NetApp Verified Architecture

Red Hat OpenShift Container Platform with NetApp HCI

NVA Deployment

Amit Borulkar, NetApp
November 2018 | NVA-1124-DEPLOY | Version 1.0

Abstract

This document provides specific deployment and configuration instructions for deploying Red Hat OpenShift Container Platform on NetApp® HCI at an enterprise scale in a risk-free and reliable manner.

In partnership with

redhat

NetApp®

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 Solution Summary

Red Hat OpenShift Container Platform with NetApp HCI is a prevalidated, best-practice data center architecture for deploying microservices workloads at an enterprise scale. This architecture is codesigned and coengineered by subject matter experts from NetApp and Red Hat to provide the advantages of open-source innovation with enterprise robustness.

NetApp HCI provides the widely recognized benefits of hyper converged solutions such as lower TCO, ease of purchasing, deployment, growth, and management for virtualized workloads. However, NetApp HCI is different in that it enables IT to scale storage and compute separately. NetApp HCI is based on the VMware platform, which is the enterprise standard for private cloud management. Red Hat OpenShift Container Platform is a comprehensive enterprise-grade platform as a service (PaaS), based on Kubernetes. Red Hat OpenShift Container Platform is frequently deployed within a VMware virtualized infrastructure to take advantage of the native high availability of VMware and the dynamic resizing of VM resources, leading to improved security, efficiency, and agility.

Together, these three technology solutions offer ease of procurement, deployment, and ongoing management and growth of your business-critical hardware, virtual resources, and enterprise applications. This architecture uses a modern container-based application deployment model while leveraging the common tools of the modern data center to preserve investments.

This document includes specific installation and configuration details for the implementation of this solution as validated. To determine the appropriate configuration for your organization, refer to the Red Hat OpenShift Container Platform with NetApp HCI Design guide.

2 Solution Overview

2.1 Business Challenges

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. For example:

- High availability (HA) at all layers in the stack
- 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

2.2 Red Hat OpenShift Container Platform with NetApp HCI

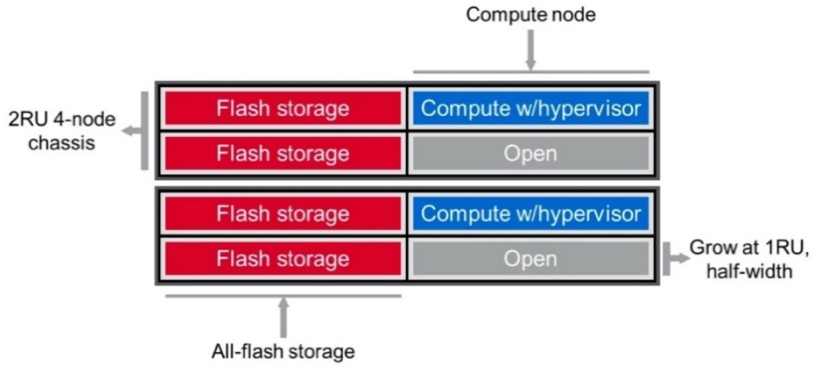
To meet these business challenges, NetApp and Red Hat have coengineered, codesigned, and validated a reference architecture that allows you to develop and deploy an application in a reliable and risk-free manner. Red Hat OpenShift Container Platform with NetApp HCI enables you to run containerized applications by using enterprise-grade Kubernetes for container orchestration with performance guarantees. Furthermore, the scale-out design of the solution allows you to start small and eventually scale out compute and storage resources independently of one another to meet the requirements of your workload. With enterprise-class support, security patches, and bug fixes at all layers in the stack, you can now focus on your application development, resulting in better business outcomes.

2.3 Solution Technology

NetApp HCI is an enterprise-scale solution that delivers compute and storage resources in an agile, scalable, easy-to-manage two-rack unit, four-node building block. It is based on the following configuration:

- NetApp H-Series all-flash storage nodes based on NetApp Element® Software
- NetApp H-Series compute nodes
- NetApp Deployment Engine (NDE) and NetApp Element vSphere plug-in, which enable deployment and management of NetApp HCI

Figure 1) NetApp HCI minimum configuration.



For full details and technical specifications of compute and storage nodes in NetApp HCI, refer to the NetApp HCI datasheet,

NetApp HCI Design Principles

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, and a simple deployment and management experience.

Predictable

One of the biggest challenges in a multitenant environment is delivering predictable performance consistently for all your workloads. Running multiple enterprise-class workloads can result in resource contention, where one workload might interfere with the performance of another. NetApp HCI alleviates this concern with QoS limits that are available natively with NetApp Element software. NetApp Element software allows the granular control of every application and volume, eliminates noisy neighbors, and satisfies all performance SLAs. NetApp HCI multitenancy capabilities can help eliminate more the majority of traditional performance-related problems.

Flexible

Earlier versions of HCI required fixed resource ratios, thereby limiting deployments to four- to eight-node configurations. In contrast, NetApp HCI scales 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.

With NetApp HCI, licensing costs are reduced. NetApp HCI is available in mix-and-match small, medium, and large storage and compute configurations. The architectural design options allow customers to confidently scale on their terms, making HCI viable for core data center applications and platforms.

NetApp HCI is architected in building blocks at either the chassis or the node level. Each chassis holds four nodes, which can be made up of storage nodes, compute nodes, or both. A minimum configuration is two chassis with six nodes, consisting of four storage nodes and two compute nodes. Two additional empty slots can be used for expansion. Compute and storage nodes can be mixed as long as best practices are followed. Resources can be scaled nondisruptively through a simple GUI-driven process.

Simple

It is imperative to automate all routine IT tasks where possible. This eliminates the risk of user error and frees up resources to focus on more complex, higher-value projects. NetApp HCI helps IT departments to become more agile and responsive by simplifying deployment and ongoing management.

The new NDE eliminates most of the manual steps involved in deploying infrastructure, such as assigning names, network settings, and IP addresses, and provisioning ESXi hosts and VMware datastores. You can expect the infrastructure to be functional in less than 30 minutes.

The VMware vCenter plug-in is intuitive and simplifies management of the solution. Additionally, a robust suite of NetApp HCI APIs enables integration into higher-level management, orchestration, backup, and disaster recovery tools.

NetApp Deployment Engine

NDE enables the quick deployment of NetApp HCI, including the NetApp Element software cluster and the VMware virtualized infrastructure. NDE simplifies day zero deployment by reducing the number of manual steps from more than 300 to fewer than 30. Because NDE is intuitive and reuses data such as user name and password, you are not required to reenter information or set credentials at varying complexity levels. Likewise, NDE handles the assignment of IP addresses, allowing you to set a scheme and pool for all resources before actual configuration. Also, preinstallation checklists facilitate successful deployments because the system automatically checks for user errors, eliminating the need for manual checks.

Figure 2) Successful NDE deployment.

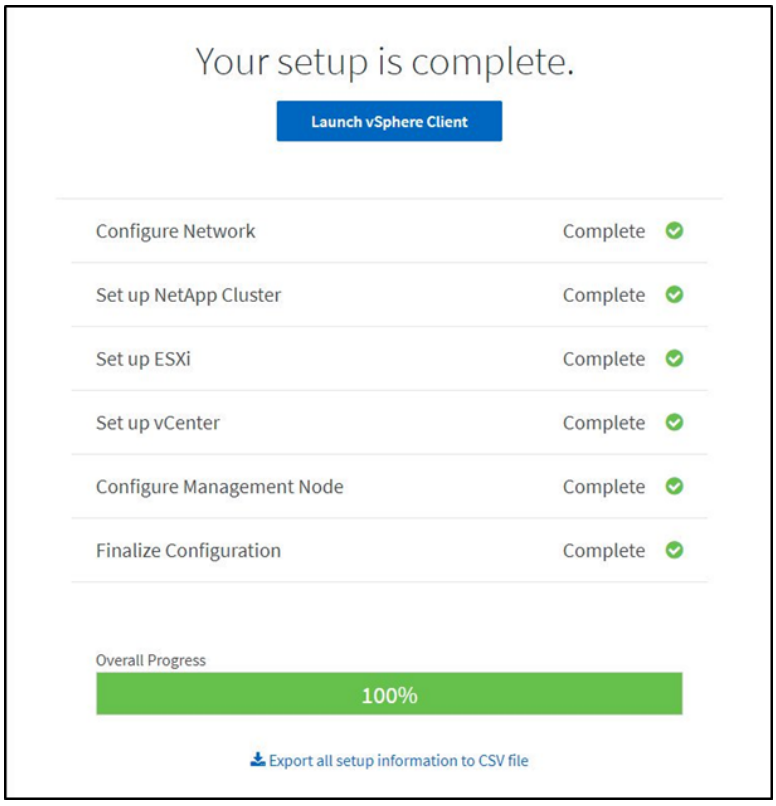


Figure 2 shows how NDE optimally configures the data and management networks, the NetApp cluster, VMware ESXi, vCenter, and other required configurations to bring your virtualized environment up-and-running in a risk-free manner. For more information about the NetApp Deployment Engine, see the NetApp HCI Deployment Engine User Guide. For more information about deploying NetApp HCI, see the NetApp HCI Deployment Guide.

NetApp Element Software Cluster QoS

A NetApp Element software cluster allows QoS to be dynamically configured on a per-volume basis. Per-volume QoS settings can be used to control storage performance based on customer-defined SLAs.

There are three configurable parameters that define QoS:

- **Min IOPS.** The minimum number of IOPS that are provided by the NetApp Element software cluster to a volume. The Min IOPS for a volume is the guaranteed level of performance for a volume. Per-volume performance does not drop below this level.
- **Max IOPS.** The maximum number of sustained IOPS that are provided by the NetApp Element software cluster to a particular volume.
- **Burst IOPS.** The maximum number of IOPS allowed in a short burst scenario. If a volume is running below the max IOPS, burst credits are accumulated. When performance levels become very high and are pushed to maximum levels, short bursts of IOPS beyond Max IOPS are allowed on the volume.

Enterprise Storage Efficiencies

The NetApp Element software cluster leverages key features to increase overall storage efficiency and performance. These features are performed inline, are enabled, and require no manual configuration by the user.

- **Deduplication.** The system stores only 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 the NetApp Element software Helix feature. This system significantly reduces capacity consumption and write operations in the system.
- **Compression.** Compression is performed inline before data is written to NVRAM. Data is compressed and stored in 4K blocks and, once compressed, it remains compressed in the system. This 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 it's needed, so that capacity is not consumed unnecessarily by overprovisioned or underutilized volumes.
- **SolidFire.** The metadata for an individual volume is stored on a metadata drive and replicated to a secondary metadata drive for redundancy.

Finally, Element was designed for automation. All storage features are available through APIs. These APIs are the only methods used by the UI to control the system. For more information, see the Element product page.

Project Trident

Trident enables microservices and containerized applications to leverage enterprise-class storage services (such as QoS, storage efficiencies, cloning, and so on) to meet their persistent storage demands. Depending on the requirements of the application, Trident can dynamically provision storage from NetApp ONTAP (AFF, FAS, ONTAP Select, ONTAP Cloud), Element (NetApp HCI, SolidFire®) and SANtricity® (E/EF-Series) data management software. Trident is a fully supported open-source project maintained by NetApp.

Trident makes use of the "storage class" object introduced in Kubernetes 1.4 to dynamically provision "persistent volume" when a "persistent volume claim" object is created. A "storage class" enables administrators to describe the classes of storage they offer. A "storage class" might specify different QoS levels, back-up policies, and so on. This reference architecture recommends creating a "storage class" that specifies Trident as the external provisioner and SolidFire SAN as the Trident storage backend. Refer to the Trident documentation for more information.

Red Hat OpenShift Container Platform

Red Hat OpenShift Container Platform unites developers and IT operations on a single platform to build, deploy, and manage applications consistently across on-premises and hybrid cloud infrastructures. Red Hat OpenShift is built on open source innovation and industry standards, including Kubernetes and Red Hat Enterprise Linux, the world's leading enterprise Linux distribution.

OpenShift is part of the Cloud Native Computing Foundation (CNCF) Certified Kubernetes program, offering portability and interoperability of your container workloads. OpenShift Container Platform provides the following capabilities:

- **Self-service provisioning.** Developers can quickly and easily create applications on demand from the tools they use most, while operations retain full control over the entire environment.
- **Persistent storage.** By providing support for persistent storage, OpenShift Container Platform allows users to run both stateful applications and cloud-native stateless applications.
- **Open-source standards.** Incorporates Open Container Initiative, compatible runtimes, and Kubernetes for container orchestration, in addition to other open-source technologies. You are not restricted to the technology or business roadmap of a specific vendor.
- **CI/CD pipelines.** OpenShift provides out-of-the-box support for continuous integration/continuous delivery pipelines so that development teams can automate every step of the application delivery process and verify that it was executed on every code or configuration change in the application.

- **Role based access control (RBAC).** Enables team and user tracking for coordinating a large developer organization.
- **Automated build and deploy.** OpenShift offers developers the choice to either build their containerized applications themselves or to have the platform build containers from the application source code or even binaries. The platform then automates deployment of these applications across the infrastructure based on predefined characteristics for allocation of resources and appropriate locations for the applications in accordance with third-party licenses.
- **Consistent environments.** OpenShift verifies that the environment provisioned for developers and across the lifecycle of the application is consistent, from the operating system to libraries to runtime version (for example, Java runtime), and even the application runtime in use (for example, Apache Tomcat). This mitigates the risks associated with inconsistent environments.
- **Configuration management.** Configuration and sensitive data management are built into the platform to offer consistent, environment agnostic configuration to the application regardless of the technologies used to build it and the environment in which it is deployed.
- **Application logs and metrics.** Rapid feedback is an important aspect of application development. OpenShift integrated monitoring and log management provide immediate metrics back to developers, allowing them to study how the application is behaving across changes and to fix issues as early as possible in the application lifecycle.
- **Security and container catalog.** OpenShift offers multitenancy and protects the user from harmful code execution by using established security with Security-Enhanced Linux (SELinux), CGroups, and Secure Computing Mode to isolate and protect containers. TLS certificates provide encryption for the various subsystems, and OpenShift provides access to Red Hat certified containers (https://access.redhat.com/containers/) that are scanned and graded with a specific emphasis on security to provide certified, trusted, and secure application containers to end users.

3 Technology Requirements

This section covers the technology requirements for Red Hat OpenShift Container Platform with the NetApp HCI validated solution. All the models of compute and storage listed on the NetApp Interoperability Matrix Tool (IMT) are supported.

Note: For more information about the technical requirements and for installation guidance on NetApp HCI, review the NetApp HCI Resources page.

3.1 Hardware Requirements

Table 1 lists the hardware components that were used to deploy the solution in a NetApp lab. The hardware components that are used in any particular implementation of the solution might vary, based on your organization's requirements.

Table 1) Hardware requirements.

Layer	Product Family	Quantity	Details
Compute	NetApp H500E	4	2 X Intel E5-2650v4 12 cores 2.2GHz 512GB RAM
Storage	NetApp H500S	4	6 x 960GB encrypting/nonencrypting

3.2 Software Requirements

Table 2 lists the software components that were required to implement the solution in a NetApp lab. The software components that are used in any particular implementation of the solution might vary, based on your requirements. All the components mentioned in IMT are supported.

Table 2) Software requirements.

Layer	Software	Version
Storage	NetApp Element OS	10.4
	NetApp Trident	18.10
NetApp HCI engine	NetApp Deployment Engine	1.4
Hypervisor and above	Hypervisor	VMware vSphere ESXi 6.5 U2
	Hypervisor Management System	VMware vCenter Server 6.5
	Red Hat Enterprise Linux	7.5
	Red Hat OpenShift Container Platform	3.10

Note: NetApp HCI is switch vendor agnostic and relies on standard enterprise-class data center switching features. The network design is described later in section 4, “Networking Topology.”

3.3 License Requirements

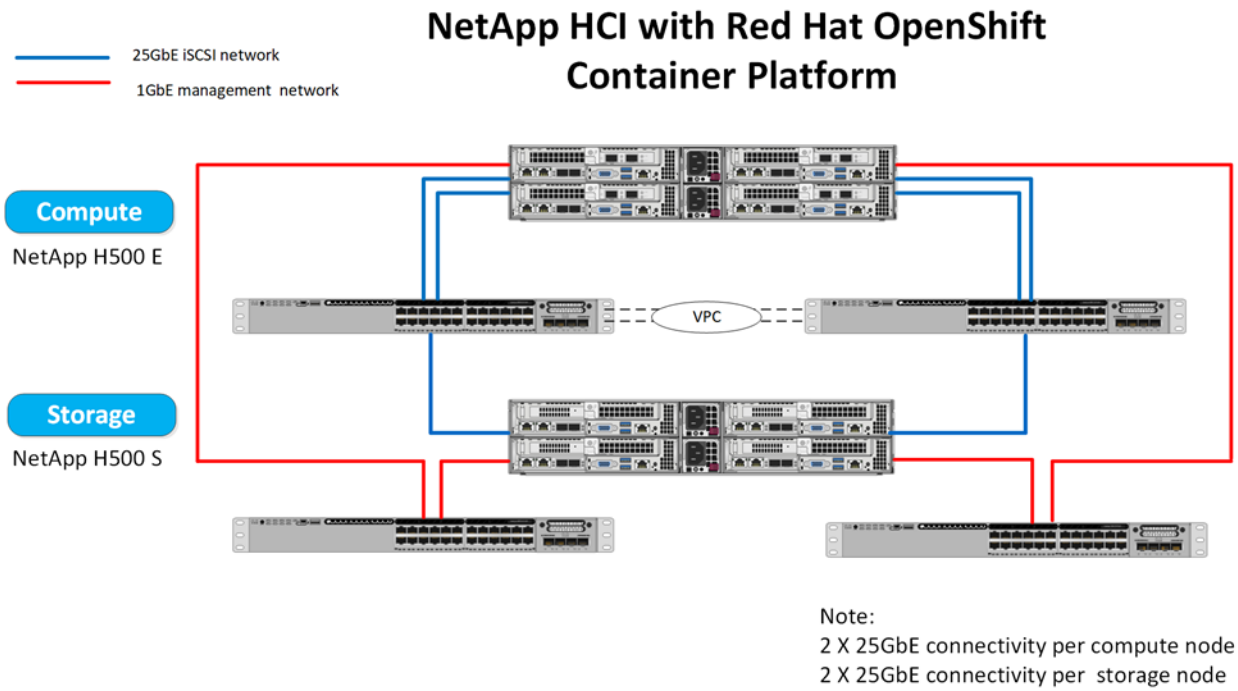
The following components in the solution have license or subscription requirements:

- VMware vSphere
- VMware vCenter Server Enterprise Plus
- Red Hat OpenShift Container Platform subscription

4 Networking Topology

This reference architecture enables end-to-end 25GbE iSCSI network, as shown in Figure 3.

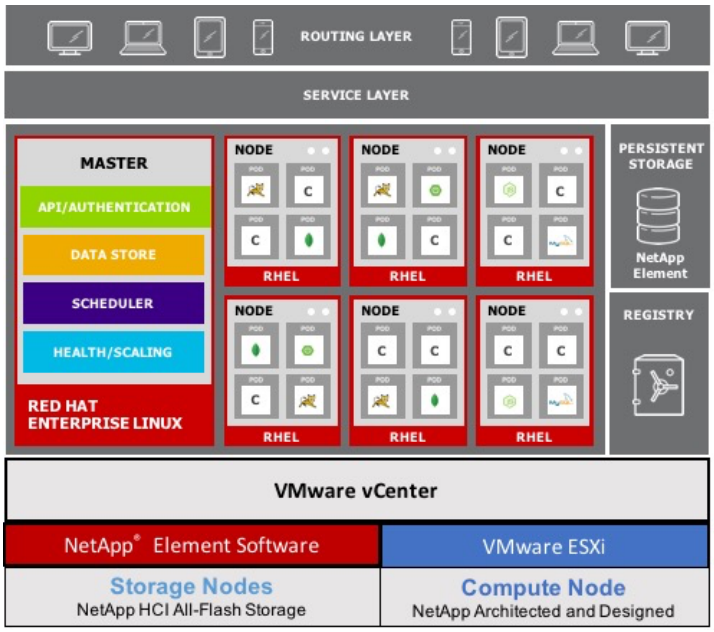
Figure 3) Network architecture.



NDE contains the following network configuration out of the box:

- Two 25GbE storage ports on the NetApp H500S nodes are configured in LACP mode, providing higher throughput and resiliency.
- Two 25GbE compute ports on the NetApp H500E nodes are configured with VMware iSCSI port-binding, providing higher throughput and resiliency.
- NDE automatically configures required port groups for all management functions, iSCSI storage access, and vMotion.
- An additional port-group with `OpenShift-Storage` maps iSCSI storage directly from RHEL nodes; this port group uses the “Route based on physical nic load” load-balancing policy.
- Jumbo frames are configured end to end.

Figure 4) Layered architecture



NetApp HCI leverages VMware ESXi as the hypervisor and VMware vCenter server for managing virtual machines, ESXi hosts, and NetApp Element Software in a centralized manner. All the OpenShift nodes are deployed as Red Hat Enterprise Linux 7.5 virtual machines on NetApp HCI.

4.1 Necessary VLANs

The different types of network traffic in Red Hat OpenShift Container Platform with NetApp HCI are isolated using the VLANs described in Table 3.

Table 3) Necessary VLANS.

Variables	Description	Value used in NetApp HCI RTP lab
`native_vlan`	Native VLAN for all the switch ports	2
`management_vlan_id`	Management network for VMware ESXi hosts and NetApp Element cluster	3287
`vmotion_vlan_id`	Network for VMware virtual machines vMotion	102
`storage_vlan_id`	The primary iSCSI storage network for NetApp Element cluster	100
`openshift_storage_vlan_id`	Network that carries the storage traffic for OpenShift tenants and other OpenShift services	200
`openshift_network`	The underlay network for OpenShift traffic	3291
`openshift_external_network`	A routable network for OpenShift egress access	3290

5 Deployment Procedures

This section contains specific configuration details and instructions for deploying Red Hat OpenShift Container Platform on NetApp HCI.

Implementing the solution involves the following actions:

- Deploying NetApp HCI
- Deploying and configuring Red Hat Enterprise Linux virtual machines for OpenShift
- Deploying Red Hat OpenShift Container Platform
- Deploying Trident
- Deploying other OpenShift components such as OpenShift Container Registry, OpenShift Logging, and OpenShift metrics

This document also demonstrates the common solution operations, such as:

- Ensuring storage quotas and limits in a multitenant environment
- Persistent volume cloning operations
- OpenShift registry operations
- Sample application deployment

5.1 NetApp HCI Deployment Using NDE 1.4

The NetApp Deployment Engine (NDE) deploys the storage and compute components in NetApp HCI and configures them in an automated manner. Consult the NetApp HCI Prerequisites Checklist to review the requirements and recommendations for NetApp HCI before you begin deployment.

For information about racking your NetApp HCI system, refer to the NetApp HCI Rail Kit Installation Flyer.

Comprehensive instructions for deploying NetApp HCI using NDE can be found in the NetApp HCI Deployment Guide.

This reference architecture uses a 2-cable configuration for NetApp HCI compute nodes. The management, vMotion, iSCSI traffic, and other VM traffics are carried over the two 25GbE ports.

Prerequisites

Switch Configuration

NetApp HCI has specific physical and network requirements for an enterprise-grade data center deployment. A prerequisite to running NDE is that the switches must be deployed and have corresponding relevant configuration for the compute and storage ports.

This solution uses Cisco Nexus OS based switches and provides relevant configuration for compute and storage ports on NetApp HCI. Complete switch configuration is beyond the scope of this document.

Note: This guide assumes that the switches have been racked and cabled, and have initial configuration such as NTP, default gateway, management IP, port descriptions, and other relevant global configuration.

Switch features such as `lacp` and `vpc` should be enabled on the switches.

Table 4 shows the variables used in the switch configuration.

Table 4) Switch configuration.

Variable	Description	Value used in NetApp HCI lab
`switch-A-mgmt0-ip`	Management IP address for switch A	172.16.15.15
`switch-B-mgmt0-ip`	Management IP address for switch B	172.16.15.14
`vpc-domain-id`	The vPC domain id	10
`controlling_storage_mgmt_ip`	IP address to access NDE	172.21.31.151

Configuring Switches

To configure switches for compute and storage ports, follow these steps.

1. From the global configuration mode, create VLANs on switch A and switch B.

```
vlan <<native_vlan>>
name native
vlan << management_vlan_id>>
name IB-Management
vlan <<vmotion_vlan_id>>
name vmotion
vlan <<storage_vlan_id>>
name storage
vlan <<openshift_network>>
name OpenShift-network
vlan <<openshift_external_network>>
name openshift-external
vlan <<openshift_storage_vlan_id>>
name OpenShift-storage
exit
```

2. Configure virtual port channels (vPCs) for switch A.

```
vpc domain <vpc-domain-id>
role priority 10
peer-keepalive destination <switch-B-mgmt0-ip> source <switch-A-mgmt0-ip>
peer-switch
peer-gateway
auto-recovery
delay restore 150
```

3. Configure virtual port channels (vPCs) for switch B.

```
vpc domain <vpc-domain-id>
role priority 20
peer-keepalive destination <switch-A-mgmt0-ip> source <switch-B-mgmt0-ip>
peer-switch
peer-gateway
auto-recovery
delay restore 150
```

4. Configure port-channel on switches A and B for the ports connecting to NetApp HCI storage nodes.

```
interface Ethernet1/45, Ethernet1/46
channel-group 10 mode active
no shutdown

interface po10
  description vPC Peer-Link
  switchport mode trunk
  switchport trunk native vlan 2
  switchport trunk allowed vlan 2,100,200
  spanning-tree port type network
```

```
  vpc peer-link
  exit
```

```
interface po117
  description HCI_Storage01_25G
  switchport mode trunk
  switchport trunk native vlan 2
  switchport trunk allowed vlan 2,100,200
  spanning-tree port type edge trunk
  mtu 9216
  vpc 117
  exit
interface Eth1/17
channel-group 117 mode active
no shutdown
exit


interface po118
  description HCI_Storage02_25G
  switchport mode trunk
  switchport trunk native vlan 2
  switchport trunk allowed vlan 2,100,200
  spanning-tree port type edge trunk
  mtu 9216
  vpc 118
  exit
interface Eth1/18
channel-group 118 mode active
no shutdown
exit

interface po119
  description HCI_Storage03_25G
  switchport mode trunk
  switchport trunk native vlan 2
  switchport trunk allowed vlan 2,100,200
  spanning-tree port type edge trunk
  mtu 9216
  vpc 119
  exit
interface Eth1/19
channel-group 119 mode active
no shutdown
exit

interface po120
  description HCI_Storage04_25G
  switchport mode trunk
  switchport trunk native vlan 2
  switchport trunk allowed vlan 2,100,200
  spanning-tree port type edge trunk
  mtu 9216
  vpc 120
  exit
interface Eth1/20
channel-group 120 mode active
no shutdown
exit
```

5. Configure the ports on switches A and B that connect to NetApp HCI compute nodes.

```
interface Ethernet1/1
  description HCI_Compute01_25G
  switchport mode trunk
  switchport trunk native vlan 2
  switchport trunk allowed vlan 2,100-102,200,3287,3291
  spanning-tree port type edge trunk
```

```
  mtu 9216

interface Ethernet1/2
  description HCI_Compute02_25G
  switchport mode trunk
  switchport trunk native vlan 2
  switchport trunk allowed vlan 2,100-102,200,3287,3291
  spanning-tree port type edge trunk
  mtu 9216

interface Ethernet1/3
  description HCI_Compute03_25G
  switchport mode trunk
  switchport trunk native vlan 2
  switchport trunk allowed vlan 2,100-102,200,3287,3291
  spanning-tree port type edge trunk
  mtu 9216

interface Ethernet1/4
  description HCI_Compute04_25G
  switchport mode trunk
  switchport trunk native vlan 2
  switchport trunk allowed vlan 2,100-102,200,3287,3291
  spanning-tree port type edge trunk
  mtu 9216
```

Note: The 1G management ports on NetApp H500S nodes are connected to the management switch, and the in-band management VLAN is tagged on the switch ports.

DNS

OpenShift Container Platform requires a fully functional DNS server in the environment. This is ideally a separate host, or a virtual machine running the DNS service that can provide name resolution to hosts and containers running on the platform. It is also used to resolve the hostnames for other NetApp HCI components such as FQDN of VMware vCenter Server instance, and ESXi hosts.

Note: You can also choose an existing DNS server in your environment.

NTP

For a smooth deployment, it is critical to verify accurate time synchronization among different components in the solution.

IP address

NDE enables assigning blocks of IP addresses to different components in NetApp HCI. Table 5 shows the number of IP addresses required.

Table 5) IP address requirements.

System component	Management network IP address	Storage network IP address	vMotion network IP address	IP address needed per component
Compute node	1	2	1	4
Storage node	1	1		2
Storage cluster	1	1		2
VMware vCenter	1			1
Management node	1			1

Note: The IP addresses do not have to be routable.

Configure LACP on Storage Nodes

Configure the Bond10G interfaces on the NetApp HCI storage nodes to operate in an LACP mode. The storage nodes can be accessed through either KVM or remote management.

NDE Access

To access NDE, assign `controlling_storage_mgmt_ip` IP address and add the corresponding VLAN tag to the Bond1G interface on one of the storage nodes. This can be done manually or through DHCP. If DHCP isn't configured in the environment, the storage node can be accessed by using KVM or through a remote console.

Note: This IP address is temporary and cannot be reused to access the management interface of the storage node. The storage node is eventually configured with a management IP address that is specified during deployment.

Final Preparations

1. Gather all relevant information about your network, any current or planned VMware infrastructure, and planned user credentials.
2. Rack, cable, and power on the NetApp HCI installation.
3. Use `show vpc` on the switches to verify that all the port channels are up.

Launching NetApp Deployment Engine to Configure NetApp HCI

Begin the launch by accessing `http://controlling_storage_mgmt_ip:442/nde`, then follow these steps:

1. Accept the end-user licenses.
2. Select "This solution doesn't require Data Fabric services."
3. Configure the vCenter server FQDN and authentication credentials for the cluster.
4. Select the 2-cable network topology. NDE autodiscovers the storage and compute nodes.
5. Configure the appropriate network node settings.

vCenter Networking

VLAN ID	Subnet	Default Gateway	FQDN	IP Address
3287	172.21.31.0/24	172.21.31.51	vcenter-hci.hci.rtp.com	172.21.31.91

Management Node Networking

VLAN ID	Subnet	Default Gateway	Hostname	IP Address
3287	172.21.31.0/24	172.21.31.51	OpenShift-NetAppHCI-mnod	172.21.31.171

Compute Node Networking

Management Network	vMotion Network	iSCSI A Network	iSCSI B Network
VLAN ID: 3287	VLAN ID: 102	VLAN ID: 100	VLAN ID: 100
Subnet: 172.21.31.0/24	Subnet: 192.168.102.0/24	Subnet: 192.168.100.0/24	Subnet: 192.168.100.0/24
Default Gateway: 172.21.31.51	Default Gateway (Optional)	Default Gateway (Optional)	Default Gateway (Optional)

Serial Number	Hostname	Management IP Address	vMotion IP Address	iSCSI A - IP Address	iSCSI B - IP Address
221818004414	OpenShift-NetAppHCI-	172.21.31.101	192.168.102.101	192.168.100.101	192.168.100.102
221820005010	OpenShift-NetAppHCI-	172.21.31.102	192.168.102.102	192.168.100.103	192.168.100.104
221820005019	OpenShift-NetAppHCI-	172.21.31.103	192.168.102.103	192.168.100.105	192.168.100.106
221820005023	OpenShift-NetAppHCI-	172.21.31.104	192.168.102.104	192.168.100.107	192.168.100.108

6. Configure the appropriate storage node settings.

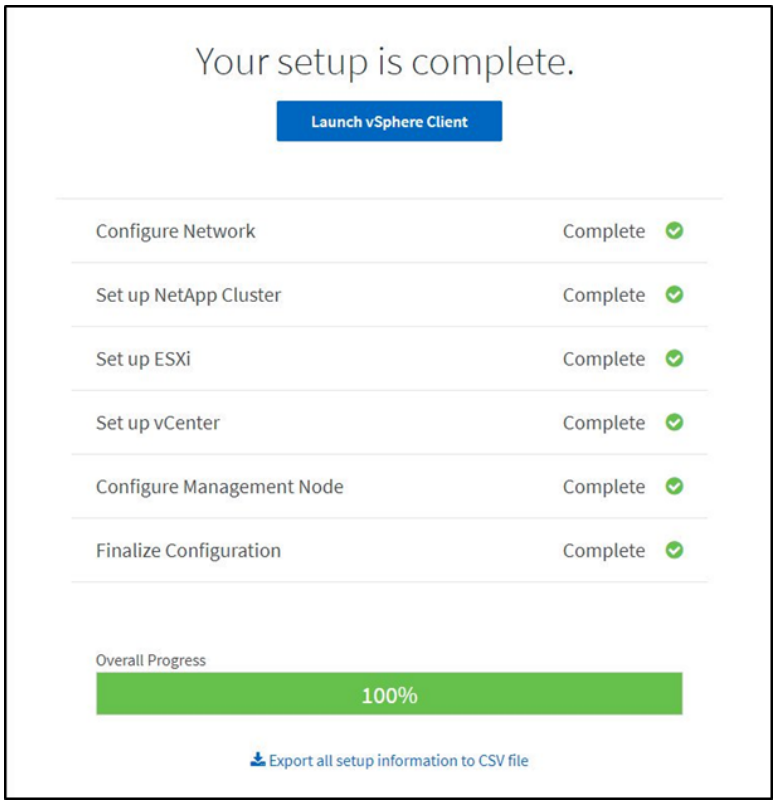
Storage Node Networking

Storage Cluster Name: OpenShift-NetAppHCI-cluster

Management Network	iSCSI Network
VLAN ID: 3287	VLAN ID: 100
Subnet: 172.21.31.0/24	Subnet: 192.168.100.0/24
Default Gateway: 172.21.31.51	Default Gateway (Optional)
Management Virtual IP (MVIP): 172.21.31.181	Storage Virtual IP (SVIP): 192.168.100.109

Serial Number	Hostname	Management IP Address	Storage (iSCSI) IP Address
221811002872	OpenShift-NetAppHCI-stg-01	172.21.31.182	192.168.100.110
221814003426	OpenShift-NetAppHCI-stg-02	172.21.31.183	192.168.100.111
221814003502	OpenShift-NetAppHCI-stg-03	172.21.31.184	192.168.100.112
221818004571	OpenShift-NetAppHCI-stg-04	172.21.31.185	192.168.100.113

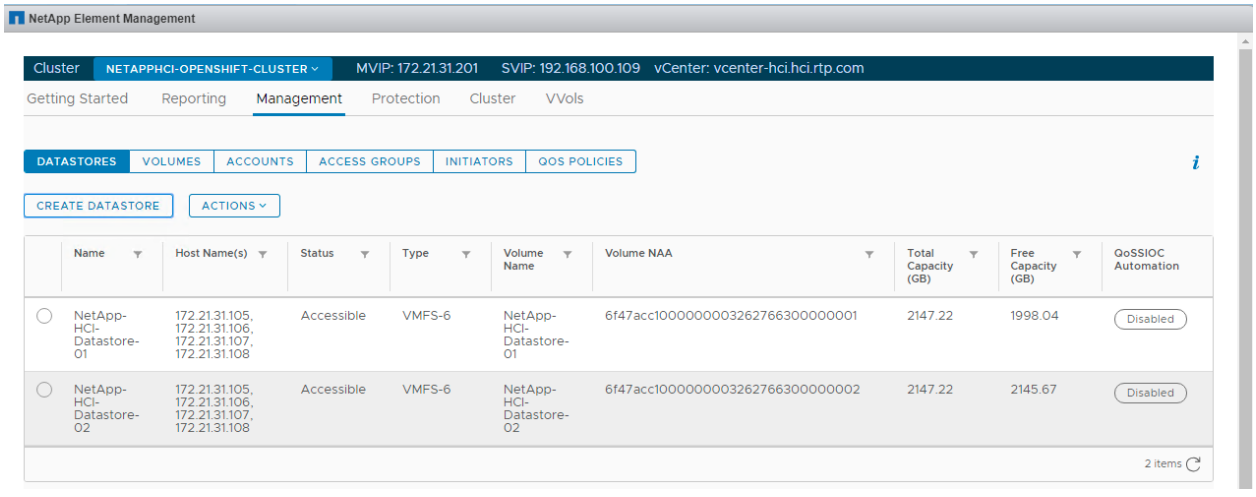
7. Export the configuration after a successful NDE deployment.



Creating an ISO Datastore (Optional)

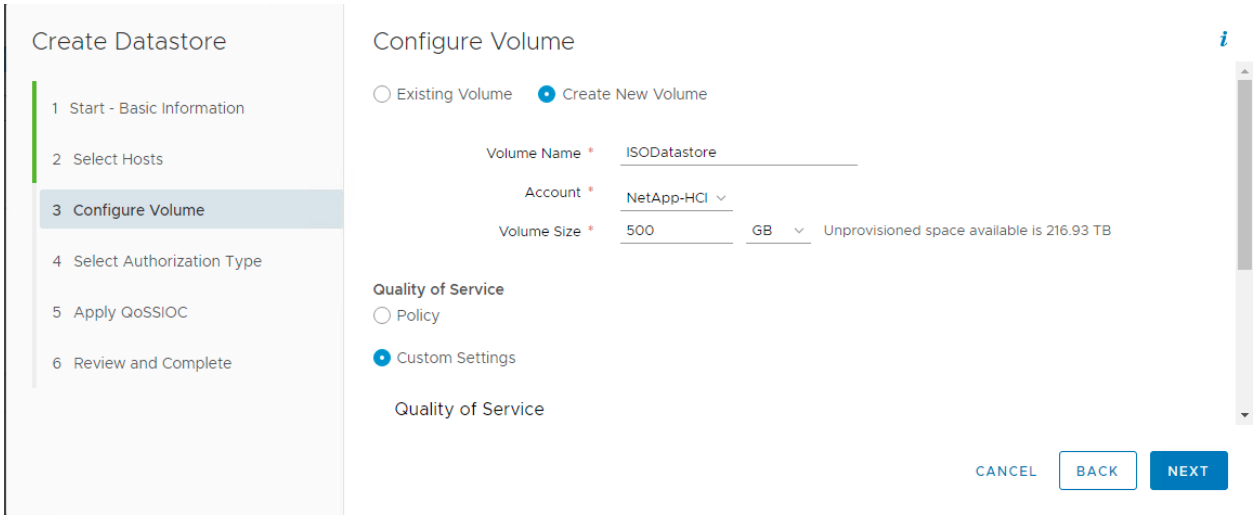
Create a datastore of size 500GB to store the various guest ISO images to be deployed on NetApp HCI.

1. From the vSphere client, navigate to NetApp Element Management.
2. Navigate to the Management and Datastores tab and click Create Datastore.

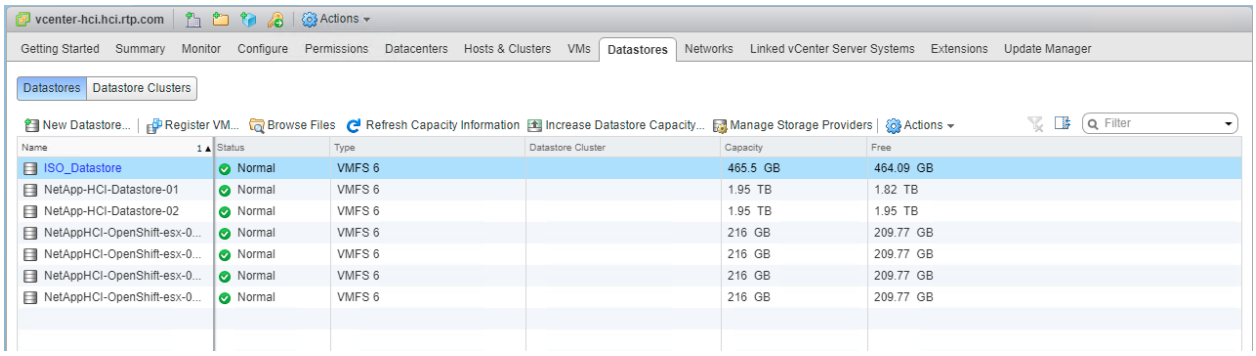


3. Enter the name ISO_Datastore and then select the NetApp HCI cluster.

4. Configure a volume for the datastore.



5. Retain the remaining default settings and create the datastore.

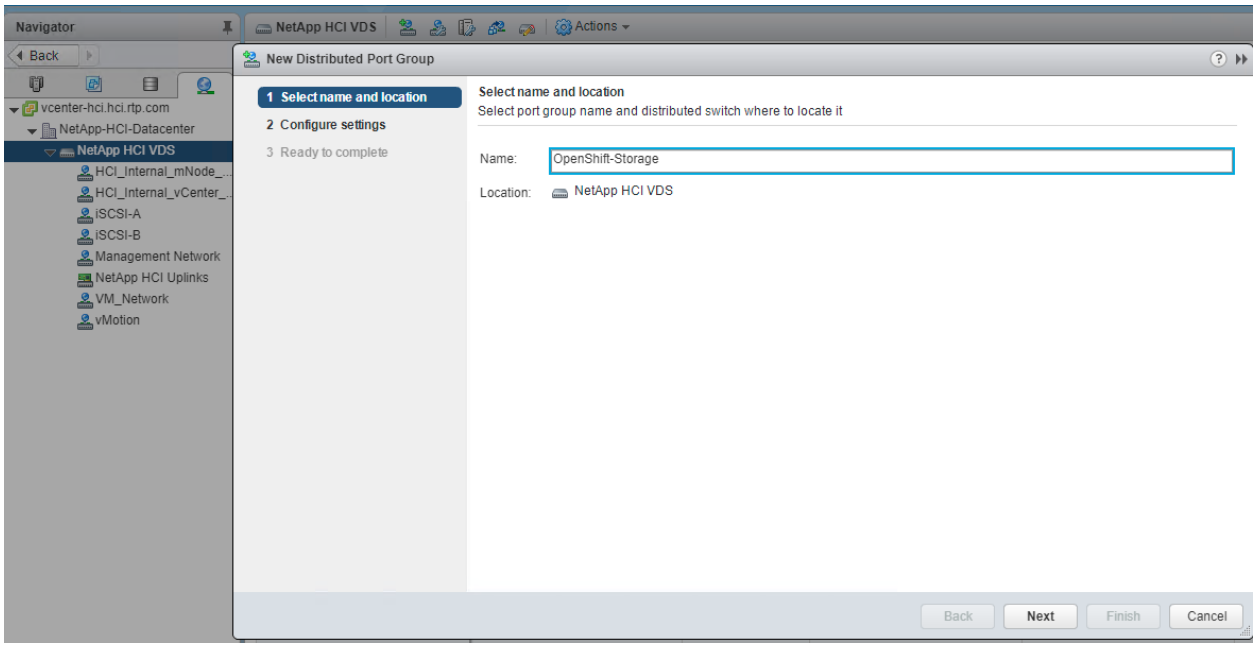


6. Upload the RHEL 7.5 image to the image datastore.

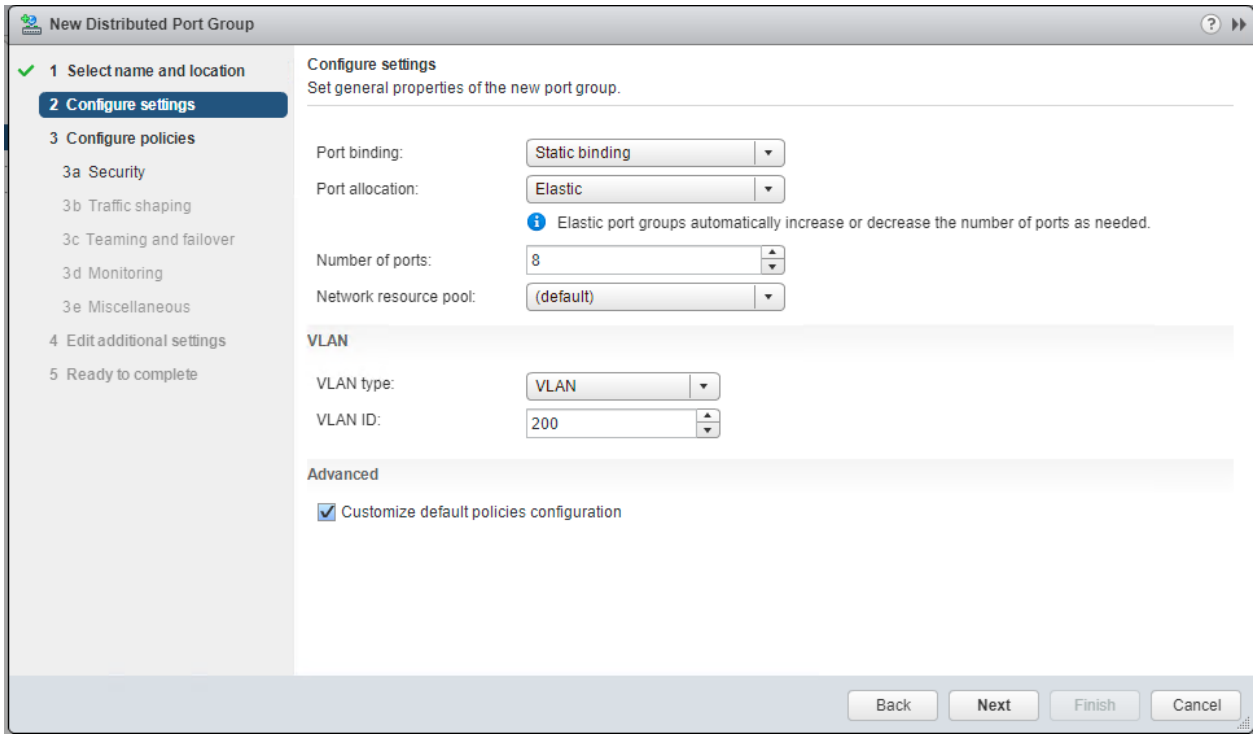
Note: This solution uses Red Hat Enterprise Linux as the guest OS for the OpenShift environment; upload the image ISO in the datastore. The current Red Hat Enterprise Linux image can be obtained from https://access.redhat.com/products/red-hat-enterprise-linux/.

Creating the Required Networks in VMware vSphere

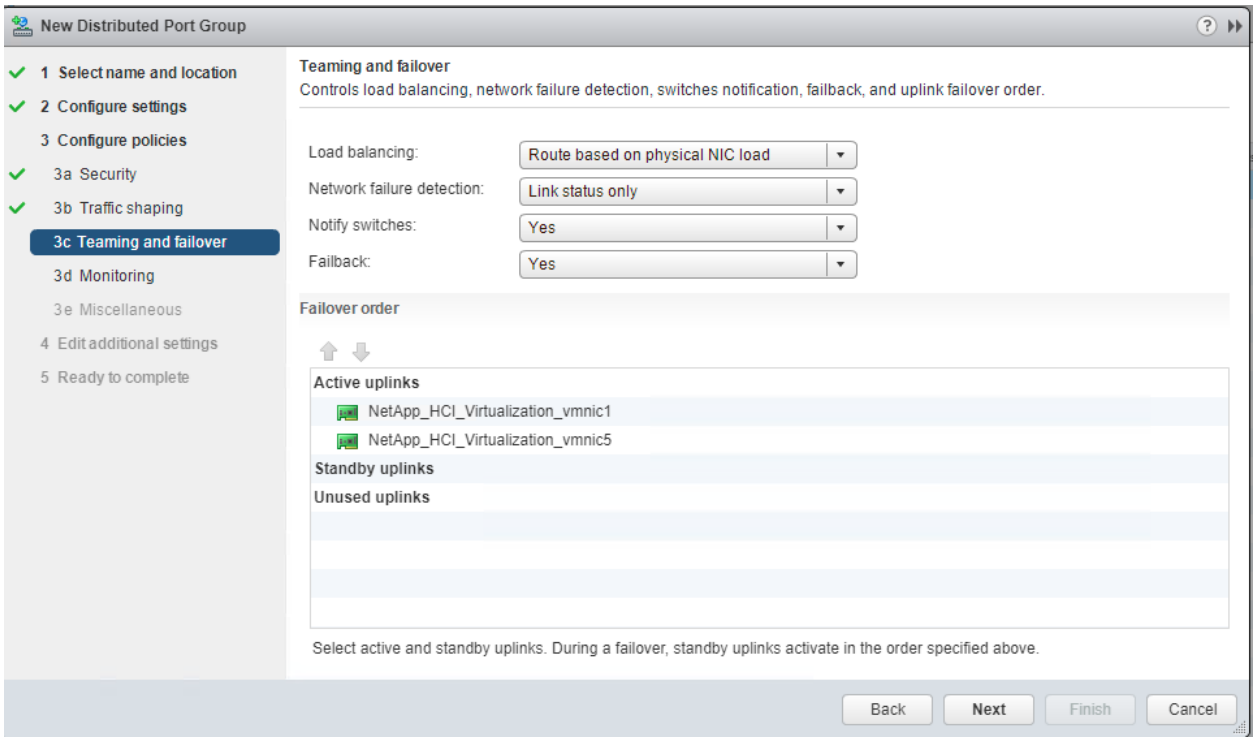
1. Navigate to vSphere networking and create a new virtual dedicated server (VDS).



2. Select VLAN type and VLAN，use openshift_storage_vlan_id as the VLAN ID, and select the customize default policies configuration.



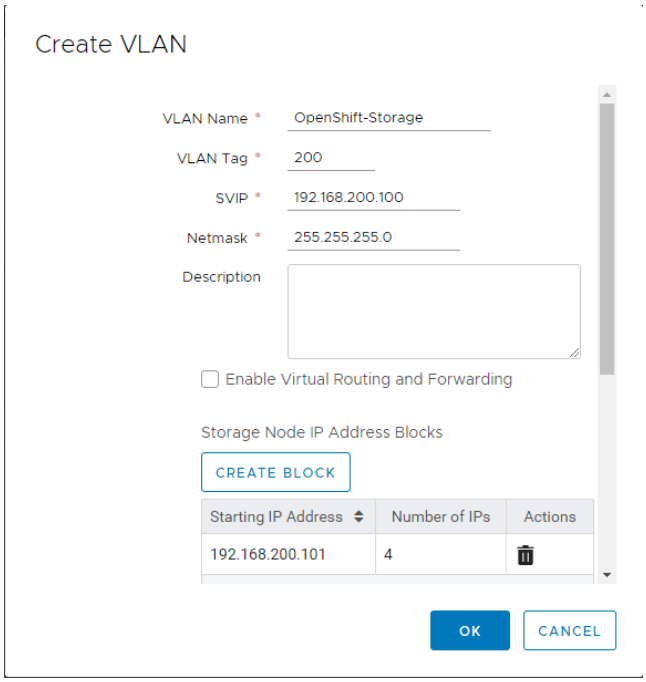
3. Accept the Security and Traffic Shaping defaults and modify Teaming and Failover to use a route based on physical NIC load.



4. Accept the rest of the default configuration and create the new distributed port group.
5. Repeat steps 1 through 4 to create openshift_network and openshift_external_network with their respective VLANS.

Creating an OpenShift Storage Network on a NetApp Element Cluster

1. From vSphere, navigate to NetApp Element Management.
2. Select Cluster and Network.
3. Create a VLAN with the required network settings and select all of the hosts in the cluster.

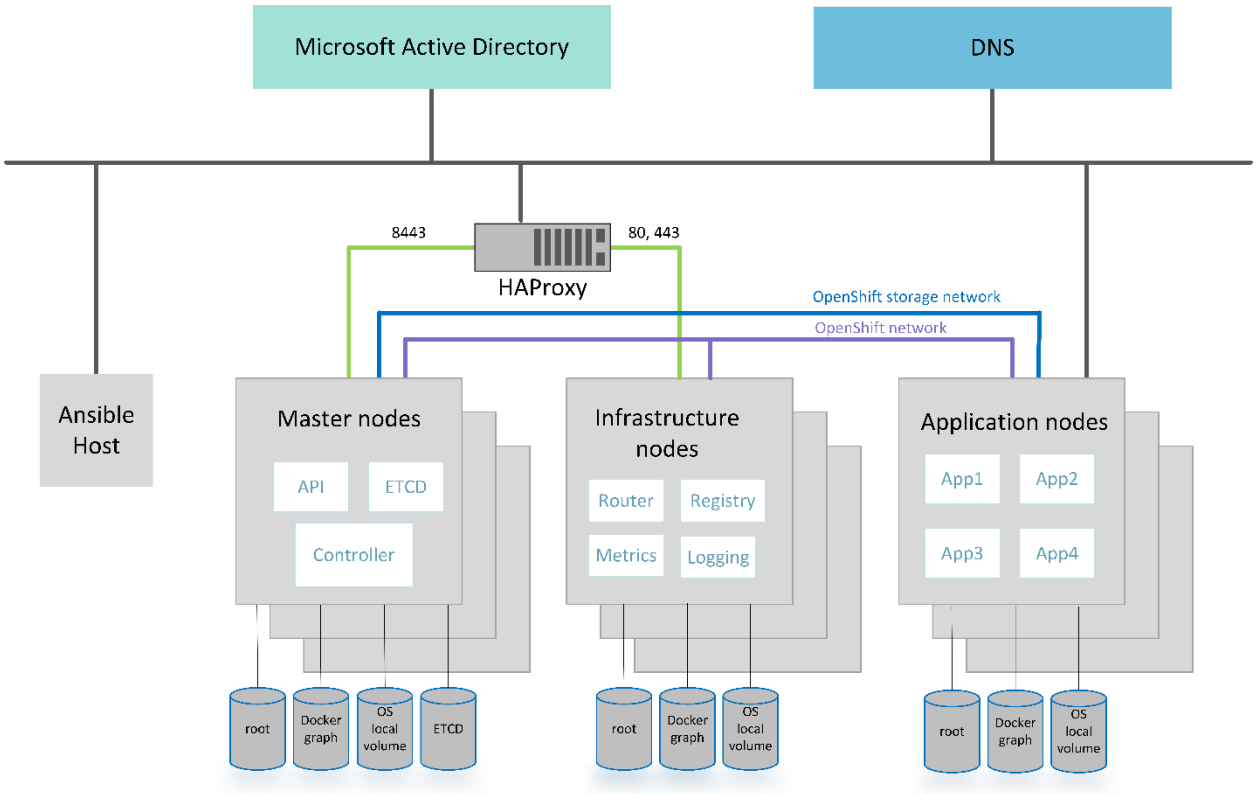


5.2 Deploy and Configure Red Hat Enterprise Linux Virtual Machines for OpenShift

OpenShift Container Platform leverages Kubernetes for managing containerized applications across a set of hosts. A highly available deployment of OpenShift requires the following components:

- Deployment instance (bastion host)
- HA proxy instance
- Three master instances
- Three infrastructure instances
- Four application instances

Figure 5) HA deployment of Red Hat OpenShift Container Platform.

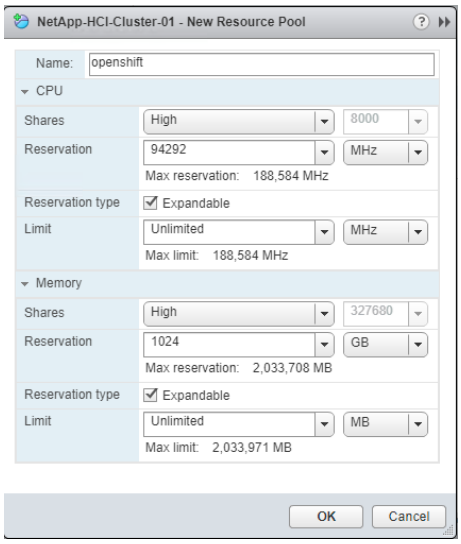


Note: Customers might choose to use an existing load balancer in their environment instead of deploying a new HA proxy instance.

Creating a VMware Resource Pool for Red Hat OpenShift Container Platform

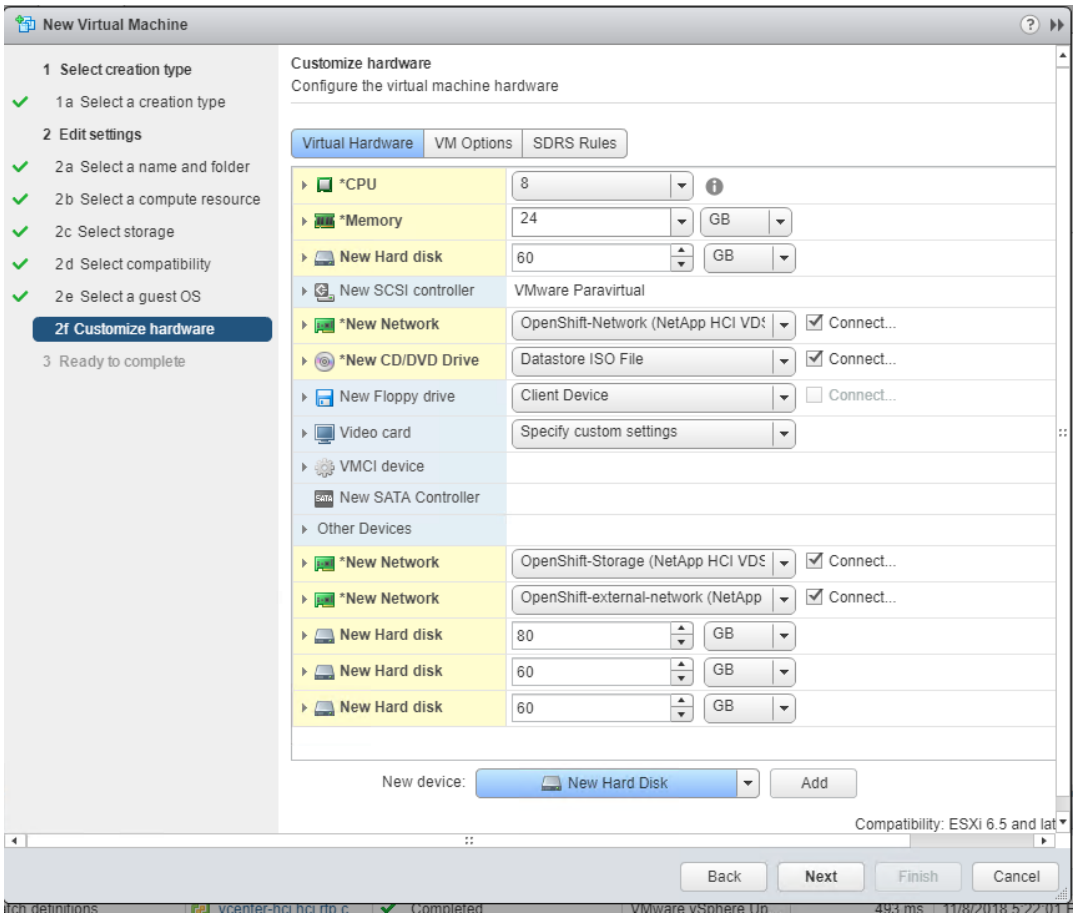
Resource pools enable isolation of resources among different logical entities in the vSphere cluster. Resource pools allow resource guarantees to workloads. As an example, this solution architecture reserves 50% of CPU and memory resources for OpenShift and allows you to expand depending on the usage. You can choose the reservation limits depending on the anticipated microservice applications and the other workloads running in OpenShift cluster. This allows you to run mixed workloads in a risk-free manner.

1. From the vSphere navigation pane, right-click the NetApp-HCI-Cluster-01 and select Create a New Resource Pool.



Creating a Virtual Machine Template for and OpenShift Cluster

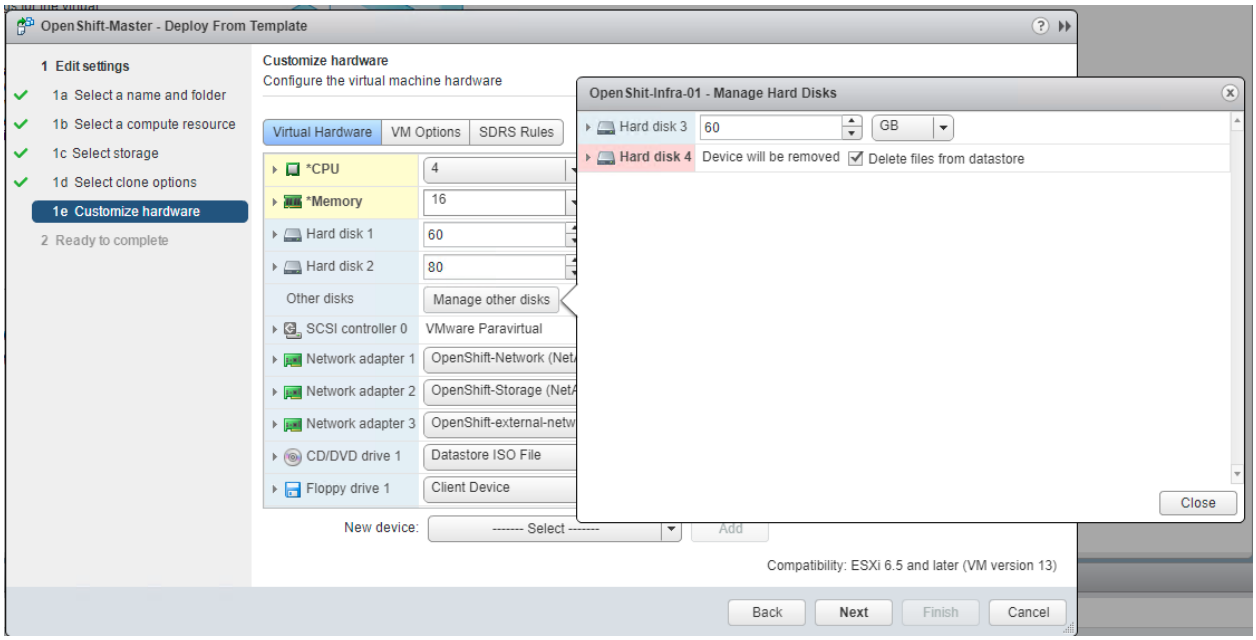
1. Right-click the OpenShift resource pool and click Create a New Virtual Machine.
2. Name the virtual machine OpenShift-Master and place it in the default folder (or optionally create a new folder called OpenShift to organize all the OpenShift-related virtual machines).
3. Select OpenShift as the compute resource pool and NetApp-HCI-Datastore-02 as the storage.
4. Select Linux and Red Hat Enterprise Linux (64 bit) as the guest OS defaults.
5. Customize hardware for the specifications.



6. Right-click the virtual machine and create a template.
7. Navigate to the OpenShift folder and right-click to create a virtual machine – openshift-master01. Select OpenShift as the compute resource pool and NetApp-HCI-Datastore-01 as the datastore for storage. Select Thin Provision as the virtual data disk format.
8. Repeat Step 7 twice more to create OpenShift-Master02 and OpenShift-Master03. Use NetApp-HCI-Datastore-02 and NetApp-HCI-Datastore-01 respectively for storage.
9. Minimum recommendations for hardware resources from Red Hat OpenShift Container Platform 3.10 can be found at https://access.redhat.com/documentation/en-us/openshift_container_platform/3.10/html-single/installing_clusters/.

OpenShift Master Hardware Resources	Value
vCPU	8
RAM	24GB
Root disk	60GB
Graph storage	80GB
OpenShift Local Volume	60GB
etcd volume	60GB

10. Repeat step 7 and customize the virtual machine's hardware to remove the etcd volume and reduce the number of vCPUs to 4. Set memory to 16GB for the OpenShift infrastructure nodes and OpenShift nodes and then create an OpenShift-Infra-01 virtual machine.

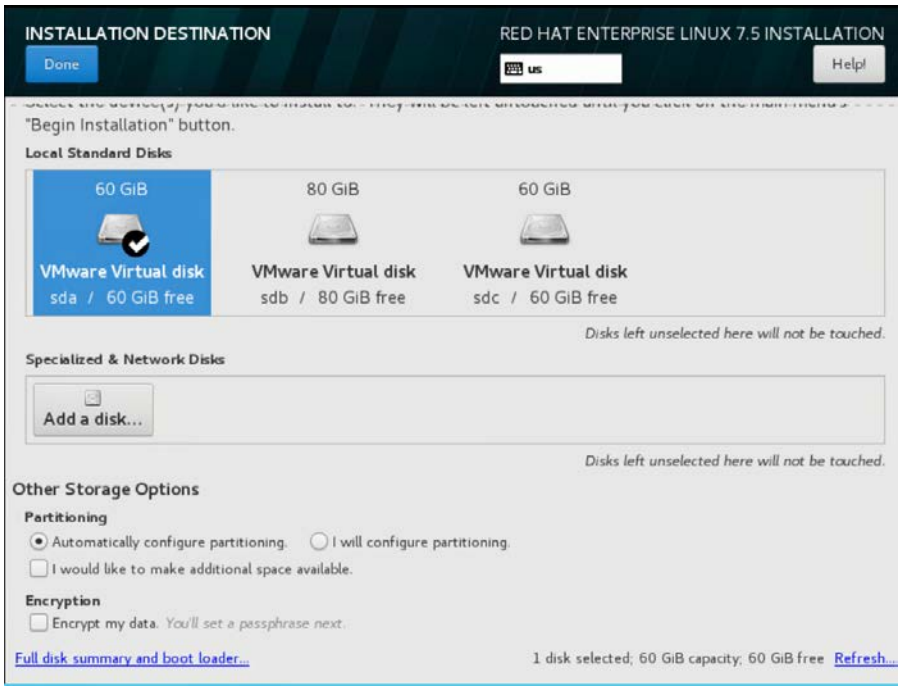


OpenShift Infrastructure Nodes and OpenShift Nodes Hardware Specifications	Value
vCPU	4
RAM	16GB
Root disk	60GB
Graph storage	80GB
OpenShift Local Volume	60GB

11. Clone OpenShift-Infra01 to create OpenShift-Infra02, OpenShift-Infra03, OpenShift-Node01, OpenShift-Node02, OpenShift-Node03, and OpenShift-Node04 VMs.
12. Similarly, deploy the bastion host and HA proxy host.

Installing Red Hat Enterprise Linux

1. Right-click one of the powered-on OpenShift nodes and connect it to the console.
2. Select English as the installation language.
3. Select sda as the rootdisk.



4. Begin the installation and specify the root password.
5. After successful installation, reboot the virtual machine.
6. Configure the network interfaces on the OpenShift virtual machines. Each OpenShift node has the following interfaces. The corresponding port groups are configured with native VLANs.

Interface	VLAN
ens192	`openshift_network`
ens224	`openshift_storage_vlan_id`
ens256	`openshift_external_network`

Note: For a sample network configuration for the network interfaces, see the appendix.

7. Configure the DNS server and add entries for corresponding hosts.

Name	Type	Data
(same as parent folder)	IPv6 Host (AAAA)	fd00:0016:0000:0000:d85c:...
bastion	Host (A)	192.168.35.9
openshift-haproxy	Host (A)	192.168.35.41
openshift-infra01	Host (A)	192.168.35.14
openshift-infra02	Host (A)	192.168.35.15
openshift-infra03	Host (A)	192.168.35.16
openshift-master01	Host (A)	192.168.35.11
openshift-master02	Host (A)	192.168.35.12
openshift-master03	Host (A)	192.168.35.13
openshift-node01	Host (A)	192.168.35.17
openshift-node02	Host (A)	192.168.35.18
sgws-primary-admin	Host (A)	192.168.250.100
vcenter-hci	Host (A)	172.21.31.151

5.3 Deploy Red Hat OpenShift Container Platform

Preparing Your Hosts

1. Log in to individual hosts and set the hostname.

```
hostnamectl set-hostname openshift-master01.hci.rtp.com
```

2. Verify host access and then run the command from the bastion host.

```
[root@bastion ~]# ssh-keygen
```

3. Distribute the SSH keys using a `for` loop

```
[root@bastion ~]# for host in openshift-master01.hci.rtp.com \
 openshift-master02.hci.rtp.com \
 openshift-master03.hci.rtp.com \
 openshift-infra01.hci.rtp.com \
 openshift-infra02.hci.rtp.com \
 openshift-infra03.hci.rtp.com \
 openshift-node01.hci.rtp.com \
 openshift-node02.hci.rtp.com \
 openshift-node03.hci.rtp.com \
 openshift-node04.hci.rtp.com \
 openshift-haproxy.hci.rtp.com
 do ssh-copy-id -i ~/.ssh/id_rsa.pub $host; \
 done
```

4. Register the bastion host.

```
[root@bastion ~]# subscription-manager register
```

5. List available subscriptions.

```
subscription-manager list --available --matches '*OpenShift*'
```

6. Attach the subscriptions output and the pool ID for OpenShift platform.

```
subscription-manager attach --pool=<pool_id>
```

7. Disable all the repositories.

```
subscription-manager repos --disable="*"
```

8. Enable OpenShift Container Platform 3.10 repositories.

```
subscription-manager repos \
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-ose-3.10-rpms" \
    --enable="rhel-7-server-ansible-2.4-rpms"
```

9. Install the following base packages.

```
yum install wget git net-tools bind-utils yum-utils iptables-services bridge-utils bash-
completion kexec-tools sos psacct
```

10. Update the system to the latest packages and reboot.

```
yum update; reboot
```

11. Install the OpenShift packages.

```
yum install openshift-ansible
```

12. Configure the `/etc/ansible/hosts` to include all the OpenShift nodes.

```
[openshift]
openshift-master01.hci.rtp.com
openshift-master02.hci.rtp.com
openshift-master03.hci.rtp.com
openshift-infra01.hci.rtp.com
```

```
openshift-infra02.hci.rtp.com
openshift-infra03.hci.rtp.com
openshift-node01.hci.rtp.com
openshift-node02.hci.rtp.com
openshift-node03.hci.rtp.com
openshift-node04.hci.rtp.com
```

13. Subscribe all the OpenShift nodes with an OpenShift license.

```
ansible openshift -m command -a "subscription-manager register --username aborulka@netapp.com --
password password"
```

14. Attach the OpenShift subscription to all the nodes.

```
ansible openshift -m command -a "subscription-manager attach --pool=pool-id”
```

15. Disable the nonrelevant repositories.

```
ansible openshift -m command -a “subscription-manager repos --disable="*"”
```

16. Enable the OpenShift repositories.

```
ansible openshift -m command -a “subscription-manager repos \
    --enable="rhel-7-server-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-ose-3.10-rpms" \
    --enable="rhel-7-server-ansible-2.4-rpms"”
```

17. Install Docker 1.13.1.

```
ansible openshift -m command -a "yum -y install docker-1.13.1"
```

18. Configure Overlay2 Graph driver for Docker and create a docker-storage-setup file.

```
DEVS=/dev/sdb
VG=docker-vg
DATA_SIZE=95%VG
CONTAINER_ROOT_LV_NAME=docker-lv
CONTAINER_ROOT_LV_MOUNT_PATH=/var/lib/docker
```

19. Copy the file to all the OpenShift nodes.

```
ansible openshift -m copy -a "src=docker-storage-setup dest=/etc/sysconfig/docker-storage-setup"
```

20. Run the docker-storage-setup script to configure the graph storage. The script creates the volume group and the logical volume, configures Docker thin-pool, and mounts it to `/var/lib/docker.`

```
ansible openshift -m command -a "systemctl restart docker-storage-setup"
```

21. Start the Docker service.

```
ansible openshift -m command -a "systemctl start docker"
```

22. Verify that the graph driver is configured to use overlay2 as the driver and to use the newly created volume group. The sections in bold indicate the configuration that you must verify.

```
[root@openshift-master02 ~]# docker info
Containers: 0
 Running: 0
 Paused: 0
 Stopped: 0
Images: 0
Server Version: 1.13.1
Storage Driver: overlay2
```

```
mount
/dev/mapper/docker--vg-docker--lv on /var/lib/docker type xfs
(rw,relatime,seclabel,attr2,inode64,prjquota)
/dev/mapper/docker--vg-docker--lv on /var/lib/docker/containers type xfs
(rw,relatime,seclabel,attr2,inode64,prjquota)
```

```
/dev/mapper/docker--vg-docker--lv on /var/lib/docker/overlay2 type xfs
(rw,relatime,seclabel,attr2,inode64,prjquota)
```

23. Configure OpenShift local volumes.

```
ansible openshift -m  lineinfile -a  "path=/etc/fstab line='/dev/sdc
/var/lib/origin/openshift.local.volumes xfs gquota 0 0'"
ansible openshift -a "mkdir -p /var/lib/origin/openshift.local.volumes"
```

24. In the Ansible hosts file, create a block for masters.

```
 [openshift]
openshift-master01.hci.rtp.com
openshift-master02.hci.rtp.com
openshift-master03.hci.rtp.com
openshift-infra01.hci.rtp.com
openshift-infra02.hci.rtp.com
openshift-infra03.hci.rtp.com
openshift-node01.hci.rtp.com
openshift-node02.hci.rtp.com
openshift-node03.hci.rtp.com
openshift-node04.hci.rtp.com


[master]
openshift-master01.hci.rtp.com
openshift-master02.hci.rtp.com
openshift-master03.hci.rtp.com
```

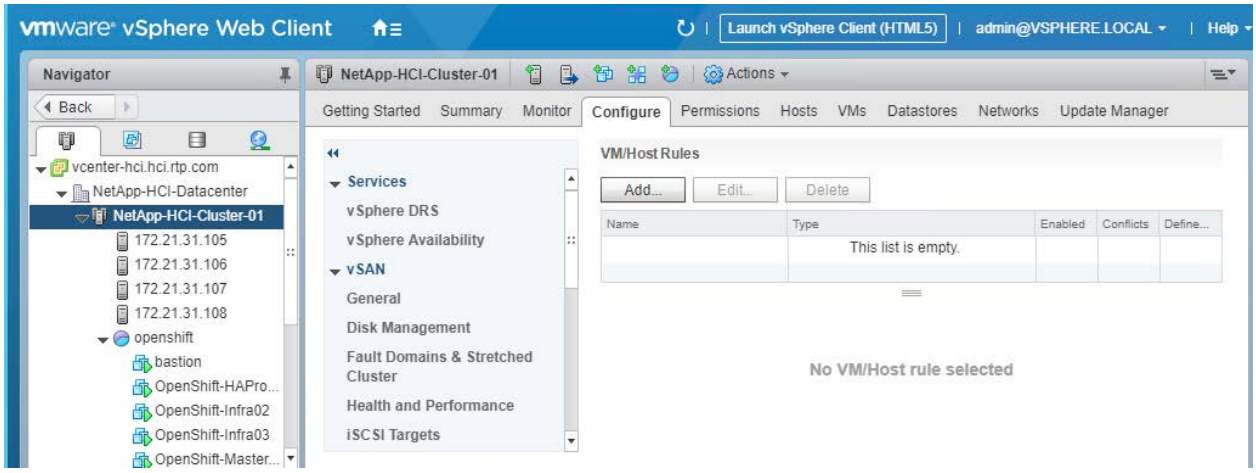
25. Configure separate storage for etcd volumes. Edit the fstab entries to mount the `/dev/sdd` partition to `/var/lib/etcd.`

```
ansible master -m  lineinfile -a  "path=/etc/fstab line='/dev/sdd /var/lib/etcd xfs defaults 0
0'"
```

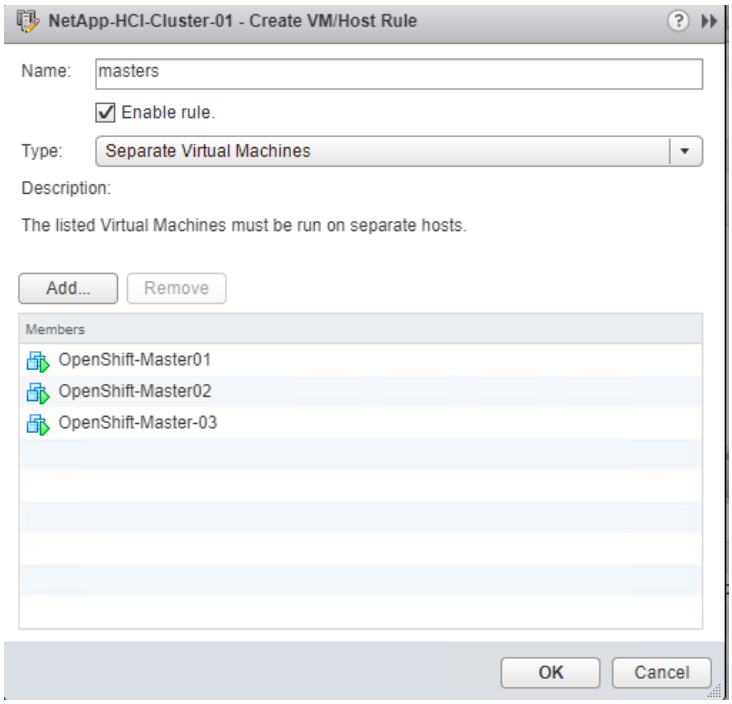
26. Partition the `/dev/sdd` to the XFS file system.

```
ansible master -a "mkfs -t xfs /dev/sdd"
```

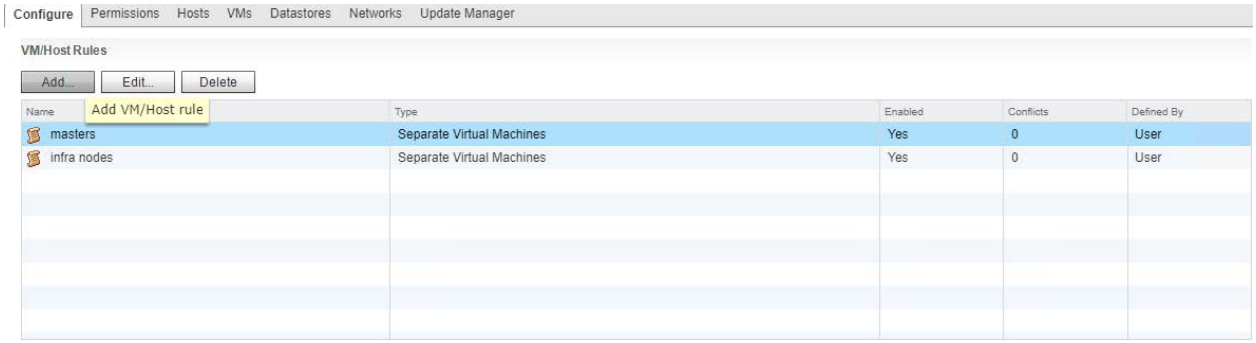
27. Configure VMware VM-VM anti-affinity rules. Navigate to the vSphere cluster and configure the tab in the vSphere client.



28. Click Add and include all the OpenShift Masters. Select the type as Separate Virtual Machines.



29. Similarly, create a rule to separate the OpenShift infrastructure nodes.



Deploy the Red Hat OpenShift Container Platform

The RPM-based installer uses Ansible installed through RPM packages to run playbooks and configuration files available on the local host.

Configuring multiple masters for high availability allows the cluster to have no single point of failure.

When configuring multiple masters, the cluster installation process supports the native HA method. This method leverages the native HA master capabilities that are built into OpenShift Container Platform and can be combined with any load-balancing solution.

If a host is defined in the `[lb]` section of the inventory file, Ansible installs and configures HAProxy automatically as the load-balancing solution. If no host is defined, it is assumed that you have preconfigured an external load-balancing solution of your choice to balance the master API (port 8443) on all master hosts.

Note: This HAProxy load balancer is intended to demonstrate the API server's HA mode. NetApp strongly recommends that you use a TCP-based load balancer or take other steps to provide a HA load balancer.

Multiple Masters Using Native HA with Colocated Clustered etcd

This section describes an example environment for three masters using the native HA method, with etcd running as a static pod on each host. There is one HAProxy load balancer, four nodes for hosting user applications, and three nodes with the `node-role.kubernetes.io/infra=true` label for hosting dedicated infrastructure.

You can see these example hosts in the `[masters]`, `[etcd]`, `[lb]`, and `[nodes]` sections of the following example inventory file

Note: Refer to Configuring Node Host Labels to understand the default node selector requirements and node label considerations beginning in OpenShift Container Platform 3.9.

To use this example, modify the file to match your environment and specifications, and save it as `/etc/ansible/hosts`.

```
# Create an OSEv3 group that contains the master, nodes, etcd, and lb groups.
# The lb group lets Ansible configure HAProxy as the load balancing solution.
# Comment lb out if your load balancer is pre-configured.
[OSEv3:children]
masters
etcd
nodes
lb

# Set variables common for all OSEv3 hosts
[OSEv3:vars]
ansible_ssh_user=root
openshift_deployment_type=openshift-enterprise
os_firewall_use_firewalld=True
os_sdn_network_plugin_name='redhat/openshift-ovs-multitenant'

# uncomment the following to enable htpasswd authentication; defaults to
DenyAllPasswordIdentityProvider
#openshift_master_identity_providers=[{'name': 'htpasswd_auth', 'login': 'true', 'challenge':
'true', 'kind': 'HTPasswdPasswordIdentityProvider'}]

# Native high availability cluster method with optional load balancer.
# If no lb group is defined installer assumes that a load balancer has
# been preconfigured. For installation the value of
# openshift_master_cluster_hostname must resolve to the load balancer
# or to one or all of the masters defined in the inventory if no load
# balancer is present.
openshift_master_cluster_method=native
openshift_master_cluster_hostname=openshift-haproxy.hci.rtp.com
openshift_master_cluster_public_hostname=openshift-haproxy.hci.rtp.com
default_subdomain=hci.rtp.com
osm_default_subdomain="apps.{{ default_subdomain }}"
openshift_enable_service_catalog=false
openshift_node_local_quota_per_fsgroup=512Mi
osm_use_cockpit=false
openshift_hosted_registry_replicas=1
# host group for masters
[masters]
openshift-master01.hci.rtp.com
openshift-master02.hci.rtp.com
openshift-master03.hci.rtp.com

# host group for etcd
[etcd]
openshift-master01.hci.rtp.com
openshift-master02.hci.rtp.com
openshift-master03.hci.rtp.com

# Specify load balancer host
[lb]
openshift-haproxy.hci.rtp.com

# host group for nodes, includes region info
```

```
[nodes]
openshift-master[01:03].hci.rtp.com openshift_node_group_name='node-config-master'
openshift-node01.hci.rtp.com openshift_node_group_name='node-config-compute'
openshift-node02.hci.rtp.com openshift_node_group_name='node-config-compute'
openshift-node03.hci.rtp.com openshift_node_group_name='node-config-compute'
openshift-node04.hci.rtp.com openshift_node_group_name='node-config-compute'
openshift-infra01.hci.rtp.com openshift_node_group_name='node-config-infra'
openshift-infra02.hci.rtp.com openshift_node_group_name='node-config-infra'
openshift-infra03.hci.rtp.com openshift_node_group_name='node-config-infra'
```

The installer uses modularized playbooks that allow you to install specific components as needed. By breaking up the roles and playbooks, you have an increased level of control during installations and the process is more efficient. The playbooks and their ordering are detailed in Running Individual Component Playbooks.

Running the RPM-Based Installer

The RPM-based installer uses Ansible installed through RPM packages to run playbooks and configuration files available on the local host.

1. Run the `prerequisites.yml` playbook. This playbook installs the required software packages, if any, and modifies the container runtimes. Unless you must configure container runtimes, run this playbook only once before you deploy a cluster for the first time.

```
[root@bastion ~]# ansible-playbook /usr/share/ansible/openshift-
ansible/playbooks/prerequisites.yml
PLAY RECAP
*********************************************************************************************
*********************************************
localhost                  : ok=12   changed=0    unreachable=0    failed=0
openshift-haproxy.hci.rtp.com : ok=37   changed=4    unreachable=0    failed=0
openshift-infra01.hci.rtp.com : ok=65   changed=6    unreachable=0    failed=0
openshift-infra02.hci.rtp.com : ok=65   changed=6    unreachable=0    failed=0
openshift-infra03.hci.rtp.com : ok=65   changed=6    unreachable=0    failed=0
openshift-master01.hci.rtp.com : ok=87   changed=6    unreachable=0    failed=0
openshift-master02.hci.rtp.com : ok=71   changed=18   unreachable=0    failed=0
openshift-master03.hci.rtp.com : ok=70   changed=6    unreachable=0    failed=0
openshift-node01.hci.rtp.com : ok=65   changed=6    unreachable=0    failed=0
openshift-node02.hci.rtp.com : ok=65   changed=6    unreachable=0    failed=0
openshift-node03.hci.rtp.com : ok=65   changed=6    unreachable=0    failed=0
openshift-node04.hci.rtp.com : ok=65   changed=6    unreachable=0    failed=0


INSTALLER STATUS
*********************************************************************************************
*****************************************
Initialization  : Complete (0:08:24)
[root@bastion ~]#
```

Note: Do not disable any health checks. These checks verify that the required OpenShift Docker images are available to download.

2. Run the OpenShift cluster deploy playbook.

```
[root@bastion ~]# ansible-playbook   /usr/share/ansible/openshift-
ansible/playbooks/deploy_cluster.yml
```

```
PLAY RECAP
*********************************************************************************************
*********************************************************************************************
**
localhost                  : ok=13   changed=0    unreachable=0    failed=0
openshift-haproxy.hci.rtp.com : ok=38   changed=10   unreachable=0    failed=0
openshift-infra01.hci.rtp.com : ok=130  changed=62   unreachable=0    failed=0
openshift-infra02.hci.rtp.com : ok=130  changed=62   unreachable=0    failed=0
openshift-infra03.hci.rtp.com : ok=130  changed=62   unreachable=0    failed=0
```

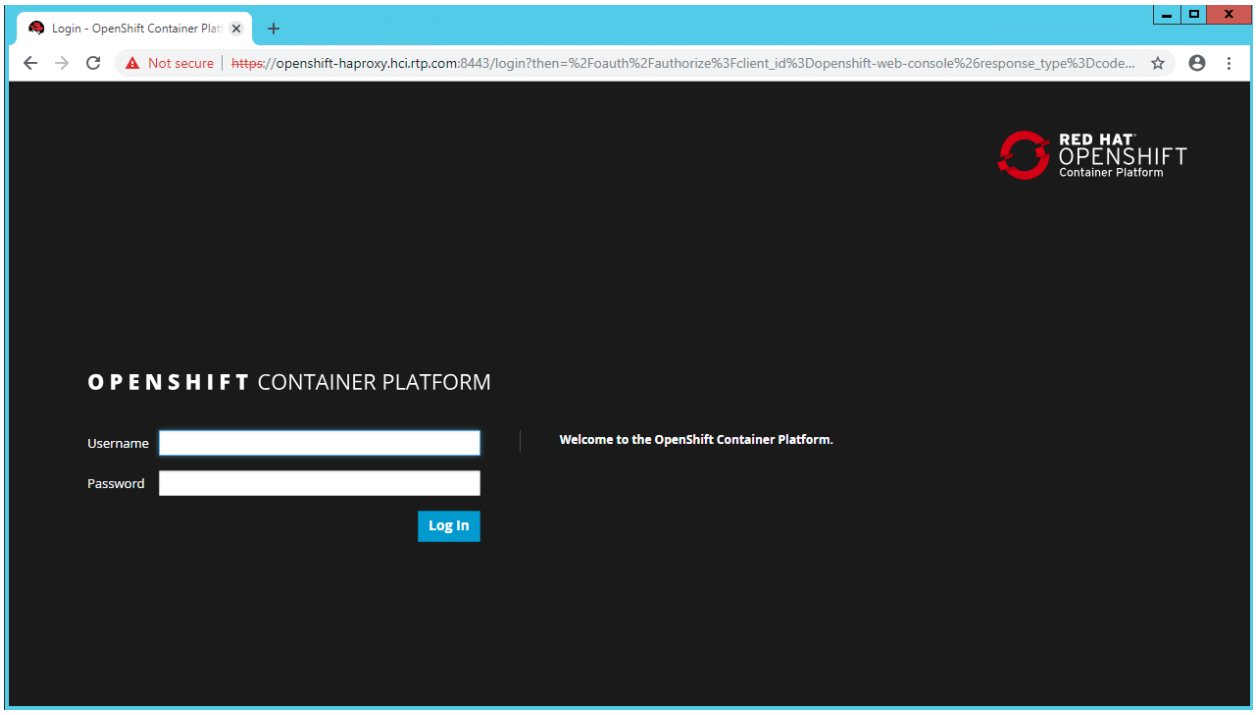
```
openshift-master01.hci.rtp.com : ok=624  changed=267  unreachable=0    failed=0
openshift-master02.hci.rtp.com : ok=305  changed=141  unreachable=0    failed=0
openshift-master03.hci.rtp.com : ok=305  changed=141  unreachable=0    failed=0
openshift-node01.hci.rtp.com : ok=130  changed=62   unreachable=0    failed=0
openshift-node02.hci.rtp.com : ok=130  changed=62   unreachable=0    failed=0
openshift-node03.hci.rtp.com : ok=130  changed=62   unreachable=0    failed=0
openshift-node04.hci.rtp.com : ok=130  changed=62   unreachable=0    failed=0


INSTALLER STATUS
*********************************************************************************************
*********************************************************************************************
Initialization              : Complete (0:00:50)
Health Check                : Complete (0:02:55)
Node Bootstrap Preparation  : Complete (1:04:53)
etcd Install                : Complete (0:01:07)
Load Balancer Install       : Complete (0:01:14)
Master Install              : Complete (0:11:45)
Master Additional Install   : Complete (0:00:53)
Node Join                   : Complete (0:01:35)
Hosted Install              : Complete (0:00:53)
Web Console Install         : Complete (0:00:32)
[root@bastion ~]#
```

Verifying the Installation

1. Verify that the master services have started, and that nodes are registered and reporting in `Ready` status. On one of the master hosts, run the following as root.

```
[root@openshift-master01 ~]# oc get nodes
NAME                           STATUS    ROLES     AGE       VERSION
openshift-infra01.hci.rtp.com    Ready     infra     9h        v1.10.0+b81c8f8
openshift-infra02.hci.rtp.com    Ready     infra     9h        v1.10.0+b81c8f8
openshift-infra03.hci.rtp.com    Ready     infra     9h        v1.10.0+b81c8f8
openshift-master01.hci.rtp.com   Ready     master    10h       v1.10.0+b81c8f8
openshift-master02.hci.rtp.com   Ready     master    10h       v1.10.0+b81c8f8
openshift-master03.hci.rtp.com   Ready     master    10h       v1.10.0+b81c8f8
openshift-node01.hci.rtp.com     Ready     compute   9h        v1.10.0+b81c8f8
openshift-node02.hci.rtp.com     Ready     compute   9h        v1.10.0+b81c8f8
openshift-node03.hci.rtp.com     Ready     compute   9h        v1.10.0+b81c8f8
openshift-node04.hci.rtp.com     Ready     compute   9h        v1.10.0+b81c8f8
[root@openshift-master01 ~]#
```

2. Verify that the web console is functional. Browse `openshift-haproxy.hci.rtp.com:9443` on a web browser.



Creating an Admin User

1. Red Hat OpenShift Container Platform can be integrated with existing authentication mechanisms in your environment such as LDAP and Keystone. For more information about configuring identity providers, see

https://docs.openshift.com/container-platform/3.10/install_config/configuring_authentication.html.

Note: This reference architecture demonstrates authentication operations with HTPasswd as the authentication mechanism. However, NetApp strongly recommends using an existing enterprise authentication paradigm in their environment.

2. Create a flat file with a username and hashed password.

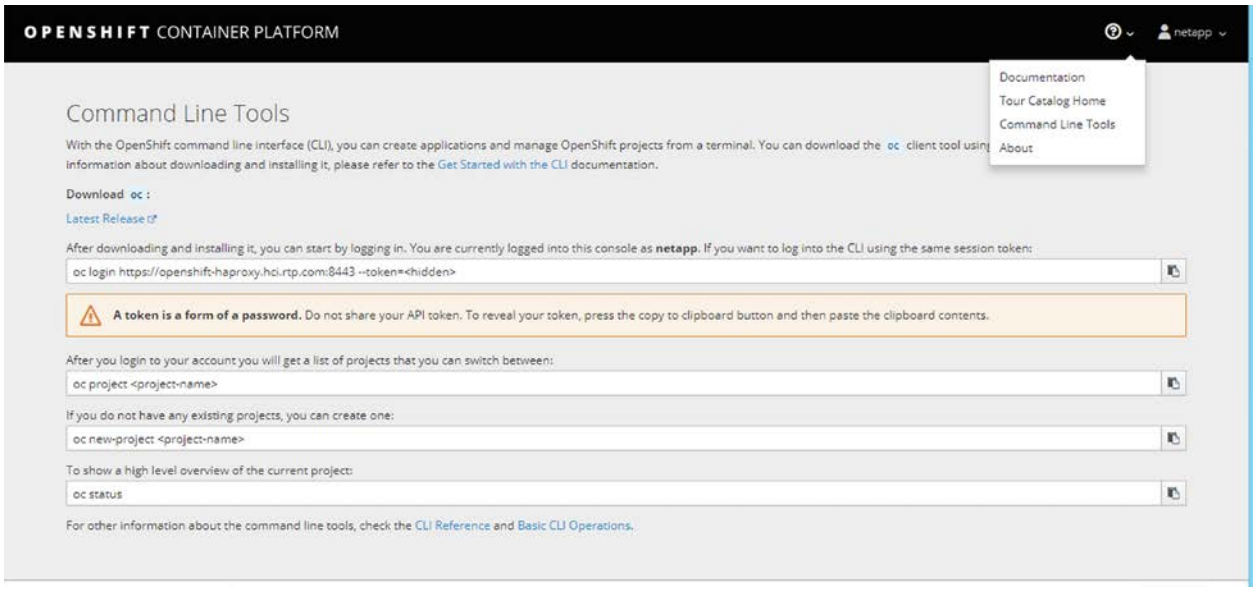
```
[root@openshift-master01 ~]# htpasswd -c /etc/origin/master/users.passwd netapp
New password:
Re-type new password:
Adding password for user netapp
```

3. Verify that you can log in to the OpenShift console using `netapp` user.
4. Escalate the `netapp` user to admin privileges.

```
oc adm policy add-cluster-role-to-user cluster-admin netapp
```

Note: Be very careful when granting the cluster administrator role to a user. Confirm that the user truly needs that level of power. This cluster administrator user can do absolutely anything to any resource on the entire cluster, which can result in destruction if not used appropriately.

5. Log in to the OpenShift console using `netapp` user and download the `oc` command line tool.



6. This example uses bastion host as the Linux client to connect to the OpenShift cluster. However, these instructions can work with any Linux host. Untar the oc command tools package.

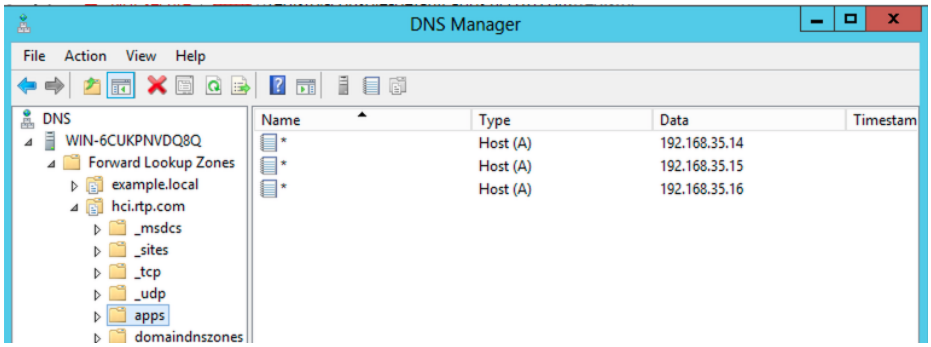
```
[root@bastion ~]# tar -xvf oc-3.10.45-linux.tar.gz
oc
[root@bastion ~]# cp oc /usr/local/sbin/
```

7. Log in to your cluster.

```
oc login https://openshift-haproxy.hci.rtp.com:8443 --token=<token-id>
```

Adding DNS Entry for OpenShift Router

OpenShift router pods are deployed on the OpenShift infrastructure nodes. Configure round-robin on your DNS server and add records to point to resolve `*.apps.hci.rtp.com` all the infrastructure nodes.

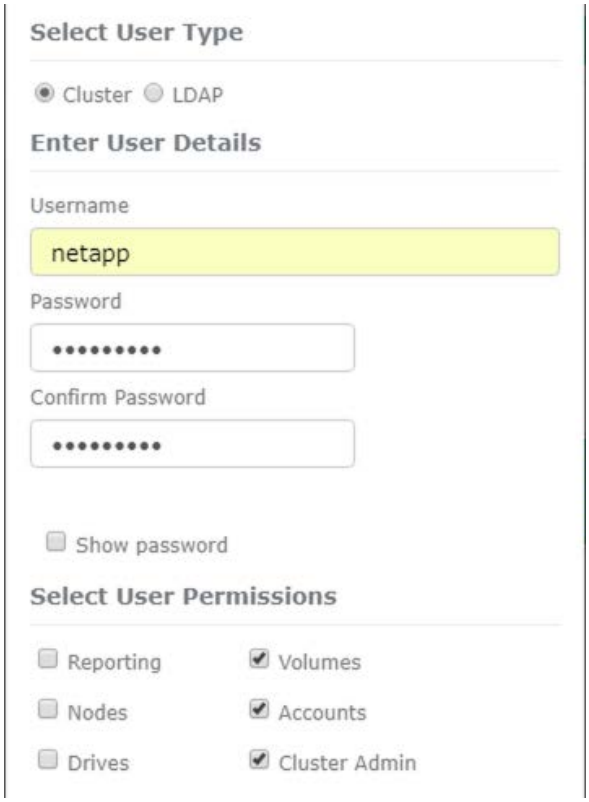


5.4 Deploy and Configure Trident

Comprehensive and detailed documentation for deploying and configuring Trident can be found in the Trident documentation. This section describes Trident deployment and configuration for NetApp HCI.

If you are using Trident, you must have privileges to volumes, accounts, and cluster administrator privileges in the Element OS. NetApp recommends creating a separate user with these privileges instead

of using the default admin user. Create the following user by accessing the NetApp Element MVIP from any browser.



1. From the bastion host, verify that the `netapp` user has sufficient privileges.

```
[root@bastion ~]# oc auth can-i '*' '*' --all-namespaces
Yes
```

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

```
[root@bastion ~]# oc run -i --tty ping --image=busybox --restart=Never --rm --   ping
172.21.31.201
If you don't see a command prompt, try pressing enter.
64 bytes from 172.21.31.201: seq=1 ttl=62 time=0.776 ms
64 bytes from 172.21.31.201: seq=2 ttl=62 time=1.157 ms
64 bytes from 172.21.31.201: seq=3 ttl=62 time=0.800 ms
64 bytes from 172.21.31.201: seq=4 ttl=62 time=1.050 ms
64 bytes from 172.21.31.201: seq=5 ttl=62 time=0.871 ms
```

3. Download Trident installer bundle from the Downloads section and extract it.

```
wget https://github.com/NetApp/trident/releases/download/v18.10.0/trident-installer-
18.10.0.tar.gz
tar -xf trident-installer-18.10.0.tar.gz
cd trident-installer
```

4. Create `setup/backend.json` file with the following configuration.

```
{
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://netapp:password@netapp_hci_element_mvip/json-rpc/8.0",
    "SVIP": "192.168.200.100:3260",
```

```
    "TenantName": "trident",
    "InitiatorIFace": "default",
    "UseCHAP": true,
    "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}},
              {"Type": "default", "Qos": {"minIOPS": 125, "maxIOPS": 2000, "burstIOPS": 4000}}]
}
```

Endpoint corresponds to the Trident user created in step 4 and the MVIP of the NetApp HCI Element cluster. SVIP corresponds to the `openshift_storage_network.`

Each NetApp HCI storage node can support 700 volumes with a maximum of 400 mounted with active I/O. Depending on whether you are using storage node NetApp H410S or NetApp H610S, each of these can support 50,000 to 100,000 IOPS per node. A default QoS band with min IOPs set to 50000/400=125 can be created so that the node doesn't exhaust out of IOPS before the storage capacity.

The types correspond to different QoS bands. New persistent volumes can be created with specific QoS settings by specifying the exact storage pool.

5. Create a new project called Trident in OpenShift and configure the OpenShift cluster to run Trident on OpenShift Infra nodes.

```
oc adm new-project trident --node-selector=node-role.kubernetes.io/infra=true
```

6. Install Trident.

```
[root@bastion trident-installer]# ./tridentctl install  -n trident
INFO Created installer service account.            serviceaccount=trident-installer
INFO Created installer cluster role.               clusterrole=trident-installer
INFO Created installer cluster role binding.       clusterrolebinding=trident-installer
INFO Added security context constraint user.       scc=privileged user=trident-installer
INFO Created installer configmap.                  configmap=trident-installer
INFO Created installer pod.                        pod=trident-installer
INFO Waiting for Trident installer pod to start.
INFO Trident installer pod started.                namespace=trident pod=trident-installer
INFO Starting storage driver.                      backend=/setup/backend.json
INFO Storage driver loaded.                        driver=solidfire-san
INFO Starting Trident installation.                namespace=trident
INFO Created service account.
INFO Created cluster role.
INFO Created cluster role binding.
INFO Added security context constraint user.       scc=anyuid user=trident
INFO Created PVC.
INFO Created iSCSI CHAP secret.                    secret=trident-chap-solidfire-192-168-200-100-
trident
INFO Controller serial numbers.                    serialNumbers=""
INFO Created PV.                                   pv=trident
INFO Waiting for PVC to be bound.                  pvc=trident
INFO Created Trident deployment.
INFO Waiting for Trident pod to start.
INFO Trident pod started.                          namespace=trident pod=trident-7b6bc9ddb5-ccknf
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                 version=18.10.0
INFO Trident installation succeeded.
INFO Waiting for Trident installer pod to finish.
INFO Trident installer pod finished.               namespace=trident pod=trident-installer
INFO Deleted installer pod.                        pod=trident-installer
INFO Deleted installer configmap.                  configmap=trident-installer
INFO In-cluster installation completed.
INFO Deleted installer cluster role binding.
INFO Deleted installer cluster role.
INFO Deleted installer service account.
INFO Removed security context constraint user.     scc=privileged user=trident-installer
[root@bastion trident-installer]#
```

7. Verify that Trident is running with the correct version.

```
[root@bastion trident-installer]# oc get pod -n trident
```

```
NAME                        READY     STATUS    RESTARTS   AGE
trident-7b6bc9ddb5-ccknf    2/2       Running   0          19m
[root@bastion trident-installer]# ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 18.10.0        | 18.10.0        |
+----------------+----------------+
[root@bastion trident-installer]#
```

8. Create a storage back end; this can be the same backend.json used in step 4.

```
./tridentctl -n trident create backend -f setup/backend.json
+--------------------------+----------------+--------+---------+
|           NAME           | STORAGE DRIVER | ONLINE | VOLUMES |
+--------------------------+----------------+--------+---------+
| solidfire_192.168.200.100 | solidfire-san  | true   |       0 |
+--------------------------+----------------+--------+---------+
```

Note: The storage back end specifies the Element cluster in NetApp HCI. We also specify sample default, bronze, silver, and gold types with corresponding QoS specs.

9. Create a storage class that specifies Trident as the provisioner and specifies the storage back end as `solidfire-san`.

```
[root@bastion trident-installer]# vi sample-input/storage-class-solidfire-bronze.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-bronze
provisioner: netapp.io/trident
parameters:
  backendType: "solidfire-san"
  IOPS: "1500"
  fsType: "ext4"
```

An OpenShift administrator can define multiple storage classes that correspond to different QoS requirements based on their applications. A relational DBMS like Oracle might have different QoS requirements than a key-value store like Redis.

Trident selects a storage back end that can satisfy all the criteria specified in the parameters section in the storage class definition. The IOPS parameter helps identify the QoS settings that should be applied to the newly created volume. It's important to note that the IOPS value specified in the storage class is not applied to the volume. Trident selects the type based on the condition `minIOPS <= IOPS <=maxIOPS` and applies the min IOPS, max IOPS, and burst IOPS to the volume as specified in the type.

Note: Therefore, NetApp recommends creating a type in your storage back end that covers the range into which the anticipated IOPS for your application would fall.

```
 [root@bastion trident-installer]# oc create -f sample-input/storage-class-solidfire-bronze.yaml
storageclass.storage.k8s.io "solidfire-bronze" created
[root@bastion trident-installer]#
```

5.5 Deploy Other OpenShift Components

Deploying OpenShift Registry

By default, OpenShift registry uses `emptydir` as the storage back end for storing the registry images. NetApp recommends patching the OpenShift registry service to use a persistent volume as the storage back end for registry.

1. Create a PVC request.

```
kind: PersistentVolumeClaim
apiVersion: v1
```

```
metadata:
  name: openshift-registry
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 500Gi
  storageClassName: solidfire-bronze
```

2. Verify that the PVC request is bound.

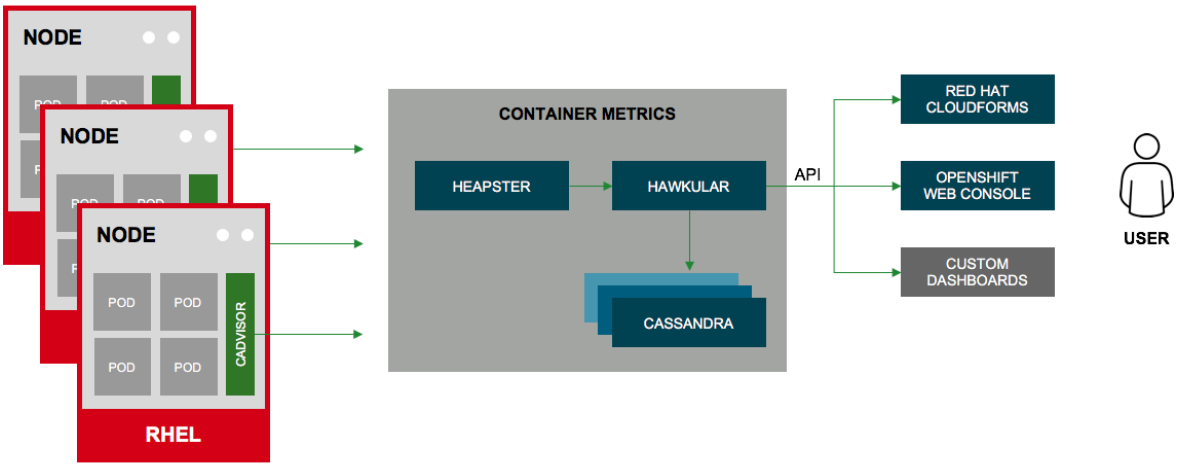
```
[root@bastion trident-installer]# oc get pvc
NAME                 STATUS    VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS   AGE
openshift-registry   Bound     default-openshift-registry-82e21   500Gi      RWO
solidfire-bronze         5s
[root@bastion trident-installer]#
```

3. Patch the docker-registry deployment configuration

```
[root@bastion trident-installer]# oc volume deploymentconfigs/docker-registry --add --
name=registry-storage -t pvc \
> --claim-name=openshift-registry --overwrite
deploymentconfig "docker-registry" updated
[root@bastion trident-installer]#
```

Deploying the OpenShift Metrics Service

In Red Hat OpenShift Container Platform 3.10, cluster metrics are monitored by using Heapster, Hawkular, and Cassandra.



The Cassandra database requires a persistent storage component to store the metrics data in a consistent and reliable manner. To deploy the metrics services, variables are configured in the Ansible inventory file (`/etc/ansible/hosts`)on the bastion host.

```
#Variables to configure metrics services
openshift_metrics_install_metrics=true
openshift_metrics_cassandra_storage_type=dynamic
openshift_metrics_cassandra_pvc_storage_class_name=solidfire-bronze
openshift_metrics_hawkular_nodeselector={"node-role.kubernetes.io/infra":"true"}
openshift_metrics_cassandra_nodeselector={"node-role.kubernetes.io/infra":"true"}
openshift_metrics_hawkular_nodeselector={"node-role.kubernetes.io/infra":"true"}
```

The variables `openshift_metrics_cassandra_pvc_storage_type` and `openshift_metrics_cassandra_pvc_storage_class_name` specify that the storage for the

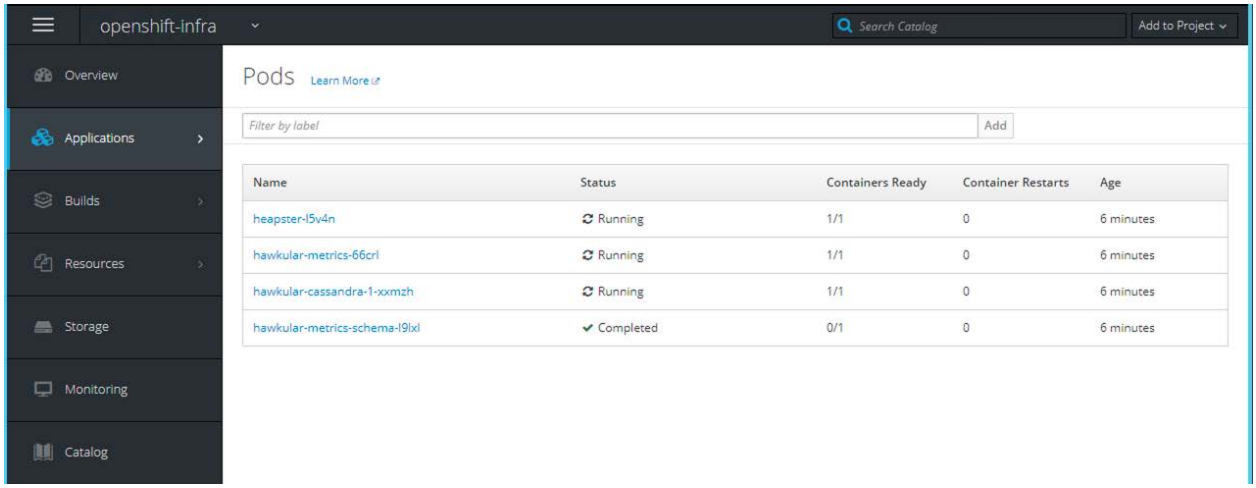
Cassandra database is provisioned dynamically by using the bronze storage class, which was configured earlier.

1. Invoke the metrics services deployment.

```
ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/openshift-metrics/config.yml
```

```
PLAY RECAP
*********************************************************************************************
**
localhost                  : ok=12   changed=0    unreachable=0    failed=0
openshift-haproxy.hci.rtp.com : ok=1    changed=0    unreachable=0    failed=0
openshift-infra01.hci.rtp.com : ok=0    changed=0    unreachable=0    failed=0
openshift-infra02.hci.rtp.com : ok=0    changed=0    unreachable=0    failed=0
openshift-infra03.hci.rtp.com : ok=0    changed=0    unreachable=0    failed=0
openshift-master01.hci.rtp.com : ok=242  changed=49   unreachable=0    failed=0
openshift-master02.hci.rtp.com : ok=35   changed=1    unreachable=0    failed=0
openshift-master03.hci.rtp.com : ok=35   changed=1    unreachable=0    failed=0
openshift-node01.hci.rtp.com : ok=0    changed=0    unreachable=0    failed=0
openshift-node02.hci.rtp.com : ok=0    changed=0    unreachable=0    failed=0
openshift-node03.hci.rtp.com : ok=0    changed=0    unreachable=0    failed=0
openshift-node04.hci.rtp.com : ok=0    changed=0    unreachable=0    failed=0


INSTALLER STATUS
*********************************************************************************************
Initialization   : Complete (0:00:27)
Metrics Install  : Complete (0:01:47)
```

2. Navigate to the openshift-infra project from OpenShift web console and verify that pods for all the services are running.



3. Select one of the pods to view the metrics captured.

4. Log in to NetApp HCI vSphere management and NetApp Element Management to verify that a volume for the Cassandra database was created.

Deploying OpenShift Logging

As an OpenShift Container Platform cluster administrator, you can deploy the EFK stack to aggregate logs for a range of OpenShift Container Platform services. Application developers can view the logs of the projects for which they have view access. The EFK stack aggregates logs from hosts and applications, whether they are coming from multiple containers or even deleted pods.

The EFK stack is a modified version of the ELK stack. It is composed of:

- Elasticsearch (ES): An object store where all logs are stored
- Fluentd: Gathers logs from nodes and feeds them to ES
- Kibana: A web UI for ES

After deployment in a cluster, the stack aggregates logs from all nodes, projects them into ES, and provides a Kibana UI to view any logs. Cluster administrators can view all logs, but application developers can view logs only for projects that they have permission to view. The stack components communicate securely. We configure a separate OpenShift cluster to aggregate logs from OpenShift cluster components.

1. The following variables are configured in the Ansible inventory file `/etc/ansible/hosts` on the bastion host.

```
#Variables to configure logging
openshift_logging_install_logging=true
openshift_logging_use_ops=true
openshift_logging_curator_nodeselector={"node-role.kubernetes.io/infra":"true"}
openshift_logging_kibana_nodeselector={"node-role.kubernetes.io/infra":"true"}
openshift_logging_fluentd_nodeselector={"node-role.kubernetes.io/infra":"true"}
openshift_logging_es_pvc_storage_class_name=solidfire-bronze
openshift_logging_es_nodeselector={"node-role.kubernetes.io/infra":"true"}
openshift_logging_es_ops_pvc_dynamic=true
```

```
openshift_logging_es_ops_nodeselector={"node-role.kubernetes.io/infra":"true"}
openshift_logging_kibana_ops_nodeselector={"node-role.kubernetes.io/infra":"true"}
openshift_logging_curator_ops_nodeselector={"node-role.kubernetes.io/infra":"true"}
```

2. Trigger the deployment.

```
ansible-playbook /usr/share/ansible/openshift-ansible/playbooks/openshift-metrics/config.yml
```

```
PLAY RECAP
*************************************************************************************************
***********************************
localhost                  : ok=12   changed=0    unreachable=0    failed=0
openshift-haproxy.hci.rtp.com : ok=1    changed=0    unreachable=0    failed=0
openshift-infra01.hci.rtp.com : ok=0    changed=0    unreachable=0    failed=0
openshift-infra02.hci.rtp.com : ok=0    changed=0    unreachable=0    failed=0
openshift-infra03.hci.rtp.com : ok=0    changed=0    unreachable=0    failed=0
openshift-master01.hci.rtp.com : ok=363  changed=104  unreachable=0    failed=0
openshift-master02.hci.rtp.com : ok=35   changed=0    unreachable=0    failed=0
openshift-master03.hci.rtp.com : ok=35   changed=0    unreachable=0    failed=0
openshift-node01.hci.rtp.com : ok=0    changed=0    unreachable=0    failed=0
openshift-node02.hci.rtp.com : ok=0    changed=0    unreachable=0    failed=0
openshift-node03.hci.rtp.com : ok=0    changed=0    unreachable=0    failed=0
openshift-node04.hci.rtp.com : ok=0    changed=0    unreachable=0    failed=0



INSTALLER STATUS
*************************************************************************************************
*****************************
Initialization   : Complete (0:00:27)
Logging Install  : Complete (0:04:35)
```

3. Verify that the pods for the logging cluster are up and running.

```
[root@bastion ~]# oc get pods -n openshift-logging
NAME                                      READY     STATUS    RESTARTS   AGE
logging-curator-1-b9pdn                   1/1       Running   0          1d
logging-curator-ops-1-7wqg6               1/1       Running   0          1d
logging-es-data-master-3ezjy36f-1-zzgdz   2/2       Running   0          1d
logging-es-ops-data-master-c48m3zte-1-zcgrk   2/2       Running   0          1d
logging-fluentd-55k8b                     1/1       Running   0          1d
logging-fluentd-5d4tf                     1/1       Running   0          1d
logging-fluentd-6chj9                     1/1       Running   0          1d
logging-fluentd-b4crq                     1/1       Running   0          1d
logging-fluentd-ckxzr                     1/1       Running   0          1d
logging-fluentd-cw8sr                     1/1       Running   0          1d
logging-fluentd-hmtrt                     1/1       Running   0          1d
logging-fluentd-lf786                     1/1       Running   0          1d
logging-fluentd-m2h4v                     1/1       Running   0          1d
logging-fluentd-ppbmp                     1/1       Running   0          1d
logging-kibana-1-4kgld                    2/2       Running   0          1d
logging-kibana-ops-1-jhz2l                2/2       Running   0          1d
[root@bastion ~]#
```

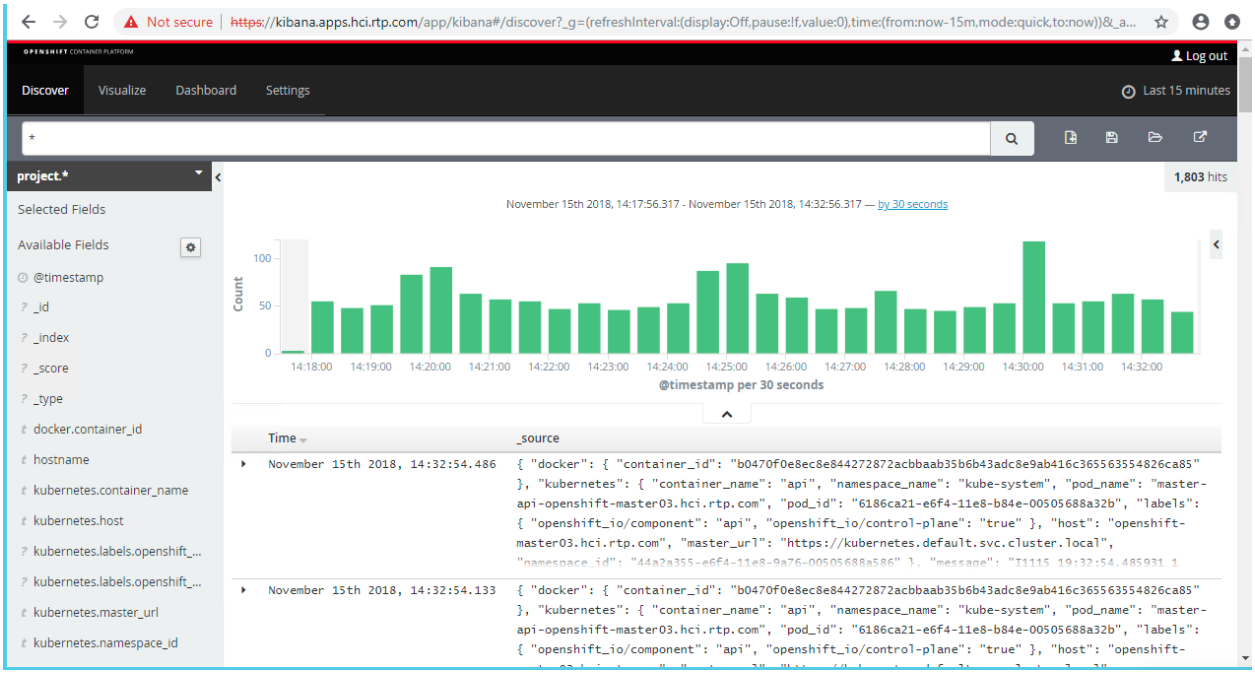
4. Verify that the volumes for Elasticsearch are created.

```
[root@bastion ~]# oc get pvc -n openshift-logging
NAME               STATUS    VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS   AGE
logging-es-0       Bound     openshift-logging-logging-es-0-9071b       10Gi       RWO
bronze         1h
logging-es-ops-0   Bound     openshift-logging-logging-es-ops-0-beacb   10Gi       RWO
bronze         12h
[root@bastion ~]#
```

5. Get the public URL for the Kibana dashboard and then verify its accessibility.

```
[root@bastion ~]# oc get routes -n openshift-logging
```

```
NAME                    HOST/PORT                          PATH      SERVICES              PORT
TERMINATION             WILDCARD
logging-kibana          kibana.apps.hci.rtp.com                      logging-kibana        <all>
reencrypt/Redirect      None
logging-kibana-ops      kibana-ops.apps.hci.rtp.com                  logging-kibana-ops    <all>
reencrypt/Redirect      None
[root@bastion ~]#
```



6 Solution Operations

6.1 Trident Storage Pools and Default Storage Class

Trident enables an OpenShift storage class to not only specify the provisioner, but also to specify attributes associated with the volumes. In the example, we defined a storage back end with `default`, `bronze`, `silver`, and `gold` as the volume types. We can create a storage class that specifies `solidfire_192.168.200.100` as the storage back end and create volumes with default QoS settings.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: bronze
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: netapp.io/trident
parameters:
  storagePools: "solidfire_192.168.200.100:default"
```

In the example, we also specified Kubernetes to use this default storage class. OpenShift defaults to this storage class if no storage class is specified in the Persistent Volume Claim request.

6.2 Quotas and Limit Ranges

You can use quotas and limit ranges to set constraints limiting the number of objects or amount of compute resources that are used in a project. This helps you to manage and allocate resources across all projects and verify that no projects are using more than is appropriate for the cluster size.

You can set limits for resources such as CPU, memory usage per container or pod, and the size of images that can be uploaded to the OpenShift registry.

You can set Persistent Volume Claim (PVC) limits for:

- The number of PVCs in a namespace
- The amount of storage each claim can request
- The amount of cumulative storage the namespace can have

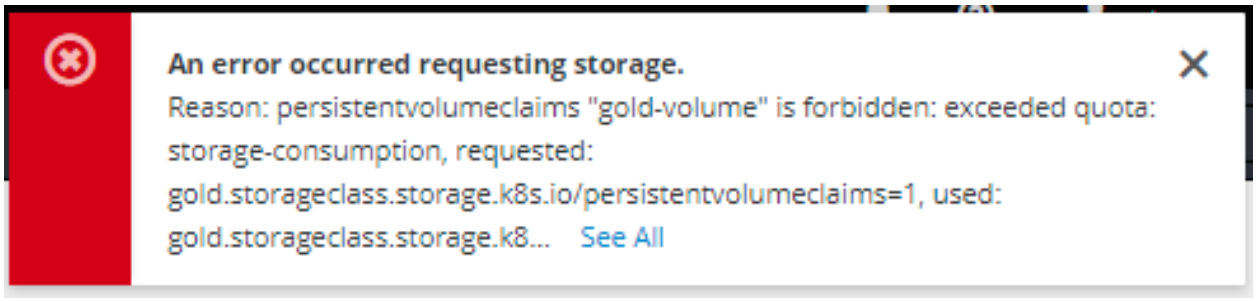
For example, the ResourceQuota object (`storage-quotas.yaml`) restricts the PVC count and the cumulative storage capacity used. This example also restricts the usage of gold storage classes per project.

```
apiVersion: v1
kind: ResourceQuota
metadata:
 name: storage-consumption
spec:
  hard:
    persistentvolumeclaims: "10"
    requests.storage: "50Gi"
    gold.storageclass.storage.k8s.io/persistentvolumeclaims: "0"
```

This limit can be enforced on a project basis by creating this object in association with a namespace.

```
[root@bastion ~]# oc create -f storage-quotas.yaml -n my-wordpress
resourcequota "storage-consumption" created
```

Note: Creating a gold storage class volume in the my-wordpress project results in an error message



6.3 Volume Clone

Volume cloning isn't exposed through the Kubernetes persistent volume framework. However, cloning operations optimize DevOps workflows such as cloning code repositories, artifacts, and other associated data in an optimized manner.

Creating a New Persistent Volume

A new persistent volume can be created by using the `trident.netapp.io/cloneFromPVC` annotation in the PVC request.

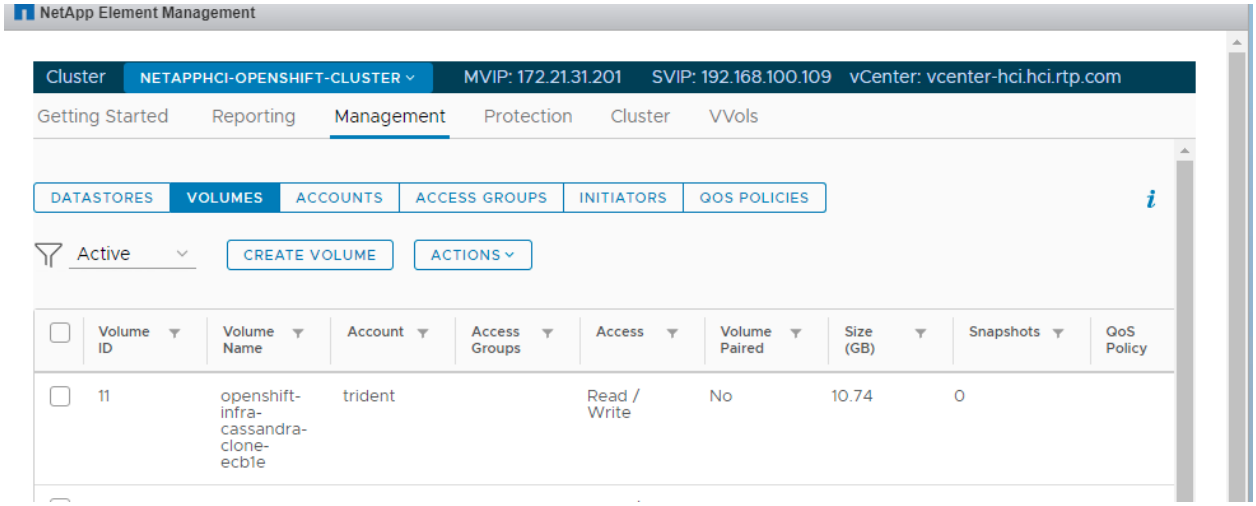
1. Identify the PVC to clone. This example clones the Cassandra volume that is used by OpenShift metrics.

```
[root@bastion ~]# oc get pvc -n openshift-infra
NAME                  STATUS    VOLUME                                     CAPACITY   ACCESS
MODES   STORAGECLASS   AGE
metrics-cassandra-1   Bound     openshift-infra-metrics-cassandra-1-33072   10Gi       RWO
bronze         2d
[root@bastion ~]#
```

2. Create a PVC request.

```
[root@bastion ~]# cat pvc-basic-clone.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: cassandra-clone
  annotations:
    trident.netapp.io/cloneFromPVC: "metrics-cassandra-1"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: bronze
[root@bastion ~]# oc create -f pvc-basic-clone.yaml -n openshift-infra
persistentvolumeclaim "cassandra-clone" created
[root@bastion ~]#
```

3. In the NetApp Element Management pane, verify that a Cassandra clone volume was created.



Note: NetApp recommends that you clone from an idle volume.

Note: PVC and its clone must be in the same OpenShift project and must use the same storage class.

6.4 Verify OpenShift Container Registry Operations

By default, Red Hat OpenShift Container Platform deployment adds only the server certificates that correspond to local cluster IPs and local cluster URLs. These certificates are added as secrets to the registry service. The certificates corresponding to different server names are shown here.

```
[root@openshift-master01 ~]# ls -l /etc/docker/certs.d/
total 0
drwxr-xr-x. 2 root root 32 Nov 12 22:23 docker-registry.default.svc:5000
drwxr-xr-x. 2 root root 27 Nov 12 11:02 redhat.com
drwxr-xr-x. 2 root root 27 Nov 12 11:02 redhat.io
drwxr-xr-x. 2 root root 27 Nov 12 11:02 registry.access.redhat.com
[root@openshift-master01 ~]#
```

Note: Creating a certificate that corresponds to a public routable host name and adding it as a secret to the registry service is out of scope for this document. For comprehensive instructions, refer to the OpenShift documentation.

Verifying Register Operations

1. Log in to one of the OpenShift nodes and pull a test image from the Red Hat registry.

```
[root@bastion trident-installer]# docker pull httpd
```

2. Log in to the OpenShift container registry.

```
[root@openshift-node01 ~]# docker login -u netapp -p `oc whoami -t`  docker-
registry.default.svc:5000
Login Succeeded
[root@openshift-node01 ~]#
```

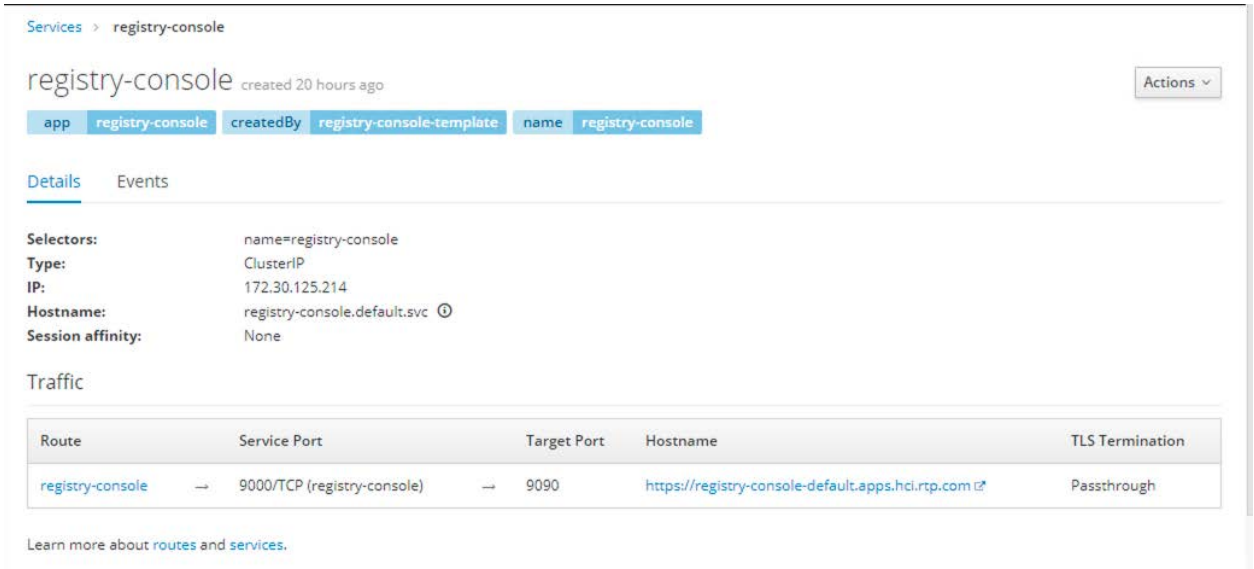
3. Tag the httpd image to the OpenShift container registry.

```
[root@openshift-node01 ~]# docker tag docker.io/httpd docker-
registry.default.svc:5000/default/httpd
```

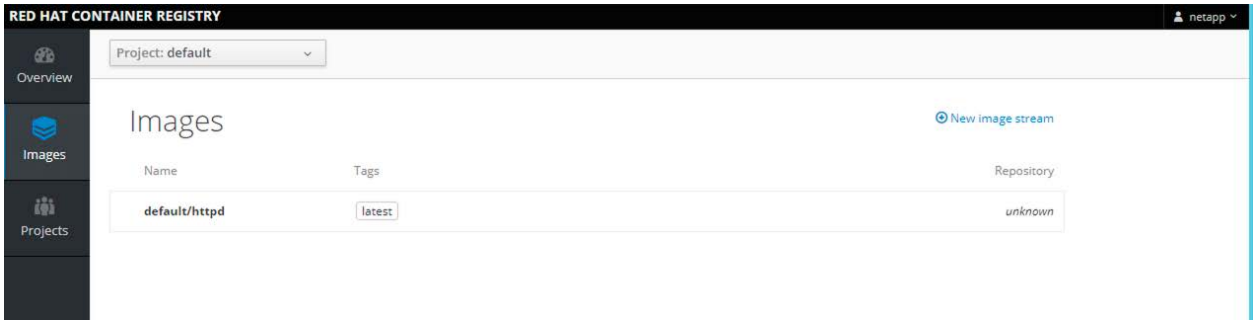
4. Verify the push operation to the OpenShift container registry console.

```
[root@openshift-node01 ~]# docker push docker-registry.default.svc:5000/default/httpd
The push refers to a repository [docker-registry.default.svc:5000/default/httpd]
d8e904686bfd: Pushed
584c122df5a0: Pushed
355bd981febe: Pushed
504b6a6a6fd2: Pushed
237472299760: Pushed
latest: digest: sha256:cd6abe0e1cae37c81a53486f216379c59aae8f9ea790923f23dcf7542853e895 size:
1367
[root@openshift-node01 ~]#
```

5. Examine the registry-console service to get the URL for the OpenShift container registry console.



6. Log in to the registry console https://registry-console-default.apps.hci.rtp.com and navigate to the Images sections. Select the default project from the drop-down list.
7. Verify that the httpd image is uploaded to the OpenShift Container Registry.



7 Conclusion

Red Hat OpenShift Container Platform with NetApp HCI enables enterprises to accelerate application development and deployment by providing an enterprise-class HA and scalable platform to run microservices and container workloads in a risk-free manner. This document provides specific implementation, configuration, and validation details to deploy the solution at an enterprise scale.

Appendix

Network Interface Configuration Files

```
[root@openshift-master01 ~]# cat /etc/sysconfig/network-scripts/ifcfg-ens192
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens192
UUID=4b513d06-c8a6-4f4a-83da-3da01b8b9160
DEVICE=ens192
ONBOOT=yes
IPADDR=192.168.35.11
PREFIX=24
GATEWAY=192.168.35.151
DNS1=192.168.35.51
```

```
[root@openshift-master01 ~]# cat /etc/sysconfig/network-scripts/ifcfg-ens224
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens224
UUID=2395fcd6-1bc8-402c-b47d-2dbd466f3276
DEVICE=ens224
ONBOOT=yes
IPADDR=192.168.200.11
PREFIX=24
```

```
[root@openshift-master01 ~]# cat /etc/sysconfig/network-scripts/ifcfg-ens256
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=ens256
UUID=252937a2-75a8-4564-9cea-c236f741c637
DEVICE=ens256
```

```
ONBOOT=yes
IPADDR=192.168.90.11
PREFIX=24
```

Where to Find Additional Information

To learn more about the information that is described in this document, review the following documents and/or websites:

- Red Hat OpenShift Container Platform with NetApp HCI Design guide
  www.netapp.com/us/media/nva-1124.pdf
- NetApp HCI
  www.netapp.com/us/products/converged-systems/hyper-converged-infrastructure.aspx
- NetApp HCI datasheet
  www.netapp.com/us/media/ds-3881.pdf
- NetApp Trident documentation
  https://netapp-trident.readthedocs.io/en/stable-v18.10/
- Red Hat OpenShift Container Platform 10
  https://access.redhat.com/documentation/en-us/openshift_container_platform/3.10/html/installing_clusters/
- Red Hat OpenShift Container Platform Scaling and Performance guide
  https://access.redhat.com/documentation/en-us/openshift_container_platform/3.10/html-single/scaling_and_performance_guide/

Version History

Version	Date	Document Version History
Version 1.0	November 2018	Initial release

Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.

Copyright Information

Trademark Information