you to select the OVA file that you just downloaded in Step 1. Click Next.



3. Click Next to continue through each step and accept the default values for each screen presented until you reach the storage selection screen. Select the VM_Datastore that was created earlier, and then click Next.

4. The next screen presented by the wizard allows you to customize the virtual networks for use in the environment. Select VM_Network for the External field and select Management_Network for the Management field. Internal and HA are used for advanced configurations for the F5 Big-IP appliance and

are not configured. These parameters can be left alone, or they can be configured to connect to non-infrastructure, distributed port groups. Click Next.

5. Review the summary screen for the appliance, and, if all the information is correct, click Finish to start the deployment.
6. After the virtual appliance is deployed, right-click it and power it up. It should receive a DHCP address on the management network. The appliance is Linux-based, and it has VMware Tools deployed, so that you can view the DHCP address it receives in the vSphere client.

7. Open a web browser and connect to the appliance at the IP address from the previous step. The default login is admin/admin, and, after the first login, the appliance immediately prompts you to change the admin password. It then returns you to a screen where you must log in with the new credentials.

8. The first screen prompts the you to complete the Setup Utility. Begin the utility by clicking Next.

9. The next screen prompts you for activation of the appliance license. Click Activate to begin. When prompted on the next page, paste either the 30-day evaluation license key you received when you registered for the download or the permanent license you acquired when you purchased the appliance. Click Next.

For the device to perform activation, the network defined on the management interface must be able to reach the internet.

10. On the next screen, the End User License Agreement (EULA) is presented. If the terms in the license are acceptable, click Accept.
11. The next screen counts the elapsed time as it verifies the configuration changes that have been made so far. Click Continue to resume with the initial configuration.
12. The Configuration Change window closes, and the Setup Utility displays the Resource Provisioning menu. This window lists the features that are currently licensed and the current resource allocations for the virtual appliance and each running service.
13. Clicking the Platform menu option on the left enables additional modification of the platform. Modifications include setting the management IP address configured with DHCP, setting the host name and the time zone the appliance is installed in, and securing the appliance from SSH accessibility.
14. Next click the Network menu, which enables you to configure standard networking features. Click Next to begin the Standard Network Configuration wizard.
15. The first page of the wizard configures redundancy; leave the defaults and click Next. The next page enables you to configure an internal interface on the load balancer. Interface 1.1 maps to the vmnic labeled Internal in the OVF deployment wizard.

    The fields in this page for Self IP Address, Netmask, and Floating IP address can be filled with a non-routable IP address for use as a placeholder. They can also be filled with an internal network that has been configured as a distributed port group for virtual guests if you are deploying the three-armed configuration. They must be completed to continue with the wizard.

16. The next page enables you to configure an external network that is used to map services to the pods deployed in Kubernetes. Select a static IP from the VM_Network range, the appropriate subnet mask, and a floating IP from that same range. Interface 1.2 maps to the vmnic labeled External in the OVF deployment wizard.
17. On the next page, you can configure an internal-HA network if you are deploying multiple virtual appliances in the environment. To proceed, you must fill the Self-IP Address and the Netmask fields, and you must select interface 1.3 as the VLAN Interface, which maps to the HA network defined by the OVF template wizard.
18. The next page enables you to configure the NTP servers. Then click Next to continue to the DNS setup.

The DNS servers and domain search list should already be populated by the DHCP server. Click Next to accept the defaults and continue.

19. For the remainder of the wizard, click Next to continue through the advanced peering setup, the configuration of which is beyond the scope of this document. Then click Finish to exit the wizard.
20. Create individual partitions for the Anthos admin cluster and each user cluster deployed in the environment. Click System in the menu on the left, navigate to Users, and click Partition List.
21. The displayed screen only shows the current common partition. Click Create on the right to create the first additional partition and name it `Anthos-Admin`. Then click Repeat, name the partition `Anthos-Cluster1`, and click the Repeat button again to name the next partition `Anthos-Cluster2`. Finally click Finished to complete the wizard. The Partition list screen returns with all the partitions now listed.

Next: Complete Anthos prerequisites.

Complete Anthos prerequisites

Now that the physical environment is set up, you can begin Anthos deployment. This starts with several prerequisites that you must meet to deploy the solution and access it afterward. Each of these steps are discussed in depth in the Anthos GKE On-Prem Guide.

To prepare your environment for the deployment of Anthos on VMware, complete the following steps:

1. Create a Google Cloud project following the instructions available here.

   > Your organization might already have a project in place intended for this purpose. Check with your cloud administration team to see if a project exists and is already configured for access to Anthos on VMware. All projects intended for use with Anthos must be whitelisted by Google. This includes the primary user account, additional team members, and the access service account created in a later step.

2. Create a deployment workstation from which to manage the installation of Anthos on VMware. The deployment workstation can be Linux, MacOS, or Windows. For the purposes of this validated deployment, Red Hat Enterprise Linux 7 was used.

   > This workstation can be hosted either internal or external to the NetApp HCI deployment. The only requirement is that it must be able to successfully communicate with the deployed VMware vCenter Server and the internet to function correctly.

3. Install Google Cloud SDK for interactions with Google Cloud. It can be downloaded as an archive of binaries for manual install or installed by either the apt-get (Ubuntu/Debian) or yum (RHEL) package managers.

```
[user@rhel7 ~]$ sudo yum install google-cloud-sdk
Failed to set locale, defaulting to C
Loaded plugins: langpacks, product-id, search-disabled-repos,
subscription-manager
Resolving Dependencies
--> Running transaction check
---> Package google-cloud-sdk.noarch 0:270.0.0-1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

==========================================================================
=========================
Package                  Arch            Version            Repository
Size
==========================================================================
=========================
Installing:
google-cloud-sdk         noarch          270.0.0-1          google-cloud-
sdk              36 M

Transaction Summary
==========================================================================
=========================
Install  1 Package

Total download size: 36 M
Installed size: 174 M
Is this ok [y/d/N]: y
Downloading packages:
6d81c821884ae40244c746f6044fc1bcd801143a0d9c8da06767036b8d090a24-google-
cloud-sdk-270.0.0-1.noar |  36 MB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : google-cloud-sdk-270.0.0-1.noarch
1/1
  Verifying  : google-cloud-sdk-270.0.0-1.noarch
1/1

Installed:
  google-cloud-sdk.noarch 0:270.0.0-1

Complete!
```

The gcloud binary must be at least version 265.0.0. You can update a manual install with a gcloud components update. However, if SDK was installed by a package manager, future updates must also be performed using that same package manager.

4. With the workstation configured, log in to Google Cloud with your credentials. To do so, enter the login command from the deployment workstation and retrieve a link that can be copied and pasted into a browser to allow interactive sign-in to Google services. After you have logged in, the web page presents a code that you can copy and paste back into the deployment workstation when prompted.

```
[user@rhel7 ~]$ gcloud auth login
Go to the following link in your browser:

    https://accounts.google.com/o/oauth2/auth?code_challenge=-
7oPNSySHr_Sd2ZZ4K83koIeGTLVcdbjc8omr6zCbAI&prompt=select_account&code_ch
allenge_method=S256&access_type=offline&redirect_uri=urn%3Aietf%3Awg%3Ao
auth%3A2.0%3Aoob&response_type=code&client_id=32655940559.apps.googleuse
rcontent.com&scope=https%3A%3F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.em
ail+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud-
platform+https%3A%6F%2Fwww.googleapis.com%2Fauth%2Fappengine.admin+https
%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcompute+https%3A%2F%2Fwww.googleapi
s.com%2Fauth%2Faccounts.reauth


Enter verification code: 6/swGAh52VVgB-
TRS5LVrSvP79ZdDlb9V6ObyUGqoY67a3zp9NPciIKsM
You are now logged in as [user@netapp.com].
Your current project is [anthos-dev].  You can change this setting by
running:
  $ gcloud config set project PROJECT_ID
```

5. Enable several APIs so that your environment can communicate with Google Cloud. The pods deployed in your clusters must be able to access https://www.googleapis.com and https://gkeconnect.googleapis.com to function as expected. Therefore, the VM_Network that the worker nodes are attached to must have internet access. To enable the necessary APIs, run the following command from the deployment workstation:

```
[user@rhel7 ~]$ gcloud services enable --project anthos-dev \
cloudresourcemanager.googleapis.com \
container.googleapis.com \
gkeconnect.googleapis.com \
gkehub.googleapis.com \
serviceusage.googleapis.com \
stackdriver.googleapis.com \
monitoring.googleapis.com \
logging.googleapis.com
```

6. Create a working directory called anthos-install, and change into that directory.

```
[user@rhel7 ~]$ mkdir anthos-install && cd anthos-install
[user@rhel7 anthos-install]$
```

7. Before you can install Anthos on VMware, you must create four service accounts, each with a specific purpose in interacting with Google Cloud. The following table lists the accounts and their purposes.

Account Name	Purpose
component-access-sa	Used to download the Anthos binaries from Cloud Storage.
connect-register-sa	Used to register Anthos clusters to the Google Cloud console.
connect-agent-sa	Used to maintain the connection between user clusters and the Google Cloud.
logging-monitoring-sa	Used to write logging and monitoring data to Stackdriver.

> Each account is assigned an email address that references your approved Google Cloud project name. The following examples all list the project Anthos-Dev, which was used during the NetApp validation. Make sure to substitute your appropriate project name in syntax examples where necessary.

```
[user@rhel7 anthos-install]$ gcloud iam service-accounts create
component-access-sa \
    --display-name "Component Access Service Account" \
    --project anthos-dev
[user@rhel7 anthos-install]$ gcloud iam service-accounts keys create
component-access-key.json \
   --iam-account component-access-sa@anthos-dev.iam.gserviceaccount.com

[user@rhel7 anthos-install]$ gcloud iam service-accounts create connect-
register-sa \
    --project anthos-dev
[user@rhel7 anthos-install]$ gcloud iam service-accounts keys create
connect-register-key.json \
   --iam-account connect-register-sa@anthos-dev.iam.gserviceaccount.com

[user@rhel7 anthos-install]$ gcloud iam service-accounts create connect-
agent-sa \
    --project anthos-dev
[user@rhel7 anthos-install]$ gcloud iam service-accounts keys create
connect-agent-key.json \
    --iam-account connect-agent-sa@anthos-dev.iam.gserviceaccount.com

[user@rhel7 anthos-install]$ gcloud iam service-accounts create logging-
monitoring-sa \
    --project anthos-dev
[user@rhel7 anthos-install]$ gcloud iam service-accounts keys create
logging-monitoring-key.json \
    --iam-account logging-monitoring-sa@anthos-
dev.iam.gserviceaccount.com
```

8. The final step needed to prepare your environment to deploy Anthos is to limit certain privileges to your service accounts. You need the associated email address for each service account listed in Step 7.
   a. Using the component-access-sa account, assign the roles for `serviceuseage.serviceUsageViewer`, `iam.serviceAccountCreator`, and `iam.roleViewer`.

```
[user@rhel7 anthos-install]$ gcloud projects add-iam-policy-binding
anthos-dev\
    --member "serviceAccount:component-access-sa@anthos-
dev.iam.gserviceaccount.com" \
    --role "roles/serviceusage.serviceUsageViewer"
[user@rhel7 anthos-install]$ gcloud projects add-iam-policy-binding
anthos-dev\
    --member "serviceAccount:component-access-sa@anthos-
dev.iam.gserviceaccount.com" \
    --role "roles/iam.serviceAccountCreator"
[user@rhel7 anthos-install]$ gcloud projects add-iam-policy-binding
anthos-dev\
    --member "serviceAccount:component-access-sa@anthos-
dev.iam.gserviceaccount.com" \
    --role "roles/iam.roleViewer"
```

b. Using the connect-register-sa service account, assign the role for `gkehub.admin`.

```
[user@rhel7 anthos-install]$ gcloud projects add-iam-policy-binding
anthos-dev \
    --member "serviceAccount:connect-register-sa@anthos-
dev.iam.gserviceaccount.com " \
    --role "roles/gkehub.admin"
```

c. Using the connect-agent-sa acccount, assign the role for `gkehub.connect`.

```
[user@rhel7 anthos-install]$ gcloud projects add-iam-policy-binding
anthos-dev \
    --member "serviceAccount:connect-agent-sa@anthos-
dev.iam.gserviceaccount.com" \
    --role "roles/gkehub.connect"
```

d. With the logging-monitoring-sa service account, assign the roles for `stackdriver.resourceMetadata.writer`, `logging.logWriter`, `monitoring.metricWriter`, and `monitoring.dashboardEditor`.

```
[user@rhel7 anthos-install]$ gcloud projects add-iam-policy-binding
anthos-dev \
    --member "serviceAccount:logging-monitoring-sa@anthos-
dev.iam.gserviceaccount.com" \
    --role "roles/stackdriver.resourceMetadata.writer"
[user@rhel7 anthos-install]$ gcloud projects add-iam-policy-binding
anthos-dev\
    --member "serviceAccount:logging-monitoring-sa@anthos-
dev.iam.gserviceaccount.com" \
    --role "roles/logging.logWriter"
[user@rhel7 anthos-install]$ gcloud projects add-iam-policy-binding
anthos-dev\
    --member "serviceAccount:logging-monitoring-sa@anthos-
dev.iam.gserviceaccount.com" \
    --role "roles/monitoring.metricWriter"
[user@rhel7 anthos-install]$ gcloud projects add-iam-policy-binding
anthos-dev\
    --member "serviceAccount:logging-monitoring-sa@anthos-
dev.iam.gserviceaccount.com" \
    --role "roles/monitoring.dashboardEditor"
```

9. Download the vCenter certificate for the VMWare CA; this is used later to authenticate to the vCenter during installation.

```
[user@rhel7 anthos-install]$ true | openssl s_client -connect anthos-
vc.cie.netapp.com:443 -showcerts 2>/dev/null | sed -ne '/-BEGIN/,/-
END/p' > vcenter.pem
```

Next: Deploy the Anthos admin workstation

7. Deploy the Anthos admin workstation

The admin workstation is a vSphere VM deployed within your NetApp HCI environment that is preinstalled with all the tools necessary to administer the Anthos on VMware solution. Follow the instructions in this section to deploy the Anthos admin workstation.

To deploy the Anthos admin workstation, complete the following steps:

1. Download the gkeadm binary into your working directory

```
[user@rhel7 anthos-install]$ gsutil cp gs://gke-on-prem-
release/gkeadm/1.6.1-gke.1/linux/gkeadm ./
[user@rhel7 anthos-install]$ chmod +x gkeadm
```

2. Use the gkeadm tool to create an admin workstation configuration file.

```
[user@rhel7 anthos-install]$ ./gkeadm create config
```

3. Two files are created: `credential.yaml` and `admin-ws-config.yaml`. Fill out each of these files.

   a. credential.yaml contains your username and passwords for your VMware vCenter server.

```
kind: CredentialFile
items:
- name: vCenter
  username: "administrator@vsphere.local"
  password: "vSphereAdminPassword"
```

   b. admin-ws-config.yaml contains other information about your vSphere environment as well as the physical and networking options for the admin-workstation VM.

```
gcp:
  # Path of the whitelisted service account's JSON key file
  whitelistedServiceAccountKeyPath: "/home/anthos-install/service-
keys/access-key.json"
# Specify which vCenter resources to use
vCenter:
  # The credentials and address GKE On-Prem should use to connect to
vCenter
  credentials:
    address: "anthos-vc.cie.netapp.com"
    datacenter: "NetApp-HCI-Datacenter-01"
    datastore: "VM_Datastore"
    cluster: "NetApp-HCI-Cluster-01"
    network: "VM_Network"
    resourcePool: "Anthos-Resource-Pool"
# Provide the path to vCenter CA certificate pub key for SSL
verification
    caCertPath: "/home/anthos-install/vcenter.pem"
# The URL of the proxy for the jump host
proxyUrl: ""
adminWorkstation:
  name: gke-admin-ws-200915-151421
  cpus: 4
  memoryMB: 8192
#The boot disk size of the admin workstation in GB. It is recommended
to use a disk with at least 50 GB to host images decompressed from
the bundle.
  diskGB: 50
```

```
# Name for the persistent disk to be mounted to the home directory
(ending in
.vmdk).
# Any directory in the supplied path must be created before
deployment.
  dataDiskName: gke-on-prem-admin-workstation-data-disk/gke-admin-ws-
200915-151421-data-disk.vmdk
# The size of the data disk in MB.
  dataDiskMB: 512
  network:
# The IP allocation mode: 'dhcp' or 'static'
    ipAllocationMode: "dhcp"
# # The host config in static IP mode. Do not include if using DHCP
  # hostConfig:
    #   # The IPv4 static IP address for the admin workstation
    #   ip: ""
    #   # The IP address of the default gateway of the subnet in
which the admin workstation
    #   # is to be created
    #   gateway: ""
    #   # The subnet mask of the network where you want to create
your admin workstation
    #   netmask: ""
    #   # The list of DNS nameservers to be used by the admin
workstation
    #   dns:
    #   - ""
  # The URL of the proxy for the admin workstation
  proxyUrl: ""
  ntpServer: ntp.ubuntu.com
```

4. Create the admin workstation.

```
[user@rhel7 anthos-install]$ ./gkeadm create admin-workstation
The output will be verbose as the workstation is created. In the end you
will be prompted with the IP address to login to the workstation if you
chose DHCP.
...
Getting ... service account...
...
********************************************************************
Admin workstation is ready to use.

Admin workstation information saved to /usr/local/google/home/me/my-
admin-workstation
This file is required for future upgrades
SSH into the admin workstation with the following command:
ssh -i /home/user/.ssh/gke-admin-workstation ubuntu@10.63.172.10
********************************************************************
```

Next: Deploy the admin and the first user cluster

8. Deploy the admin cluster

All Kubernetes clusters deployed as a part of the Anthos solution are deployed from the Anthos admin workstation that you just created. A user logs into the admin workstation using SSH, the public key created in a previous step, and the IP address provided at the end of the VM deployment. An admin cluster controls all actions in an Anthos environment. The admin cluster must be deployed first, and then individual user clusters can be deployed for specific workload needs.

> There are specific procedures for deploying clusters that use static IP addresses here, and procedures for environments with DHCP can be found here. In this guide, we use the second set of instructions for ease of deployment.

To deploy the admin cluster, complete the following steps:

1. Log into your admin-workstation using the SSH command prompted at the end of the deployment. After successful authentication, you can list the files in the home directory, which are used to create the admin cluster and additional clusters later on. The directory also includes the copied vCenter cert and the access key for Anthos that was created in earlier steps.

```
[user@rhel7 anthos-install]$ ssh -i ~/.ssh/gke-admin-workstation
ubuntu@10.63.172.10

Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1001-gkeop x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
Last login: Fri Jan 29 15:46:35 2021 from 10.249.129.216

ubuntu@gke-admin-200915-151421:~$ ls
admin-cluster.yaml
user-cluster.yaml
vcenter.pem
component-access-key.json
```

2. Use scp to copy the remaining keys for your Anthos account over from the workstation you deployed the admin-workstation from.

```
ubuntu@gke-admin-200915-151421:~$ scp user@rhel7:~/anthos-
install/connect-register-key.json ./
ubuntu@gke-admin-200915-151421:~$ scp user@rhel7:~/anthos-
install/connect-agent-key.json  ./
ubuntu@gke-admin-200915-151421:~$ scp user@rhel7:~/anthos-
install/logging-monitoring-key.json ./
```

3. Edit the admin-cluster.yaml file so that it is specific to the deployed environment. The file is very large, so we will address it by sections.
   a. Most of the information is already filled in by default based on the configuration used to deploy the admin-workstation by gkeadm. This first section confirms the information for the version of Anthos being deployed and the vCenter instance it is deployed on. It also allows you to define a local data disk (VMDK) for Kubernetes object data.

```
apiVersion: v1
kind: AdminCluster
# (Required) Absolute path to a GKE bundle on disk
bundlePath: /var/lib/gke/bundles/gke-onprem-vsphere-1.6.0-gke.7-
full.tgz
# (Required) vCenter configuration
vCenter:
  address: anthos-vc.cie.netapp.com
  datacenter: NetApp-HCI-Datacenter-01
  cluster: NetApp-HCI-Cluster-01
  resourcePool: Anthos-Resource-Pool
  datastore: VM_Datastore
  # Provide the path to vCenter CA certificate pub key for SSL
verification
  caCertPath: "/home/ubuntu/vcenter.pem"
  # The credentials to connect to vCenter
  credentials:
    username: administrator@vsphere.local
    password: "vSphereAdminPassword"
  # Provide the name for the persistent disk to be used by the
deployment (ending
  # in .vmdk). Any directory in the supplied path must be created
before deployment
  dataDisk: "admin-cluster-disk.vmdk"
```

b. FIll out the networking section next, and select whether you are using static or DHCP mode. If you are using static addresses, you must create an IP-block file based on the instructions linked to above, and add it to the config file.

> If static IPs are used in a deployment, the items under the host configuration are global. This includes static IPs for clusters or those used for SeeSaw load balancers, which are configured later.

```
# (Required) Network configuration
network:
# (Required) Hostconfig for static addresseses on Seesaw LB's
  hostConfig:
    dnsServers:
    - "10.61.184.251"
    - "10.61.184.252"
    ntpServers:
    - "0.pool.ntp.org"
    - "1.pool.ntp.org"
    - "2.pool.ntp.org"
    searchDomainsForDNS:
    - "cie.netapp.com"
  ipMode:
    # (Required) Define what IP mode to use ("dhcp" or "static")
    type: dhcp
    # # (Required when using "static" mode) The absolute or relative
path to the yaml file
    # # to use for static IP allocation
    # ipBlockFilePath: ""
  # (Required) The Kubernetes service CIDR range for the cluster.
Must not overlap
  # with the pod CIDR range
  serviceCIDR: 10.96.232.0/24
  # (Required) The Kubernetes pod CIDR range for the cluster. Must
not overlap with
  # the service CIDR range
  podCIDR: 192.168.0.0/16
  vCenter:
    # vSphere network name
    networkName: VM_Network
```

c. Fill out the load balancer section next. This can vary depending on the type of load balancer being deployed.

   Seesaw example:

```
loadBalancer:
  # (Required) The VIPs to use for load balancing
  vips:
    # Used to connect to the Kubernetes API
    controlPlaneVIP: "10.63.172.155"
    # # (Optional) Used for admin cluster addons (needed for multi
cluster features). Must
    # # be the same across clusters
```

```
    # # addonsVIP: "10.63.172.153"
  # (Required) Which load balancer to use "F5BigIP" "Seesaw" or
"ManualLB". Uncomment
  # the corresponding field below to provide the detailed spec
  kind: Seesaw
  # # (Required when using "ManualLB" kind) Specify pre-defined
nodeports
  # manualLB:
  #   # NodePort for ingress service's http (only needed for user
cluster)
  #   ingressHTTPNodePort: 0
  #   # NodePort for ingress service's https (only needed for user
cluster)
  #   ingressHTTPSNodePort: 0
  #   # NodePort for control plane service
  #   controlPlaneNodePort: 30968
  #   # NodePort for addon service (only needed for admin cluster)
  #   addonsNodePort: 31405
  # # (Required when using "F5BigIP" kind) Specify the already-
existing partition and
  # # credentials
  # f5BigIP:
  #   address:
  #   credentials:
  #     username:
  #     password:
  #   partition:
  #   # # (Optional) Specify a pool name if using SNAT
  #   # snatPoolName: ""
  # (Required when using "Seesaw" kind) Specify the Seesaw configs
  seesaw:
  # (Required) The absolute or relative path to the yaml file to use
for IP allocation
  #  for LB VMs. Must contain one or two IPs.
  ipBlockFilePath: "admin-seesaw-block.yaml"
  #   (Required) The Virtual Router IDentifier of VRRP for the Seesaw
group. Must
  #   be between 1-255 and unique in a VLAN.
    vrid: 100
  #   (Required) The IP announced by the master of Seesaw group
    masterIP: "10.63.172.151"
  #   (Required) The number CPUs per machine
    cpus: 1
  #   (Required) Memory size in MB per machine
    memoryMB: 2048
  #   (Optional) Network that the LB interface of Seesaw runs in
```

```
(default: cluster
  #   network)
    vCenter:
  #   vSphere network name
      networkName: VM_Network
  #   (Optional) Run two LB VMs to achieve high availability
(default: false)
    enableHA: false
```

d. For a SeeSaw load balancer, you must create an additional external file to supply the static IP information for the load balancer. Create the file `admin-seesaw-block.yaml`, which was referenced in this configuration section.

```
blocks:
  - netmask: "255.255.255.0"
    gateway: "10.63.172.1"
    ips:
    - ip: "10.63.172.152"
      hostname: "admin-seesaw-vm"
```

F5 BigIP Example:

```
# (Required) Load balancer configuration
loadBalancer:
  # (Required) The VIPs to use for load balancing
  vips:
    # Used to connect to the Kubernetes API
    controlPlaneVIP: "10.63.172.155"
    # # (Optional) Used for admin cluster addons (needed for multi
cluster features). Must
    # # be the same across clusters
    # # addonsVIP: "10.63.172.153"
  # (Required) Which load balancer to use "F5BigIP" "Seesaw" or
"ManualLB". Uncomment
  # the corresponding field below to provide the detailed spec
  kind: F5BigIP
  # # (Required when using "ManualLB" kind) Specify pre-defined
nodeports
  # manualLB:
  #   # NodePort for ingress service's http (only needed for user
cluster)
  #   ingressHTTPNodePort: 0
  #   # NodePort for ingress service's https (only needed for user
cluster)
  #   ingressHTTPSNodePort: 0
```

```
  #   # NodePort for control plane service
  #   controlPlaneNodePort: 30968
  #   # NodePort for addon service (only needed for admin cluster)
  #   addonsNodePort: 31405
  # # (Required when using "F5BigIP" kind) Specify the already-
existing partition and
  # # credentials
  f5BigIP:
    address: "172.21.224.21"
    credentials:
      username: "admin"
      password: "admin-password"
    partition: "Admin-Cluster"
  #   # # (Optional) Specify a pool name if using SNAT
  #   # snatPoolName: ""
  # (Required when using "Seesaw" kind) Specify the Seesaw configs
  # seesaw:
    # (Required) The absolute or relative path to the yaml file to
use for IP allocation
    # for LB VMs. Must contain one or two IPs.
    #  ipBlockFilePath: ""
    # (Required) The Virtual Router IDentifier of VRRP for the Seesaw
group. Must
    # be between 1-255 and unique in a VLAN.
    #  vrid: 0
    # (Required) The IP announced by the master of Seesaw group
    #  masterIP: ""
    # (Required) The number CPUs per machine
    #  cpus: 4
    # (Required) Memory size in MB per machine
    #   memoryMB: 8192
    # (Optional) Network that the LB interface of Seesaw runs in
(default: cluster
    # network)
    #   vCenter:
      # vSphere network name
      #     networkName: VM_Network
    # (Optional) Run two LB VMs to achieve high availability
(default: false)
    #   enableHA: false
```

e. The last section of the admin config file contains additional options that can be tuned to fit the specific deployment environment. These include enabling anti-affinity groups if Anthos is being deployed on less than three ESXi servers. You can also configure proxies, private docker registries, and the connections to Stackdriver and Google Cloud for auditing.

```
antiAffinityGroups:
  # Set to false to disable DRS rule creation
  enabled: false
# (Optional) Specify the proxy configuration
proxy:
  # The URL of the proxy
  url: ""
  # The domains and IP addresses excluded from proxying
  noProxy: ""
# # (Optional) Use a private Docker registry to host GKE images
# privateRegistry:
#   # Do not include the scheme with your registry address
#   address: ""
#   credentials:
#     username: ""
#     password: ""
#   # The absolute or relative path to the CA certificate for this
registry
#   caCertPath: ""
# (Required): The absolute or relative path to the GCP service
account key for pulling
# GKE images
gcrKeyPath: "/home/ubuntu/component-access-key.json"
# (Optional) Specify which GCP project to connect your logs and
metrics to
stackdriver:
  projectID: "anthos-dev"
  # A GCP region where you would like to store logs and metrics for
this cluster.
  clusterLocation: "us-east1"
  enableVPC: false
  # The absolute or relative path to the key file for a GCP service
account used to
  # send logs and metrics from the cluster
  serviceAccountKeyPath: "/home/ubuntu/logging-monitoring-key.json"
# # (Optional) Configure kubernetes apiserver audit logging
# cloudAuditLogging:
#   projectid: ""
#   # A GCP region where you would like to store audit logs for this
cluster.
#   clusterlocation: ""
#   # The absolute or relative path to the key file for a GCP service
account used to
#   # send audit logs from the cluster
#   serviceaccountkeypath: ""
```

The deployment detailed in this document is a minimum configuration for validation that requires the disabling of anti-affinity rules. NetApp recommends leaving this option set to true in production deployments.

By default, Anthos on VMware uses a pre-existing, Google-owned container image registry that requires no additional setup. If you choose to use a private Docker registry for deployment, then you must configure that registry separately based on instructions found here. This step is beyond the scope of this deployment guide.

4. When edits to the admin-cluster.yaml file are complete, be sure to check for proper syntax and spacing.

```
ubuntu@gke-admin-200915-151421:~$ gkectl check-config –config admin-
cluster.yaml
```

5. After the configuration check has passed and any identified issues have been remedied, you can then stage the deployment of the cluster. Since we have already checked the validation of the config file, we can skip those steps by passing the `--skip-validation-all` flag.

```
ubuntu@gke-admin-200915-151421:~$ gkectl prepare --config admin-
cluster.yaml --skip-validation-all
```

6. If you are using a SeeSaw load balancer, you must create one before deploying the cluster itself (otherwise skip this step).

```
ubuntu@gke-admin-200915-151421:~$ gkectl create loadbalancer --config
admin-cluster.yaml
```

7. You can now stand up the admin cluster. This is done with the `gkectl create` admin command, which can use the `--skip-validation-all` flag to speed up deployment.

```
ubuntu@gke-admin-200915-151421:~$ gkectl create admin --config admin-
cluster.yaml --skip-validation-all
```

8. When the cluster is deployed, it creates the kubeconfig file in the local directory. This file can be used the check the status of the cluster using kubectl or run diagnostics with gkectl.

```
ubuntu@gke-admin-ws-200915-151421:~ $ kubectl get nodes --kubeconfig
kubeconfig
NAME                                      STATUS   ROLES    AGE
VERSION
gke-admin-master-gkvmp                    Ready    master   5m
v1.18.6-gke.6600
gke-admin-node-84b77ff5c7-6zg59           Ready    <none>   5m
v1.18.6-gke.6600
gke-admin-node-84b77ff5c7-8jdmz           Ready    <none>   5m
v1.18.6-gke.6600
ubuntu@gke-admin-ws-200915-151421:~$ gkectl diagnose cluster --
kubeconfig kubeconfig
Diagnosing admin cluster "gke-admin-gkvmp"...- Validation Category:
Admin Cluster VCenter
Checking Credentials...SUCCESS
Checking Version...SUCCESS
Checking Datacenter...SUCCESS
Checking Datastore...SUCCESS
Checking Resource pool...SUCCESS
Checking Folder...SUCCESS
Checking Network...SUCCESS- Validation Category: Admin Cluster
Checking cluster object...SUCCESS
Checking machine deployment...SUCCESS
Checking machineset...SUCCESS
Checking machine objects...SUCCESS
Checking kube-system pods...SUCCESS
Checking storage...SUCCESS
Checking resource...System pods on UserMaster cpu resource request
report: total 1754m nodeCount 2 min 877m max 877m avg 877m tracked
amount in bundle 4000m
System pods on AdminNode cpu resource request report: total 2769m
nodeCount 2 min 1252m max 1517m avg 1384m tracked amount in bundle 4000m
System pods on AdminMaster cpu resource request report: total 923m
nodeCount 1 min 923m max 923m avg 923m tracked amount in bundle 4000m
System pods on UserMaster memory resource request report: total
4524461824 nodeCount 2 min 2262230912 max 2262230912 avg 2262230912
tracked amount in bundle 8192Mi
System pods on AdminNode memory resource request report: total 6876Mi
nodeCount 2 min 2174Mi max 4702Mi avg 3438Mi tracked amount in bundle
16384Mi
System pods on AdminMaster memory resource request report: total 465Mi
nodeCount 1 min 465Mi max 465Mi avg 465Mi tracked amount in bundle
16384Mi
SUCCESS
Cluster is healthy.
```

Next: Deploy user clusters.

9. Deploy Additional User Clusters: NetApp HCI with Anthos

With Anthos, organizations can scale their environments to incorporate multiple user clusters and segregate workloads between teams. A single admin cluster can support up to five user clusters, and each user cluster can support up to twenty-five nodes.

To add additional user clusters to your deployment, complete the following steps:

1. Copy the `config.yaml` file to a new file named `anthos-cluster02-config.yaml`.

   ```
   ubuntu@Anthos-Admin-Workstation:~$ cp config.yaml anthos-cluster02-
   config.yaml
   ```

2. Make the following edits to the newly created file:
   1. Comment out the sections that refer to the existing admin cluster with (#).
   2. When you get to the `usercluster` section, update the following fields:
      1. Update the partition name under the `bigip` section.
      2. Update the `controlplanvip` and `ingressvip` values under the `vip` section.
      3. Update the `clustername` value.

      ```
       usercluster:
        # In-Cluster vCenter configuration
        vcenter:
          # If specified it overwrites the network field in global
      vcenter configuration
          network: ""
        # # The absolute or relative path to the yaml file to use for
      static IP allocation.
        # # Do not include if using DHCP
        # ipblockfilepath: ""
        # # Specify pre-defined nodeports if using "manual" load
      balancer mode
        # manuallbspec:
        #   ingresshttpnodeport: 30243
        #   ingresshttpsnodeport: 30879
        #   controlplanenodeport: 30562
        #   addonsnodeport: 0
        # Specify the already-existing partition and credentials to use
      with F5
        bigip:
          # To re-use credentials across clusters we recommend using
      YAML node anchors.
      ```

```
    # See https://yaml.org/spec/1.2/spec.html#id2785586
    credentials:
      address: "172.21.224.22"
      username: "admin"
      password: "NetApp!23"
    partition: "Anthos-Cluster02-Part"
    # # Optionally specify a pool name if using SNAT
    # snatpoolname: ""
  # The VIPs to use for load balancing
  vips:
    # Used to connect to the Kubernetes API
    controlplanevip: "10.63.172.108"
    # Shared by all services for ingress traffic
    ingressvip: "10.63.172.109"
    # # Used for admin cluster addons (needed for multi cluster
features). Must be the same
    # # across clusters
    # addonsvip: ""
  # A unique name for this cluster
  clustername: "anthos-cluster02"
  # User cluster master nodes must have either 1 or 3 replicas
  masternode:
    cpus: 4
    memorymb: 8192
    # How many machines of this type to deploy
    replicas: 1
  # The number of worker nodes to deploy and their size. Min. 2
replicas
  workernode:
    cpus: 4
    memorymb: 8192
    # How many machines of this type to deploy
    replicas: 3
  # The Kubernetes service CIDR range for the cluster
  serviceiprange: 10.96.0.0/12
  # The Kubernetes pod CIDR range for the cluster
  podiprange: 192.168.0.0/16
```

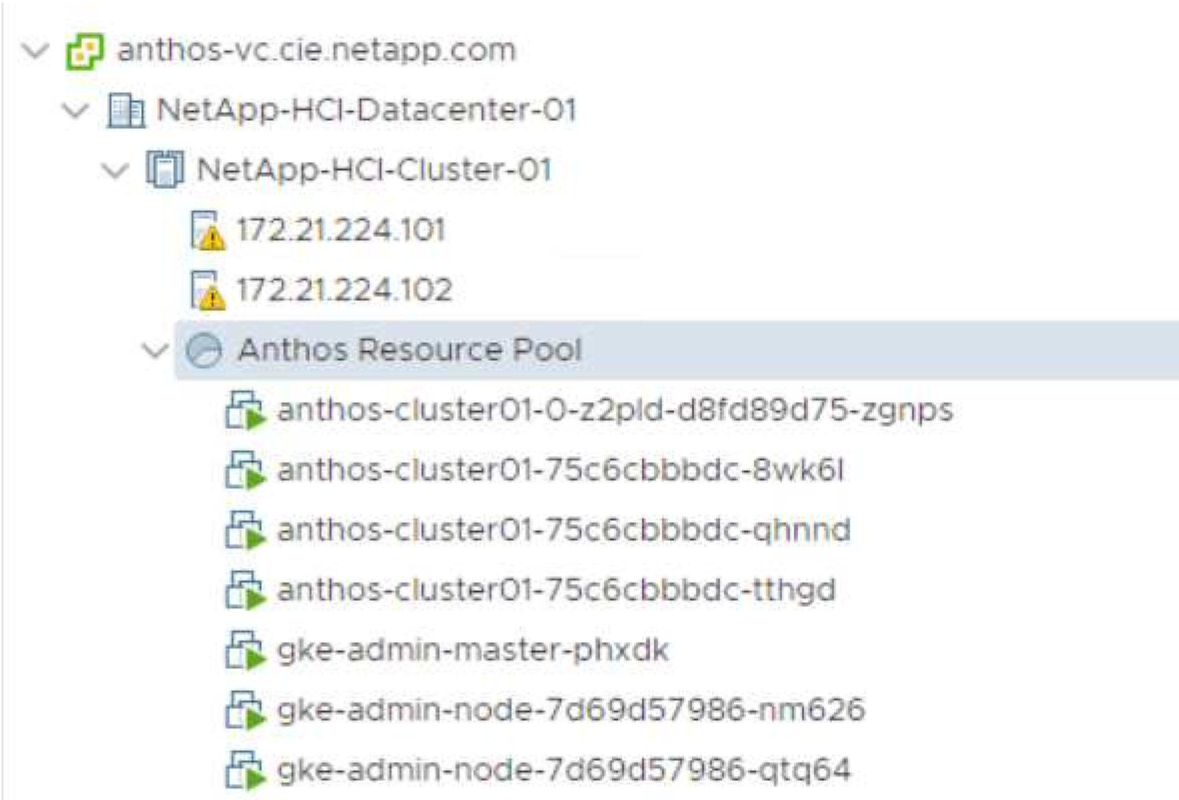
3. Run the following command to check the config file again to verify that there are no syntax errors. Because you have removed the admin section, you must reference the `kubeconfig` file for the admin cluster named `kubeconfig` (found in the working directory).

```
ubuntu@Anthos-Admin-Workstation:~$ gkectl check-config --config anthos-
cluster02-config.yaml --kubeconfig kubeconfig
- Validation Category: Config Check
    - [SUCCESS] Config

- Validation Category: Docker Registry
    - [SUCCESS] gcr.io/gke-on-prem-release access

- Validation Category: vCenter
    - [SUCCESS] Credentials
    - [SUCCESS] Datacenter
    - [SUCCESS] Datastore
    - [FAILURE] Data Disk: vCenter data disk already exists
    - [SUCCESS] Resource Pool
    - [SUCCESS] Network

- Validation Category: F5 BIG-IP
    - [SUCCESS] Credentials
    - [SUCCESS] Partition

- Validation Category: Network Configuration
    - [SUCCESS] CIDR, VIP and static IP (availability and overlapping)

- Validation Category: VIPs
    - [SUCCESS] ping (availability)

- Validation Category: Node IPs
    - [SUCCESS] ping (availability)

Some validations FAILED or SKIPPED. Check report above.
```

4. If all the checks succeed as expected, you can deploy this new user cluster in a manner very similar to the first cluster creation, referencing the `kubeconfig` file from the admin cluster.

```
ubuntu@Anthos-Admin-Workstation:~$ gkectl create cluster --config
anthos-cluster02-config.yaml --kubeconfig kubeconfig
```

5. As with the previous deployment, the process runs for several minutes and can be monitored on screen and in vCenter by watching the resource pool as the VMs populate. When complete, you should be able to see the new user cluster (four nodes).



6. You can access and execute commands against the deployed user cluster using the kubectl command line tool and the `kubeconfig` file generated by the process (stored in the working directory).

```
ubuntu@Anthos-Admin-Workstation:~$ kubectl get nodes --kubeconfig
anthos-cluster02-kubeconfig
NAME                              STATUS    ROLES     AGE      VERSION
anthos-cluster02-84744f5bd8-8rqk6  Ready     <none>    9m16s    v1.13.7-
gke.20
anthos-cluster02-84744f5bd8-fl786  Ready     <none>    9m28s    v1.13.7-
gke.20
anthos-cluster02-84744f5bd8-fnsmp  Ready     <none>    9m21s    v1.13.7-
gke.20
```

10. Enable access to the cluster with the GKE console

After clusters are deployed and registered with Google Cloud, they must be logged into with the Google Cloud console to be managed and to receive additional cluster details. The official procedure to gain access to Anthos user clusters after they are deployed is detailed here.

> ℹ️ The project and the specific user must be whitelisted to access on-premises clusters in the Google Cloud console and use Anthos on VMware services. If you are unable to see the clusters after they are deployed, you might need to open a support ticket with Google.

The non-whitelisted view looks like this:

The following figures provides a view of clusters.

To enable access to your user clusters using the GKE console, complete the following steps:

1. Create a `node-reader.yaml` file that allows you to access the cluster.

```
kind: clusterrole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
 name: node-reader
rules:
- apiGroups: [""]
  resources: ["nodes"]
  verbs: ["get", "list", "watch"]
```

2. Apply this file to the cluster that you want to log into with the `kubectl` command.

```
ubuntu@Anthos-Admin-Workstation:~$ kubectl apply -f node-reader.yaml
--kubeconfig anthos-cluster01-kubeconfig
clusterrole.rbac.authorization.k8s.io/node-reader created
```

3. Create a Kubernetes service account (KSA) that you can use to log in. Name this account after the user that uses this account to log into the cluster.

```
ubuntu@Anthos-Admin-Workstation:~$ kubectl create serviceaccount netapp-
user --kubeconfig anthos-cluster01-kubeconfig
serviceaccount/netapp-user created
```

4. Create cluster role-binding resources to bind both the view and newly created node-reader roles to the newly created KSA.

```
ubuntu@Anthos-Admin-Workstation:~$ kubectl create clusterrolebinding
netapp-user-view --clusterrole view --serviceaccount default:netapp-user
--kubeconfig anthos-cluster01-kubeconfig
clusterrolebinding.rbac.authorization.k8s.io/netapp-user-view created
ubuntu@Anthos-Admin-Workstation:~$ kubectl create clusterrolebinding
netapp-user-node-reader --clusterrole node-reader -
-serviceaccount default:netapp-user --kubeconfig anthos-cluster01-
kubeconfig
clusterrolebinding.rbac.authorization.k8s.io/netapp-user-node-reader
created
```

5. If you need to extend permissions further, you can grant the KSA user a role with cluster admin

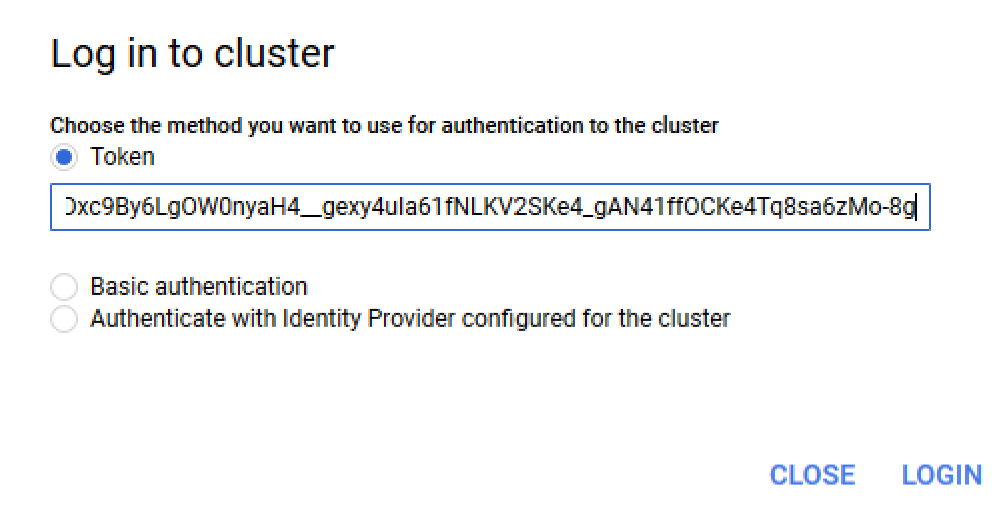
permissions in a similar manner.

```
ubuntu@Anthos-Admin-Workstation:~$ kubectl create clusterrolebinding
netapp-user-admin --clusterrole cluster-admin --serviceaccount
default:netapp-user --kubeconfig anthos-cluster01-kubeconfig
clusterrolebinding.rbac.authorization.k8s.io/netapp-user-admin created
```

6. With the KSA account created and assigned with correct permissions, you can create a bearer token to allow access with the GKE Console. To do so, set a system variable for the secret name, and pass that variable through a `kubectl` command to generate the token.

```
ubuntu@Anthos-Admin-Workstation:~$ SECRET_NAME=$(kubectl get
serviceaccount netapp-user --kubeconfig anthos-cluster01-kubeconfig -o
jsonpath='{$.secrets[0].name}')
ubuntu@Anthos-Admin-Workstation:~$ kubectl get secret ${SECRET_NAME}
--kubeconfig anthos-cluster01-kubeconfig -o jsonpath='{$.data.token}' |
base64 -d
eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2N
vdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJkZWZhdWx
0Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZWNyZXQubmFtZSI6Im5ldGFwcC1
1c2VyLXRva2VuLWJxd3piIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWN
lLWFjY291bnQubmFtZSI6Im5ldGFwcC11c2VyIiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWN
jb3VudC9zZXJ2aWNlLWFjY291bnQudWlkIjoiNmIzZTFiZjQtMDE3NS0xMWVhLWEzMGUtNmF
iZmRlYjYwNDBmIiwic3ViIjoic3lzdGVtOnNlcnZpY2VhY2NvdW50OmRlZmF1bHQ6bmV0YXB
wLXVzZXIifQ.YrHn4kYlb3gwxVKCLyo7p6J1f7mwwIgZqNw9eTvIkt4PfyR4IJHxQwawnJ4T
6RljIFcbVSQwvWI1yGuTJ98lADdcwtFXHoEfMcOa6SIn4OMVw1d5BGloaESn8150VCK3xES2
DHAmLexFBqhVBgckZ0E4fZDvn4EhYvtFVpKlRbSyaE-
DHD59P1bIgPdioiKREgbOddKdMn6XTVsuip4V4tVKhktcdRNRAuw6cFDY1fPol3BFHr2aNBI
e6lFLkUqvQN-
9nMd63JGdHL4hfXu6PPDxc9By6LgOW0nyaH4__gexy4uIa61fNLKV2SKe4_gAN41ffOCKe4T
q8sa6zMo-8g
```

7. With this token, you can visit the Google Cloud Console and log in to the cluster by clicking the login button and pasting in the token.



1. After login is complete, you see a green check mark next to the cluster name, and information is displayed about the physical environment. Clicking the cluster name displays more verbose information.

Next: Install and Configure NetApp Trident Storage Provisioner.

11. Install and configure NetApp Trident storage provisioner

Trident is a storage orchestrator for containers. With Trident, microservices and containerized applications can take advantage of enterprise-class storage services provided by the full NetApp portfolio of storage systems for persistent storage mounts. Depending on an application’s requirements, Trident dynamically provisions storage for ONTAP-based products such as NetApp AFF and FAS systems and Element storage systems like NetApp SolidFire and NetApp HCI.

To install Trident on the deployed user cluster and provision a persistent volume, complete the following steps:

> The following instructions are screen-capped from a Trident 21.01 install, but the same steps to manually deploy the Trident Operator also apply to the current 21.04 release.

1. Download the installation archive to the admin workstation and extract the contents. The current version of Trident is 21.04, which can be downloaded here.

```
ubuntu@gke-admin-ws-200915-151421:~$ wget
https://github.com/NetApp/trident/releases/download/v21.01.0/trident-
installer-21.01.0.tar.gz
--2021-02-17 12:40:42--
https://github.com/NetApp/trident/releases/download/v21.01.0/trident-
installer-21.01.0.tar.gz
Resolving github.com (github.com)... 140.82.121.4
Connecting to github.com (github.com)|140.82.121.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-
releases.githubusercontent.com/77179634/0a63b600-6273-11eb-98df-
3d542851f6ff?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAIWNJYAX4CSVEH53A%2F20210217%2Fus-east-
```

```
1%2Fs3%2Faws4_request&X-Amz-Date=20210217T173945Z&X-Amz-Expires=300&X-
Amz-
Signature=58f26bcac7eeee64673a84d46696490acec357b97a651af42653f973b778ee
88&X-Amz-
SignedHeaders=host&actor_id=0&key_id=0&repo_id=77179634&response-
content-disposition=attachment%3B%20filename%3Dtrident-installer-
21.01.0.tar.gz&response-content-type=application%2Foctet-stream
[following]
--2021-02-17 12:40:43--  https://github-
releases.githubusercontent.com/77179634/0a63b600-6273-11eb-98df-
3d542851f6ff?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAIWNJYAX4CSVEH53A%2F20210217%2Fus-east-
1%2Fs3%2Faws4_request&X-Amz-Date=20210217T173945Z&X-Amz-Expires=300&X-
Amz-
Signature=58f26bcac7eeee64673a84d46696490acec357b97a651af42653f973b778ee
88&X-Amz-
SignedHeaders=host&actor_id=0&key_id=0&repo_id=77179634&response-
content-disposition=attachment%3B%20filename%3Dtrident-installer-
21.01.0.tar.gz&response-content-type=application%2Foctet-stream
Resolving github-releases.githubusercontent.com (github-
releases.githubusercontent.com)... 185.199.111.154, 185.199.108.154,
185.199.109.154, ...
Connecting to github-releases.githubusercontent.com (github-
releases.githubusercontent.com)|185.199.111.154|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 38527217 (37M) [application/octet-stream]
Saving to: ‘trident-installer-21.01.0.tar.gz’

100%[====================================================================
==============================================>] 38,527,217  84.9MB/s
in 0.4s

2021-02-17 12:40:44 (84.9 MB/s) - ‘trident-installer-21.01.0.tar.gz’
saved [38527217/38527217]
```

2. Extract the Trident install from the downloaded bundle.

```
ubuntu@gke-admin-ws-200915-151421:~$ tar -xf trident-installer-
21.01.0.tar.gz
ubuntu@gke-admin-ws-200915-151421:~$ cd trident-installer
```

3. First set the location of the user cluster’s `kubeconfig` file as an environment variable so that you don’t have to reference it, because Trident has no option to pass this file.

```
ubuntu@gke-admin-ws-200915-151421:~/trident-installer$ export
KUBECONFIG=~/anthos-cluster01-kubeconfig
```

4. The `trident-installer` directory contains manifests for defining all the required resources. Using the appropriate manifests, create the `TridentOrchestrator` custom resource definition.

```
ubuntu@gke-admin-ws-200915-151421:~/trident-installer$ kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
customresourcedefinition.apiextensions.k8s.io/tridentorchestrators.tride
nt.netapp.io created
```

5. If a Trident namespace does not exist, create one in your cluster using the provided manifest.

```
ubuntu@gke-admin-ws-200915-151421:~/trident-installer$ kubectl apply -f
deploy/namespace.yaml
namespace/trident created
```

6. Create the resources required for the Trident operator deployment, such as a `ServiceAccount` for the operator, a `ClusterRole` and `ClusterRoleBinding` to the `ServiceAccount`, a dedicated `PodSecurityPolicy`, or the operator itself.

```
ubuntu@gke-admin-ws-200915-151421:~/trident-installer$ kubectl create -f
deploy/bundle.yaml
serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created
```

7. You can check the status of the operator after it's deployed with the following commands:

```
ubuntu@gke-admin-ws-200915-151421:~/trident-installer$ kubectl get
deployment -n trident
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator   1/1     1            1           54s
ubuntu@gke-admin-ws-200915-151421:~/trident-installer$ kubectl get pods
-n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-operator-5c8bbf6754-h957z   1/1     Running   0          68s
```

8. With the operator deployed, we can now use it to install Trident. This requires creating a `TridentOrchestrator`.

```
ubuntu@gke-admin-ws-200915-151421:~/trident-installer$ kubectl create -f
deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created
ubuntu@gke-admin-ws-200915-151421:~/trident-installer$ kubectl describe
torc trident
Name:         trident
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  trident.netapp.io/v1
Kind:         TridentOrchestrator
Metadata:
  Creation Timestamp:  2021-02-17T18:25:43Z
  Generation:          1
  Managed Fields:
    API Version:  trident.netapp.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      f:spec:
        .:
        f:debug:
        f:namespace:
    Manager:      kubectl
    Operation:    Update
    Time:         2021-02-17T18:25:43Z
    API Version:  trident.netapp.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      f:status:
        .:
        f:currentInstallationParams:
          .:
          f:IPv6:
          f:autosupportHostname:
          f:autosupportImage:
          f:autosupportProxy:
          f:autosupportSerialNumber:
          f:debug:
          f:enableNodePrep:
          f:imagePullSecrets:
          f:imageRegistry:
          f:k8sTimeout:
          f:kubeletDir:
          f:logFormat:
          f:silenceAutosupport:
```

```
          f:tridentImage:
        f:message:
        f:namespace:
        f:status:
        f:version:
    Manager:         trident-operator
    Operation:       Update
    Time:            2021-02-17T18:25:43Z
  Resource Version:  14836643
  Self Link:
/apis/trident.netapp.io/v1/tridentorchestrators/trident
  UID:               0e5f2c3b-6ca2-4b85-8453-0382e1426160
Spec:
  Debug:      true
  Namespace:  trident
Status:
  Current Installation Params:
    IPv6:
    Autosupport Hostname:
    Autosupport Image:
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:
    Enable Node Prep:
    Image Pull Secrets:         <nil>
    Image Registry:
    k8sTimeout:
    Kubelet Dir:
    Log Format:
    Silence Autosupport:
    Trident Image:
  Message:                      Installing Trident
  Namespace:                    trident
  Status:                       Installing
  Version:
Events:
  Type    Reason      Age   From                        Message
  ----    ------      ----  ----                        -------
  Normal  Installing  23s   trident-operator.netapp.io  Installing
Trident
  Normal  Installed   15s   trident-operator.netapp.io  Trident
installed
```

9. You can verify that Trident is successfully installed by checking the pods that are running in the namespace or by using the tridentctl binary to check the installed version.

```
ubuntu@gke-admin-ws-200915-151421:~/trident-installer$ kubectl get pod
-n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-2cp7x                   2/2     Running   0          4m16s
trident-csi-2xr5h                   2/2     Running   0          4m16s
trident-csi-bnwvh                   2/2     Running   0          4m16s
trident-csi-d6cfc6bb-lxm2p          6/6     Running   0          4m16s
trident-operator-5c8bbf6754-h957z   1/1     Running   0          8m55s

ubuntu@gke-admin-ws-200915-151421:~/trident-installer$ ./tridentctl -n
trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.01.1        | 21.01.1        |
+----------------+----------------+
```

10. The next step in enabling Trident integration with the NetApp HCI solution and Anthos is to create a backend that enables communication with the storage system. NetApp has been validated for several different protocols through the Anthos-ready partner storage validation program. This allows NetApp Trident to provide support in Anthos environments for NFS through our ONTAP platforms and iSCSI from both the ONTAP and Element storage used in NetApp HCI.

    A NetApp HCI platform deploys with NetApp Element storage by default. In this guide we configure a backend for this system specifically. In addition to this, a customer can choose to connect to a remote ONTAP storage system or deploy an ONTAP Select software-defined storage system as a virtual appliance in VMware vSphere to provide additional NFS and iSCSI services. The configuration of each of these additional storage backends is beyond the scope of this guide.

11. There are sample backend files available in the downloaded installation archive in the `sample-input` folder. Copy `backend-solidfire.json` to your working directory and edit it to provide information detailing the storage system environment. For Element-based iSCSI connections, copy and edit the `backend-solidfire.json` file.

```
ubuntu@gke-admin-ws-200915-151421:~/trident-installer$ cp sample-
input/backend-solidfire.json ./
ubuntu@gke-admin-ws-200915-151421:~/trident-installer$ $ vi backend-
solidfire.json
```

    a. Edit the user, password, and MVIP value on the EndPoint line.

    b. Edit the SVIP value.

```
 {
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://trident:password@172.21.224.150/json-
rpc/8.0",
    "SVIP": "10.63.172.100:3260",
    "TenantName": "trident",
    "Types": [{"Type": "Bronze", "Qos": {"minIOPS": 1000, "maxIOPS":
2000, "burstIOPS": 4000}},
              {"Type": "Silver", "Qos": {"minIOPS": 4000, "maxIOPS":
6000, "burstIOPS": 8000}},
              {"Type": "Gold", "Qos": {"minIOPS": 6000, "maxIOPS":
8000, "burstIOPS": 10000}}]
}
```

12. With this backend file in place, run the following command to create your first backend.

```
ubuntu@gke-admin-ws-200915-151421:~/trident-installer$ ./tridentctl -n
trident create backend -f backend.json
+-------------------+----------------
+--------------------------------------+--------+---------+
|    NAME           | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| solidfire-backend | solidfire-san  | a5f9e159-c8f4-4340-a13a-
c615fef0f433 | online |       0 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

13. With the backend created, you must next create a storage class. Just as with the backend, there is a sample storage class file that can be edited for the environment available in the sample-inputs folder. Copy it to the working directory and make necessary edits to reflect the backend created.

```
ubuntu@gke-admin-ws-200915-151421:~/trident-installer$ cp sample-
input/storage-class-csi.yaml.templ ./storage-class-basic.yaml
ubuntu@gke-admin-ws-200915-151421:~/trident-installer$ vi storage-class-
basic.yaml
```

14. The only edit that must be made to this file is to define the `backendType` value to the name of the storage driver from the newly created backend. Also note the name-field value that must be referenced in a later step.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "solidfire-san"
```

15. Run the `kubectl` command to create the storage class.

```
ubuntu@gke-admin-ws-200915-151421:~/trident-installer$ kubectl create -f
sample-input/storage-class-basic.yaml
```

16. With the storage class created, you must then create the first persistent volume claim (PVC). There is a sample `pvc-basic.yaml` file that can be used to perform this action located in sample-inputs as well. The only edit that must be made to this file is ensuring that the storageClassName field matches the one just created.

```
ubuntu@gke-admin-ws-200915-151421:~/trident-installer$ vi sample-
input/pvc-basic.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

17. Create the PVC by issuing the `kubectl` command. Creation can take some time depending on the size of the backing volume being created, so you can watch the process as it completes.

```
ubuntu@gke-admin-ws-200915-151421:~/trident-installer$ kubectl create -f
sample-input/pvc-basic.yaml

ubuntu@gke-admin-ws-200915-151421:~/trident-installer$ kubectl get pvc
--watch
NAME      STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS   AGE
basic     Pending
basic         1s
basic     Pending   pvc-2azg0d2c-b13e-12e6-8d5f-5342040d22bf   0
basic         5s
basic     Bound     pvc-2azg0d2c-b13e-12e6-8d5f-5342040d22bf   1Gi
RWO            basic           7s
```

Next: Reference videos.

Video demos

The following videos demonstrate some of the capabilities documented in this NVA.

- Deploying an application from the Google Cloud Application Marketplace to Anthos:

► https://docs.netapp.com/us-en/hci-solutions//media/Anthos-Deploy-App-Demo.mp4 *(video)*

- Dynamic scaling of Kubernetes clusters deployed on Anthos on VMware:

► https://docs.netapp.com/us-en/hci-solutions//media/Anthos-Scaling-Demo.mp4 *(video)*

- Using NetApp Trident to provision and attach a persistent volume to a Kubernetes pod on Anthos:

► https://docs.netapp.com/us-en/hci-solutions//media/Anthos-Trident-Demo.mp4 *(video)*

NVA-1149: NetApp HCI for Red Hat OpenShift on Red Hat Virtualization

Alan Cowles and Nikhil M Kulkarni, NetApp

NetApp HCI for Red Hat OpenShift on Red Hat Virtualization (RHV) is a best-practice deployment guide for the fully automated install of Red Hat OpenShift through the Installer Provisioned Infrastructure (IPI) method onto the verified enterprise architecture of NVA-1148: NetApp HCI with Red Hat Virtualization. The purpose of this NetApp Verified Architecture deployment guide is to provide a concise set of verified instructions to be followed for the deployment of the solution. The architecture and deployment methods described in this document have been validated jointly by subject matter experts at NetApp and Red Hat to provide a best-practice implementation of the solution.

Use Cases

The NetApp HCI for Red Hat OpenShift on RHV solution is architected to deliver exceptional value for customers with the following use cases:

- Infrastructure to scale on demand with NetApp HCI
- Enterprise virtualized workloads in RHV
- Enterprise containerized workloads in Red Hat OpenShift

Business Value

Enterprises are increasingly adopting DevOps practices to create new products, shorten release cycles, and rapidly add new features. Because of their innate agile nature, containers and microservices play a crucial role in supporting DevOps practices. However, practicing DevOps at a production scale in an enterprise environment presents its own challenges and imposes certain requirements on the underlying infrastructure, such as the following:

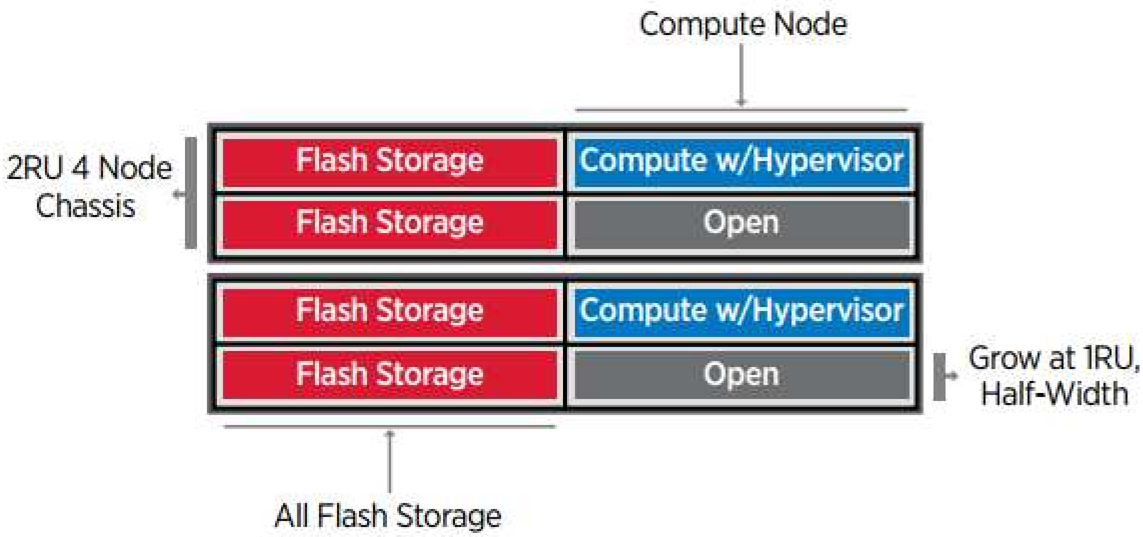
- High availability at all layers in the stack
- Ease of deployment procedures
- Nondisruptive operations and upgrades
- API-driven and programmable infrastructure to keep up with microservices agility
- Multitenancy with performance guarantees
- Ability to run virtualized and containerized workloads simultaneously
- Ability to scale infrastructure independently based on workload demands

NetApp HCI for Red Hat OpenShift on RHV acknowledges these challenges and presents a solution that helps address each concern by implementing the fully automated deployment of RedHat OpenShift IPI on the RHV enterprise hypervisor. The remainder of this document details the components used in this verified architecture.

Technology Overview

NetApp HCI

NetApp HCI is an enterprise-scale, disaggregated hybrid cloud infrastructure (HCI) solution that delivers compute and storage resources in an agile, scalable, and easy-to-manage two-rack unit (2RU), four-node building block. It can also be configured with 1RU compute and server nodes. The minimum deployment depicted in the figure below consists of four NetApp HCI storage nodes and two NetApp HCI compute nodes. The compute nodes are installed as Red Hat Virtualization Hosts (RHV-H) hypervisors in a high-availability (HA) cluster. This minimum deployment can be easily scaled to fit customer enterprise workload demands by adding additional NetApp HCI storage or compute nodes to expand available resources.



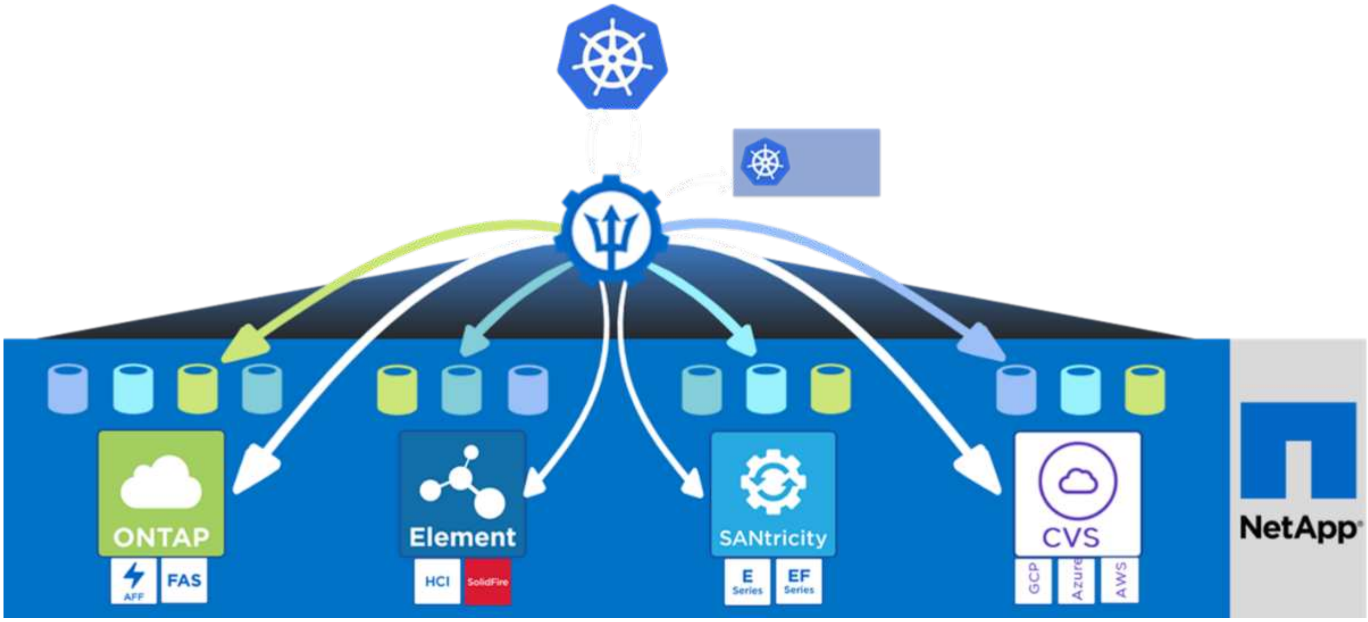
The design for NetApp HCI for Red Hat Virtualization consists of the following components in a minimum starting configuration:

- NetApp H-Series all-flash storage nodes running NetApp Element software
- NetApp H-Series compute nodes running the Red Hat Virtualization RHV-H hypervisor

For more information about compute and storage nodes in NetApp HCI, see NetApp HCI Datasheet.

NetApp Trident

Trident is a NetApp open-source and fully supported storage orchestrator for containers and Kubernetes distributions, including Red Hat OpenShift. It works with the entire NetApp storage portfolio, including the NetApp Element storage system that is deployed as a part of the NetApp HCI solution. Trident provides the ability to accelerate the DevOps workflow by allowing end users to provision and manage storage from their NetApp storage systems, without requiring intervention from a storage administrator. An administrator can configure a number of storage backends based on project needs, and storage system models that allow for any number of advanced storage features, such as: compression, specific disk types, or QoS levels that guarantee a certain performance. After they are defined, these backends can be leveraged by developers as part of their projects to create persistent volume claims (PVCs) and attach persistent storage to their containers on demand.



Red Hat Virtualization

RHV is an enterprise virtual data center platform that runs on Red Hat Enterprise Linux (RHEL) and uses the KVM hypervisor.

For more information about RHV, see the Red Hat Virtualization website.

RHV provides the following features:

- **Centralized management of VMs and hosts.** The RHV manager runs as a physical or virtual machine (VM) in the deployment and provides a web-based GUI for the management of the solution from a central interface.
- **Self-hosted engine.** To minimize the hardware requirements, RHV allows RHV Manager (RHV-M) to be deployed as a VM on the same hosts that run guest VMs.
- **High availability.** In event of host failures, to avoid disruption, RHV allows VMs to be configured for high availability. The highly available VMs are controlled at the cluster level using resiliency policies.
- **High scalability.** A single RHV cluster can have up to 200 hypervisor hosts enabling it to support requirements of massive VMs to hold resource-greedy, enterprise-class workloads.
- **Enhanced security.** Inherited from RHV, Secure Virtualization (sVirt) and Security Enhanced Linux (SELinux) technologies are employed by RHV for the purposes of elevated security and hardening for the hosts and VMs. The key advantage from these features is logical isolation of a VM and its associated resources.

Red Hat Virtualization Manager

RHV-M provides centralized enterprise-grade management for the physical and logical resources within the RHV virtualized environment. A web-based GUI with different role-based portals are provided to access RHV-M features.

RHV-M exposes configuration and management of RHV resources via open-source, community-driven RESTful API. It also supports full-fledged integration with Red Hat CloudForms and Red Hat Ansible for automation and orchestration.

Red Hat Virtualization Hosts

Hosts (also called hypervisors) are the physical servers that provide hardware resources for the VMs to run on. Kernel-based Virtual Machine (KVM) provides full virtualization support, and Virtual Desktop Server Manager (VDSM) is the host agent that is responsible for communication of the hosts with the RHV-M.

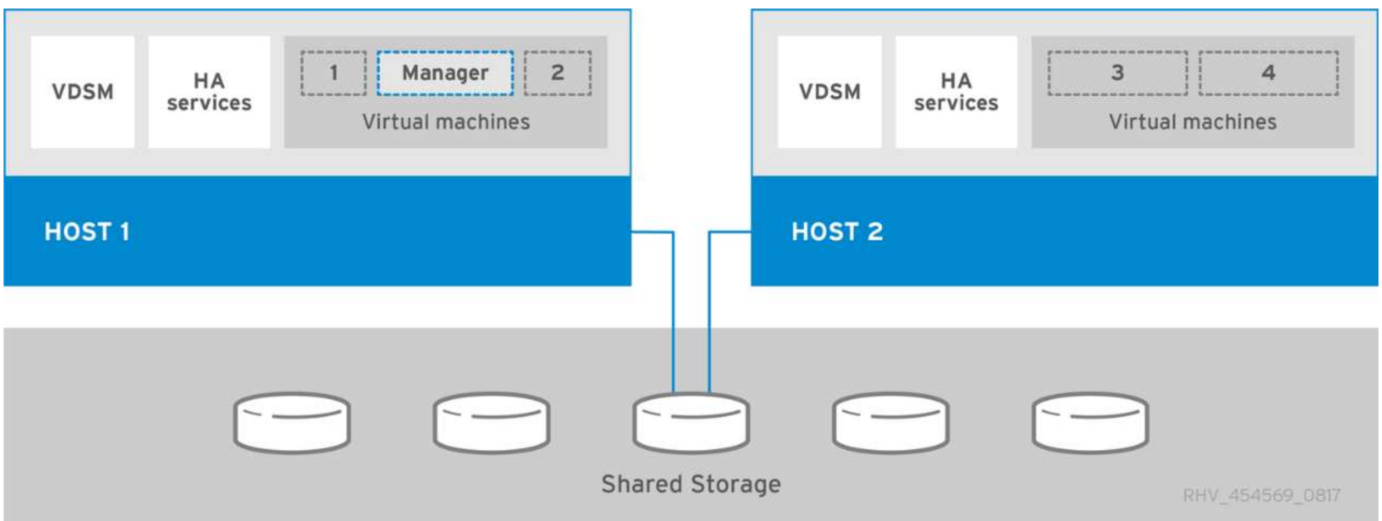
Two types of hosts are supported in RHV are RHV-H and RHEL hosts:

- RHV-H is a light-weight minimal operating system based on RHEL, optimized for ease of setting up physical servers as RHV hypervisors.
- RHEL hosts are servers that run the standard RHEL operating system and are later configured with the required subscriptions to install the packages required to permit the physical servers to be used as RHV hosts.

Red Hat Virtualization Architecture

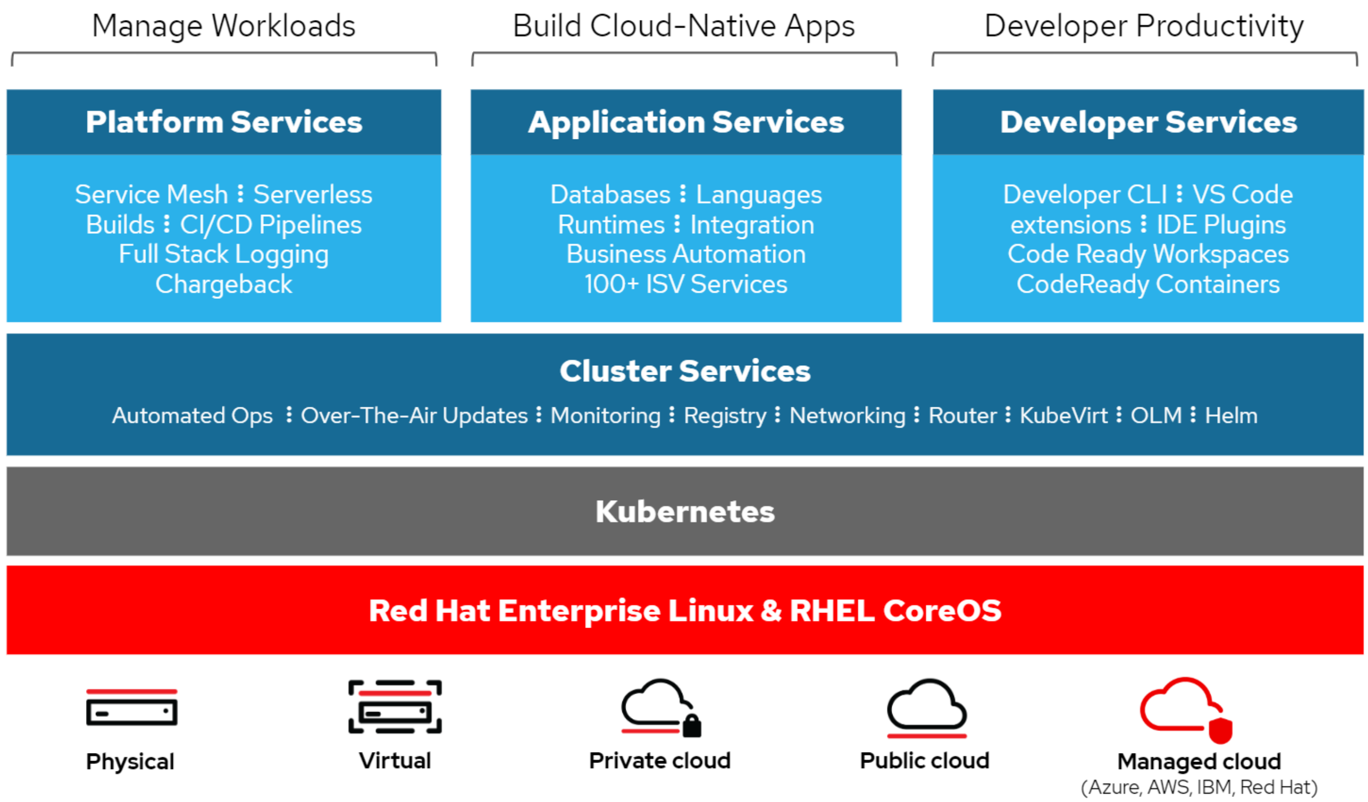
RHV can be deployed in two different architectures: with the RHV-M as a physical server in the infrastructure or with the RHV-M configured as a self-hosted engine. The self-hosted engine deployment, where the RHV-M is a VM hosted in the same environment as other VMs, is recommended and used specifically in this deployment guide.

A minimum of two self-hosted nodes are required for high availability of guest VMs and RHV-M as depicted in the figure below. For ensuring the high availability of the manager VM, HA services are enabled and run on all the self-hosted engine nodes.



Red Hat OpenShift Container Platform

Red Hat OpenShift Container Platform is a fully supported enterprise Kubernetes platform. Red Hat makes several enhancements to open-source Kubernetes to deliver an application platform with all the components fully integrated to build, deploy, and manage containerized applications. With Red Hat OpenShift 4.4, the installation and management processes have been streamlined through the IPI method which has been deployed in this solution. By leveraging this deployment method, a fully functional OpenShift cluster providing metering and monitoring at both the cluster and application level can be fully configured and deployed on top of Red Hat Virtualization in less than an hour. OpenShift nodes are based upon RHEL CoreOS, an immutable system image designed to run containers, based on RHEL, which can be upgraded or scaled easily on demand as the needs of the end user require, helping to deliver the benefits of the public cloud to the local data center.



Architectural Overview: NetApp HCI for Red Hat OpenShift on RHV

Hardware Requirements

The following table lists the minimum number of hardware components that are required to implement the solution. The hardware components that are used in specific implementations of the solution might vary based on customer requirements.

Hardware	Model	Quantity
NetApp HCI compute nodes	NetApp H410C	2
NetApp HCI storage nodes	NetApp H410S	4
Data switches	Mellanox SN2010	2
Management switches	Cisco Nexus 3048	2

Software Requirements

The following table lists the software components that are required to implement the solution. The software components that are used in any implementation of the solution might vary based on customer requirements.

Software	Purpose	Version
NetApp HCI	Infrastructure (compute/storage)	1.8
NetApp Element	Storage	12.0
NetApp Trident	Storage orchestration	20.04
RHV	Virtualization	4.3.9

Software	Purpose	Version
Red Hat OpenShift	Container orchestration	4.4.6

Design Considerations: NetApp HCI for Red Hat OpenShift on RHV

Network Design

The Red Hat OpenShift on RHV on HCI solution uses two data switches to provide primary data connectivity at 25Gbps. It also uses two additional management switches that provide connectivity at 1Gbps for in-band management for the storage nodes and out-of-band management for IPMI functionality. OCP uses the logical network on the RHV for the cluster management. This section describes the arrangement and purpose of each virtual network segment used in the solution and outlines the pre-requisites for deployment of the solution.

VLAN Requirements

The NetApp HCI for Red Hat OpenShift on RHV solution is designed to logically separate network traffic for different purposes by using virtual local area networks (VLANs). NetApp HCI requires a minimum of three network segments. However, this configuration can be scaled to meet customer demands or to provide further isolation for specific network services. The following table lists the VLANs that are required to implement the solution, as well as the specific VLAN IDs that are used later in the verified architecture deployment.

VLANs	Purpose	VLAN ID
Out-of-band management network	Management for HCI nodes and IPMI	16
In-band management network	Management for HCI nodes, ovirtmgmt, and VMs	1172
Storage network	Storage network for NetApp Element	3343
Migration network	Network for virtual guest migration	3345

Network Infrastructure Support Resources

The following infrastructure should be in place prior to the deployment of the OpenShift Container Platform (OCP) on Red Hat Virtualization on NetApp HCI solution:

- At least one DNS server which provides a full host-name resolution that is accessible from the in-band management network and the VM network.
- At least one NTP server that is accessible from the in-band management network and the VM network.
- (Optional) Outbound internet connectivity for both the in-band management network and the VM network.
- RHV cluster should have at least 28x vCPUs, 112GB RAM, and 840GB of available storage (depending on the production workload requirements).

Deploying NetApp HCI for Red Hat OpenShift on RHV

Deployment Summary: NetApp HCI for Red Hat OpenShift on RHV

The detailed steps provided in this section provide a validation for the minimum hardware and software configuration required to deploy and validate the NetApp HCI for Red Hat

OpenShift on RHV solution.

Deploying Red Hat OpenShift Container Platform through IPI on Red Hat Virtualization consists of the following steps:

1. Create storage network VLAN
2. Download OpenShift installation files
3. Download CA cert from RHV
4. Register API/Apps in DNS
5. Generate and add SSH private key
6. Install OpenShift Container Platform
7. Access console/web console
8. Configure worker nodes to run storage services
9. Download and install Trident through Operator

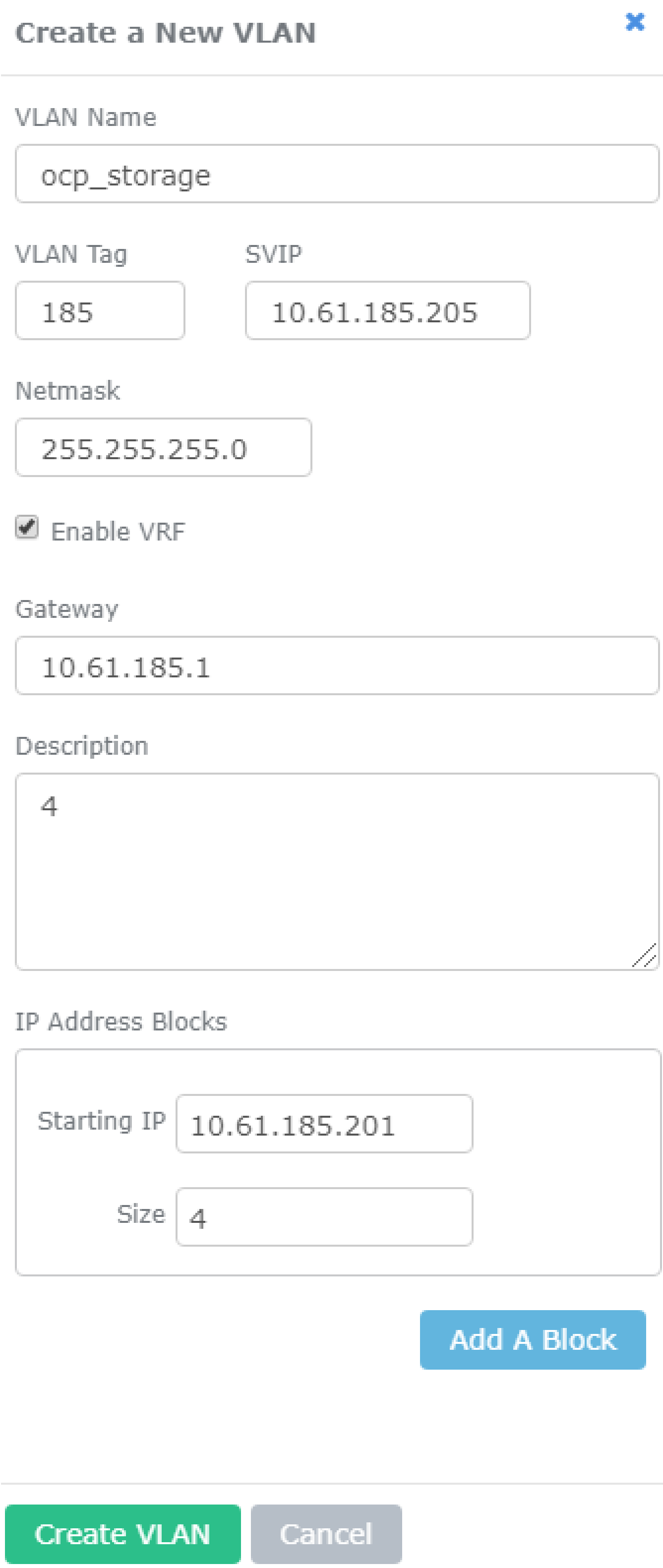
1. Create Storage Network VLAN: NetApp HCI for Red Hat OpenShift on RHV

To create a storage network VLAN, complete the following steps:

To support Element storage access for NetApp Trident to attach persistent volumes to pods deployed in OpenShift, the machine network being used for each worker in the OCP deployment must be able to reach the storage resources. If the machine network cannot access the Element storage network by default, an additional network/VLAN can be created in the Element cluster to allow access:

1. Using any browser, log in to the Element Cluster at the cluster’s MVIP.
2. Navigate to Cluster > Network and click Create VLAN.
3. Before you provide the details, reserve at least five IP addresses from the network that is reachable from the OCP network (one for the virtual network storage VIP and one for virtual network IP on each storage node).

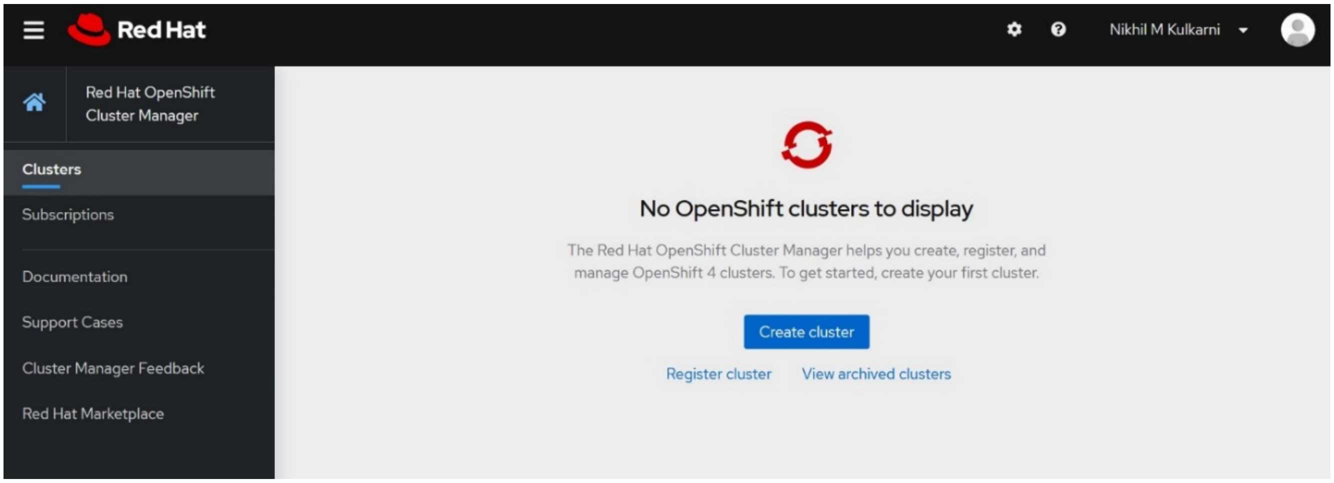
   Enter a VLAN name of your choice, enter the VLAN ID, SVIP, and netmask, select the Enable VRF option, and enter the gateway IP for the network. In the IP address blocks, enter the starting IP of the other addresses reserved for the storage nodes. In this example, the size is four because there are four storage nodes in this cluster. Click Create VLAN.



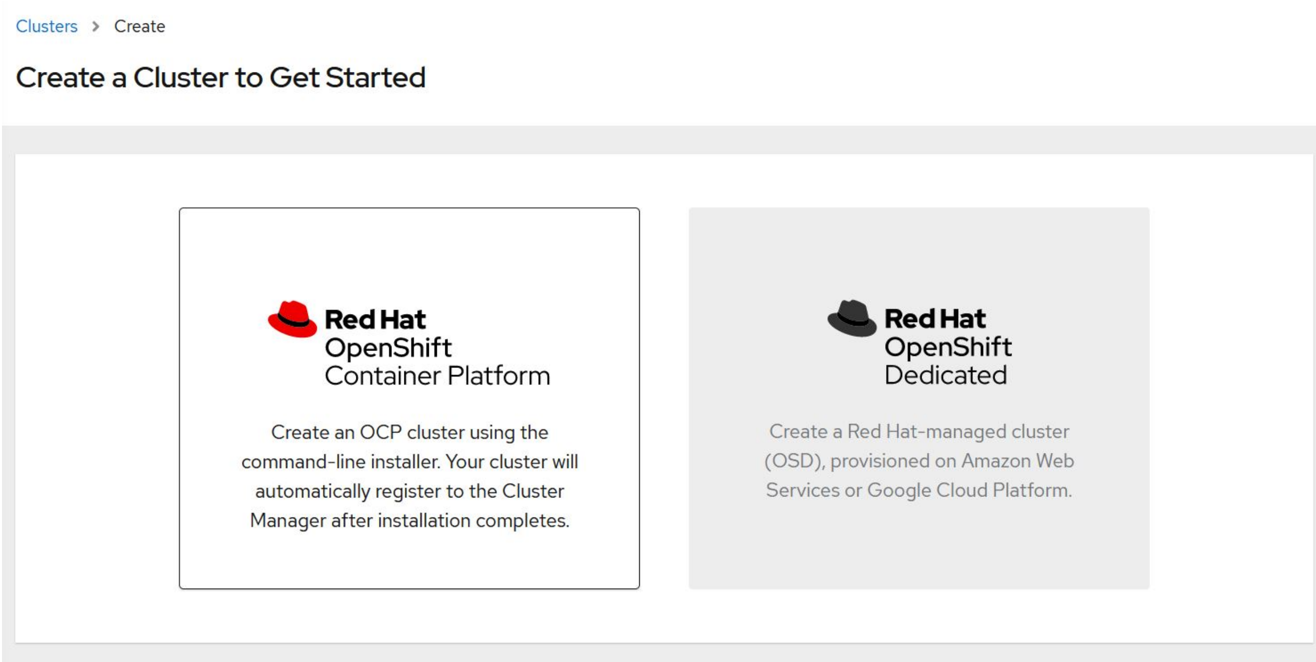
2. Download OpenShift Installation Files: NetApp HCI for Red Hat OpenShift on RHV

To download the OpenShift installation files, complete the following steps:

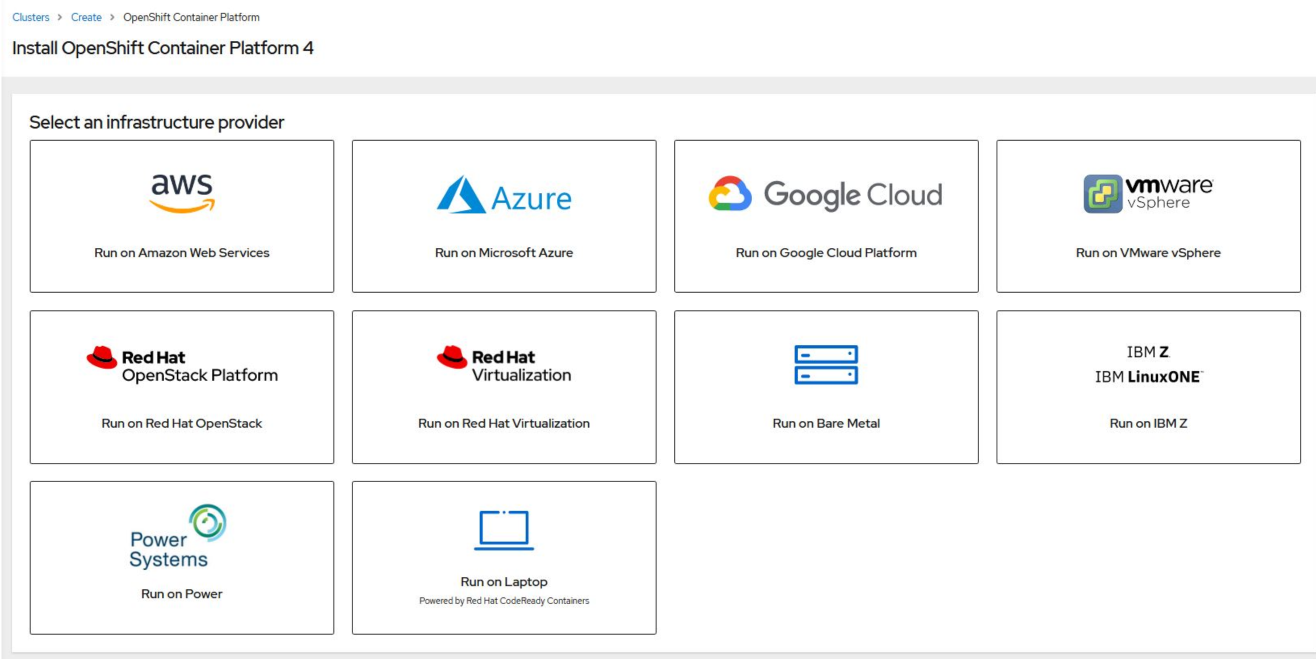
1. Go to the Red Hat login page and log in with your Red Hat credentials.
2. On the Clusters page, click Create Cluster.



3. Select OpenShift Container Platform.

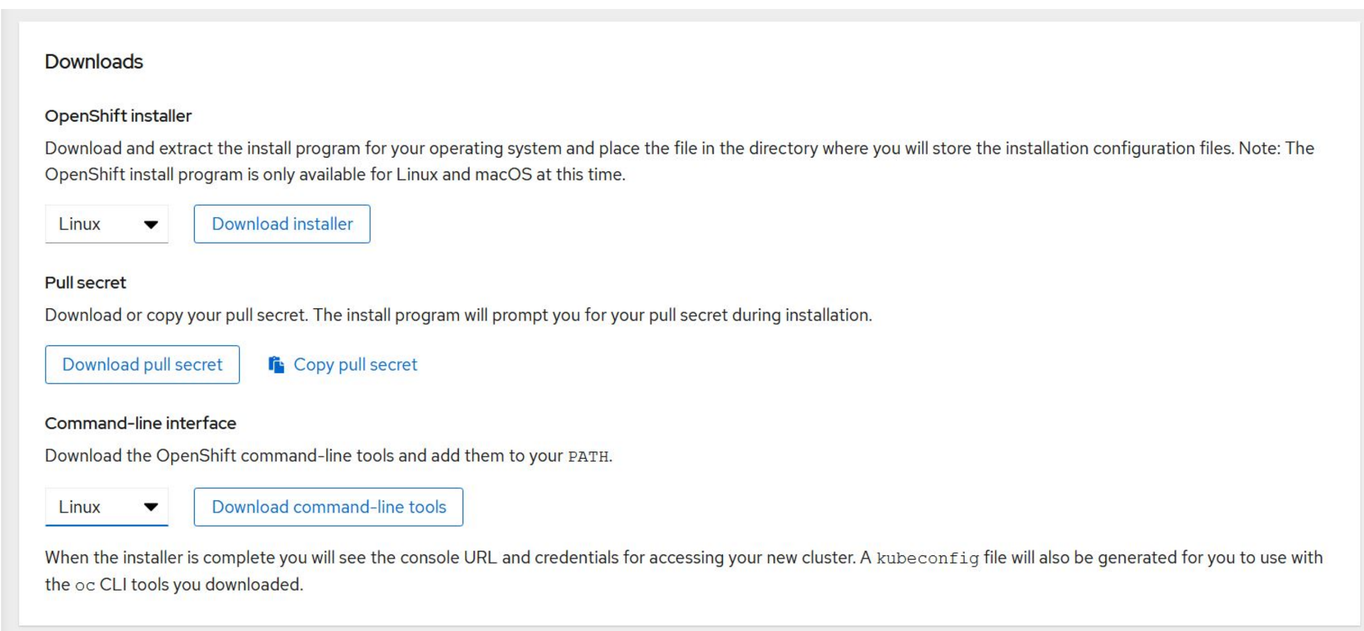


4. Select Run on Red Hat Virtualization.



5. The next page allows you to download the OpenShift installer (available for Linux and MacOS), a unique pull secret that is required to create the `install-config` file and the `oc` command-line tools (available for Linux, Windows, and MacOS).

   Download the files, transfer them to a RHEL administrative workstation from where you can run the OpenShift installation, or download these files directly using wget or curl on a RHEL administrative workstation.



3. Download CA Certificate from RHV: NetApp HCI for Red Hat OpenShift on RHV

To download the CA certificate from RHV, complete the following steps:

1. In order to access the RHV manager from the RHEL machine during the deployment process, the CA certificate trust must be updated on the machine to trust connections to RHV-M. To download the RHV

Manager’s CA certificate, run the following commands:

```
sudo curl -k 'https://<engine-fqdn>/ovirt-engine/services/pki-
resource?resource=ca-certificate&format=X509-PEM-CA' -o /tmp/ca.pem
[user@rhel7 ~]$  sudo curl -k 'https://rhv-m.cie.netapp.com/ovirt-
engine/services/pki-resource?resource=ca-certificate&format=X509-PEM-CA'
-o /tmp/ca.pem
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
100  1376  100  1376    0     0   9685      0 --:--:-- --:--:-- --:--:--
9690
```

2. Copy the CA certificate to the directory for server certificates and update the CA trust.

```
[user@rhel7 ~]$  sudo cp /tmp/ca.pem /etc/pki/ca-
trust/source/anchors/ca.pem
[user@rhel7 ~]$ sudo update-ca-trust
```

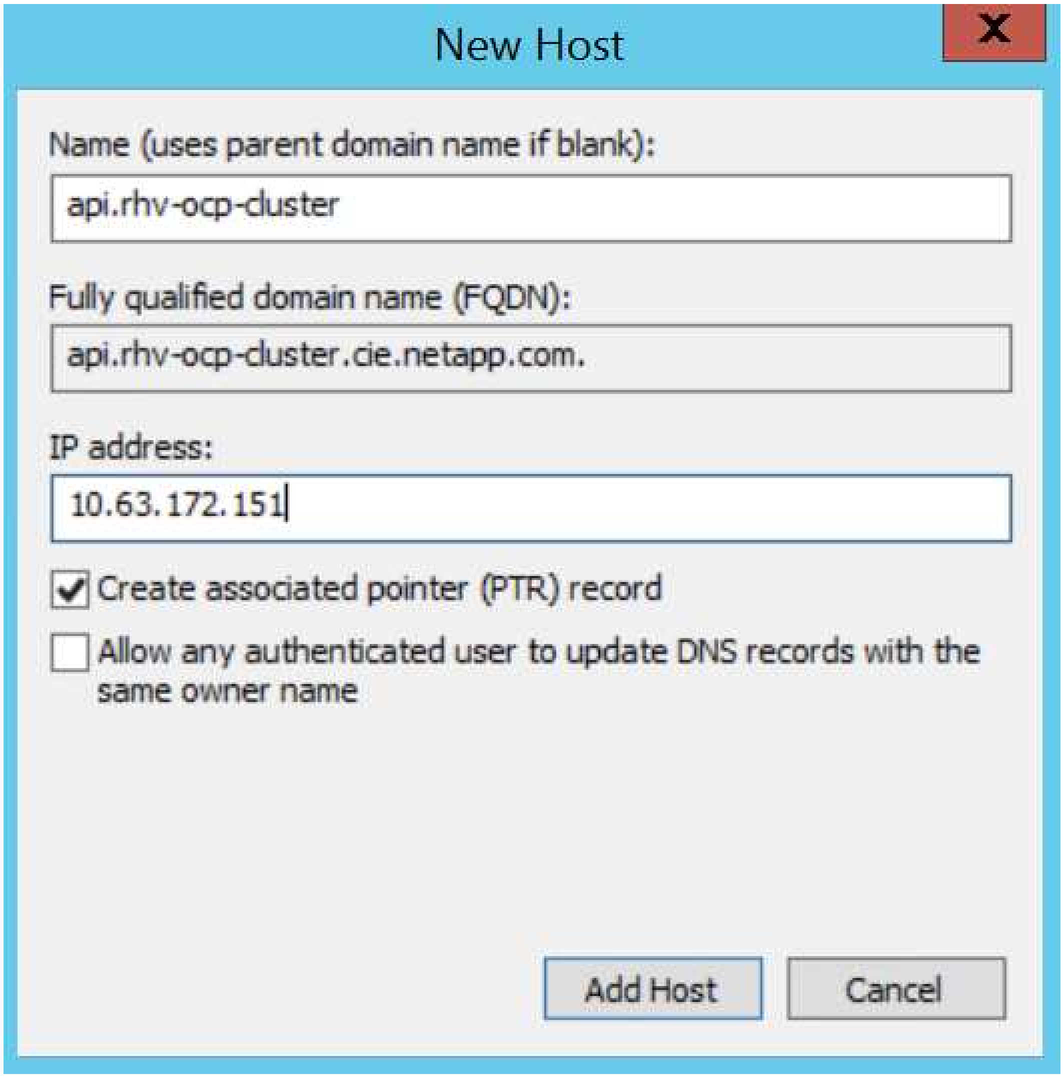
4. Register API/Apps in DNS: NetApp HCI for Red Hat OpenShift on RHV

To register API/Apps in DNS, complete the following steps:

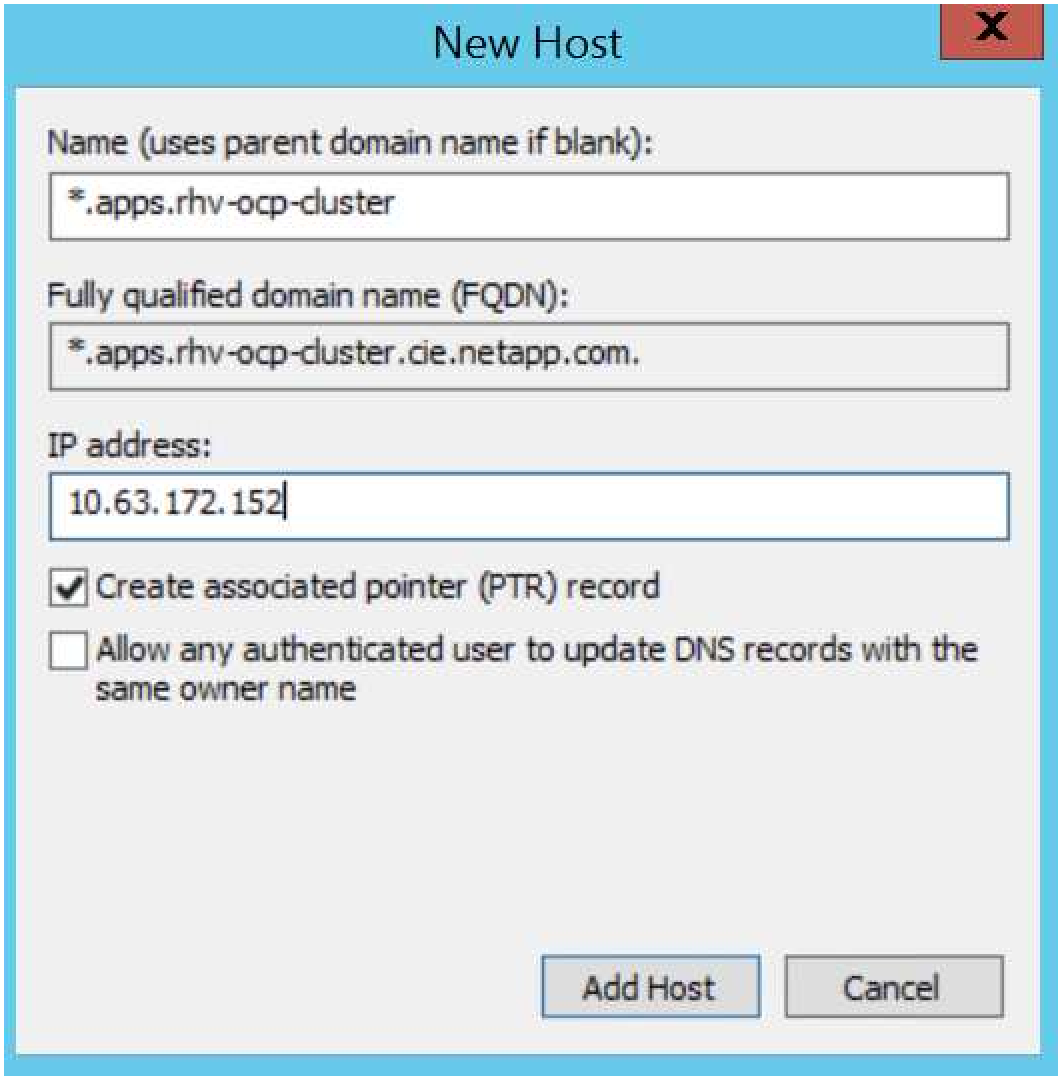
1. Reserve three static IP addresses from the network being used for OCP: the first IP address for OpenShift Container Platform REST API, the second IP address for pointing to the wildcard application ingress, and the third IP address for the internal DNS service. The first two IPs require an entry in the DNS server.

   > The default value of the `machineNetwork` subnet as created by IPI during OpenShift install is `10.0.0.0/16`. If the IPs you intend to use for your cluster’s management network fall outside of this range, you might need to customize your deployment and edit these values before deploying the cluster. For more information, see the section Use a Custom Install File for OpenShift Deployment.

2. Configure the API domain name by using the format `api.<openshift-cluster-name>.<base-domain>` pointing to the reserved IP.



3. Configure the wildcard application ingress domain name by using the format `*.apps.<openshift-cluster-name>.<base-domain>` pointing to the reserved IP.



5. Generate and Add SSH Private Key: NetApp HCI for Red Hat OpenShift on RHV

To generate and add an SSH private key, complete the following steps:

1. For the installation debugging or disaster recovery on the OpenShift cluster, you must provide an SSH key to both the `ssh-agent` and the installation program. Create an SSH key if one does not already exist for password-less authentication on the RHEL machine.

```
[user@rhel7 ~]$  ssh-keygen -t rsa -b 4096 -N '' -f ~/.ssh/id_rsa
```

2. Start the `ssh-agent` process and configure it as a background running task.

```
[user@rhel7 ~]$  eval "$(ssh-agent -s)"
Agent pid 31874
```

3. Add the SSH private key that you created in step 2 to the `ssh-agent` , which enables you to SSH directly to the nodes without having to interactively pass the key.

```
[user@rhel7 ~]$  ssh-add ~/.ssh/id_rsa
```

6. Install OpenShift Container Platform: NetApp HCI for Red Hat OpenShift on RHV

To install OpenShift Container Platform, compete the following steps:

1. Create a directory for OpenShift installation and transfer the downloaded files to it. Extract the OpenShift installer files from the tar archive.

```
[user@rhel7 ~]$ mkdir openshift-deploy
[user@rhel7 ~]$ cd openshift-deploy
[user@rhel7 openshift-deploy]$ tar xvf openshift-install-linux.tar.gz
README.md
openshift-install
[user@rhel7 openshift-deploy]$ ls -la
total 453260
drwxr-xr-x.  2 user user       146 May 26 16:01 .
dr-xr-x---. 16 user user      4096 May 26 15:58 ..
-rw-r--r--.  1 user user  25249648 May 26 15:59 openshift-client-
linux.tar.gz
-rwxr-xr-x.  1 user user 354664448 Apr 27 01:37 openshift-install
-rw-r--r--.  1 user user  84207215 May 26 16:00 openshift-install-
linux.tar.gz
-rw-r--r--.  1 user user      2736 May 26 15:59 pull-secret.txt
-rw-r--r--.  1 user user       706 Apr 27 01:37 README.md
```

The installation program creates several files in the directory used for installation of the cluster. Both the installation program and the files created by the installation program must be kept even after the cluster is up.

The binary files that you previously downloaded, such as `openshift-install` or `oc`, can be copied to a directory that is in the user’s path (for example, `/usr/local/bin`) to make them easier to run.

2. Create the cluster by running the `openshift-install create cluster` command and respond to the installation program prompts. Pass the SSH public key, select ovirt from the platform, provide the RHV infrastructure details, provide the three reserved IP addresses and the downloaded pull secret to the installation program prompts. After all the inputs are provided, the installation program creates and configures a bootstrap machine with a temporary Kubernetes control plane which then creates and configures the master VMs with the production Kubernetes control plane. The control plane on the master nodes creates and configures the worker VMs.

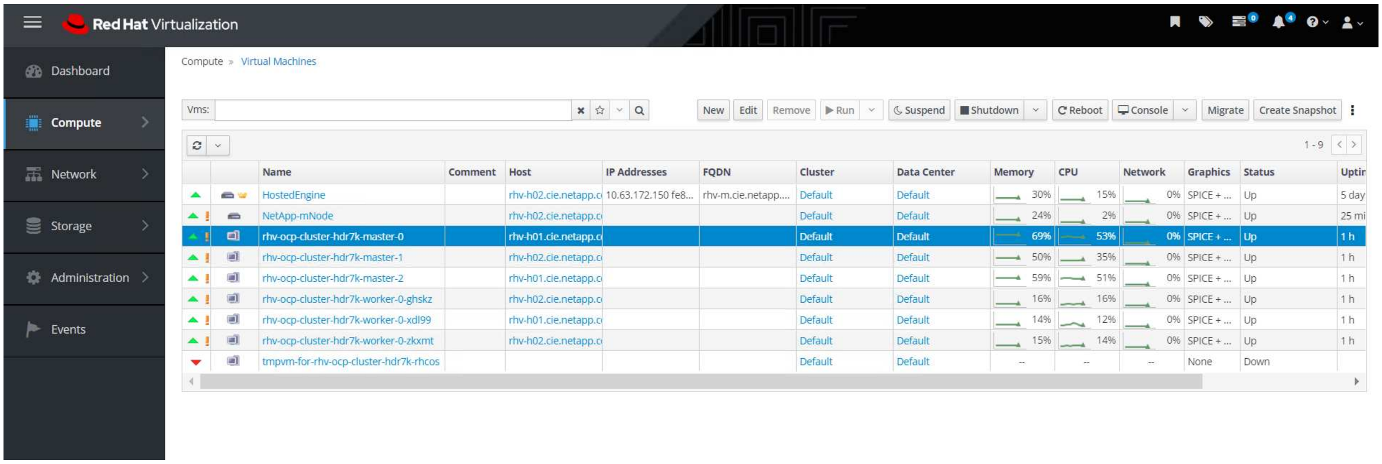
   It can take approximately 30–45 minutes to get the complete cluster up and running.

```
[user@rhel7 openshift-deploy]$ ./openshift-install create cluster
--dir=/home/user/openshift-deploy --log-level=info                    ?
SSH Public Key /home/user/.ssh/id_rsa.pub
? Platform ovirt
? oVirt cluster Default
? oVirt storage domain data_domain
? oVirt network ovirtmgmt
? Internal API virtual IP 10.63. 172.151
? Internal DNS virtual IP 10.63. 172.153
? Ingress virtual IP 10.63. 172.152
? Base Domain cie.netapp.com
? Cluster Name rhv-ocp-cluster
? Pull Secret [? for help]
************************************************************************
************************************************************************
************************************************************************
************************************************************************
********
INFO Obtaining RHCOS image file from 'https://releases-art-
rhcos.svc.ci.openshift.org/art/storage/releases/rhcos-
4.4/44.81.202004250133-0/x86_64/rhcos-44.81.202004250133-0-
openstack.x86_64.qcow2.gz?sha256=f8a44e0ea8cc45882dc22eb632a63afb90b4148
39b8aa92f3836ede001dfe9cf'
INFO The file was found in cache: /home/user/.cache/openshift-
installer/image_cache/e263efbc53c0caf612bcfaad10e3dff0. Reusing...
INFO Creating infrastructure resources...
INFO Waiting up to 20m0s for the Kubernetes API at https://api.rhv-ocp-
cluster.cie.netapp.com:6443...
INFO API v1.17.1 up
INFO Waiting up to 40m0s for bootstrapping to complete...
INFO Destroying the bootstrap resources...
INFO Waiting up to 30m0s for the cluster at https://api.rhv-ocp-
cluster.cie.netapp.com:6443 to initialize...
INFO Waiting up to 10m0s for the openshift-console route to be
created...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run
'export KUBECONFIG=/home/user/openshift-deploy/auth/kubeconfig'
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.rhv-ocp-cluster.cie.netapp.com
INFO Login to the console with user: kubeadmin, password: NtsqU-p3qUb-
8Hscu-JfAq7
```

3. When the cluster deployment is complete, the directions for accessing the OpenShift cluster, including a link to its web console and credentials for the kubeadmin user, are displayed. Make sure to take a note of

these details.

4. Log in to the RHV Manager and observe that the VMs relating to the OCP cluster are up and running.



7. Access Console/Web Console: NetApp HCI for Red Hat OpenShift on RHV

To access the console or web console, complete the following steps:

1. To access the OCP cluster through the CLI, extract the `oc` command-line tools tar file and place its content in a directory that is in the user's path.

```
[user@rhel7 openshift-deploy]$ tar xvf openshift-client-linux.tar.gz
README.md
oc
kubectl
[user@rhel7 openshift-deploy]$ echo $PATH
/usr/local/bin: /usr/local/sbin:/sbin:/bin:/usr/sbin:/usr/bin

[user@rhel7 openshift-deploy]$ cp oc /usr/local/bin
```

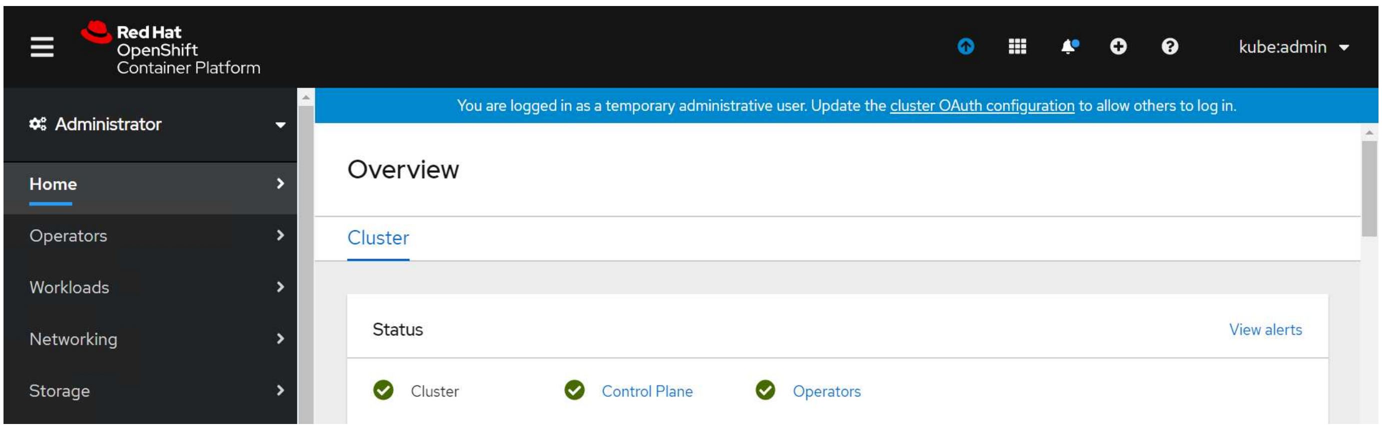
2. To interact with the cluster through the CLI, you can use the `kubeconfig` file provided by the IPI process located in the `/auth` directory inside the folder from where you launched the installation program. To easily interact with the cluster, export the file that is created in the directory. After a successful cluster deployment, the file location and the following command are displayed.

```
[user@rhel7 openshift-deploy]$ export KUBECONFIG=/home/user/openshift-
deploy/auth/kubeconfig
```

3. Verify whether you have access to the cluster and whether the nodes are in the Ready state.

```
[user@rhel7 openshift-deploy]$ oc get nodes
NAME                                 STATUS   ROLES    AGE   VERSION
rhv-ocp-cluster-hdr7k-master-0       Ready    master   93m   v1.17.1
rhv-ocp-cluster-hdr7k-master-1       Ready    master   93m   v1.17.1
rhv-ocp-cluster-hdr7k-master-2       Ready    master   93m   v1.17.1
rhv-ocp-cluster-hdr7k-worker-0-ghskz Ready    worker   83m   v1.17.1
rhv-ocp-cluster-hdr7k-worker-0-xdl99 Ready    worker   86m   v1.17.1
rhv-ocp-cluster-hdr7k-worker-0-zkxmt Ready    worker   85m   v1.17.1
```

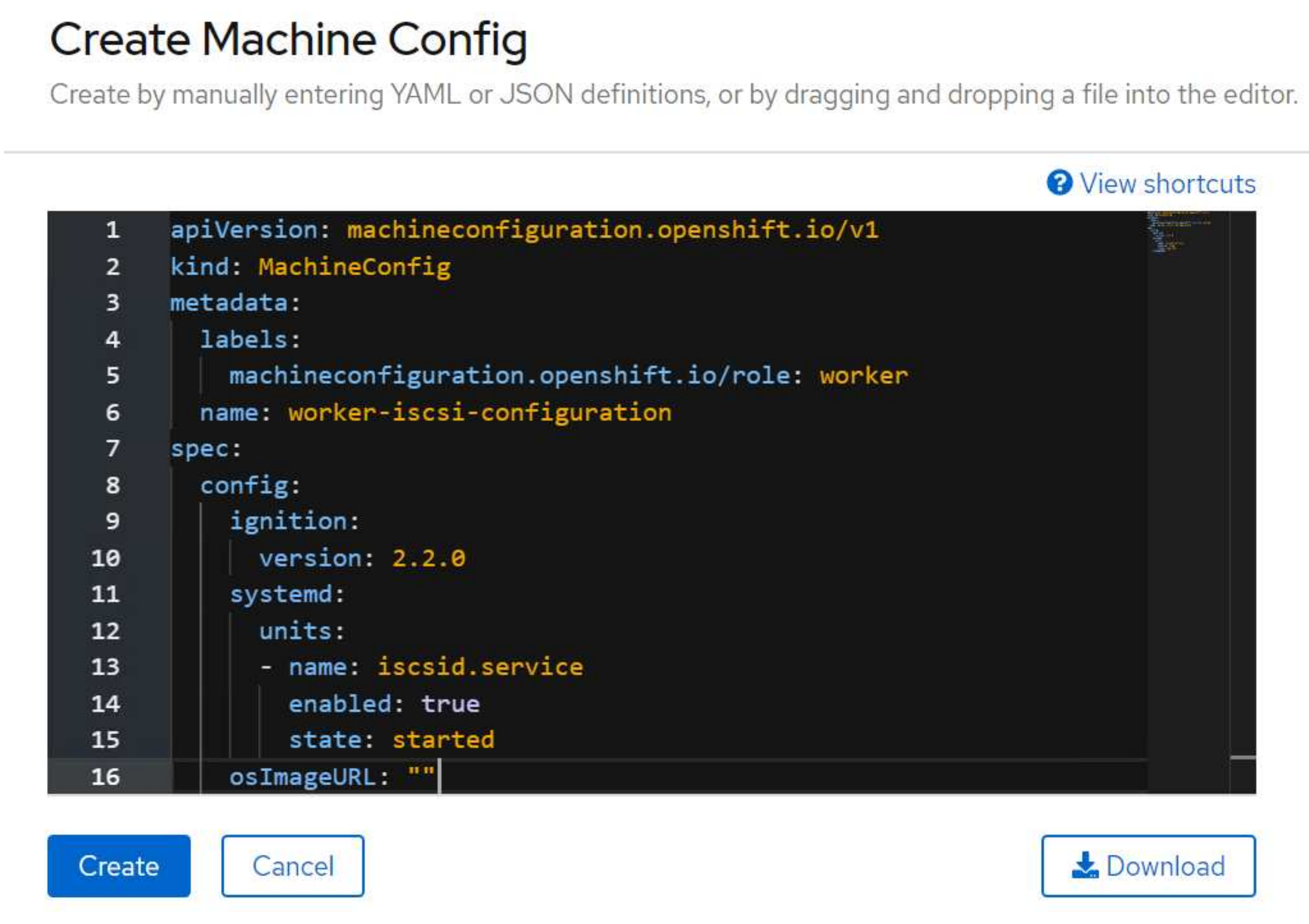
4. Log in to the web console URL by using the credentials, both of which were provided after the successful deployment of the cluster, and then verify GUI access to the cluster.



8. Configure Worker Nodes to Run Storage Services: NetApp HCI for Red Hat OpenShift on RHV

To configure the worker nodes to run storage services, complete the following steps:

1. To access storage from the Element system, each of the worker nodes must have iSCSI available and running as a service. To create a machine configuration that can enable and start the iscisd service, log in to the OCP web console and navigate to Compute > Machine Configs and click Create Machine Config. Paste the YAML file and click Create.



2. After the configuration is created, it will take approximately 20–30 minutes to apply the configuration to the worker nodes and reload them. Verify whether the machine config is applied by using `oc get mcp` and make sure that the machine config pool for workers is updated. You can also log in to the worker nodes to confirm that the iscsid service is running.

```
[user@rhel7 openshift-deploy]$ oc get mcp
NAME     CONFIG                                    UPDATED   UPDATING
DEGRADED
master   rendered-master-a520ae930e1d135e0dee7168   True      False
False
worker   rendered-worker-de321b36eeba62df41feb7bc   True      False
False
[user@rhel7 openshift-deploy]$ ssh core@10.63. 172.22 sudo systemctl
status iscsid
● iscsid.service - Open-iSCSI
   Loaded: loaded (/usr/lib/systemd/system/iscsid.service; enabled;
vendor preset: disabled)
   Active: active (running) since Tue 2020-05-26 13:36:22 UTC; 3 min ago
     Docs: man:iscsid(8)
           man:iscsiadm(8)
 Main PID: 1242 (iscsid)
   Status: "Ready to process requests"
    Tasks: 1
   Memory: 4.9M
      CPU: 9ms
   CGroup: /system.slice/iscsid.service
           └─1242 /usr/sbin/iscsid -f
```

It is also possible to confirm that the MachineConfig has been successfully applied and services have been started as expected by running the `oc debug` command with the appropriate flags.

9. Download and Install NetApp Trident: NetApp HCI for Red Hat OpenShift on RHV

To download and install NetApp Trident, complete the following steps:

1. Make sure that the user that is logged in to the OCP cluster has sufficient privileges for installing Trident.

```
[user@rhel7 openshift-deploy]$ oc auth can-i '*' '*' --all-namespaces
 yes
```

2. Verify that you can download an image from the registry and access the MVIP of the NetApp Element cluster.

```
[user@rhel7 openshift-deploy]$ oc run -i --tty ping --image=busybox
--restart=Never --rm -- ping 10.63.172.140
If you don't see a command prompt, try pressing enter.
64 bytes from 10.63.172.140: seq=1 ttl=63 time=0.312 ms
64 bytes from 10.63.172.140: seq=2 ttl=63 time=0.271 ms
64 bytes from 10.63.172.140: seq=3 ttl=63 time=0.254 ms
64 bytes from 10.63.172.140: seq=4 ttl=63 time=0.309 ms
64 bytes from 10.63.172.140: seq=5 ttl=63 time=0.319 ms
64 bytes from 10.63.172.140: seq=6 ttl=63 time=0.303 ms
^C
--- 10.63.172.140 ping statistics ---
7 packets transmitted, 7 packets received, 0% packet loss
round-trip min/avg/max = 0.254/0.387/0.946 ms
pod "ping" deleted
```

3. Download the Trident installer bundle using the following commands and extract it to a directory.

```
[user@rhel7 ~]$ wget
[user@rhel7 ~]$ tar -xf trident-installer-20.04.0.tar.gz
[user@rhel7 ~]$ cd trident-installer
```

4. The Trident installer contains manifests for defining all the required resources. Using the appropriate manifests, create the TridentProvisioner custom resource definition.

```
[user@rhel7 trident-installer]$ oc create -f
deploy/crds/trident.netapp.io_tridentprovisioners_crd_post1.16.yaml

customresourcedefinition.apiextensions.k8s.io/tridentprovisioners.triden
t.netapp.io created
```

5. Create a Trident namespace, which is required for the Trident operator.

```
[user@rhel7 trident-installer]$ oc create namespace trident
namespace/trident created
```

6. Create the resources required for the Trident operator deployment, such as a ServiceAccount for the operator, a ClusterRole and ClusterRoleBinding to the ServiceAccount, a dedicated PodSecurityPolicy, or the operator itself.

```
[user@rhel7 trident-installer]$ oc kustomize deploy/ >
deploy/bundle.yaml
[user@rhel7 trident-installer]$ oc create -f deploy/bundle.yaml
serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created
```

7. Verify that the Trident operator is deployed.

```
[user@rhel7 trident-installer]$ oc get deployment -n trident
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator   1/1     1            1           56s
[user@rhel7 trident-installer]$ oc get pods -n trident
NAME                              READY   STATUS    RESTARTS   AGE
trident-operator-564d7d66f-qrz7v   1/1     Running   0          71s
```

8. After the Trident operator is installed, install Trident using this operator. In this example, TridentProvisioner custom resource (CR) was created. The Trident installer comes with definitions for creating a TridentProvisioner CR. These can be modified based on the requirements.

```
[user@rhel7 trident-installer]$ oc create -f
deploy/crds/tridentprovisioner_cr.yaml
tridentprovisioner.trident.netapp.io/trident created
```

9. Approve the Trident serving CSR certificates by using `oc get csr -o name | xargs oc adm certificate approve`.

```
[user@rhel7 trident-installer]$ oc get csr -o name | xargs oc adm
certificate approve
certificatesigningrequest.certificates.k8s.io/csr-4b7zh approved
certificatesigningrequest.certificates.k8s.io/csr-4hkwc approved
certificatesigningrequest.certificates.k8s.io/csr-5bgh5 approved
certificatesigningrequest.certificates.k8s.io/csr-5g4d6 approved
certificatesigningrequest.certificates.k8s.io/csr-5j9hz approved
certificatesigningrequest.certificates.k8s.io/csr-5m8qb approved
certificatesigningrequest.certificates.k8s.io/csr-66hv2 approved
certificatesigningrequest.certificates.k8s.io/csr-6rdgg approved
certificatesigningrequest.certificates.k8s.io/csr-6t24f approved
certificatesigningrequest.certificates.k8s.io/csr-76wgv approved
certificatesigningrequest.certificates.k8s.io/csr-78qsq approved
certificatesigningrequest.certificates.k8s.io/csr-7r58n approved
certificatesigningrequest.certificates.k8s.io/csr-8ghmk approved
certificatesigningrequest.certificates.k8s.io/csr-8sn5q approved
```

10. Verify that Trident 20.04 is installed by using the TridentProvisioner CR, and verify that the pods related to Trident are.

```
[user@rhel7 trident-installer]$ oc get tprov -n trident
NAME      AGE
trident   9m49s

[user@rhel7 trident-installer]$ oc describe tprov trident -n trident
Name:         trident
Namespace:    trident
Labels:       <none>
Annotations:  <none>
API Version:  trident.netapp.io/v1
Kind:         TridentProvisioner
Metadata:
  Creation Timestamp:  2020-05-26T18:49:19Z
  Generation:          1
  Resource Version:    640347
  Self Link:
/apis/trident.netapp.io/v1/namespaces/trident/tridentprovisioners/triden
t
  UID:                 52656806-0414-4ed8-b355-fc123fafbf4e
Spec:
  Debug:  true
Status:
  Message:  Trident installed
  Status:   Installed
  Version:  v20.04
```

```
Events:
  Type    Reason      Age                    From
Message
  ----    ------      ----                   ----
-------
  Normal  Installing  9m32s                  trident-operator.netapp.io
Installing Trident
  Normal  Installed   3m47s (x5 over 8m56s)  trident-operator.netapp.io
Trident installed
[user@rhel7 trident-installer]$ oc get pods -n trident
NAME                               READY   STATUS    RESTARTS   AGE
trident-csi-7f769c7875-s6fmt       5/5     Running   0          10m
trident-csi-cp7wg                  2/2     Running   0          10m
trident-csi-hhx94                  2/2     Running   0          10m
trident-csi-l72bt                  2/2     Running   0          10m
trident-csi-xfl9d                  2/2     Running   0          10m
trident-csi-xrhqx                  2/2     Running   0          10m
trident-csi-zb7ws                  2/2     Running   0          10m
trident-operator-564d7d66f-qrz7v   1/1     Running   0          27m

[user@rhel7 trident-installer]$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 20.04.0        | 20.04.0        |
+----------------+----------------+
```

11. Create a storage backend that will be used by Trident to provision volumes. The storage backend specifies the Element cluster in NetApp HCI. You also can specify sample bronze, silver, and gold types with corresponding QoS specs.

```
[user@rhel7 trident-installer]$ vi backend.json
{
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://admin: admin- password@10.63.172.140/json-
rpc/8.0",
    "SVIP": "10.61.185.205:3260",
    "TenantName": "trident",
    "Types": [{"Type": "Bronze", "Qos": {"minIOPS": 1000, "maxIOPS":
2000, "burstIOPS": 4000}},
              {"Type": "Silver", "Qos": {"minIOPS": 4000, "maxIOPS":
6000, "burstIOPS": 8000}},
              {"Type": "Gold", "Qos": {"minIOPS": 6000, "maxIOPS": 8000,
"burstIOPS": 10000}}]
}
[user@rhel7 trident-installer]$ ./tridentctl -n trident create backend
-f backend.json
+-------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME           | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------------+----------------
+--------------------------------------+--------+---------+
| solidfire_10.61.185.205 | solidfire-san  | 40f48d99-5d2e-4f6c-89ab-
8aee2be71255 | online |       0 |
+-------------------------+----------------
+--------------------------------------+--------+---------+
```

Modify the `backend.json` to accommodate the details or requirements of your environment for the following values:

- Endpoint corresponds to the credentials and the MVIP of the NetApp HCI Element cluster.
- SVIP corresponds to the SVIP configured over the VM network in the section titled Create Storage Network VLAN.
- Types corresponds to different QoS bands. New persistent volumes can be created with specific QoS settings by specifying the exact storage pool.

12. Create a StorageClass that specifies Trident as the provisioner and the storage backend as `solidfire-san`.

```
[user@rhel7 trident-installer]$ vi storage-class-basic.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: basic-csi
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "solidfire-san"
  provisioningType: "thin"

[user@rhel7 trident-installer]$ oc create -f storage-class-basic.yaml
storageclass.storage.k8s.io/basic created
```

In this example, the StorageClass created is set as a default, however an OpenShift administrator can define multiple storage classes corresponding to different QoS requirements and other factors based upon their applications. Trident selects a storage backend that can satisfy all the criteria specified in the parameters section in the storage class definition. End users can then provision storage as needed, without administrative intervention.

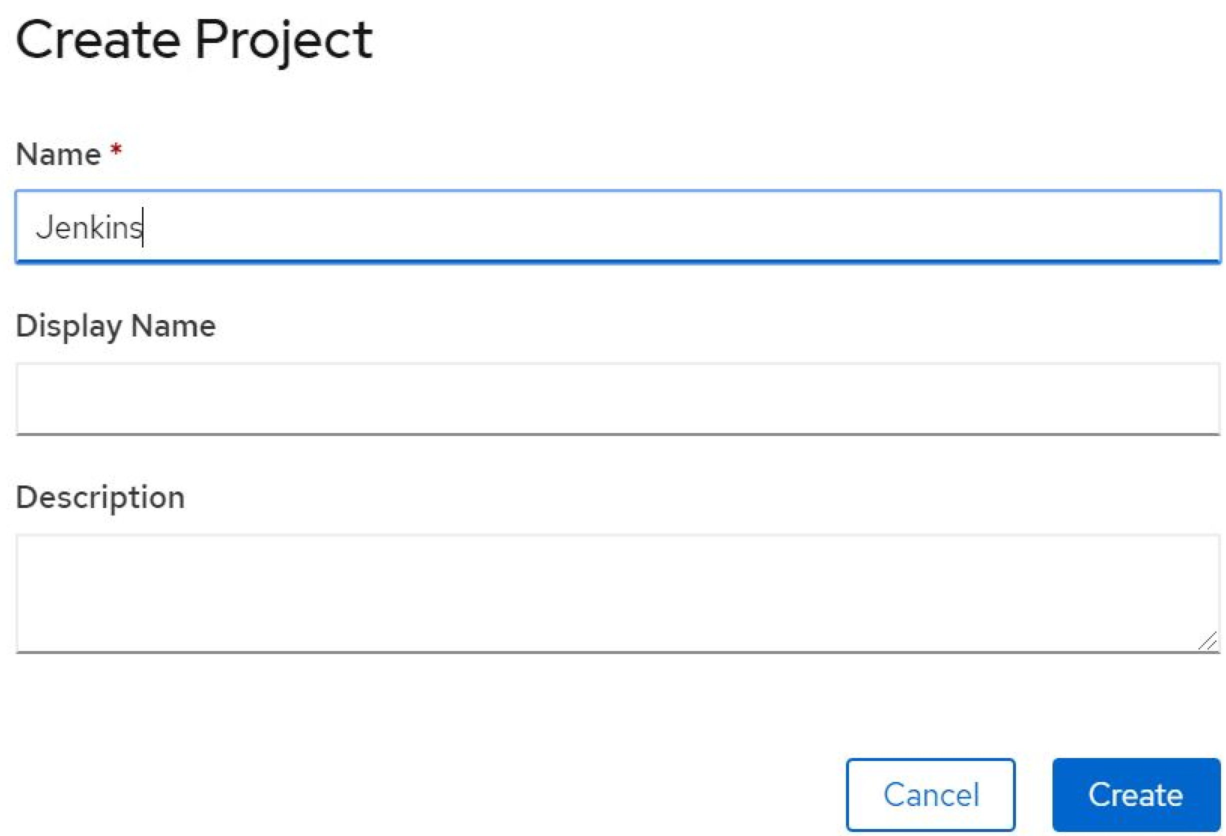
Validation Results: NetApp HCI for Red Hat OpenShift on RHV

This section provides the steps to deploy a continuous integration/continuous delivery or deployment (CI/CD) pipeline with Jenkins in order to validate the operation of the solution.

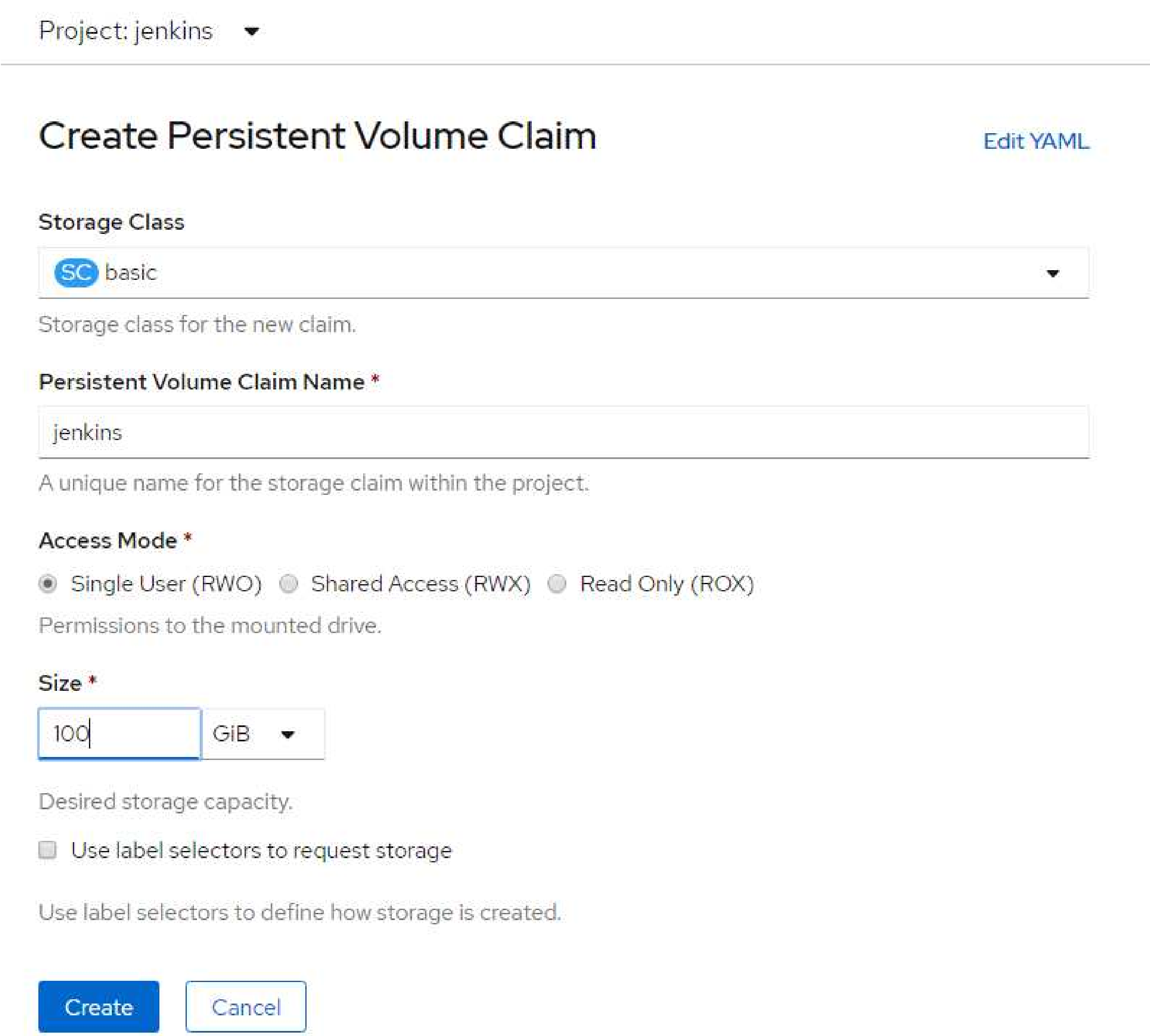
Create the Resources Required for Jenkins Deployment

To create the resources required for deploying the Jenkins application, complete the following steps:

1. Create a new project named Jenkins.



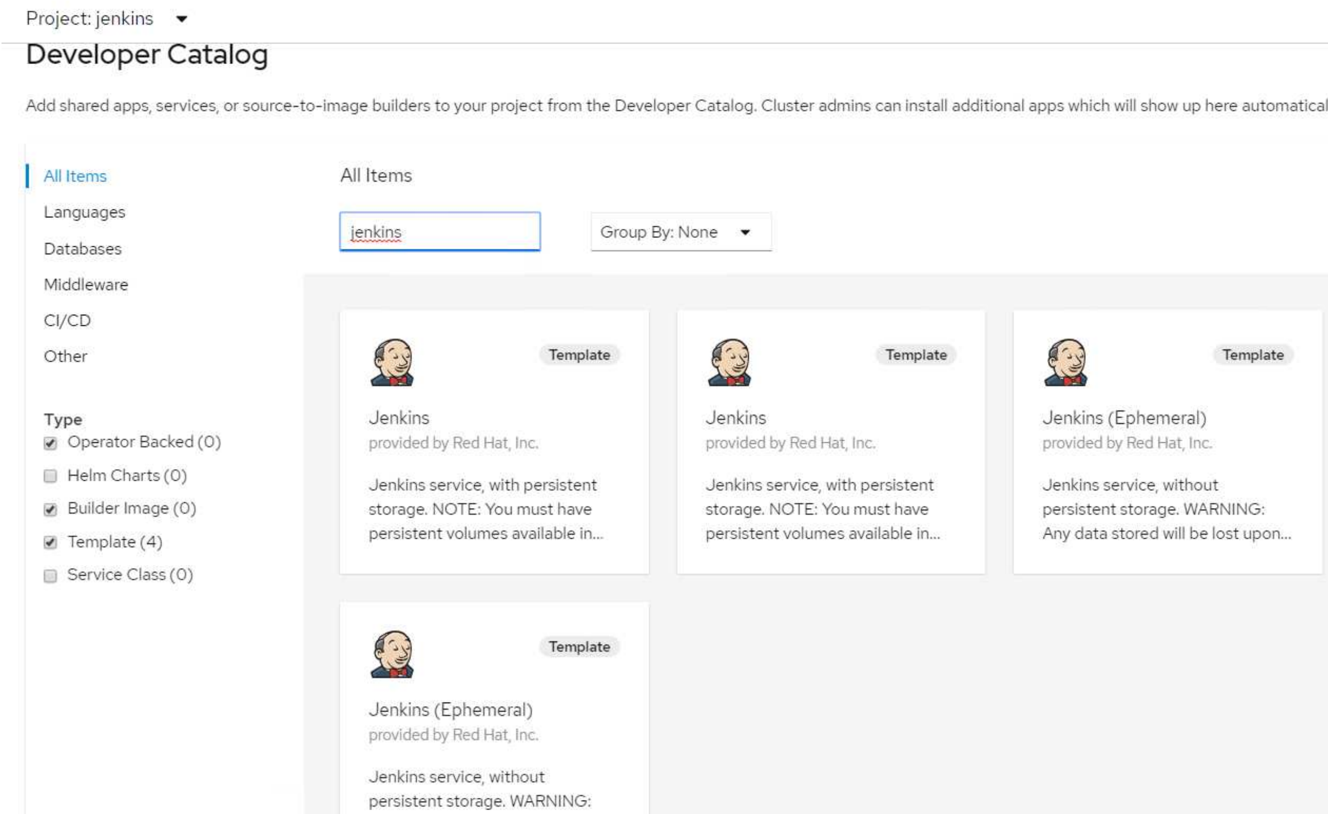
2. In this example, we deployed Jenkins with persistent storage. To support the Jenkins build, create the PVC. Navigate to Storage > Persistent Volume Claims and click Create Persistent Volume Claim. Select the storage class that was created, make sure that the Persistent Volume Claim Name is jenkins, select the appropriate size and access mode, and then click Create.



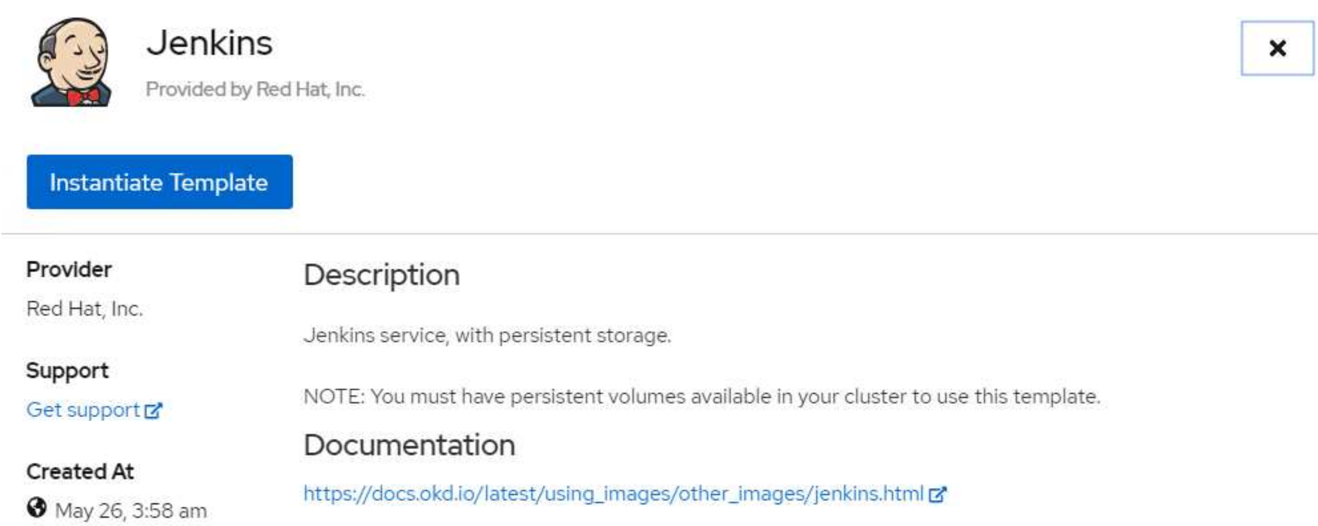
Deploy Jenkins with Persistent Storage

To deploy Jenkins with persistent storage, complete the following steps:

1. In the upper left corner, change the role from Administrator to Developer. Click +Add and select From Catalog. In the Filter by Keyword bar, search jenkins. Select Jenkins Service, with Persistent Storage.

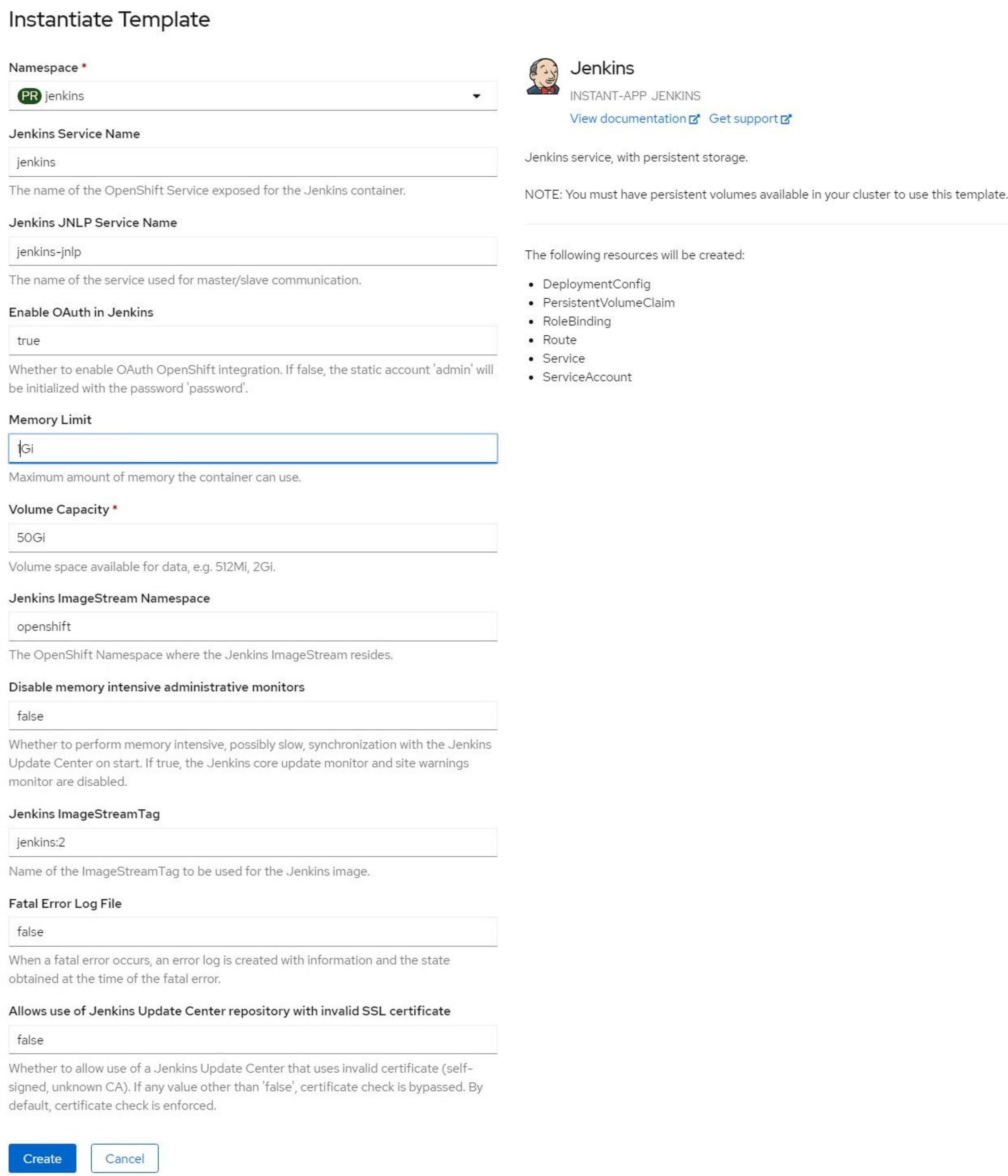


2. Click Instantiate Template.

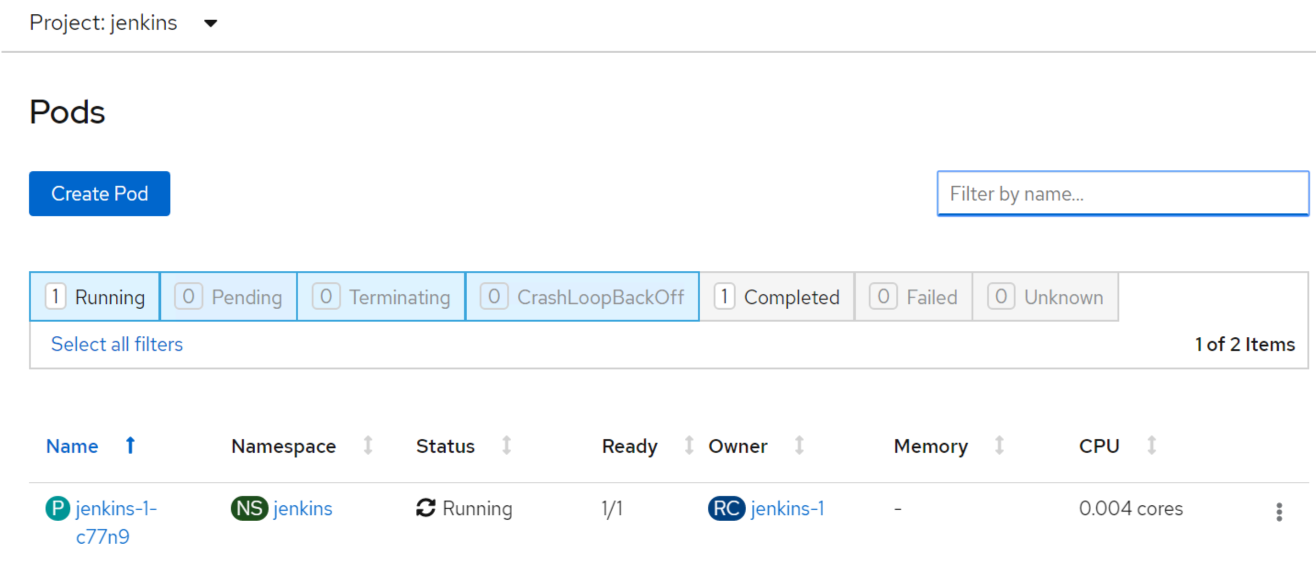


3. By default, the details for the Jenkins application are populated. Based on your requirements, modify the parameters, and click Create. This process creates all the required resources for supporting Jenkins on

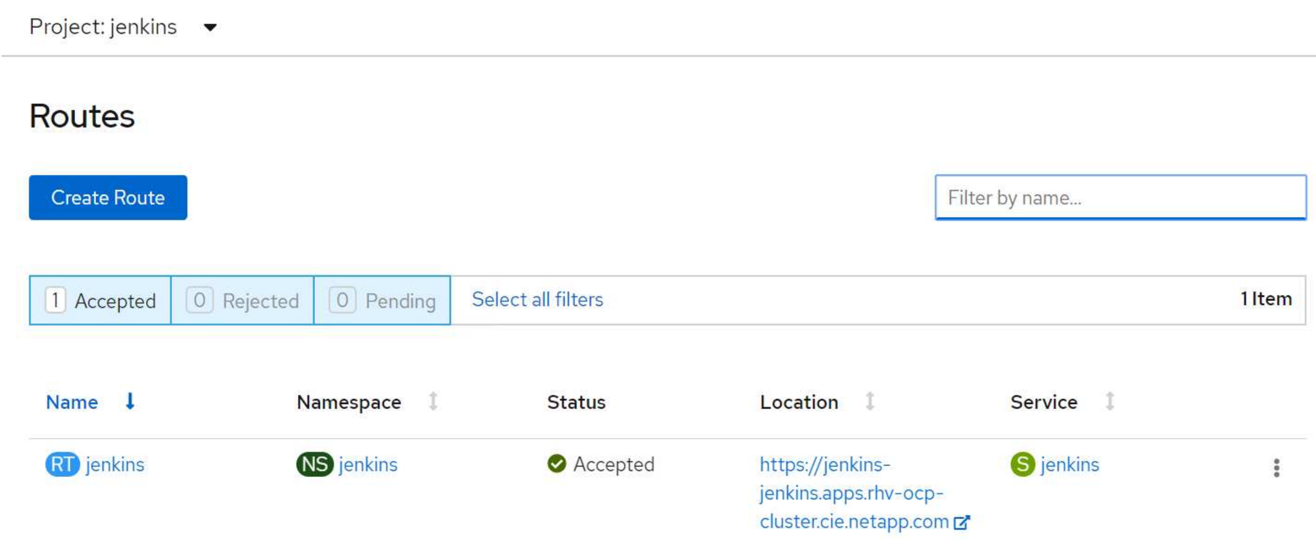
OpenShift.



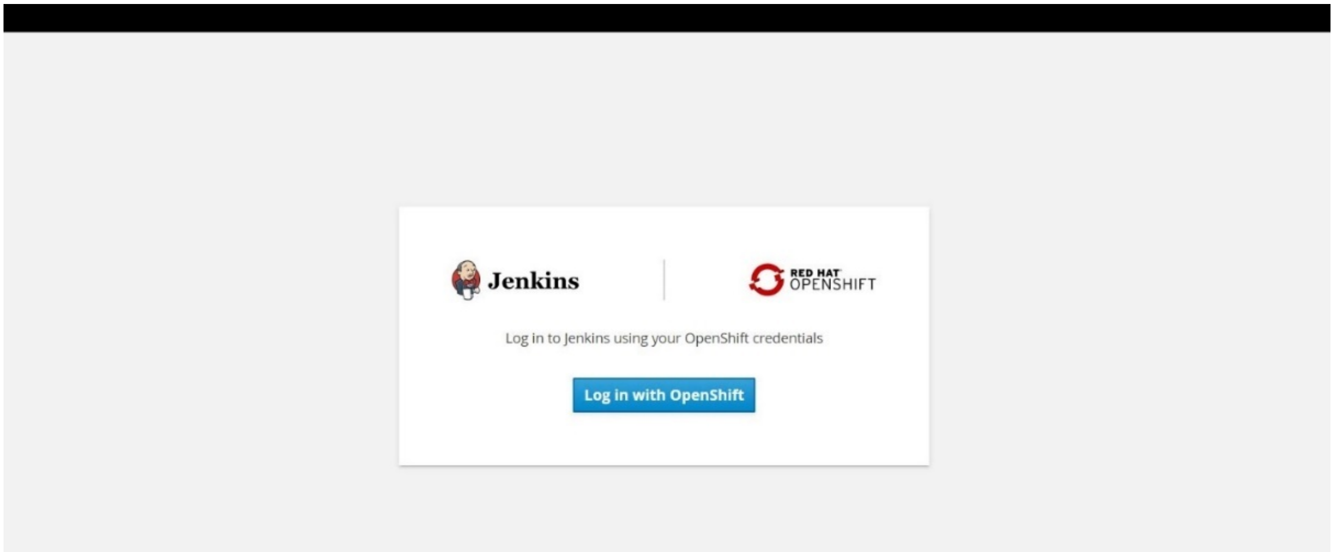
4. The Jenkins pods take approximately 10–12 minutes to enter the Ready state.



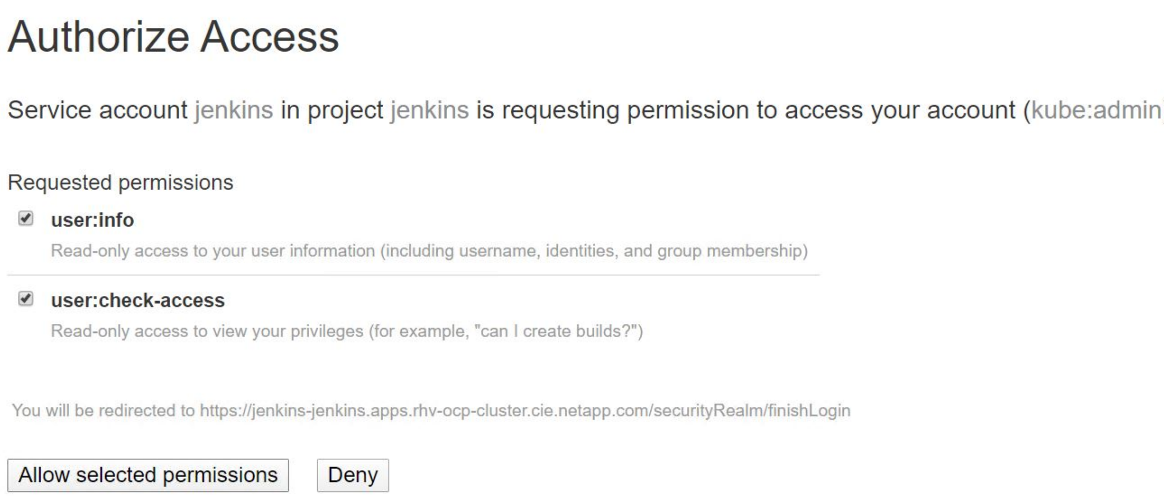
5. After the pods are instantiated, navigate to Networking > Routes. To open the Jenkins webpage, click the URL provided for the jenkins route.



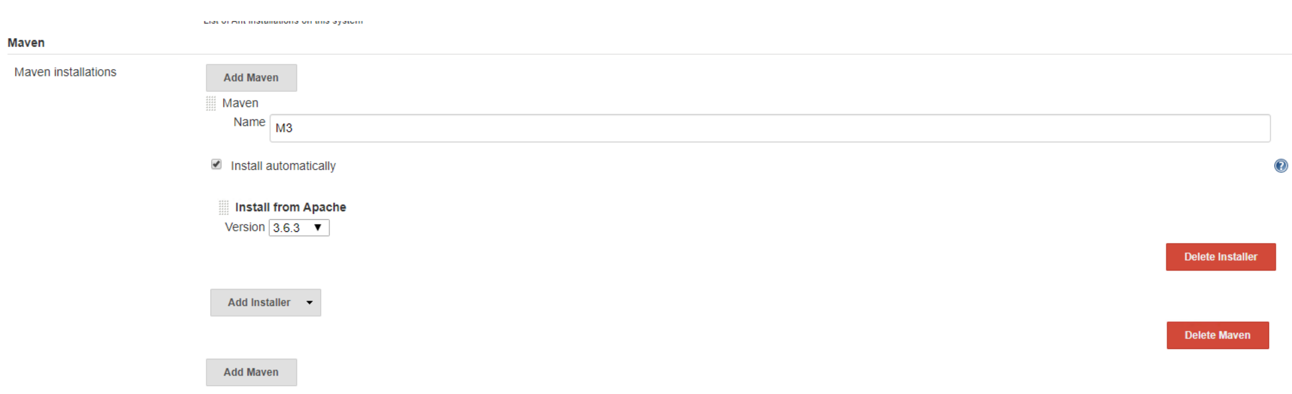
6. Because the OpenShift OAuth was used while creating the Jenkins app, click Log in with OpenShift.



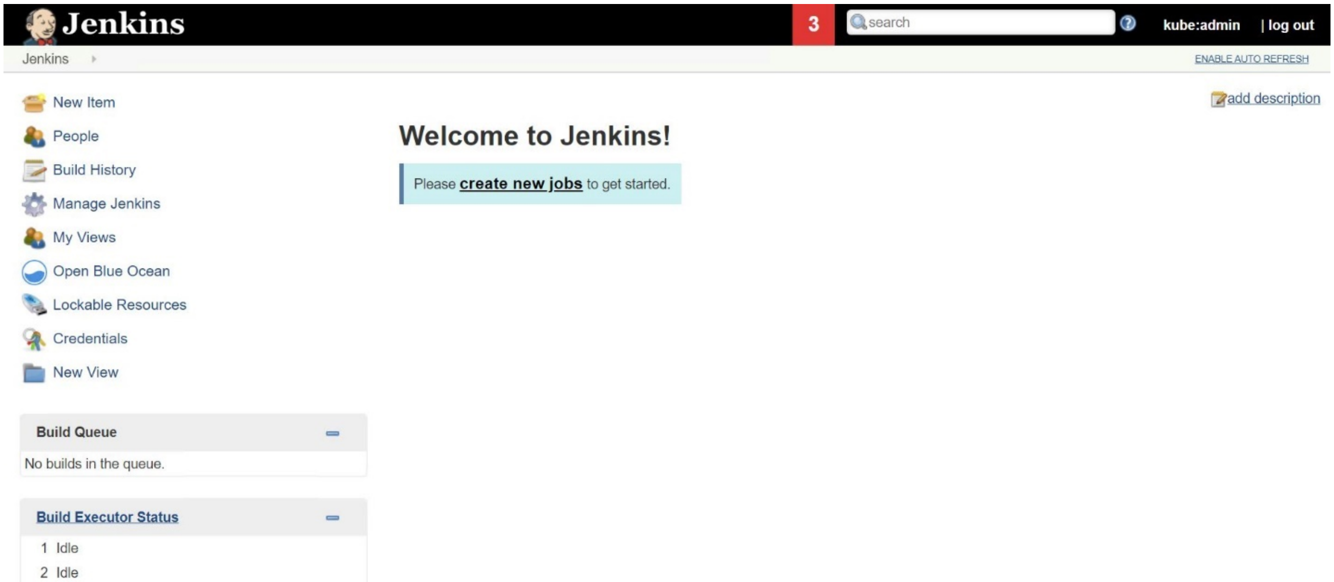
7. Authorize jenkins service-account to access the OpenShift users.



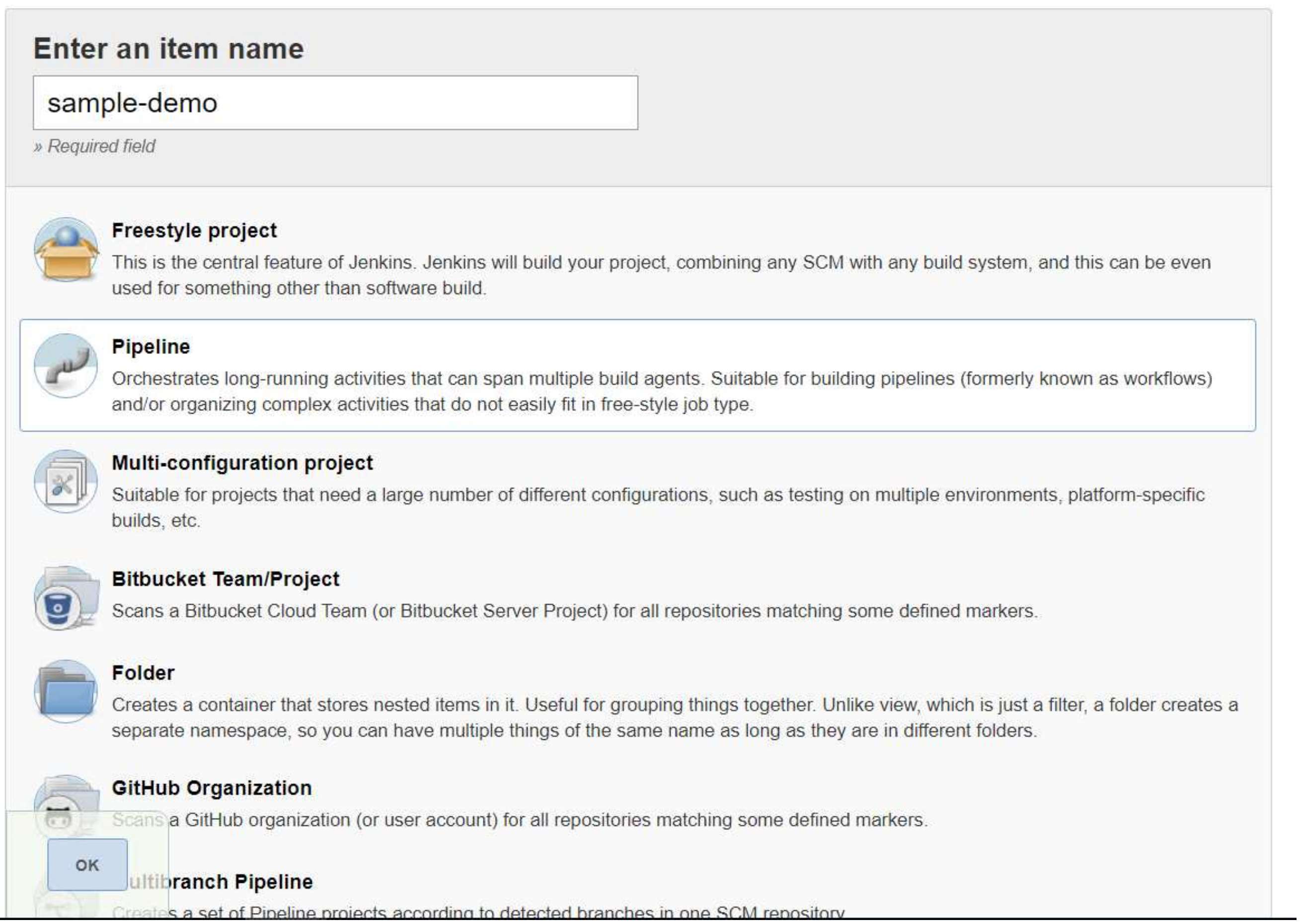
8. The Jenkins welcome page is displayed. Because we are using a Maven build, complete the Maven installation first. Navigate to Manage Jenkins > Global Tool Configuration, then in the Maven subhead, click Add Maven. Enter the name of your choice and make sure that the Install Automatically option is selected. Click Save.



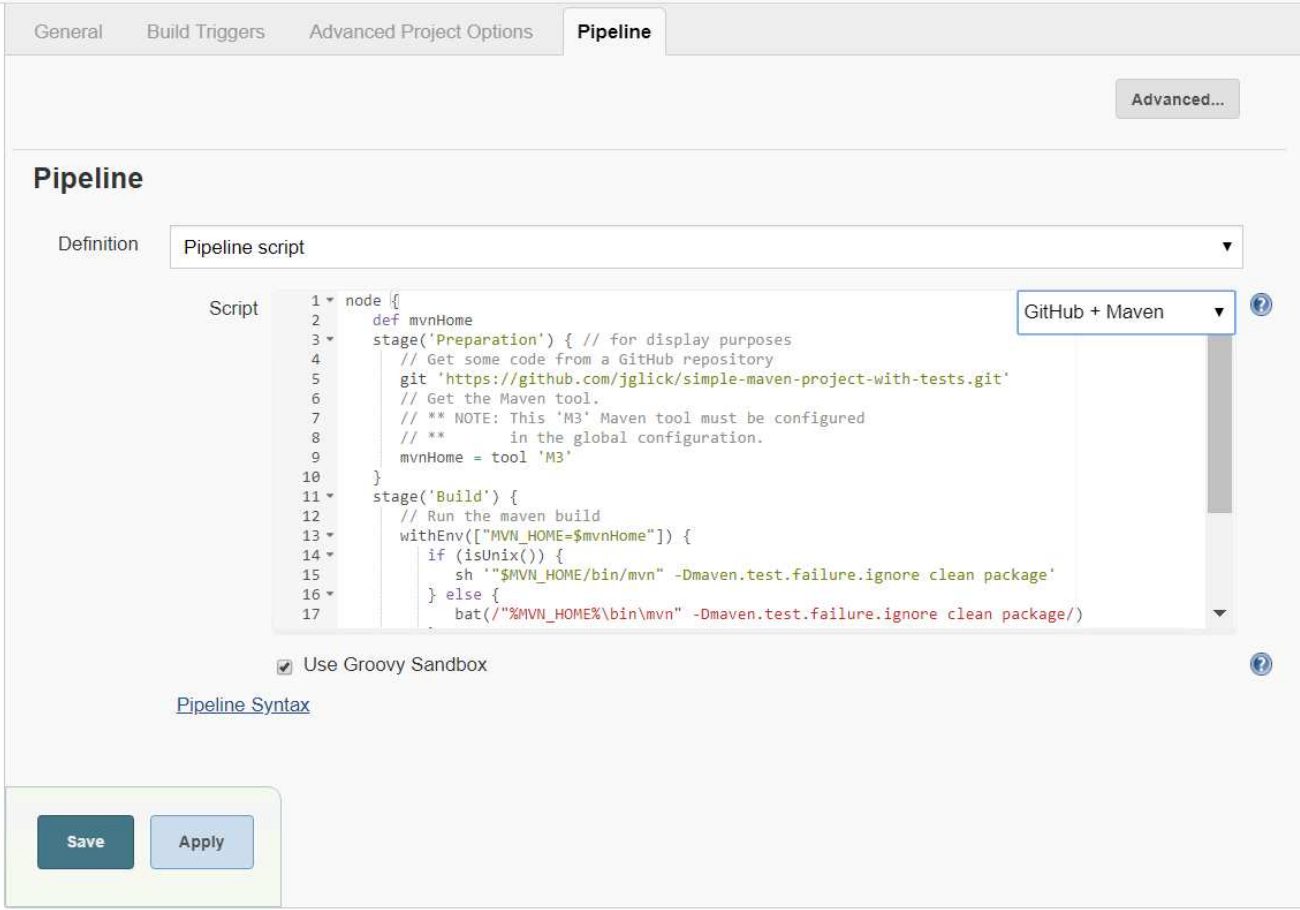
9. You can now create a pipeline to demonstrate the CI/CD workflow. On the home page, click Create New Jobs or New Item from the left-hand menu.



10. On the Create Item page, enter the name of your choice, select Pipeline, and click Ok.

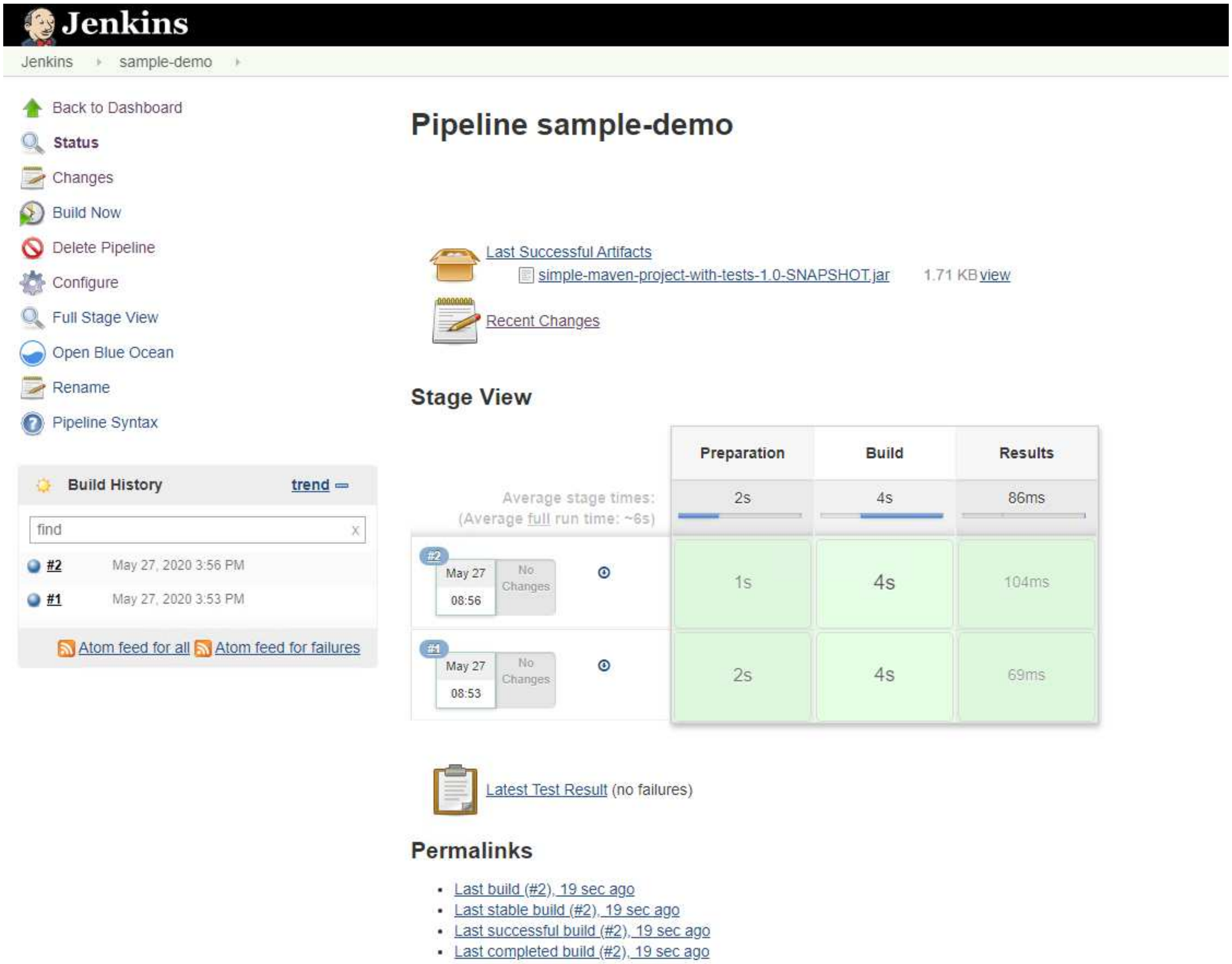


11. Select the Pipeline tab. From the Try Sample Pipeline drop- down menu, select Github + Maven. The code is automatically populated. Click Save.



12. Click Build Now to trigger the development through the preparation, build, and testing phase. It can take several minutes to complete the whole build process and display the results of the build.

13. Whenever there are any code changes, the pipeline can be rebuilt to patch the new version of software enabling continuous integration and continuous delivery. Click Recent Changes to track the changes from the previous version.



Best Practices for Production Deployments: NetApp HCI for Red Hat OpenShift on RHV

This section lists several best practices that an organization should take into consideration before deploying this solution into production.

Deploy OpenShift to an RHV Cluster of at Least Three Nodes

The verified architecture described in this document presents the minimum hardware deployment suitable for HA operations by deploying two RHV-H hypervisor nodes and ensuring a fault tolerant configuration where both hosts can manage the hosted-engine and deployed VMs can migrate between the two hypervisors. Because Red Hat OpenShift initially deploys with three master nodes, it is ensured in a two-node configuration that at least two masters will occupy the same node, which can lead to a possible outage for OpenShift if that specific node becomes unavailable. Therefore, it is a Red Hat best practice that at least three RHV-H hypervisor nodes be deployed as part of the solution so that the OpenShift masters can be distributed evenly, and the solution receives an added degree of fault tolerance.

Configure Virtual Machine/Host Affinity

Ensuring the distribution of the OpenShift masters across multiple hypervisor nodes can be achieved by enabling VM/host affinity. Affinity is a way to define rules for a set of VMs and/or hosts that determine whether the VMs run together on the same host or hosts in the group or on different hosts. It is applied to VMs by creating affinity groups that consist of VMs and/or hosts with a set of identical parameters and conditions. Depending on whether the VMs in an affinity group run on the same host or hosts in the group or separately on different hosts, the parameters of the affinity group can define either positive affinity or negative affinity. The conditions defined for the parameters can be either hard enforcement or soft enforcement. Hard enforcement ensures that the VMs in an affinity group always follows the positive/negative affinity strictly without any regards to external conditions. Soft enforcement, on the other hand, ensures that a higher preference is set out for the VMs in an affinity group to follow the positive/negative affinity whenever feasible. In a two or three hypervisor configuration as described in this document soft affinity is the recommended setting, in larger clusters hard affinity can be relied on to ensure OpenShift nodes are distributed. To configure affinity groups, see the Red Hat 6.11. Affinity Groups documentation.

Use a Custom Install File for OpenShift Deployment

IPI makes the deployment of OpenShift clusters extremely easy through the interactive wizard discussed earlier in this document. However, it is possible that there are some default values that might need to be changed as a part of a cluster deployment. In these instances, the wizard can be run and tasked without immediately deploying a cluster, but instead outputting a configuration file from which the cluster can be deployed later. This is very useful if any IPI defaults need to be changed, or if a user wants to deploy multiple identical clusters in their environment for other uses such as multitenancy. For more information about creating a customized install configuration for OpenShift, see Red Hat OpenShift Installing a Cluster on RHV with Customizations.

Videos and Demos: NetApp HCI for Red Hat OpenShift on RHV

The following video demonstrates some of the capabilities documented in this document:

[] | *NetApp HCI for Red Hat OpenShift on Red Hat Virtualization*

Additional Information: NetApp HCI for Red Hat OpenShift on RHV

To learn more about the information described in this document, review the following websites:

- NetApp HCI Documentation https://www.netapp.com/us/documentation/hci.aspx
- NetApp Trident Documentation https://netapp-trident.readthedocs.io/en/stable-v20.04/
- Red Hat Virtualization Documentation https://access.redhat.com/documentation/en-us/red_hat_virtualization/4.3/
- Red Hat OpenShift Documentation https://access.redhat.com/documentation/en-us/openshift_container_platform/4.4/