Containers

NetApp Solutions

NetApp
October 21, 2021

Table of Contents

Containers

Archived Solutions

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:

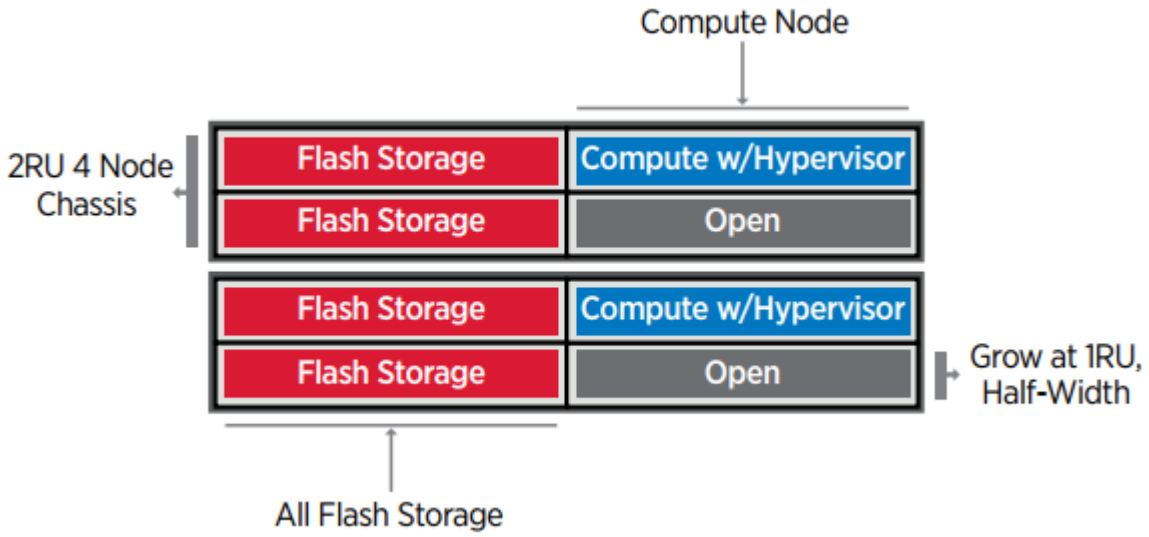
- High availability at all layers in the stack
- Ease of deployment procedures
- Nondisruptive operations and upgrades
- API-driven and programmable infrastructure to keep up with microservices agility
- Multitenancy with performance guarantees
- Ability to run virtualized and containerized workloads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



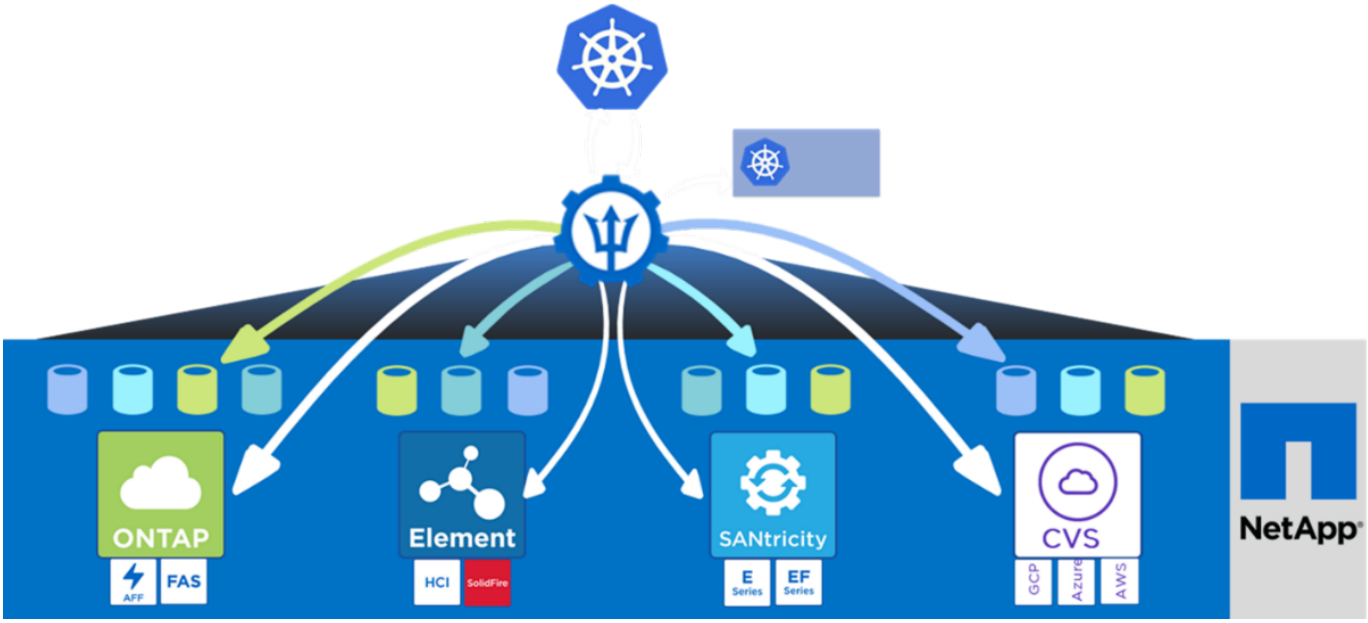
The design for NetApp HCI for Red Hat Virtualization consists of the following components in a minimum starting configuration:

- 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.

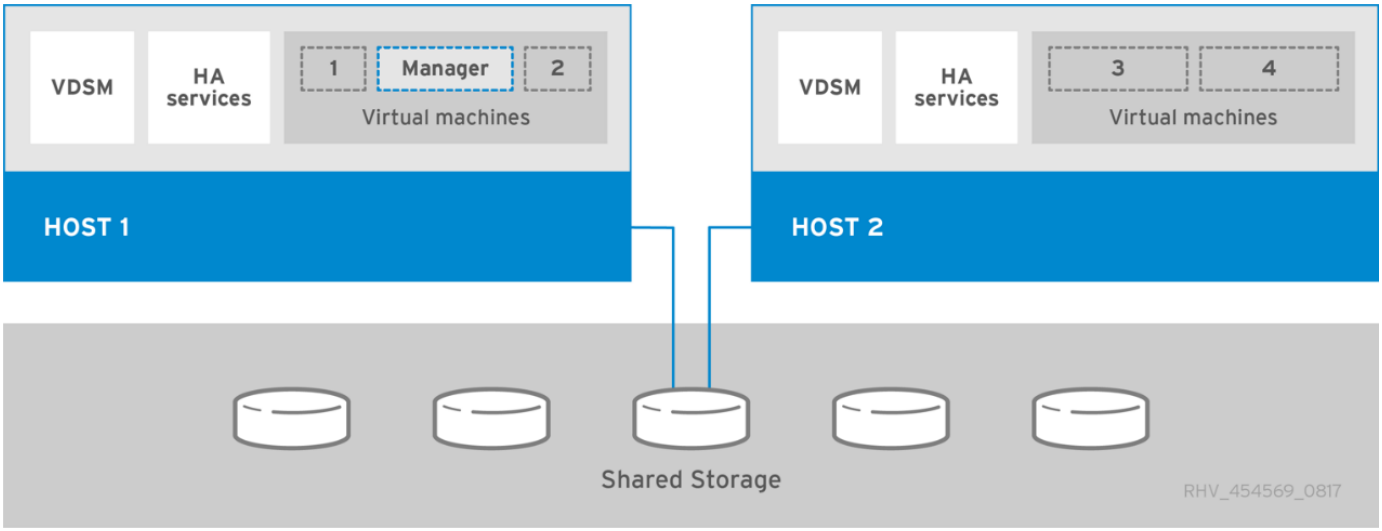
Two types of hosts are supported in RHV are RHV-H and RHEL hosts:

- 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

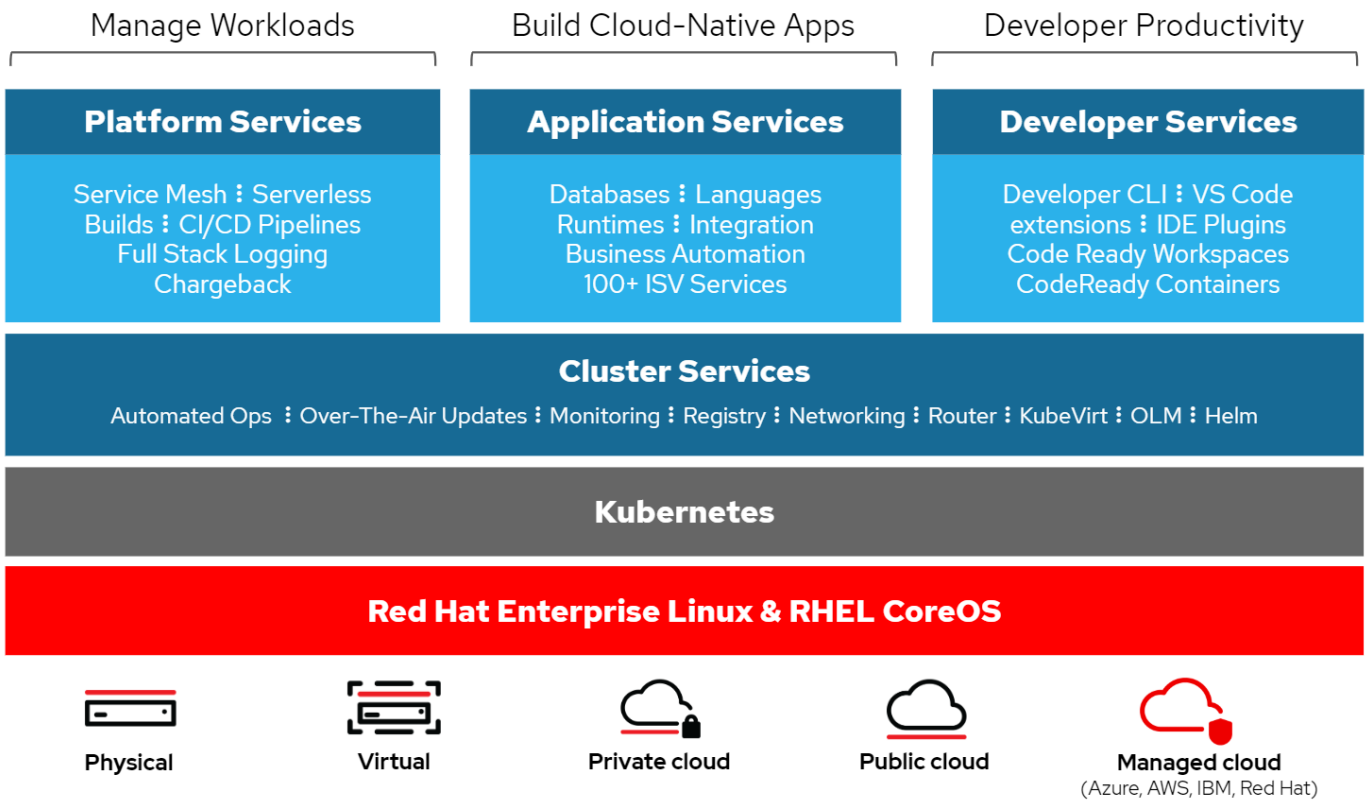
RHV can be deployed in two different architectures: with the RHV-M as a physical server in the infrastructure or with the RHV-M configured as a self-hosted engine. The self-hosted engine deployment, where the RHV-M is a VM hosted in the same environment as other VMs, is recommended and used specifically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



Next: Architectural Overview: NetApp HCI for Red Hat OpenShift on RHV.

Abstract

This NetApp HCI for Red Hat OpenShift on Red Hat Virtualization (RHV) deployment guide is for the fully automated installation of Red Hat OpenShift through the Installer Provisioned Infrastructure (IPI) method onto the verified enterprise architecture of NetApp HCI for Red Hat Virtualization described in NVA-1148: NetApp HCI with Red Hat Virtualization. This reference document provides deployment validation of the Red Hat OpenShift solution, integration of the NetApp Trident storage orchestrator, and a solution verification consisting of an example application deployment.

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
Red Hat OpenShift	Container orchestration	4.4.6

Next: Design Considerations: NetApp HCI for Red Hat OpenShift on RHV

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).

Next: 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

Next: Validation Results: NetApp HCI for Red Hat OpenShift on RHV

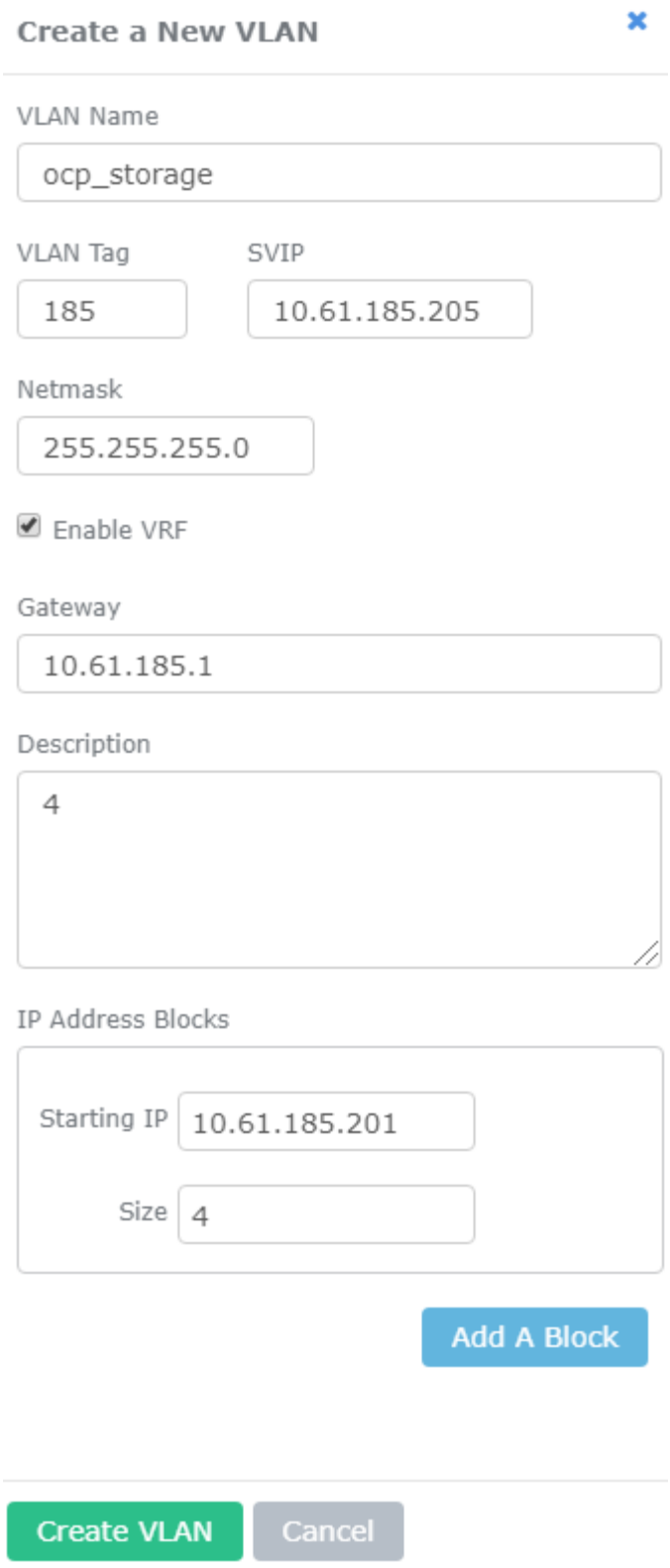
1. Create Storage Network VLAN: NetApp HCI for Red Hat OpenShift on RHV

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.



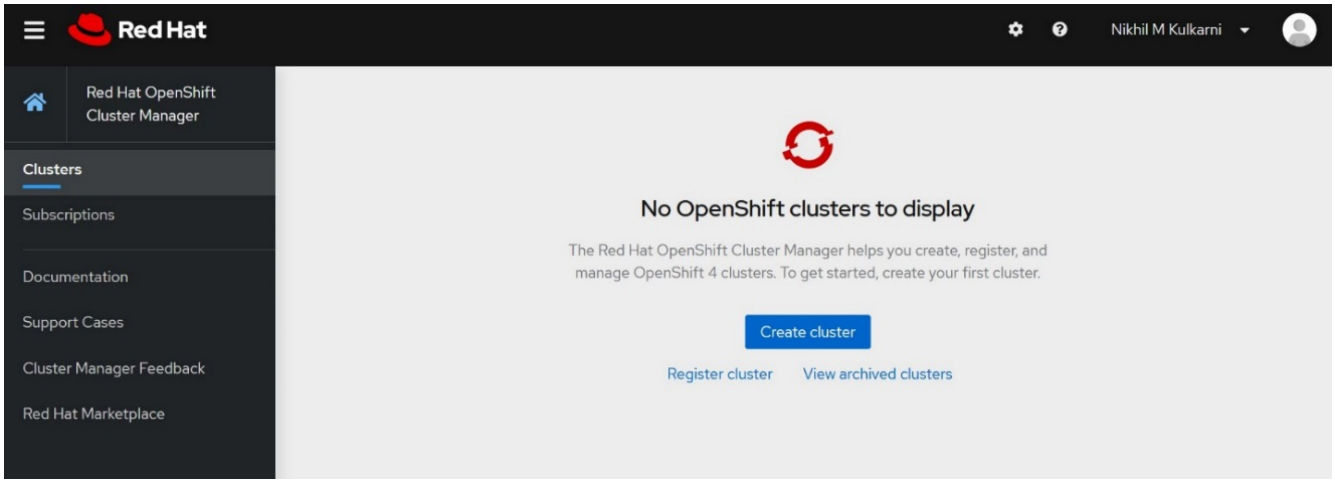
Next: 2. Download OpenShift Installation Files

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

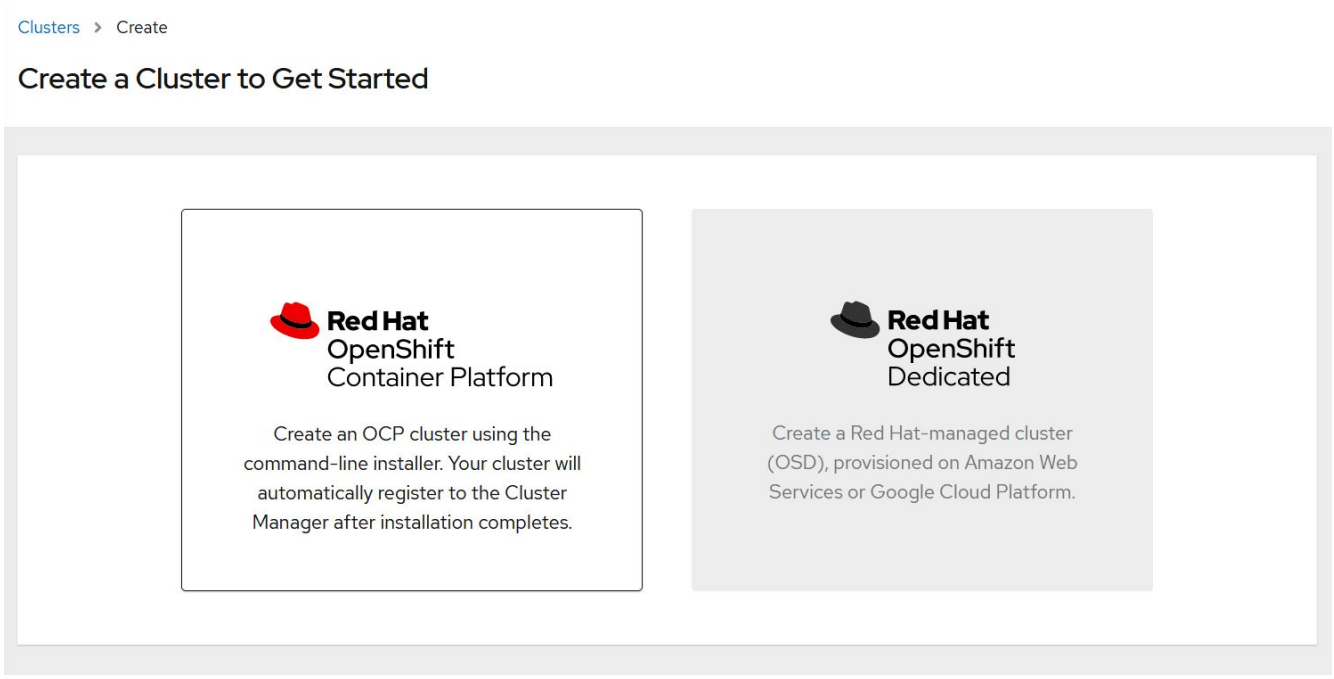
To download the OpenShift installation files, complete the following steps:

1. Go to the Red Hat login page and log in with your Red Hat credentials.

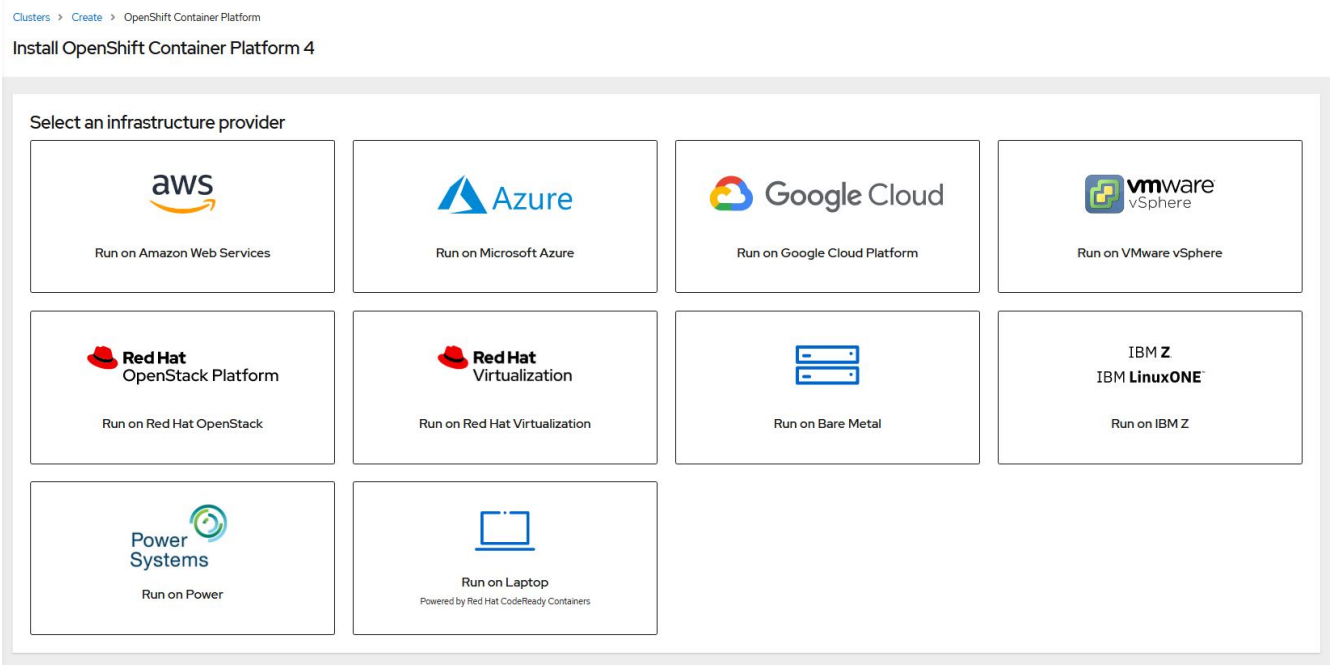
2. On the Clusters page, click Create Cluster.



3. Select OpenShift Container Platform.

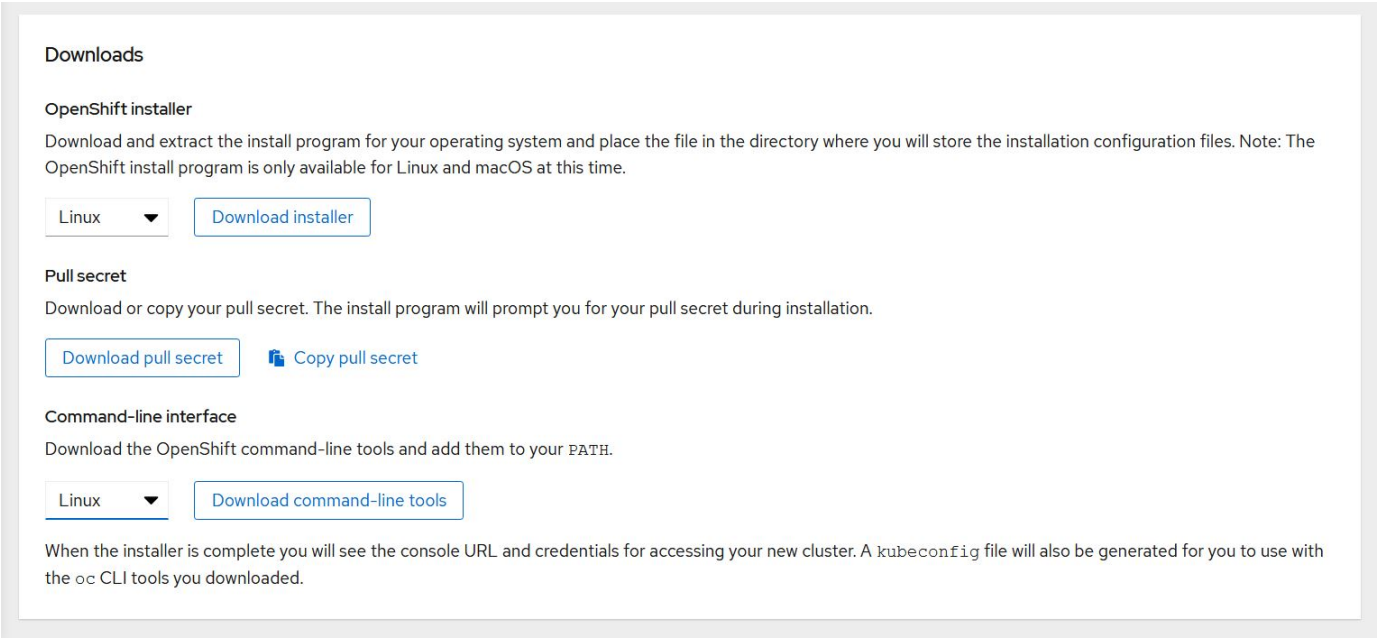


4. Select Run on Red Hat Virtualization.



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.



Next: 3. Download CA Certificate from RHV

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
```

Next: 4. Register API/Apps in DNS

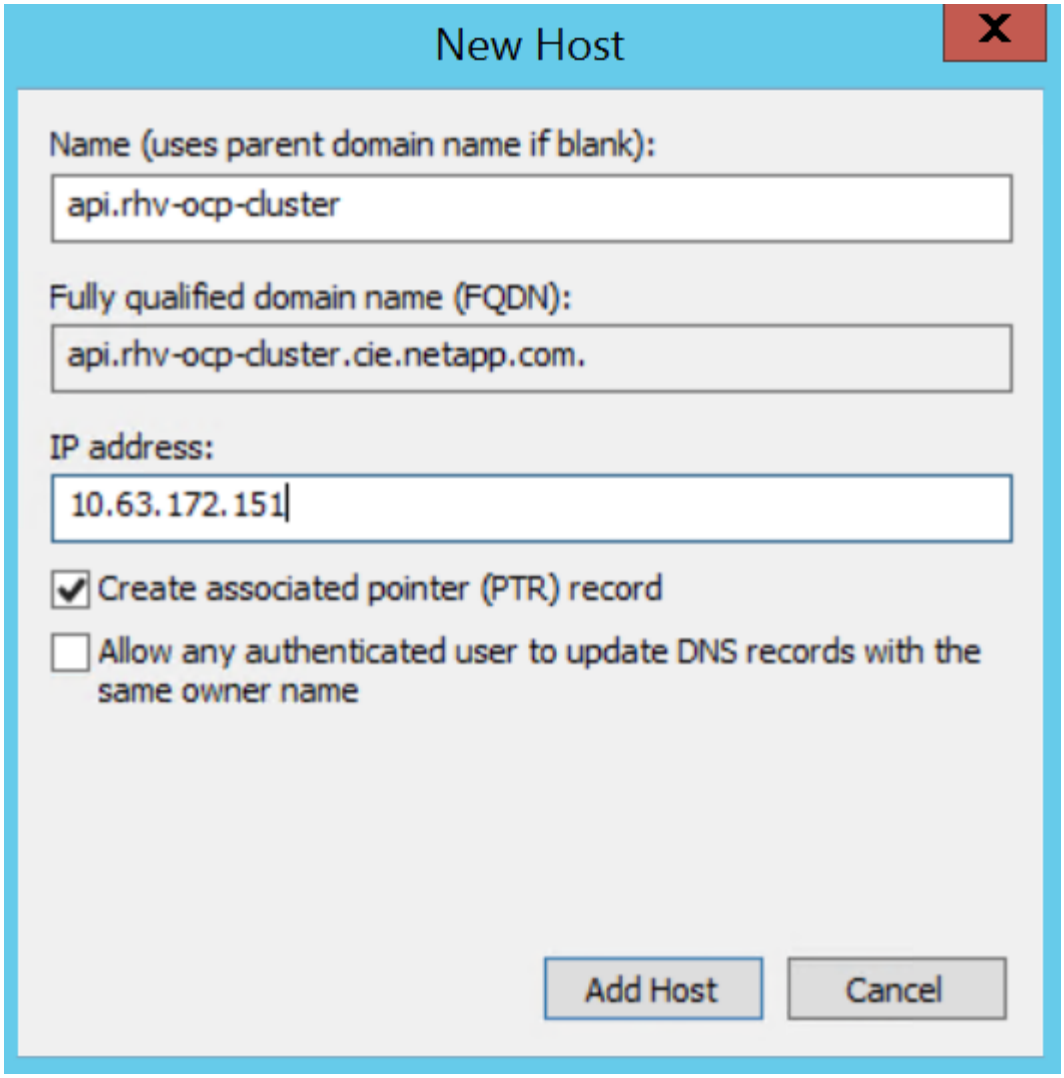
4. Register API/Apps in DNS: NetApp HCI for Red Hat OpenShift on RHV

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

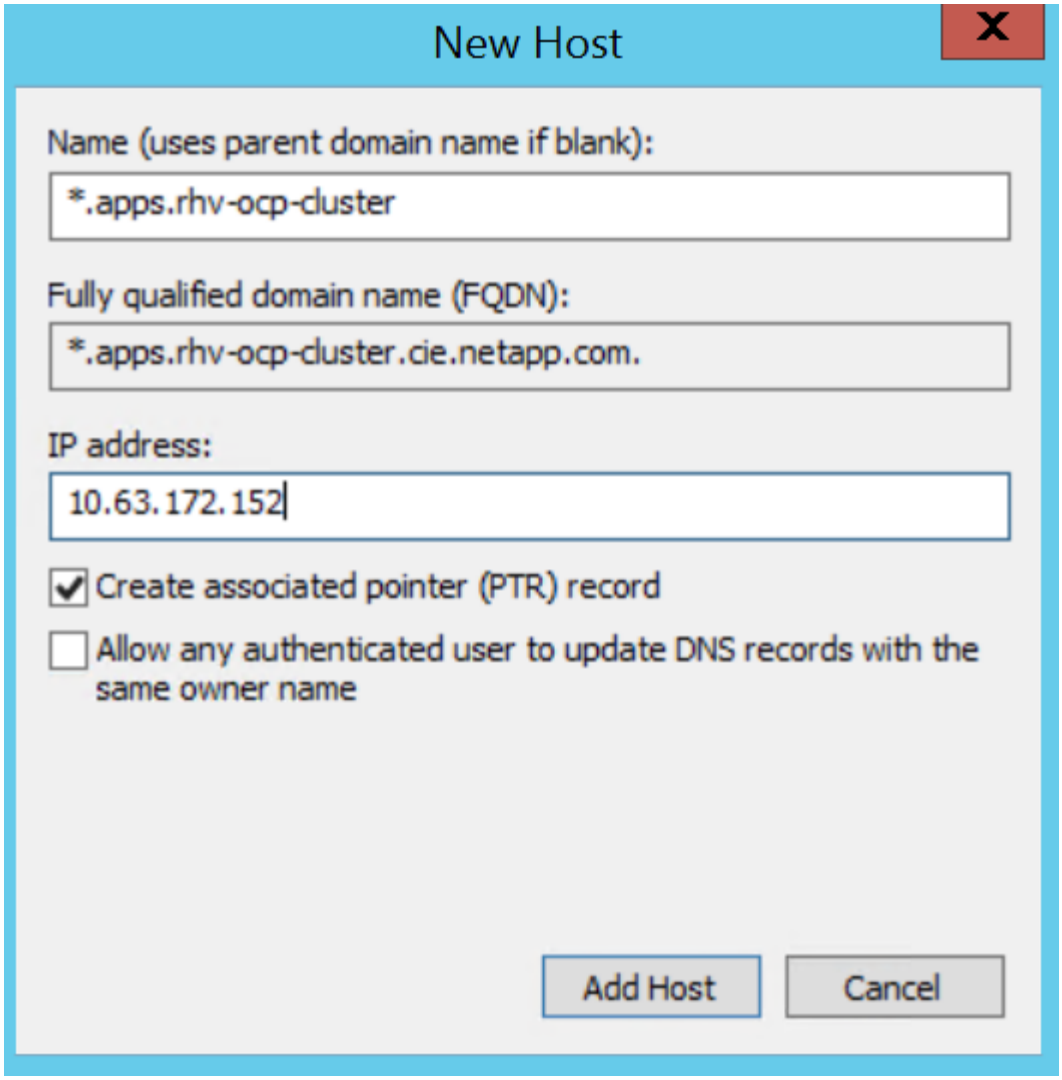
1. Reserve three static IP addresses from the network being used for OCP: the first IP address for OpenShift Container Platform REST API, the second IP address for pointing to the wildcard application ingress, and the third IP address for the internal DNS service. The first two IPs require an entry in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



Next: 5. Generate and Add SSH Private Key

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
```

Next: 6. Install OpenShift Container Platform

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.

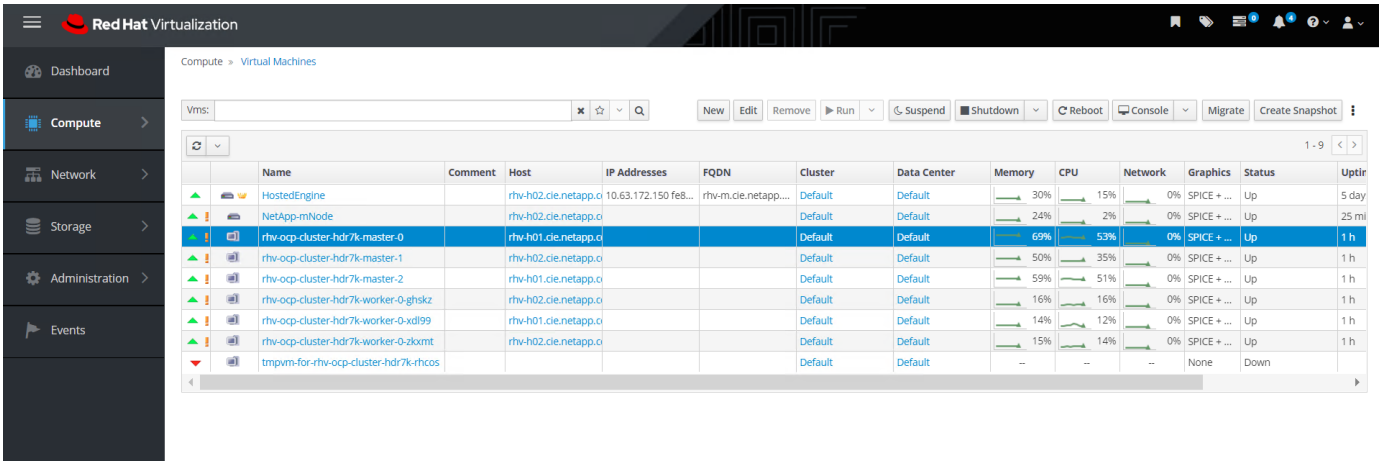
   It can take approximately 30–45 minutes to get the complete cluster up and running.

```
[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.



Next: 7. Access Console/Web Console

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
```

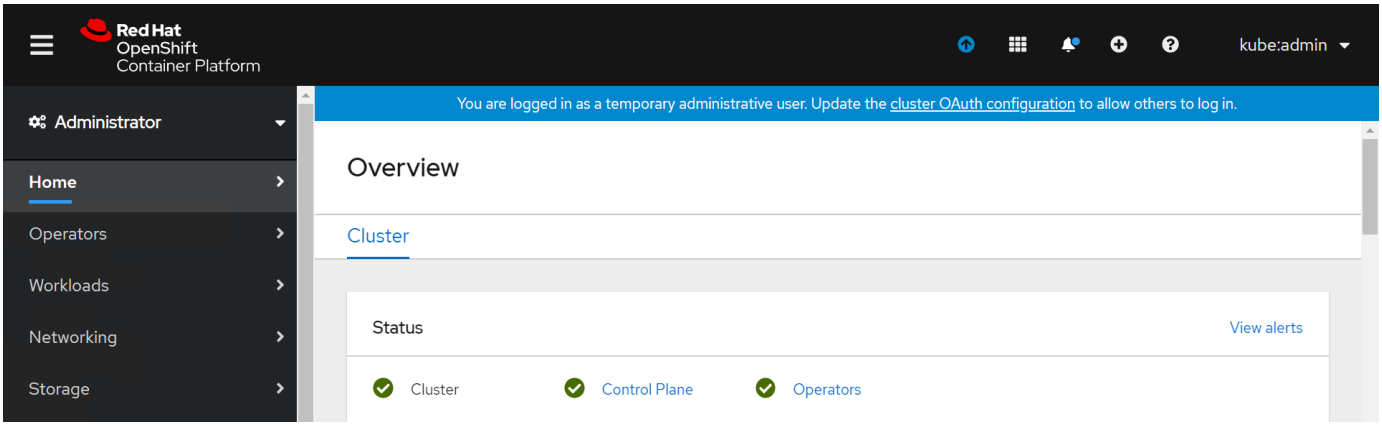
2. To interact with the cluster through the CLI, you can use the `kubeconfig` file provided by the IPI process located in the `/auth` directory inside the folder from where you launched the installation program. To easily interact with the cluster, export the file that is created in the directory. After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

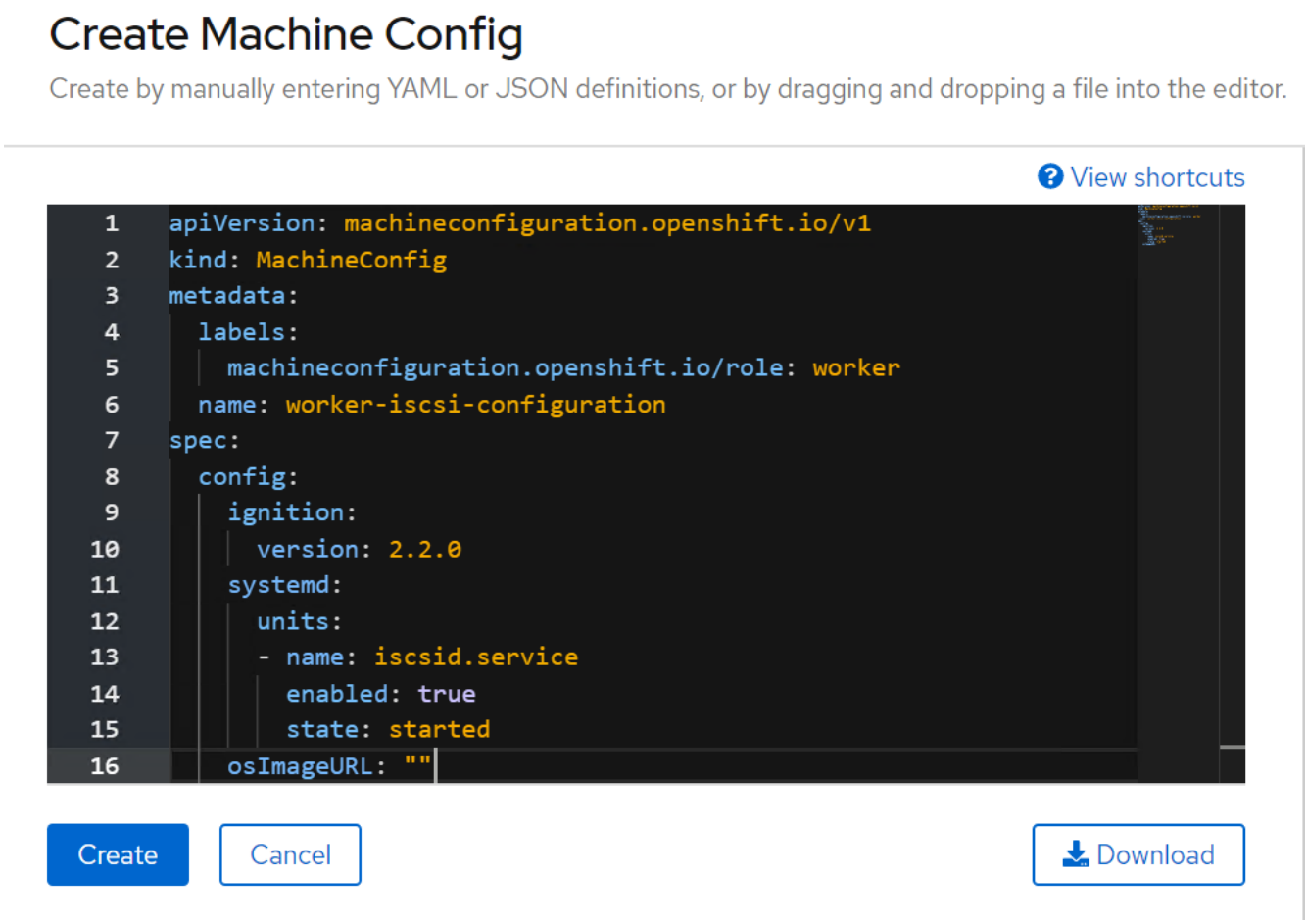


Next: 8. Configure Worker Nodes to Run Storage Services

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.

Next: 9. Download and Install NetApp Trident

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.

Next: Validation Results: NetApp HCI for Red Hat OpenShift on RHV

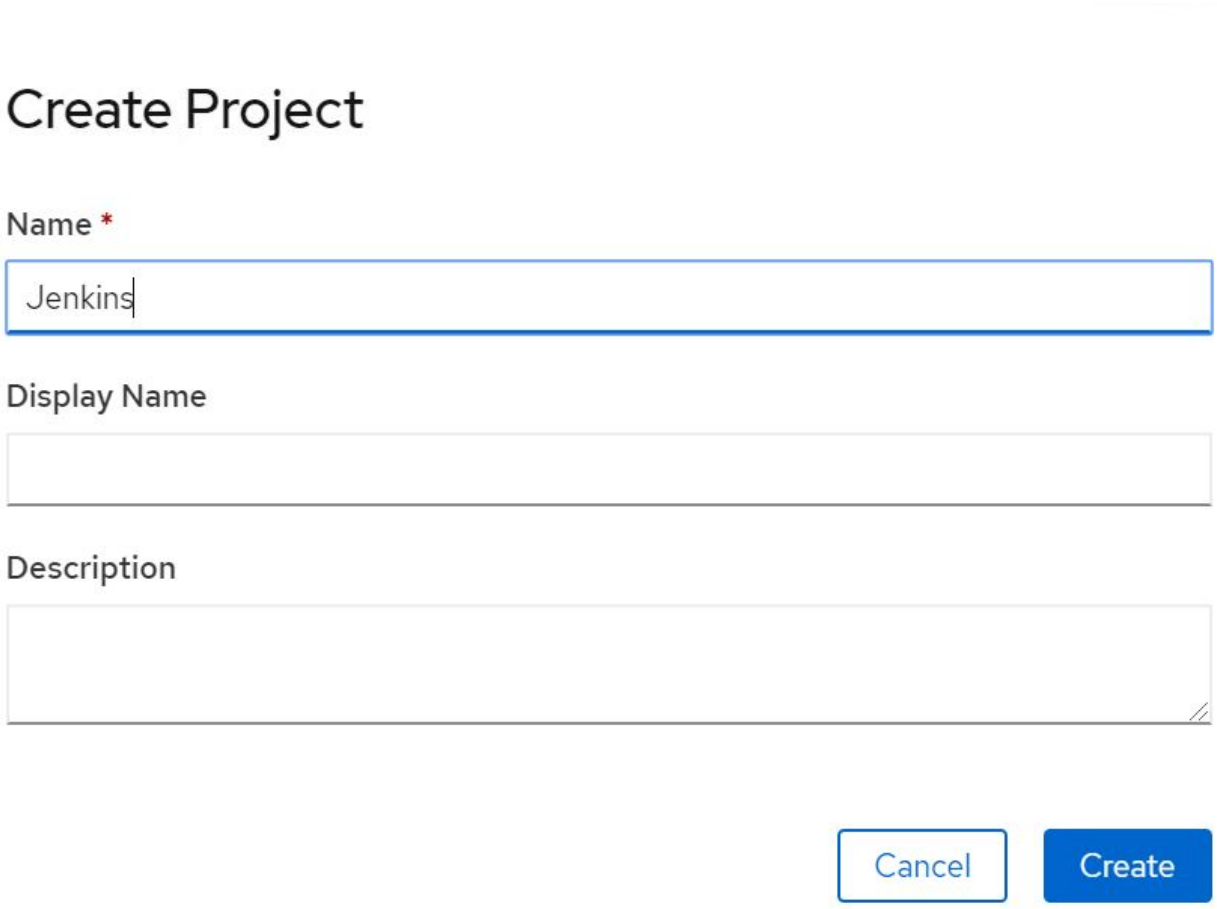
Validation Results: NetApp HCI for Red Hat OpenShift on RHV

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.

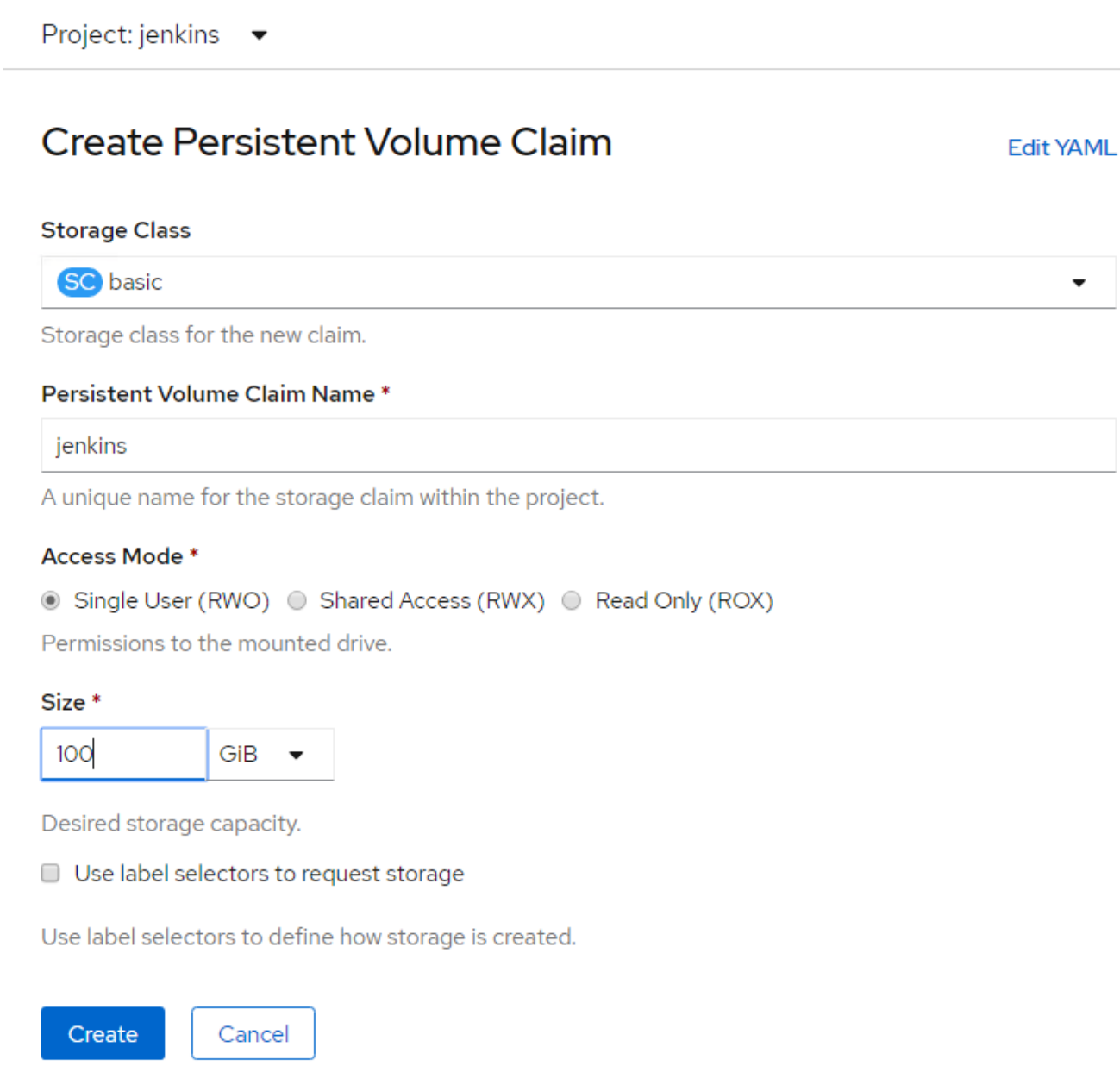
Create the Resources Required for Jenkins Deployment

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

1. Create a new project named Jenkins.



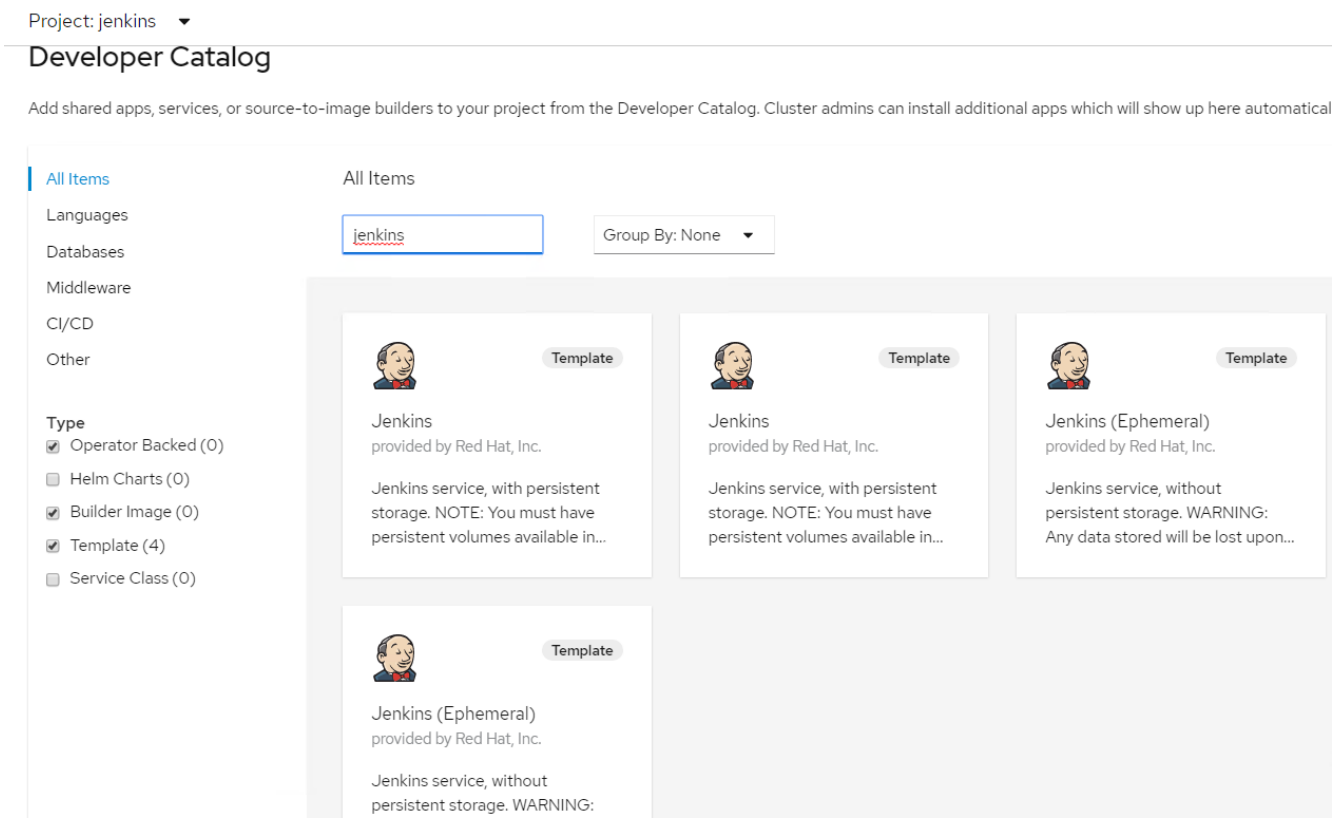
2. In this example, we deployed Jenkins with persistent storage. To support the Jenkins build, create the PVC. Navigate to Storage > Persistent Volume Claims and click Create Persistent Volume Claim. Select the storage class that was created, make sure that the Persistent Volume Claim Name is jenkins,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



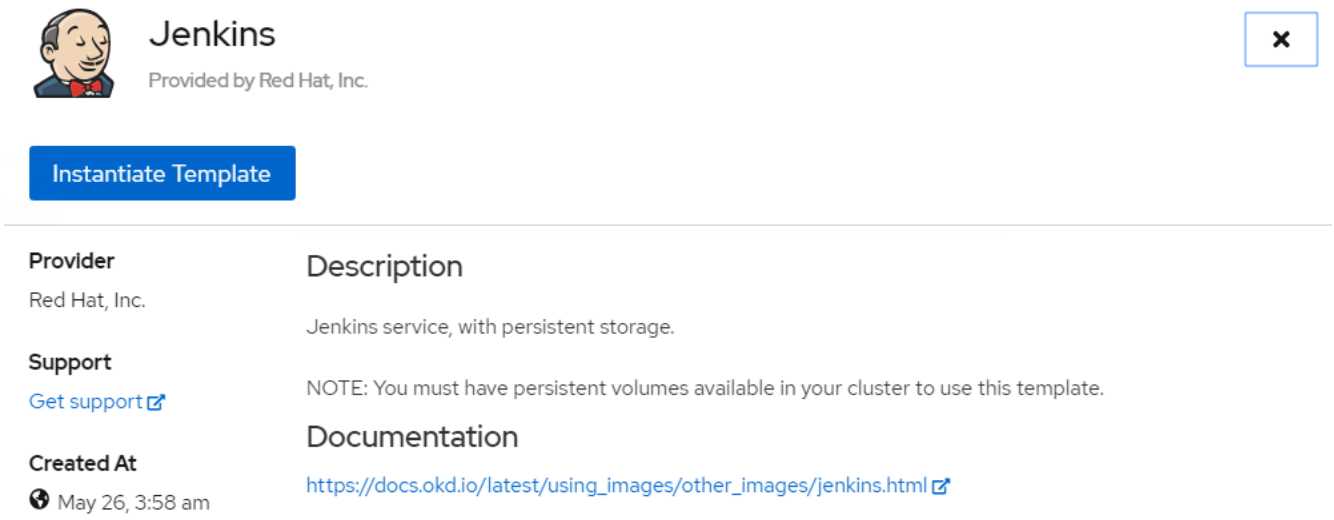
Deploy Jenkins with Persistent Storage

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.

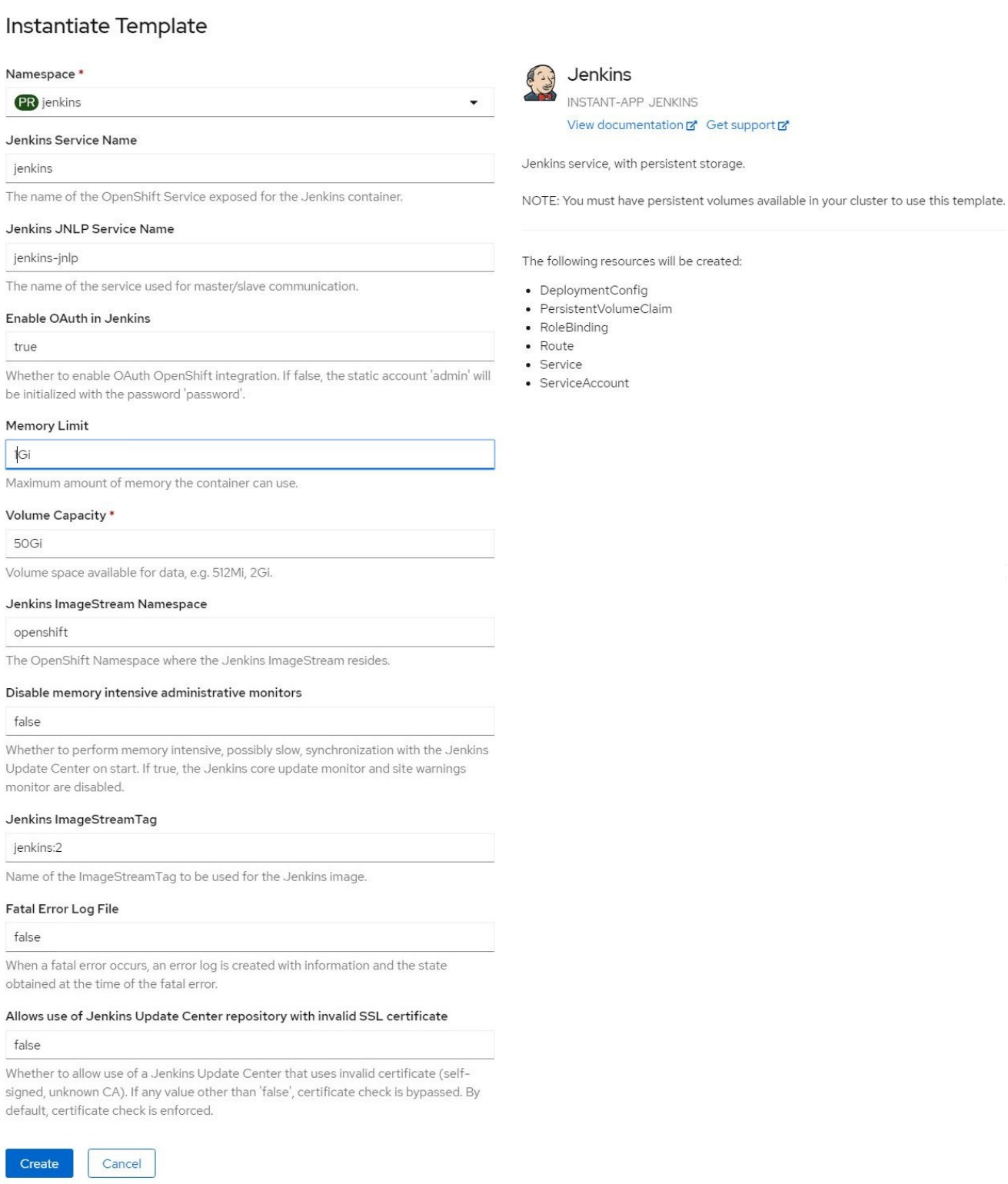


2. Click Instantiate Template.

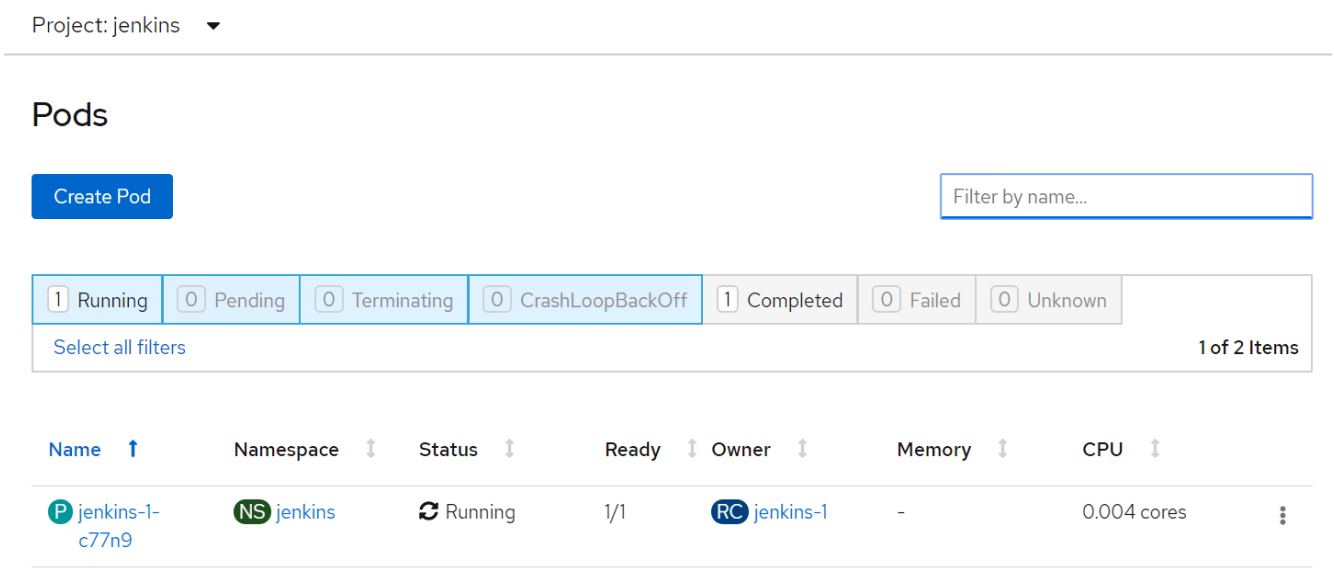


3. By default, the details for the Jenkins application are populated. Based on your requirements, modify the parameters, and click Create. This process creates all the required resources for supporting Jenkins on

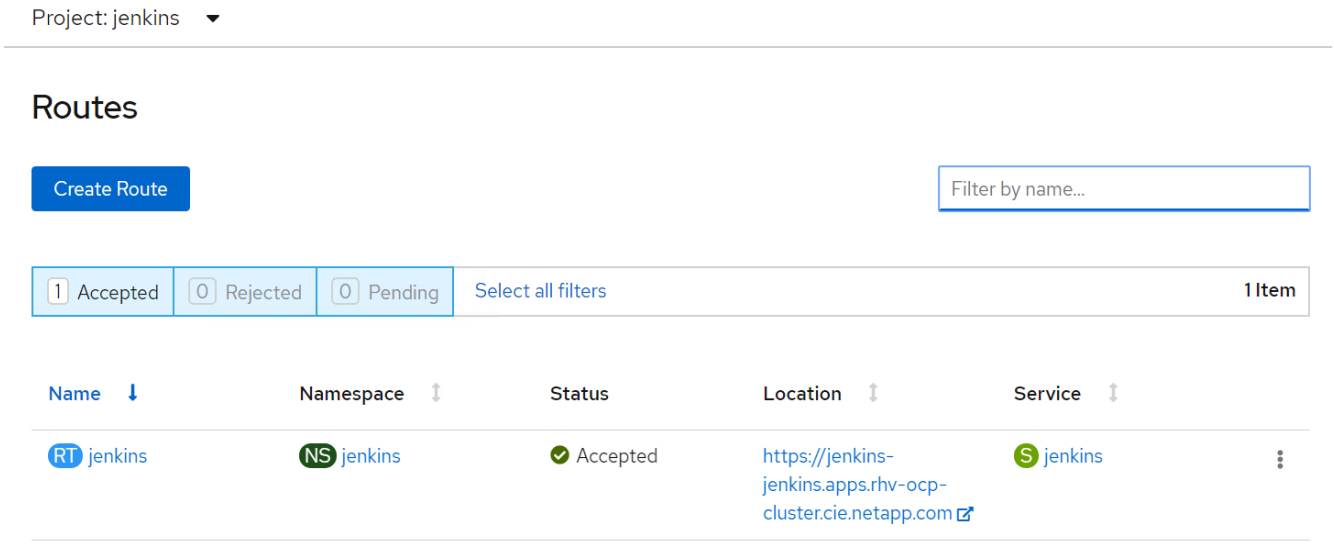
OpenShift.



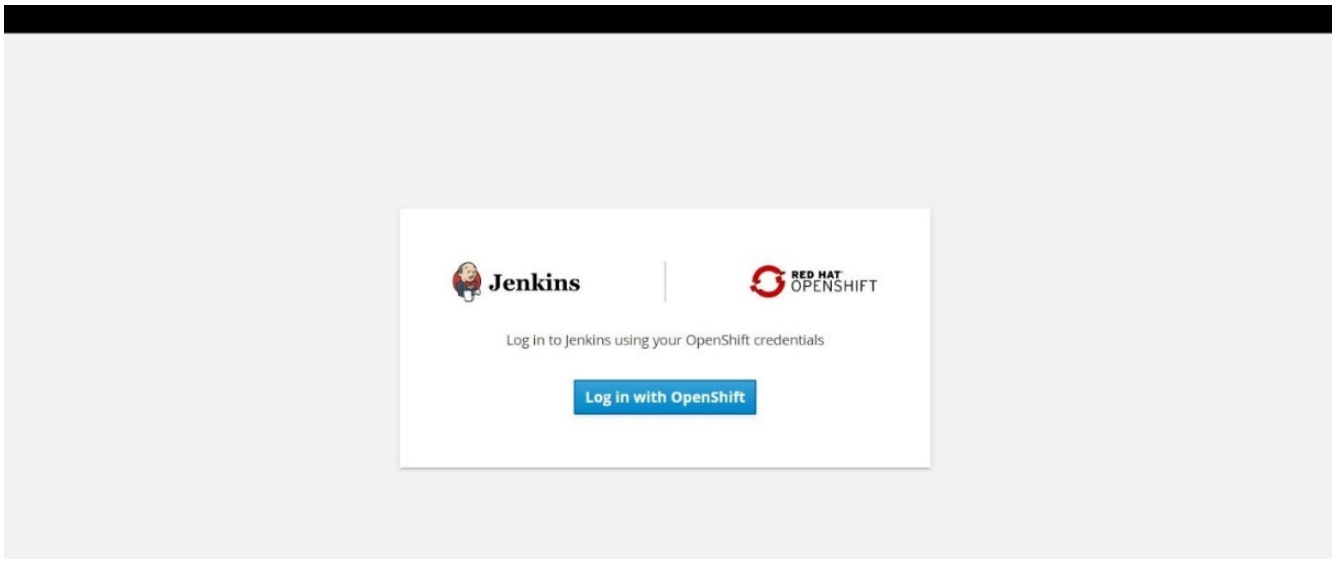
4. The Jenkins pods take approximately 10–12 minutes to enter the Ready state.



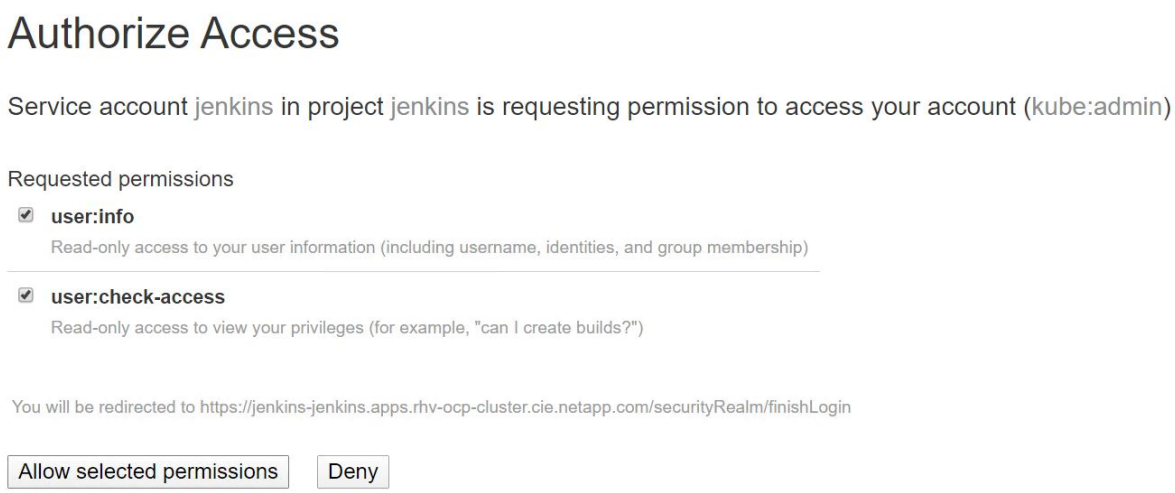
5. After the pods are instantiated, navigate to Networking > Routes. To open the Jenkins webpage, click the URL provided for the jenkins route.



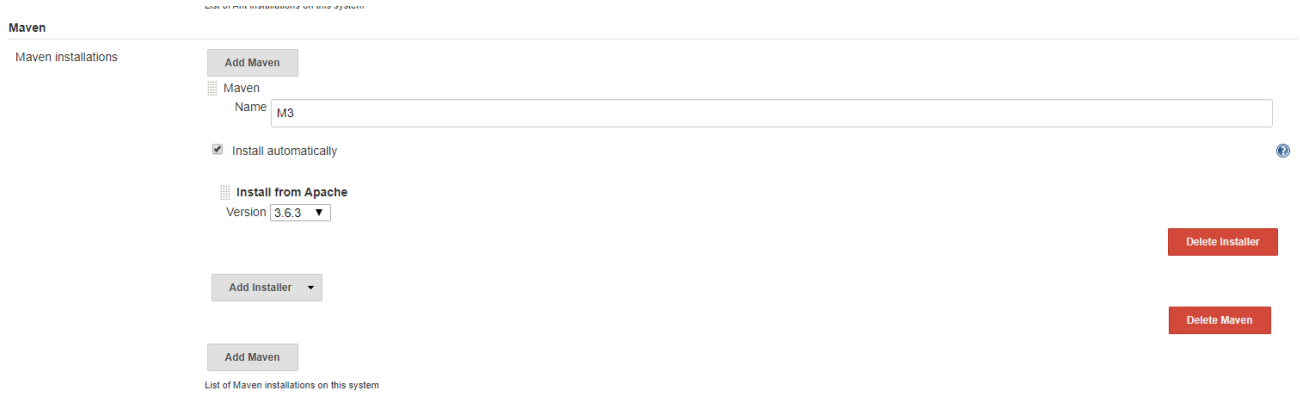
6. Because the OpenShift OAuth was used while creating the Jenkins app, click Log in with OpenShift.



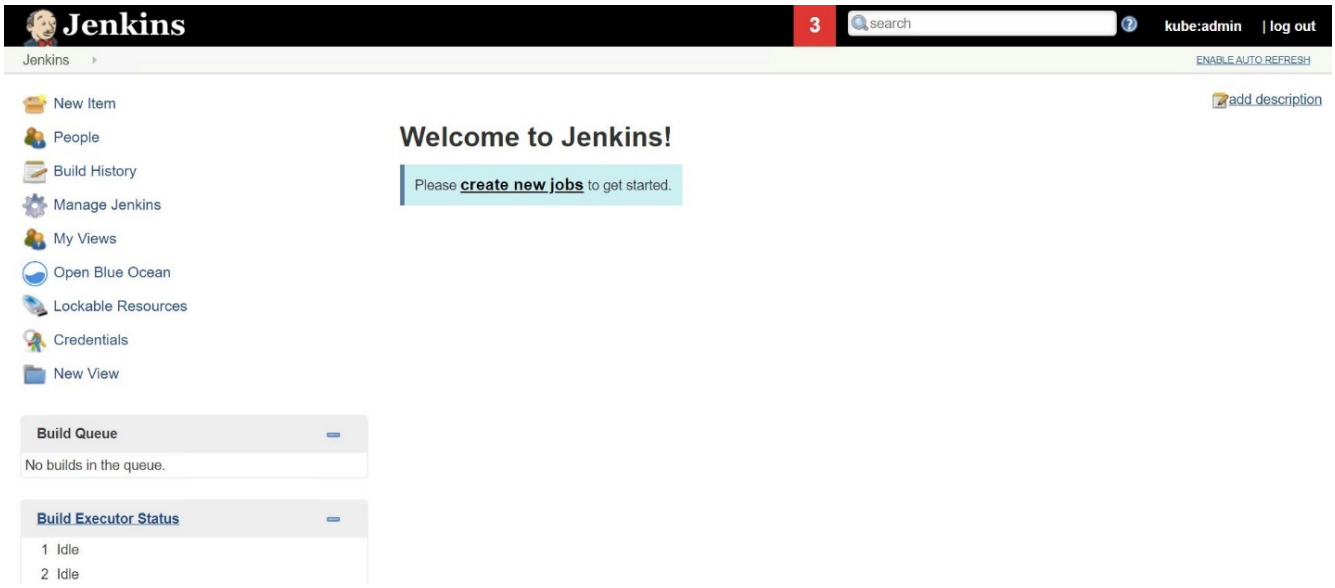
7. Authorize jenkins service-account to access the OpenShift users.



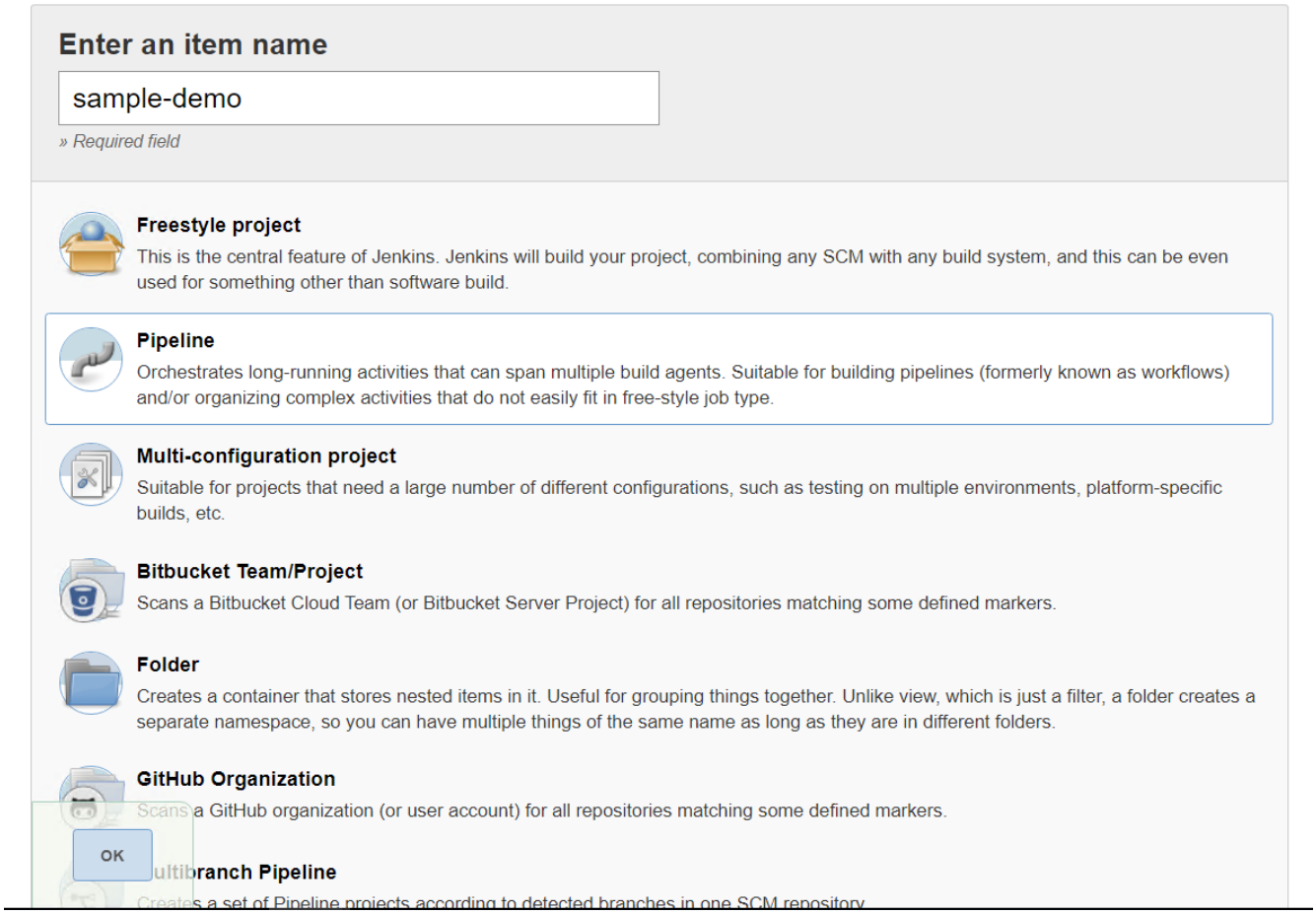
8. The Jenkins welcome page is displayed. Because we are using a Maven build, complete the Maven installation first. Navigate to Manage Jenkins > Global Tool Configuration, then in the Maven subhead, click Add Maven. Enter the name of your choice and make sure that the Install Automatically option is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



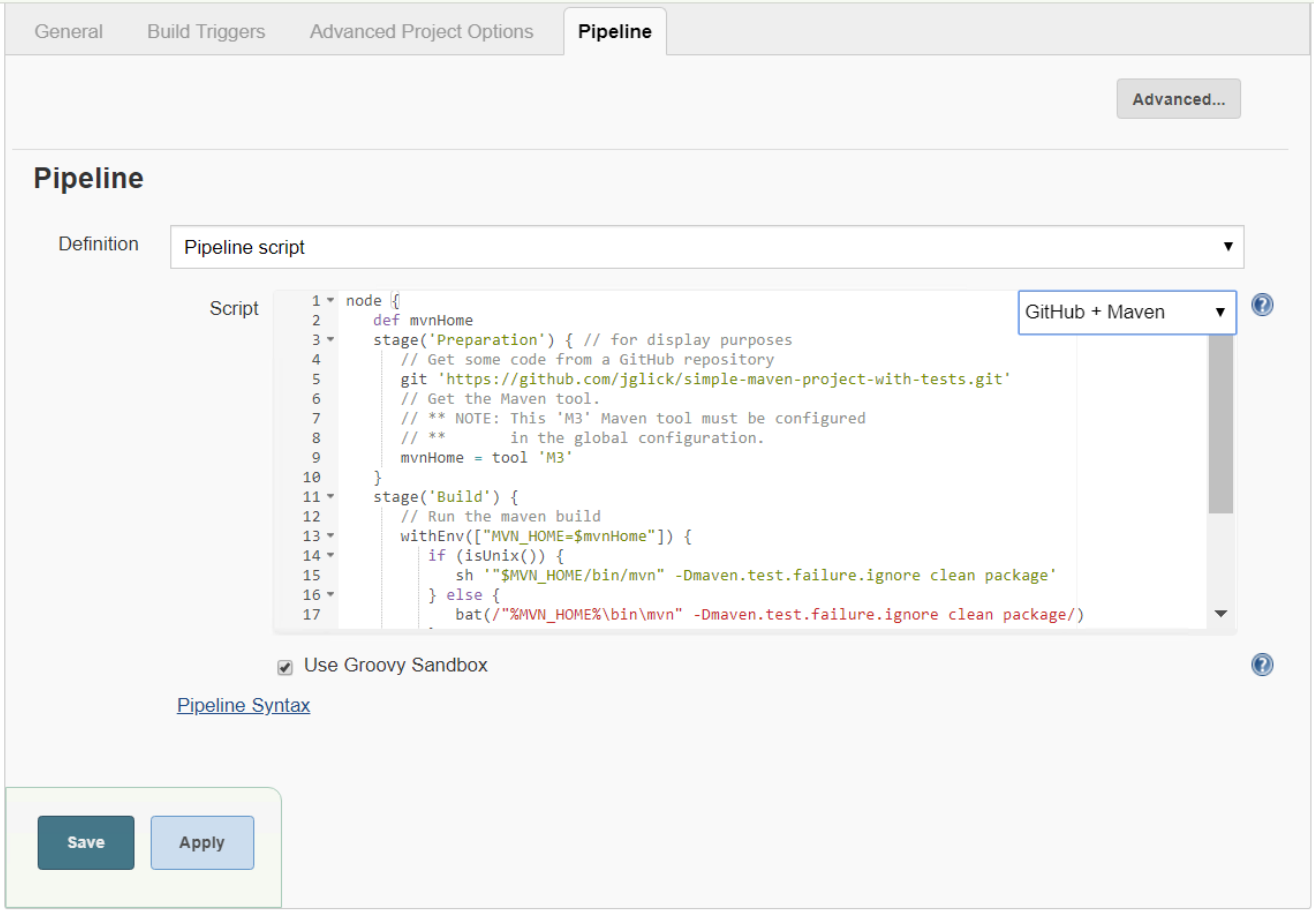
9. You can now create a pipeline to demonstrate the CI/CD workflow. On the home page, click Create New Jobs or New Item from the left- hand menu.



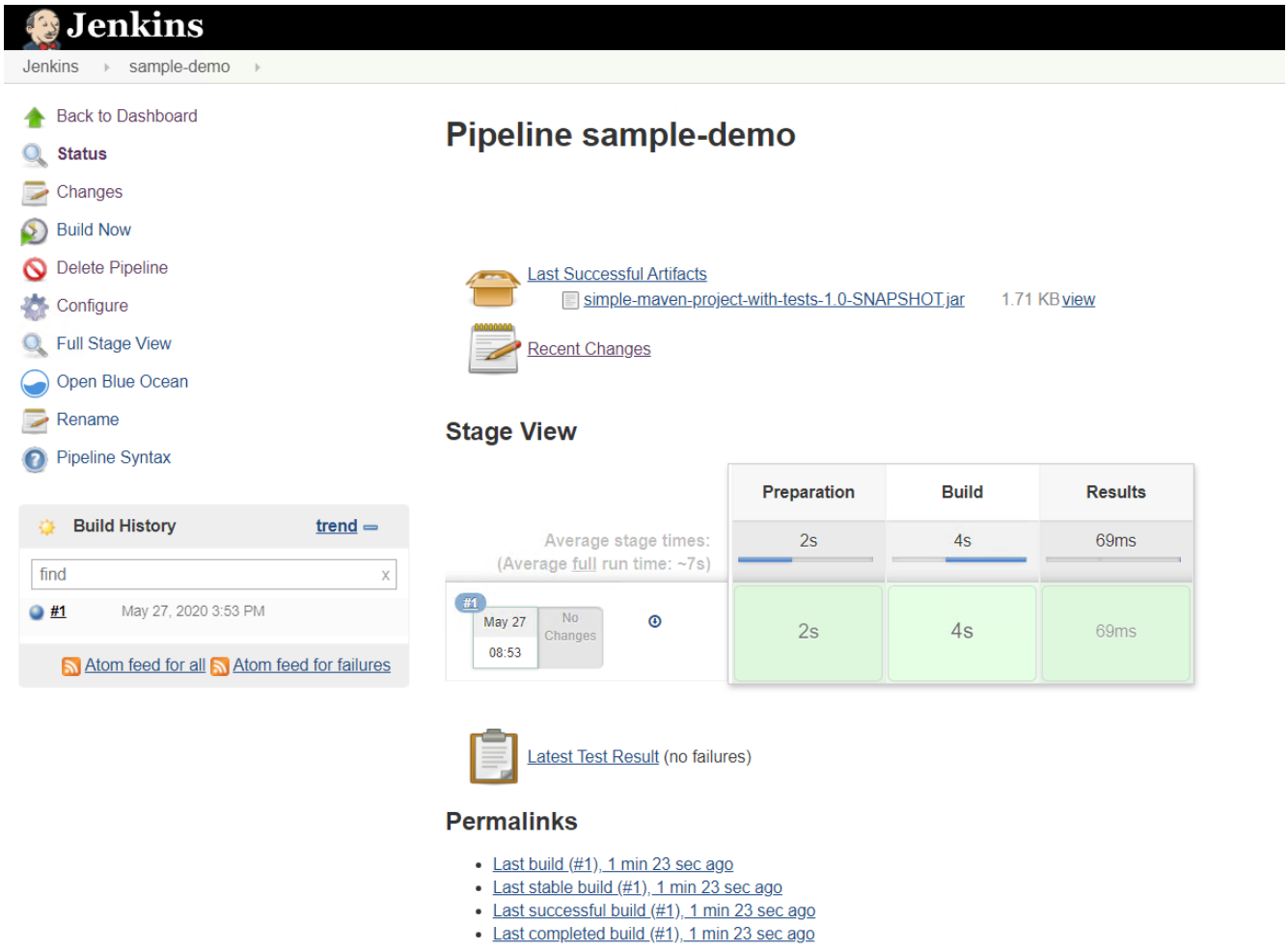
10. On the Create Item page, enter the name of your choice, select Pipeline, and click Ok.



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.

Next: Best Practices for Production Deployments

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.

Next: Videos and Demos: NetApp HCI for Red Hat OpenShift on Red Hat Virtualization

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*

Next: Additional Information: 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/

NVA-1160: Red Hat OpenShift with NetApp

Alan Cowles and Nikhil M Kulkarni, NetApp

This reference document provides deployment validation of the Red Hat OpenShift solution, deployed through Installer Provisioned Infrastructure (IPI) in several different data center environments as validated by NetApp. It also details storage integration with NetApp storage systems by making use of the Astra Trident storage orchestrator for the management of persistent storage. Lastly, a number of solution validations and real world use cases are explored and documented.

Use cases

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

- Easy to deploy and manage Red Hat OpenShift deployed using IPI (Installer Provisioned Infrastructure) on bare metal, Red Hat OpenStack Platform, Red Hat Virtualization, and VMware vSphere.
- Combined power of enterprise container and virtualized workloads with Red Hat OpenShift deployed virtually on OSP, RHV, or vSphere, or on bare metal with OpenShift Virtualization.
- Real world configuration and use cases highlighting the features of Red Hat OpenShift when used with NetApp storage and Astra Trident, the open source storage orchestrator for Kubernetes.

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
- Non-disruptive 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

Red Hat OpenShift with NetApp acknowledges these challenges and presents a solution that helps address each concern by implementing the fully automated deployment of RedHat OpenShift IPI in the customer's choice of data center environment.

Technology overview

The Red Hat OpenShift with NetApp solution is comprised of the following major components:

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.

For more information visit the OpenShift website here.

NetApp storage systems

NetApp has several storage systems perfect for enterprise data centers and hybrid cloud deployments. The NetApp portfolio includes NetApp ONTAP, NetApp Element, and NetApp e-Series storage systems, all of which can provide persistent storage for containerized applications.

For more information visit the NetApp website here.

NetApp storage integrations

NetApp Astra Control Center offers a rich set of storage and application-aware data management services for stateful Kubernetes workloads, deployed in an on-prem environment and powered by trusted NetApp data protection technology.

For more information, visit the NetApp Astra website here.

Astra Trident is an open-source and fully-supported storage orchestrator for containers and Kubernetes distributions, including Red Hat OpenShift.

For more information, visit the Astra Trident website here.

Advanced configuration options

This section is dedicated to customizations that real world users would likely need to perform when deploying this solution into production, such as creating a dedicated private image registry or deploying custom load balancer instances.

Current support matrix for validated releases

Technology	Purpose	Software version
NetApp ONTAP	Storage	9.8, 9.9.1
NetApp Element	Storage	12.3
NetApp Astra Control Center	Application Aware Data Management	21.08.65
NetApp Astra Trident	Storage Orchestration	21.07.1
Red Hat OpenShift	Container orchestration	4.6 EUS, 4.7
Red Hat OpenStack Platform	Private Cloud Infrastructure	16.1
Red Hat Virtualization	Data center virtualization	4.4
VMware vSphere	Data center virtualization	6.7U3

Next: Red Hat OpenShift Overview.

OpenShift Overview

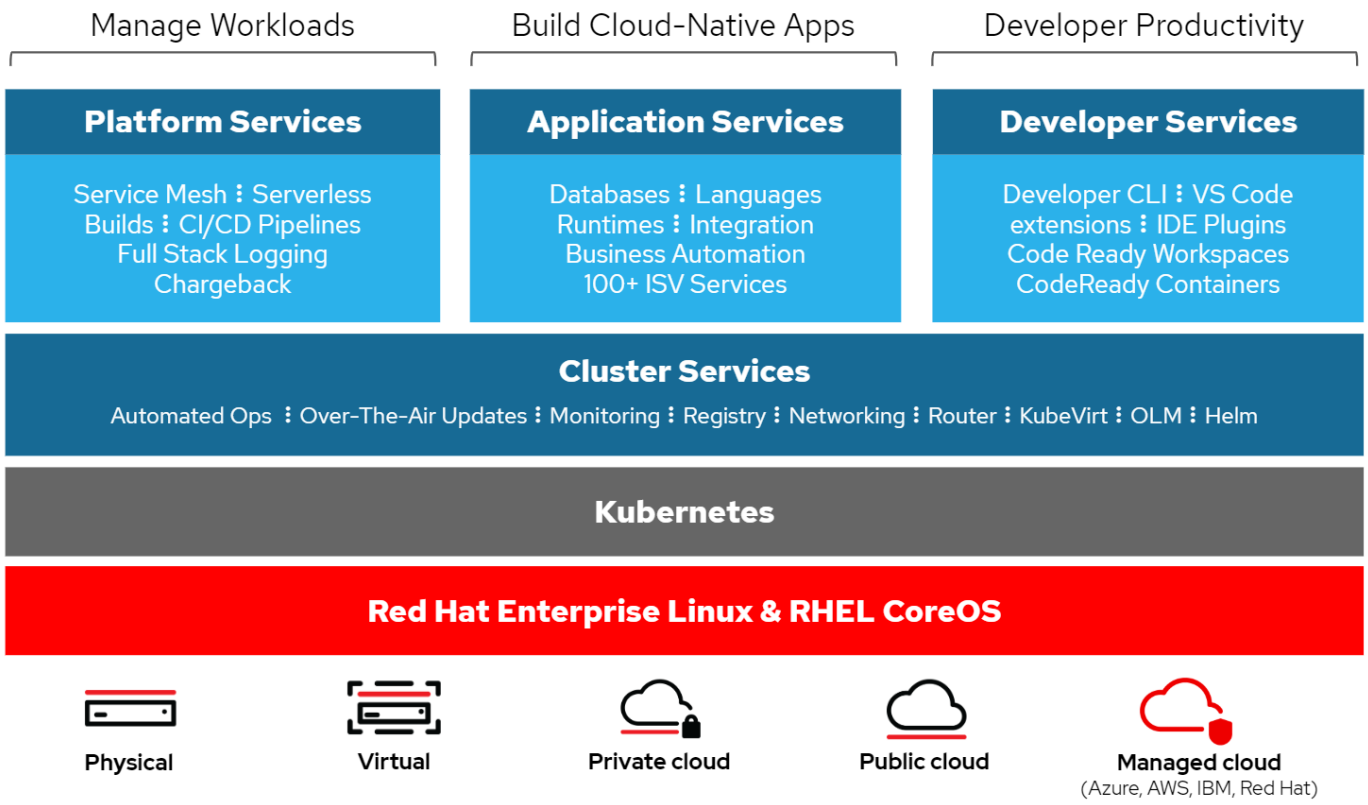
The Red Hat OpenShift Container Platform unites development and IT operations on a single platform to build, deploy, and manage applications consistently across on-premises and hybrid cloud infrastructures. Red Hat OpenShift is built on open-source innovation and industry standards, including Kubernetes and Red Hat Enterprise Linux CoreOS, the world's leading enterprise Linux distribution designed for container-based workloads. OpenShift is part of the Cloud Native Computing Foundation (CNCF) Certified Kubernetes program, providing portability and interoperability of container workloads.

Red Hat OpenShift provides the following capabilities:

- **Self-service provisioning.** Developers can quickly and easily create applications on demand from the tools that they use most, while operations retain full control over the entire environment.
- **Persistent storage.** By providing support for persistent storage, OpenShift Container Platform allows you

to run both stateful applications and cloud-native stateless applications.

- **Continuous integration and continuous development (CI/CD).** This source-code platform manages build and deployment images at scale.
- **Open-source standards.** These standards incorporate the Open Container Initiative (OCI) and Kubernetes for container orchestration, in addition to other open-source technologies. You are not restricted to the technology or to the business roadmap of a specific vendor.
- **CI/CD pipelines.** OpenShift provides out-of-the-box support for CI/CD pipelines so that development teams can automate every step of the application delivery process and make sure it's executed on every change that is made to the code or configuration of the application.
- **Role-Based Access Control (RBAC).** This feature provides team and user tracking to help organize a large developer group.
- **Automated build and deploy.** OpenShift gives developers the option to build their containerized applications or have the platform build the containers from the application source code or even the binaries. The platform then automates deployment of these applications across the infrastructure based on the characteristic that was defined for the applications. For example, how quantity of resources that should be allocated and where on the infrastructure they should be deployed in order for them to be compliant with third-party licenses.
- **Consistent environments.** OpenShift makes sure that the environment provisioned for developers and across the lifecycle of the application is consistent from the operating system, to libraries, runtime version (for example, Java runtime), and even the application runtime in use (for example, tomcat) in order to remove the risks originated from inconsistent environments.
- **Configuration management.** Configuration and sensitive data management is built in to the platform to make sure that a consistent and environment agnostic application configuration is provided to the application no matter which technologies are used to build the application or which environment it is deployed.
- **Application logs and metrics.** Rapid feedback is an important aspect of application development. OpenShift integrated monitoring and log management provides immediate metrics back to developers in order for them to study how the application is behaving across changes and be able 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 (seccomp) to isolate and protect containers. It also provides encryption through TLS certificates for the various subsystems and access to Red Hat certified containers (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.



Deployment methods for Red Hat OpenShift

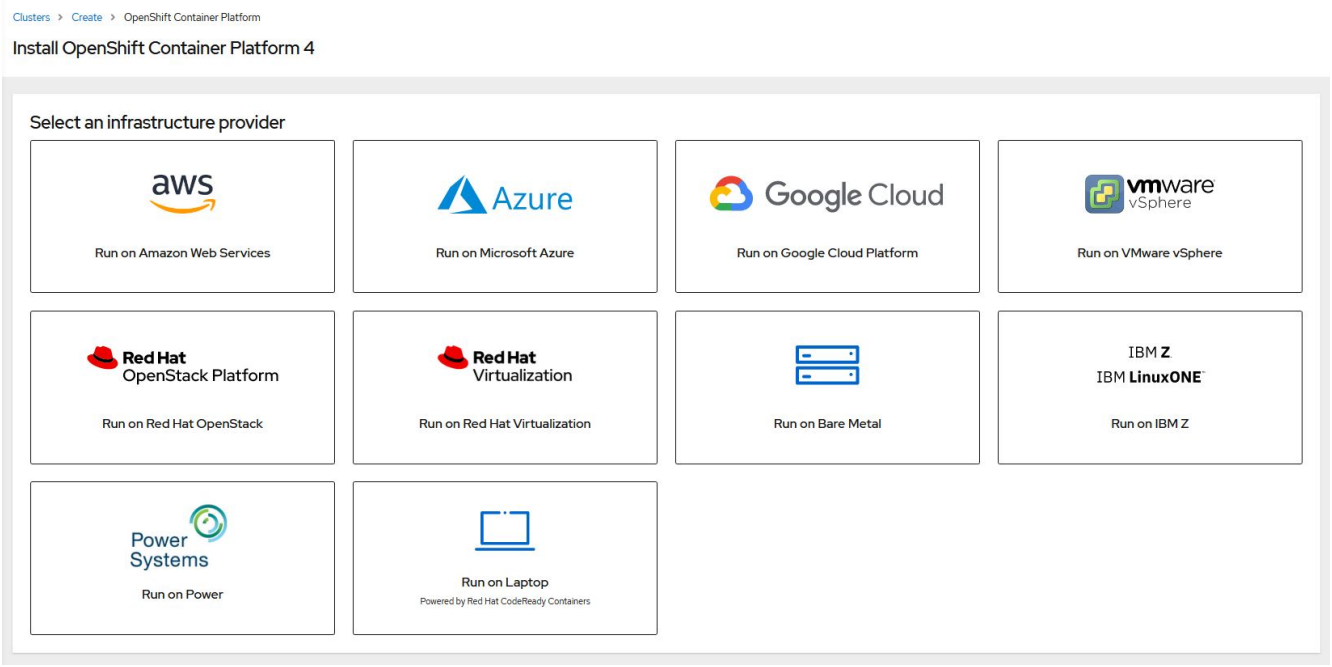
Starting with Red Hat OpenShift 4, the deployment methods for OpenShift include manual deployments using User Provisioned Infrastructure (UPI) for highly customized deployments or fully automated deployments using Installer Provisioned Infrastructure (IPI).

The IPI installation method is the preferred method in most cases because it allows for the rapid deployment of OCP clusters for dev, test, and production environments.

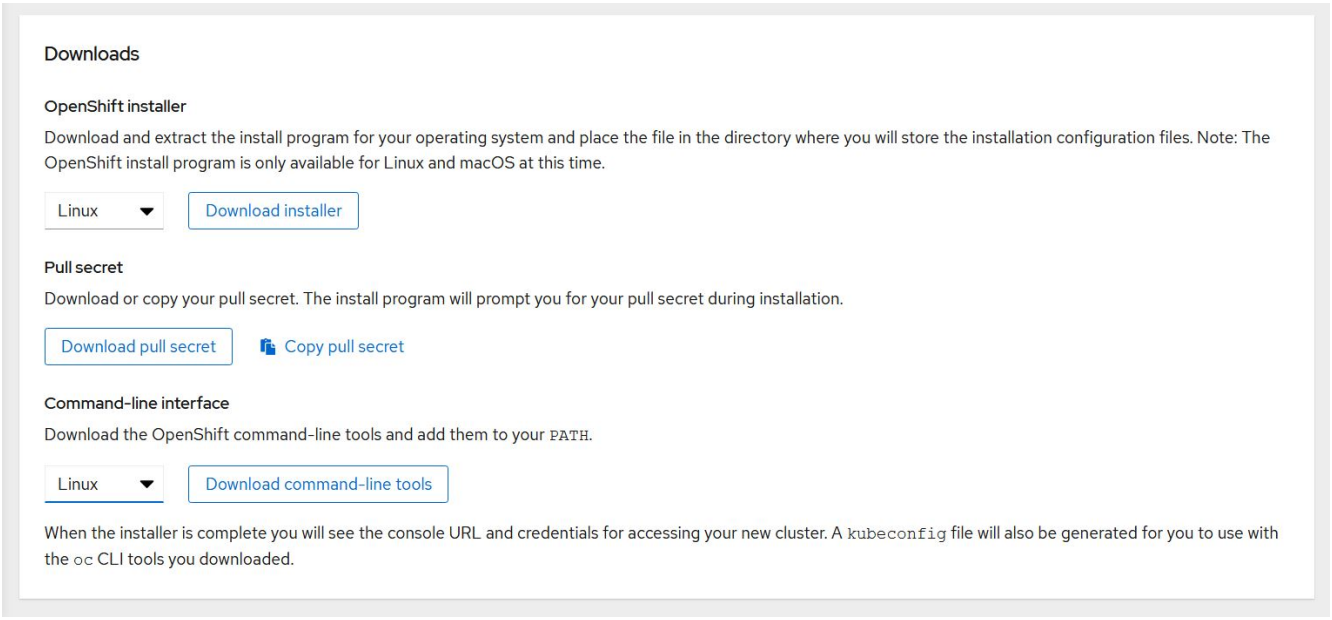
IPI installation of Red Hat OpenShift

The Installer Provisioned Infrastructure (IPI) deployment of OpenShift involves these high-level steps:

1. Visit the Red Hat OpenShift website and login with your SSO credentials.
2. Select the environment that you would like to deploy Red Hat OpenShift into.



3. On the next screen download the installer, the unique pull secret, and the CLI tools for management.



4. Follow the installation instructions provided by Red Hat to deploy to your environment of choice.

NetApp validated OpenShift deployments

NetApp has tested and validated the deployment of Red Hat OpenShift in its labs using the Installer Provisioned Infrastructure (IPI) deployment method in each of the following data center environments:

- OpenShift on Bare Metal
- OpenShift on Red Hat OpenStack Platform
- OpenShift on Red Hat Virtualization
- OpenShift on VMware vSphere

Next: NetApp Storage Overview.

OpenShift on Bare Metal

OpenShift on Bare Metal provides an automated deployment of the OpenShift Container Platform on commodity servers.

OpenShift on Bare Metal is similar to virtual deployments of OpenShift, which provide ease of deployment, rapid provisioning, and scaling of OpenShift clusters, while supporting virtualized workloads for applications that are not ready to be containerized. By deploying on bare metal, you do not require the extra overhead necessary to manage the host hypervisor environment in addition to the OpenShift environment. By deploying directly on bare metal servers, you can also reduce the physical overhead limitations of having to share resources between the host and OpenShift environment.

OpenShift on Bare Metal provides the following features:

- **IPI or assisted installer deployment.** With an OpenShift cluster deployed by Installer Provisioned Infrastructure (IPI) on bare metal servers, customers can deploy a highly versatile, easily scalable OpenShift environment directly on commodity servers, without the need to manage a hypervisor layer.
- **Compact cluster design.** To minimize the hardware requirements, OpenShift on bare metal allows for users to deploy clusters of just 3 nodes, by enabling the OpenShift control plane nodes to also act as worker nodes and host containers.
- **OpenShift virtualization.** OpenShift can run virtual machines within containers by using OpenShift Virtualization. This container-native virtualization runs the KVM hypervisor inside of a container, and attaches persistent volumes for VM storage.
- **AI/ML-optimized infrastructure.** Deploy applications like Kubeflow for machine learning applications by incorporating GPU-based worker nodes to your OpenShift environment and leveraging OpenShift Advanced Scheduling.

Network design

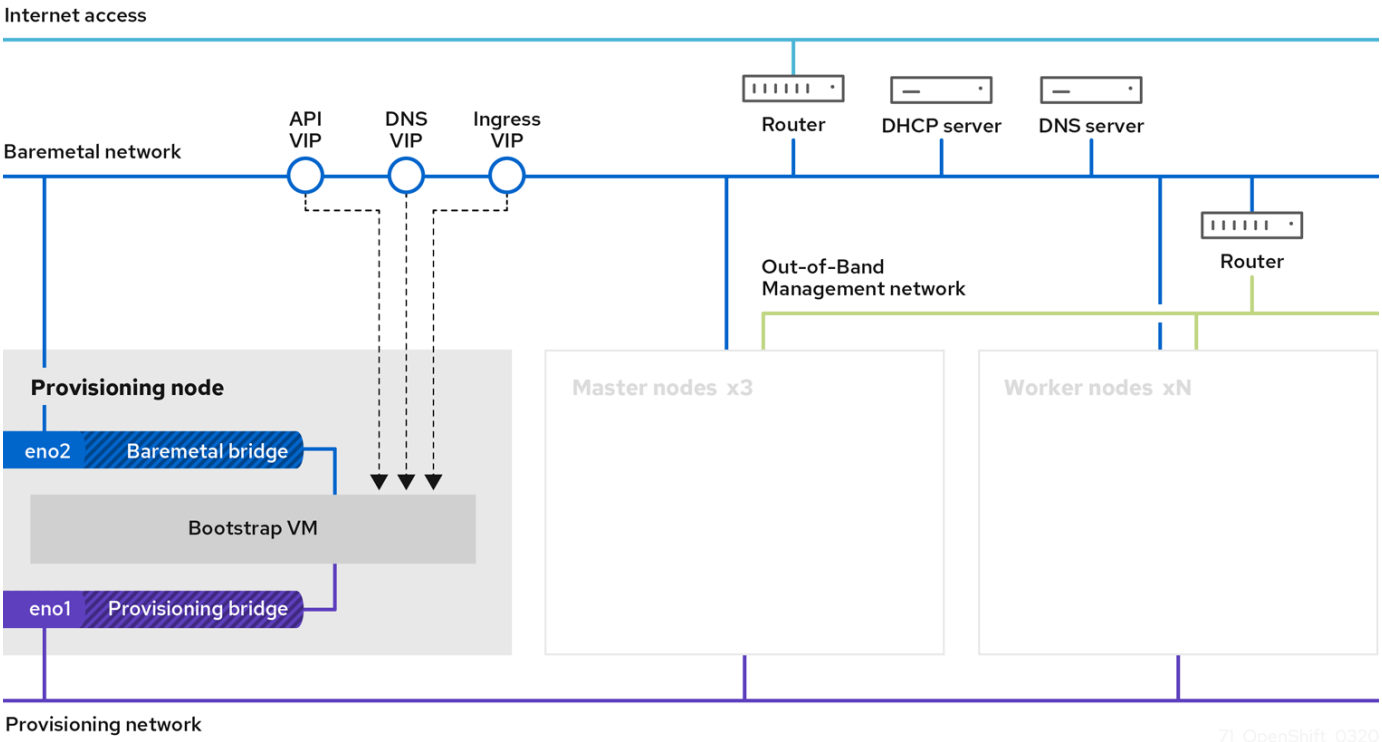
The Red Hat OpenShift on NetApp solution uses two data switches to provide primary data connectivity at 25Gbps. It also uses two management switches that provide connectivity at 1Gbps for in-band management for the storage nodes and out-of-band management for IPMI functionality.

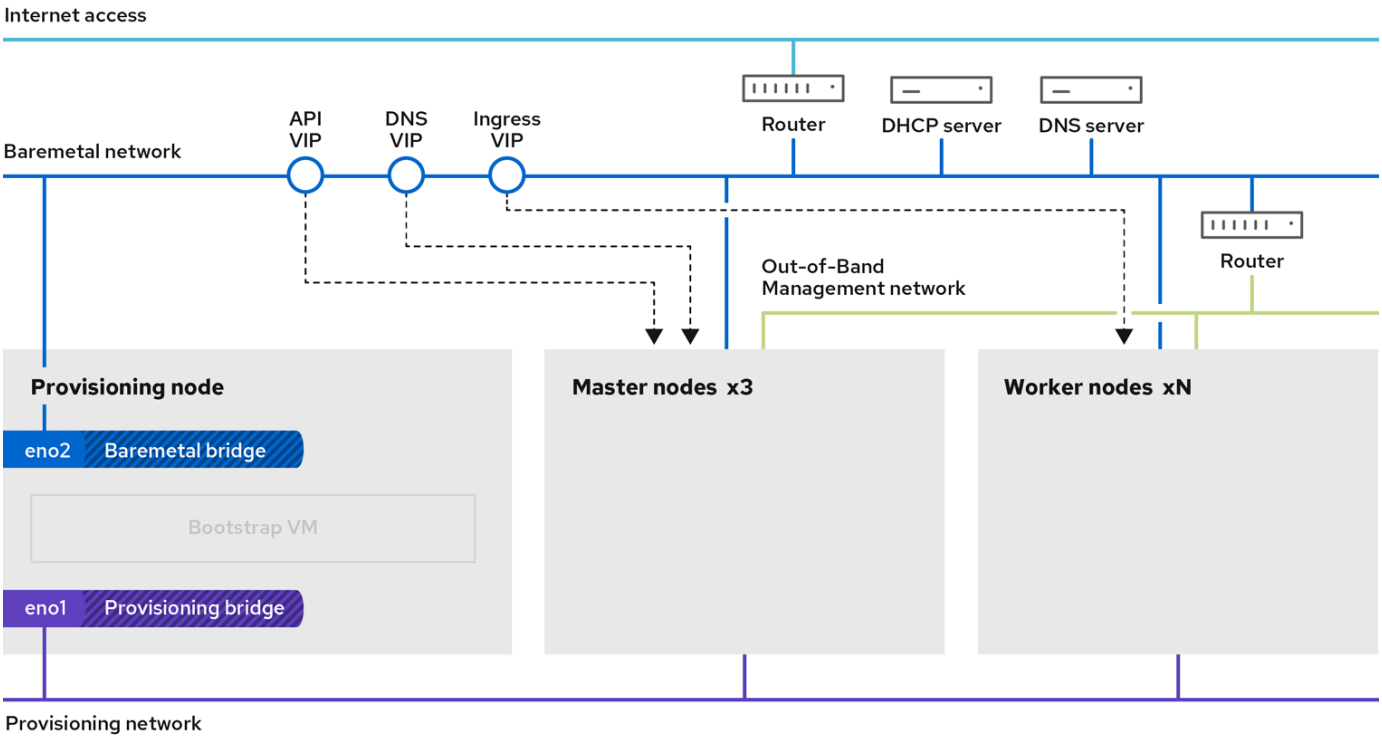
For OpenShift bare-metal IPI deployment, you must create a provisioner node, a Red Hat Enterprise Linux 8 machine that must have network interfaces attached to separate networks.

- **Provisioning network.** This network is used to boot the bare-metal nodes and install the necessary images and packages to deploy the OpenShift cluster.
- **Bare-metal network.** This network is used for public-facing communication of the cluster after it is deployed.

For the setup of the provisioner node, the customer creates bridge interfaces that allow the traffic to route properly on the node itself and on the Bootstrap VM that is provisioned for deployment purposes. After the cluster is deployed, the API and ingress VIP addresses are migrated from the bootstrap node to the newly deployed cluster.

The following images depict the environment both during IPI deployment and after the deployment is complete.





VLAN requirements

The Red Hat OpenShift with NetApp solution is designed to logically separate network traffic for different purposes by using virtual local area networks (VLANs).

VLANs	Purpose	VLAN ID
Out-of-band management network	Management for bare metal nodes and IPMI	16
Bare-metal network	Network for OpenShift services once cluster is available	181
Provisioning network	Network for PXE boot and installation of bare metal nodes via IPI	3485

Although each of these networks is virtually separated by VLANs, each physical port must be set up in Access Mode with the primary VLAN assigned, because there is no way to pass a VLAN tag during a PXE boot sequence.

Network infrastructure support resources

The following infrastructure should be in place prior to the deployment of the OpenShift container platform:

- At least one DNS server that provides a full host-name resolution 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.

Next: NetApp storage overview.

OpenShift on Red Hat OpenStack Platform

The Red Hat OpenStack Platform delivers an integrated foundation to create, deploy, and scale a secure and reliable private OpenStack cloud.

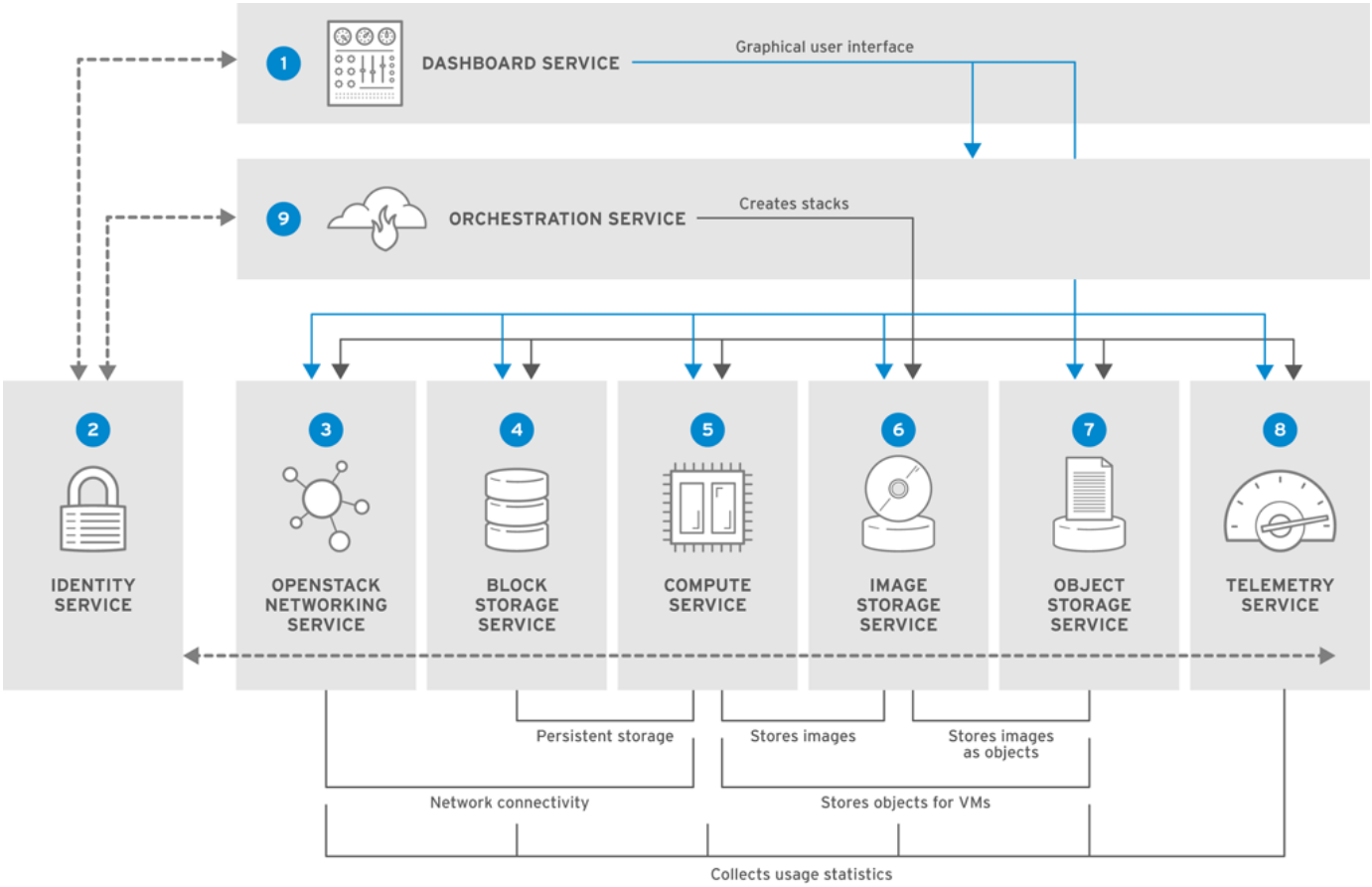
OSP is an infrastructure-as-a-service (IaaS) cloud implemented by a collection of control services that manage compute, storage, and networking resources. The environment is managed using a web-based interface that allows administrators and users to control, provision, and automate OpenStack resources. Additionally, the OpenStack infrastructure is facilitated through an extensive command line interface and API enabling full automation capabilities for administrators and end-users.

The OpenStack project is a rapidly developed community project that provides updated releases every six months. Initially Red Hat OpenStack Platform kept pace with this release cycle by publishing a new release along with every upstream release and providing long term support for every third release. Recently, with the OSP 16.0 release (based on OpenStack Train), Red Hat has chosen not to keep pace with release numbers but instead has backported new features into sub-releases. The most recent release is Red Hat OpenStack Platform 16.1, which includes backported advanced features from the Ussuri and Victoria releases upstream.

For more information about OSP see the Red Hat OpenStack Platform website.

OpenStack services

OpenStack Platform services are deployed as containers, which isolates services from one another and enables easy upgrades. The OpenStack Platform uses a set of containers built and managed with Kolla. The deployment of services is performed by pulling container images from the Red Hat Custom Portal. These service containers are managed using the Podman command and are deployed, configured, and maintained with Red Hat OpenStack Director.



Service	Project name	Description
Dashboard	Horizon	Web browser-based dashboard that you use to manage OpenStack services.
Identity	Keystone	Centralized service for authentication and authorization of OpenStack services and for managing users, projects, and roles.
OpenStack networking	Neutron	Provides connectivity between the interfaces of OpenStack services.
Block storage	Cinder	Manages persistent block storage volumes for virtual machines (VMs).
Compute	Nova	Manages and provisions VMs running on compute nodes.
Image	Glance	Registry service used to store resources such as VM images and volume snapshots.
Object storage	Swift	Allows users to storage and retrieve files and arbitrary data.

Telemetry	Ceilometer	Provides measurements of use of cloud resources.
Orchestration	Heat	Template-based orchestration engine that supports automatic creation of resource stacks.

Network design

The Red Hat OpenShift with NetApp 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.

IPMI functionality is required by Red Hat OpenStack Director to deploy Red Hat OpenStack Platform using the Ironic bare-metal provision service.

VLAN requirements

Red Hat OpenShift with NetApp is designed to logically separate network traffic for different purposes by using virtual local area networks (VLANs). 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 while validating the solution at NetApp.

VLANs	Purpose	VLAN ID
Out-of-band management network	Network used for management of physical nodes and IPMI service for Ironic.	16
Storage infrastructure	Network used for controller nodes to map volumes directly to support infrastructure services like Swift.	201
Storage Cinder	Network used to map and attach block volumes directly to virtual instances deployed in the environment.	202
Internal API	Network used for communication between the OpenStack services using API communication, RPC messages, and database communication.	301
Tenant	Neutron provides each tenant with their own networks via tunneling through VXLAN. Network traffic is isolated within each tenant network. Each tenant network has an IP subnet associated with it, and network namespaces mean that multiple tenant networks can use the same address range without causing conflicts.	302

VLANs	Purpose	VLAN ID
Storage management	OpenStack Object Storage (Swift) uses this network to synchronize data objects between participating replica nodes. The proxy service acts as the intermediary interface between user requests and the underlying storage layer. The proxy receives incoming requests and locates the necessary replica to retrieve the requested data.	303
PXE	The OpenStack Director provides PXE boot as a part of the Ironic bare metal provisioning service to orchestrate the installation of the OSP Overcloud.	3484
External	Publicly available network which hosts the OpenStack Dashboard (Horizon) for graphical management and allows for public API calls to manage OpenStack services.	3485
In-band management network	Provides access for system administration functions such as SSH access, DNS traffic, and Network Time Protocol (NTP) traffic. This network also acts as a gateway for non-controller nodes.	3486

Network infrastructure support resources

The following infrastructure should be in place prior to the deployment of the OpenShift Container Platform:

- At least one DNS server which provides a full host-name resolution.
- At least three NTP servers which can keep time synchronized for the servers in the solution.
- (Optional) Outbound internet connectivity for the OpenShift environment.

Best practices for production deployments

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

Deploy OpenShift to an OSP private cloud with at least three compute nodes

The verified architecture described in this document presents the minimum hardware deployment suitable for HA operations by deploying three OSP controller nodes and two OSP compute nodes. This architecture ensures a fault tolerant configuration in which both compute nodes can launch virtual instances and deployed VMs can migrate between the two hypervisors.

Because Red Hat OpenShift initially deploys with three master nodes, a two-node configuration might cause at least two masters to 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 to deploy at least three OSP compute nodes so that the OpenShift masters can be distributed evenly and the solution receives an added degree of fault tolerance.

Configure virtual machine/host affinity

Distributing 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. In the Red Hat OpenStack Platform, host affinity and anti-affinity rules can be created and enforced by creating server groups and configuring filters so that instances deployed by Nova in a server group deploy on different compute nodes.

A server group has a default maximum of 10 virtual instances that it can manage placement for. This can be modified by updating the default quotas for Nova.

There is a specific hard affinity/anti-affinity limit for OSP server groups; if there not enough resources to deploy on separate nodes or not enough resources to allow sharing of nodes, the VM fails to boot.

To configure affinity groups, see How do I configure Affinity and Anti-Affinity for OpenStack instances?.

Use a custom install file for OpenShift deployment

IPI makes the deployment of OpenShift clusters easy through the interactive wizard discussed earlier in this document. However, it is possible that you might need to change some default values as a part of a cluster deployment.

In these instances, you can run and task the wizardwithout immediately deploying a cluster; instead it creates a configuration file from which the cluster can be deployed later. This is very useful if you need to change any IPI defaults, or if you want to deploy multiple identical clusters in your 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 OpenStack with Customizations.

Next: NetApp Storage Overview.

OpenShift on Red Hat Virtualization

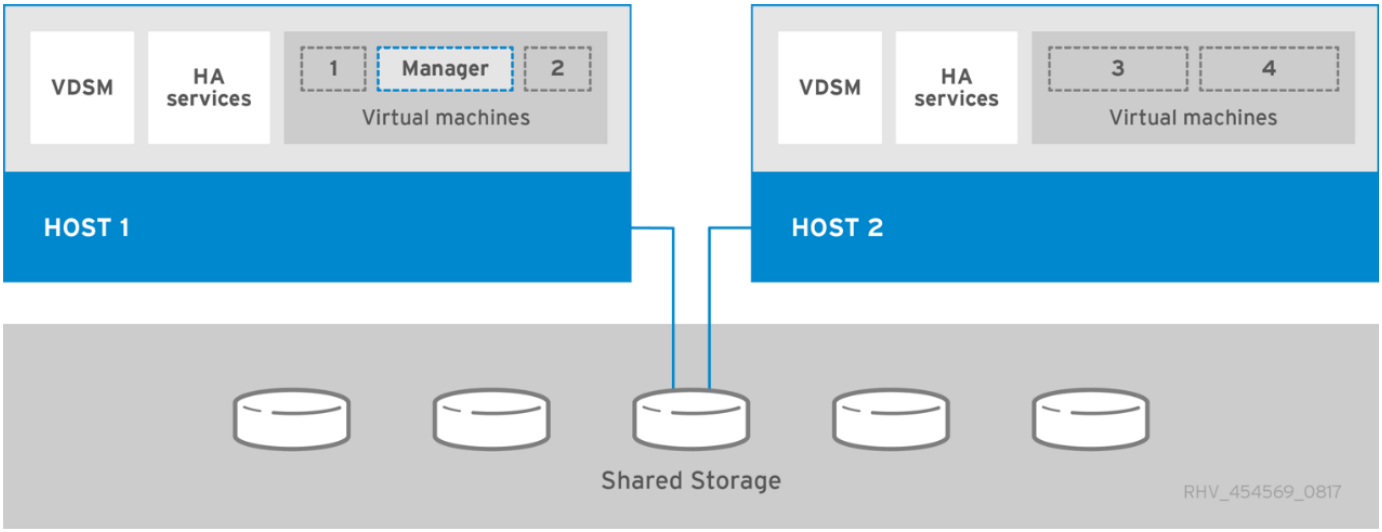
Red Hat Virtualization (RHV) is an enterprise virtual data center platform that runs on Red Hat Enterprise Linux (RHEL) and uses the KVM hypervisor.

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 hardware requirements, RHV allows RHV Manager (RHV-M) to be deployed as a VM on the same hosts that run guest VMs.

- **High availability.** To avoid disruption in event of host failures, 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 host 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.



Network design

The Red Hat OpenShift on NetApp 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 of the storage nodes and out-of-band management for IPMI functionality. OCP uses the virtual machine logical network on RHV for cluster management. This section describes the arrangement and purpose of each virtual network segment used in the solution and outlines the prerequisites for deploying the solution.

VLAN requirements

Red Hat OpenShift on RHV is designed to logically separate network traffic for different purposes by using virtual local area networks (VLANs). 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 while validating the solution at NetApp.

VLANs	Purpose	VLAN ID
Out-of-band management network	Management for physical nodes and IPMI	16
VM Network	Virtual guest network access	1172
In-band management network	Management for RHV-H nodes, RHV-Manager, and ovirtmgmt network	3343
Storage network	Storage network for NetApp Element iSCSI	3344

VLANs	Purpose	VLAN ID
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:

- At least one DNS server providing 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.

Best practices for production deployments

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

You can distribute the OpenShift masters across multiple hypervisor nodes 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 or negative affinity strictly without any regards to external conditions. Soft enforcement ensures that a higher preference is set for the VMs in an affinity group to follow the positive or negative affinity whenever feasible. In the two or three hypervisor configuration described in this document, soft affinity is the recommended setting. In larger clusters, hard affinity can correctly distribute OpenShift nodes.

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 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 cluster deployment.

In these instances, you can run and task the wizard without immediately deploying a cluster. Rather, a configuration file is created from which the cluster can be deployed later. This is very useful if you want to change any IPI defaults or if you wants to deploy multiple identical clusters in your 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.

Next: NetApp storage overview.

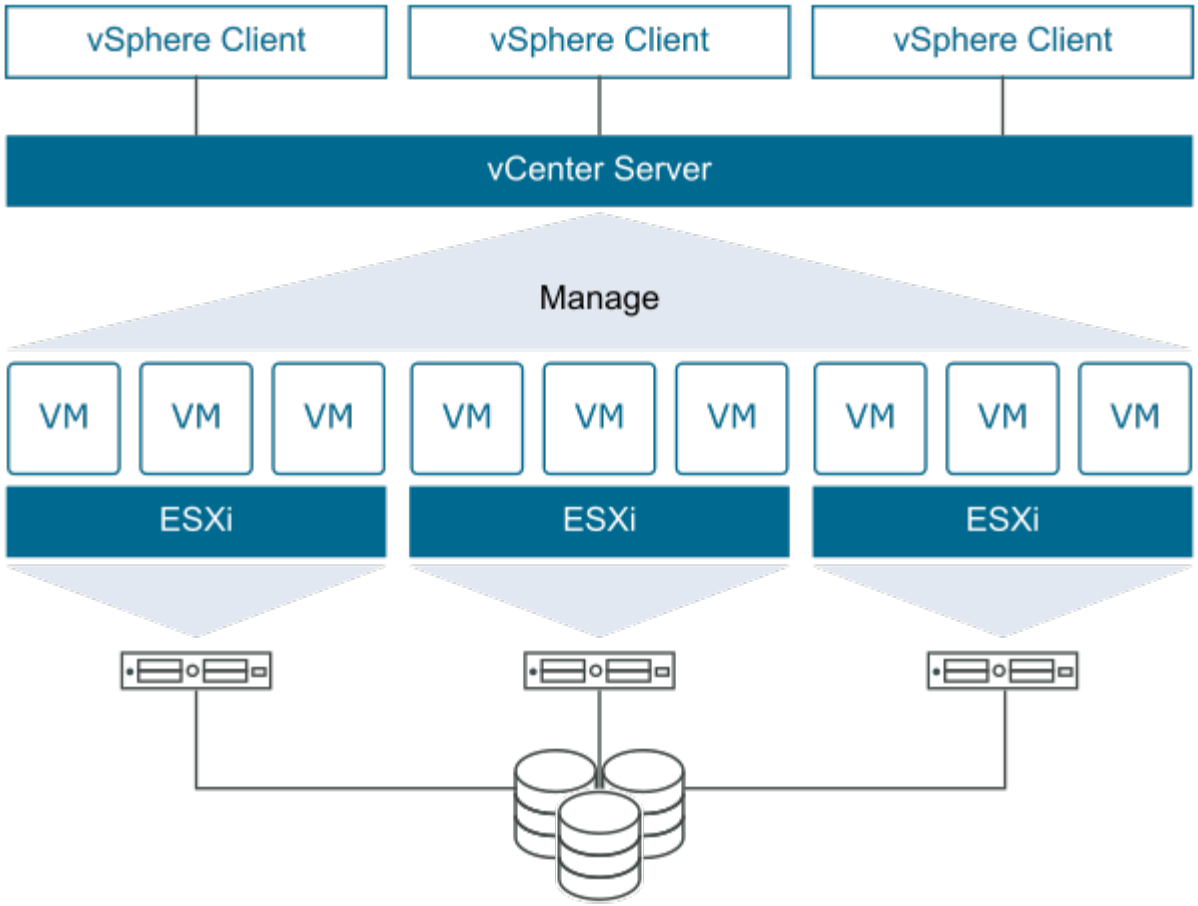
OpenShift on VMware vSphere

VMware vSphere is a virtualization platform for centrally managing a large number of virtualized servers and networks running on the ESXi hypervisor.

For more information about VMware vSphere, see the VMware vSphere website.

VMware vSphere provides the following features:

- **VMware vCenter Server.** VMware vCenter Server provides unified management of all hosts and VMs from a single console and aggregates performance monitoring of clusters, hosts, and VMs.
- **VMware vSphere vMotion.** VMware vCenter allows you to hot migrate VMs between nodes in the cluster upon request in a nondisruptive manner.
- **vSphere High Availability.** To avoid disruption in the event of host failures, VMware vSphere allows hosts to be clustered and configured for High Availability. VMs that are disrupted by host failure are rebooted shortly on other hosts in the cluster, restoring services.
- **Distributed Resource Scheduler (DRS).** A VMware vSphere cluster can be configured to load balance the resource needs of the VMs it is hosting. VMs with resource contentions can be hot migrated to other nodes in the cluster to make sure that enough resources are available.



Network design

The Red Hat OpenShift on NetApp 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 VM logical network on VMware vSphere for its cluster management. This section describes the arrangement and purpose of each virtual network segment used in the solution and outlines the prerequisites for deployment of the solution.

VLAN requirements

Red Hat OpenShift on VMware vSphere is designed to logically separate network traffic for different purposes by using virtual local area networks (VLANs). 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 while validating the solution at NetApp.

VLANs	Purpose	VLAN ID
Out-of-band management network	Management for physical nodes and IPMI	16
VM Network	Virtual guest network access	181
Storage network	Storage network for ONTAP NFS	184
Storage network	Storage network for ONTAP iSCSI	185

VLANs	Purpose	VLAN ID
In-band management network	Management for ESXi Nodes, VCenter Server, ONTAP Select	3480
Storage network	Storage network for NetApp Element iSCSI	3481
Migration network	Network for virtual guest migration	3482

Network infrastructure support resources

The following infrastructure should be in place prior to the deployment of the OpenShift Container Platform:

- At least one DNS server providing 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.

Best practices for production deployments

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

Deploy OpenShift to an ESXi 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 ESXi hypervisor nodes and ensuring a fault tolerant configuration by enabling VMware vSphere HA and VMware vMotion. This configuration allows deployed VMs to migrate between the two hypervisors and reboot should one host become unavailable.

Because Red Hat OpenShift initially deploys with three master nodes, at least two masters in a two-node configuration can occupy the same node under some circumstances, 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 ESXi hypervisor nodes must be deployed so that the OpenShift masters can be distributed evenly, which provides an added degree of fault tolerance.

Configure virtual machine and host affinity

Ensuring the distribution of the OpenShift masters across multiple hypervisor nodes can be achieved by enabling VM and host affinity.

Affinity or anti-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.

To configure affinity groups, see the vSphere 6.7 Documentation: Using DRS Affinity Rules.

Use a custom install file for OpenShift deployment

IPI makes the deployment of OpenShift clusters easy through the interactive wizard discussed earlier in this document. However, it is possible that you might need to change some default values as a part of a cluster

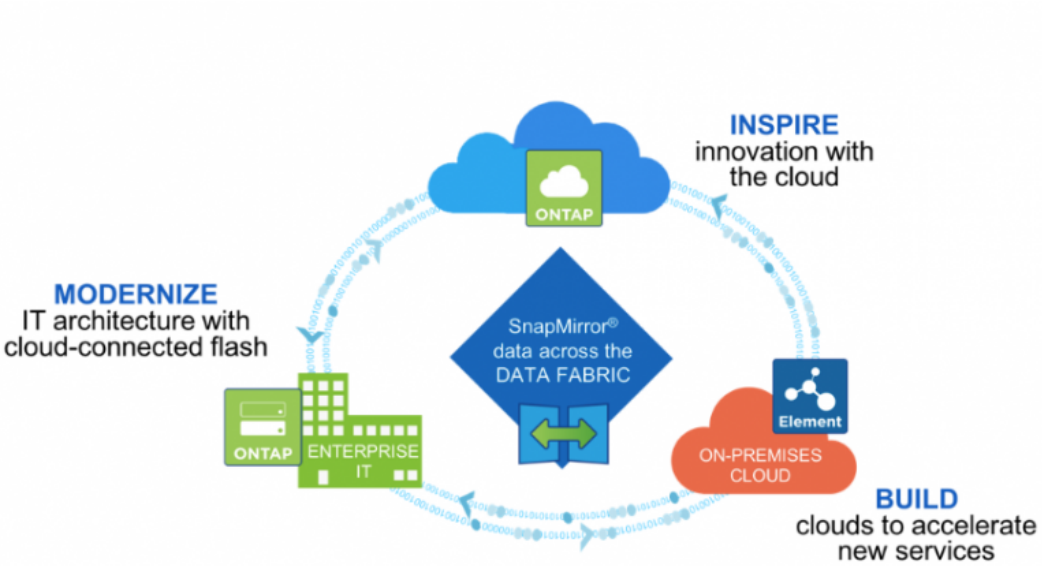
deployment.

In these instances, you can run and task the wizard without immediately deploying a cluster, but instead the wizard creates a configuration file from which the cluster can be deployed later. This is very useful if you need to changes any IPI defaults, or if you want to deploy multiple identical clusters in your 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 vSphere with Customizations.

Next: NetApp Storage Overview.

NetApp Storage Overview

NetApp has several storage platforms that are qualified with our Astra Trident Storage Orchestrator to provision storage for applications deployed on Red Hat OpenShift.



- AFF and FAS systems run NetApp ONTAP and provide storage for both file-based (NFS) and block-based (iSCSI) use cases.
- Cloud Volumes ONTAP and ONTAP Select provide the same benefits in the cloud and virtual space respectively.
- NetApp Cloud Volumes Service (AWS/GCP) and Azure NetApp Files provide file-based storage in the cloud.
- NetApp Element storage systems provide for block-based (iSCSI) use cases in a highly scalable environment.

Each storage system in the NetApp portfolio can ease both data management and movement between on-premises sites and the cloud, ensuring that your data is where your applications are.

The following pages have additional information about the NetApp storage systems validated in the Red Hat OpenShift with NetApp solution:

- NetApp ONTAP
- NetApp Element

Next: NetApp Storage Integrations Overview

NetApp ONTAP

NetApp ONTAP is a powerful storage-software tool with capabilities such as an intuitive GUI, REST APIs with automation integration, AI-informed predictive analytics and corrective action, non-disruptive hardware upgrades, and cross-storage import.

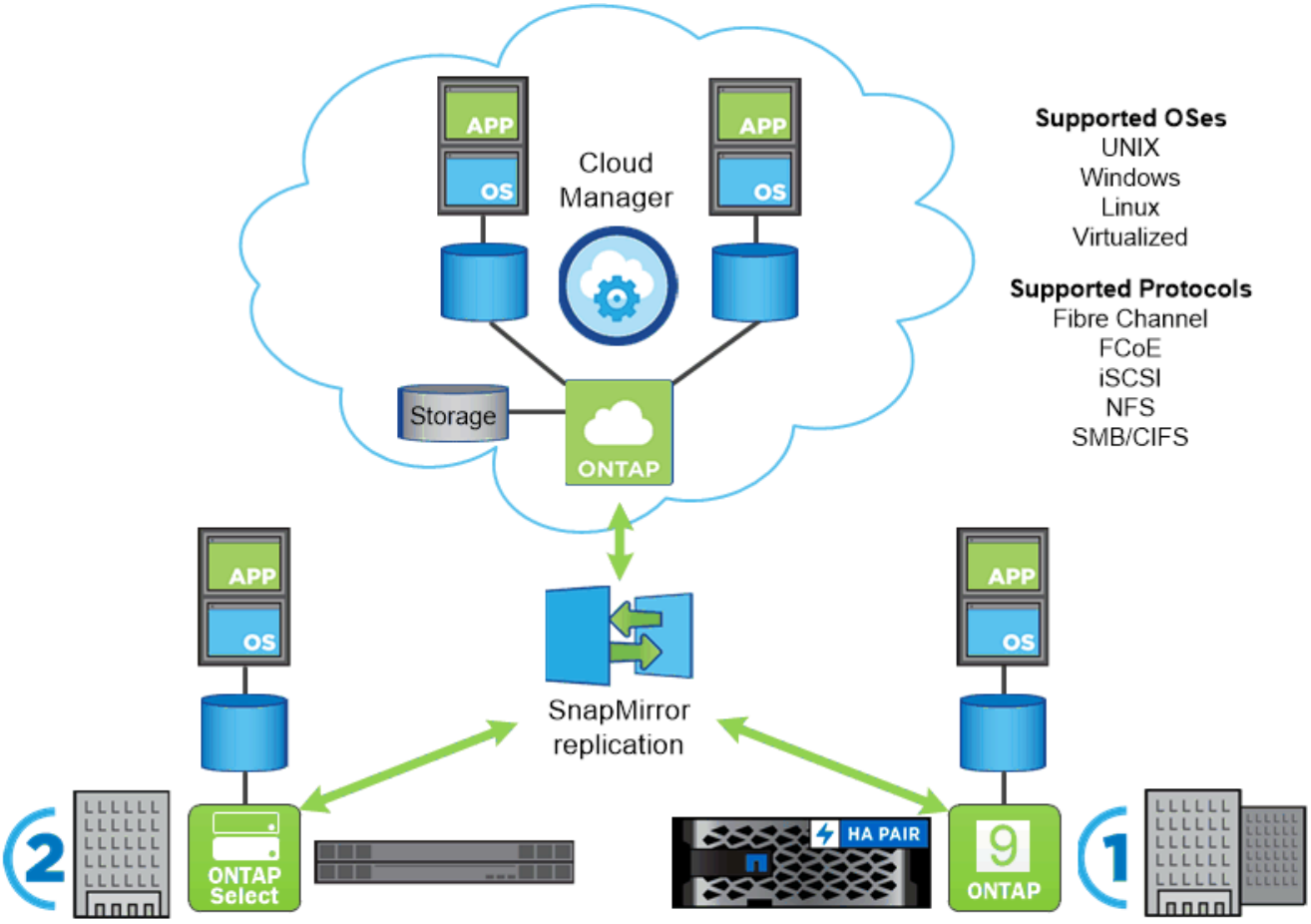
For more information about the NetApp ONTAP storage system, visit the NetApp ONTAP website.

ONTAP provides the following features:

- A unified storage system with simultaneous data access and management of NFS, CIFS, iSCSI, FC, FCoE, and FC-NVMe protocols.
- Different deployment models include on-premises on all-flash, hybrid, and all-HDD hardware configurations; VM-based storage platforms on a supported hypervisor such as ONTAP Select; and in the cloud as Cloud Volumes ONTAP.
- Increased data storage efficiency on ONTAP systems with support for automatic data tiering, inline data compression, deduplication, and compaction.
- Workload-based, QoS-controlled storage.
- Seamless integration with a public cloud for tiering and protection of data. ONTAP also provides robust data protection capabilities that sets it apart in any environment:
  - **NetApp Snapshot copies.** A fast, point-in-time backup of data using a minimal amount of disk space with no additional performance overhead.
  - **NetApp SnapMirror.** Mirrors the Snapshot copies of data from one storage system to another. ONTAP supports mirroring data to other physical platforms and cloud-native services as well.
  - **NetApp SnapLock.** Efficiently administration of non-rewritable data by writing it to special volumes that cannot be overwritten or erased for a designated period.
  - **NetApp SnapVault.** Backs up data from multiple storage systems to a central Snapshot copy that serves as a backup to all designated systems.
  - **NetApp SyncMirror.** Provides real-time, RAID-level mirroring of data to two different plexes of disks that are connected physically to the same controller.
  - **NetApp SnapRestore.** Provides fast restoration of backed-up data on demand from Snapshot copies.
  - **NetApp FlexClone.** Provides instantaneous provisioning of a fully readable and writeable copy of a NetApp volume based on a Snapshot copy.

For more information about ONTAP, see the ONTAP 9 Documentation Center.

NetApp ONTAP is available on-premises, virtualized, or in the cloud.



NetApp platforms

NetApp AFF/FAS

NetApp provides robust all-flash (AFF) and scale-out hybrid (FAS) storage platforms that are tailor-made with low-latency performance, integrated data protection, and multi-protocol support.

Both systems are powered by NetApp ONTAP data management software, the industry's most advanced data-management software for highly-available, cloud-integrated, simplified storage management to deliver enterprise-class speed, efficiency, and security your data fabric needs.

For more information about NETAPP AFF/FAS platforms, click here.

ONTAP Select

ONTAP Select is a software-defined deployment of NetApp ONTAP that can be deployed onto a hypervisor in your environment. It can be installed on VMware vSphere or on KVM and provides the full functionality and experience of a hardware-based ONTAP system.

For more information about ONTAP Select, click here.

Cloud Volumes ONTAP

NetApp Cloud Volumes ONTAP is a cloud-deployed version of NetApp ONTAP available to be deployed in a number of public clouds, including: Amazon AWS, Microsoft Azure, and Google Cloud.

For more information about Cloud Volumes ONTAP, click here.

Next: NetApp Storage Integrations Overview

NetApp Element: Red Hat OpenShift with NetApp

NetApp Element software provides modular, scalable performance, with each storage node delivering guaranteed capacity and throughput to the environment. NetApp Element systems can scale from 4 to 100 nodes in a single cluster and offer a number of advanced storage management features.

For more information about NetApp Element storage systems, visit the NetApp Solidfire website.

iSCSI login redirection and self-healing capabilities

NetApp Element software leverages the iSCSI storage protocol, a standard way to encapsulate SCSI commands on a traditional TCP/IP network. When SCSI standards change or when the performance of Ethernet networks 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 the 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 non-disruptive 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 particular 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.
- **VRF-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 in-service provider environments where scale and preservation of IPspace 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:

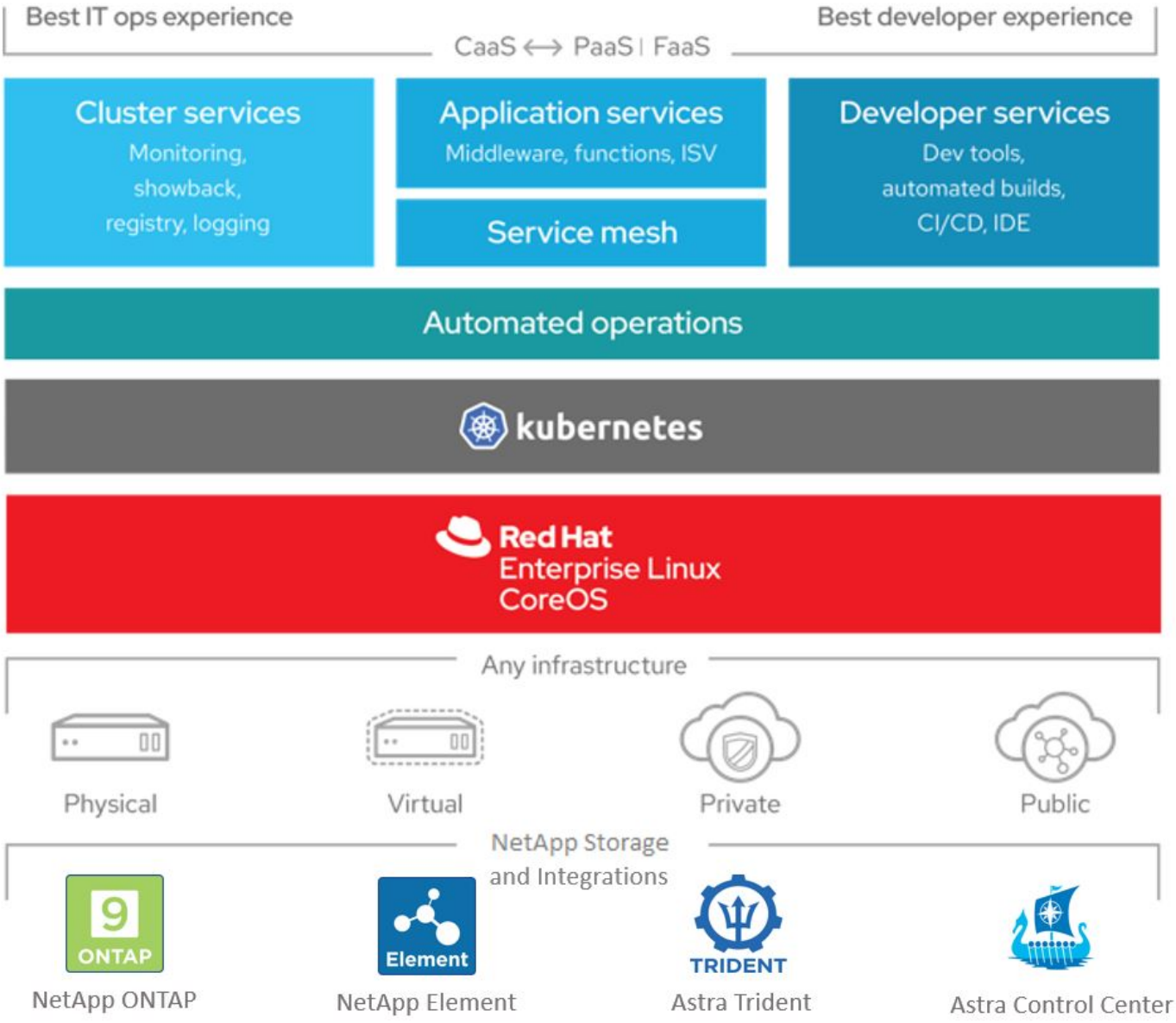
- **Deduplication.** The system only stores unique 4K blocks. Any duplicate 4K blocks are automatically associated to an already stored version of the data. Data is on block drives and is mirrored by using the NetApp Element software Helix data protection. This system significantly reduces capacity consumption 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.

Element was designed for automation. All the storage features are available through APIs. These APIs are the only method that the UI uses to control the system.

Next: NetApp Storage Integrations Overview.

NetApp Storage Integration Overview

NetApp provides a number of products to help you with orchestrating and managing persistent data in container based environments, such as Red Hat OpenShift.



NetApp Astra Control offers a rich set of storage and application-aware data management services for stateful Kubernetes workloads, powered by NetApp data protection technology. The Astra Control Service is available to support stateful workloads in cloud-native Kubernetes deployments. The Astra Control Center is available to support stateful workloads in on-premises deployments, like Red Hat OpenShift. For more information visit the NetApp Astra Control website here.

NetApp Astra Trident is an open-source and fully-supported storage orchestrator for containers and Kubernetes distributions, including Red Hat OpenShift. For more information, visit the Astra Trident website here.

The following pages have additional information about the NetApp products that have been validated for

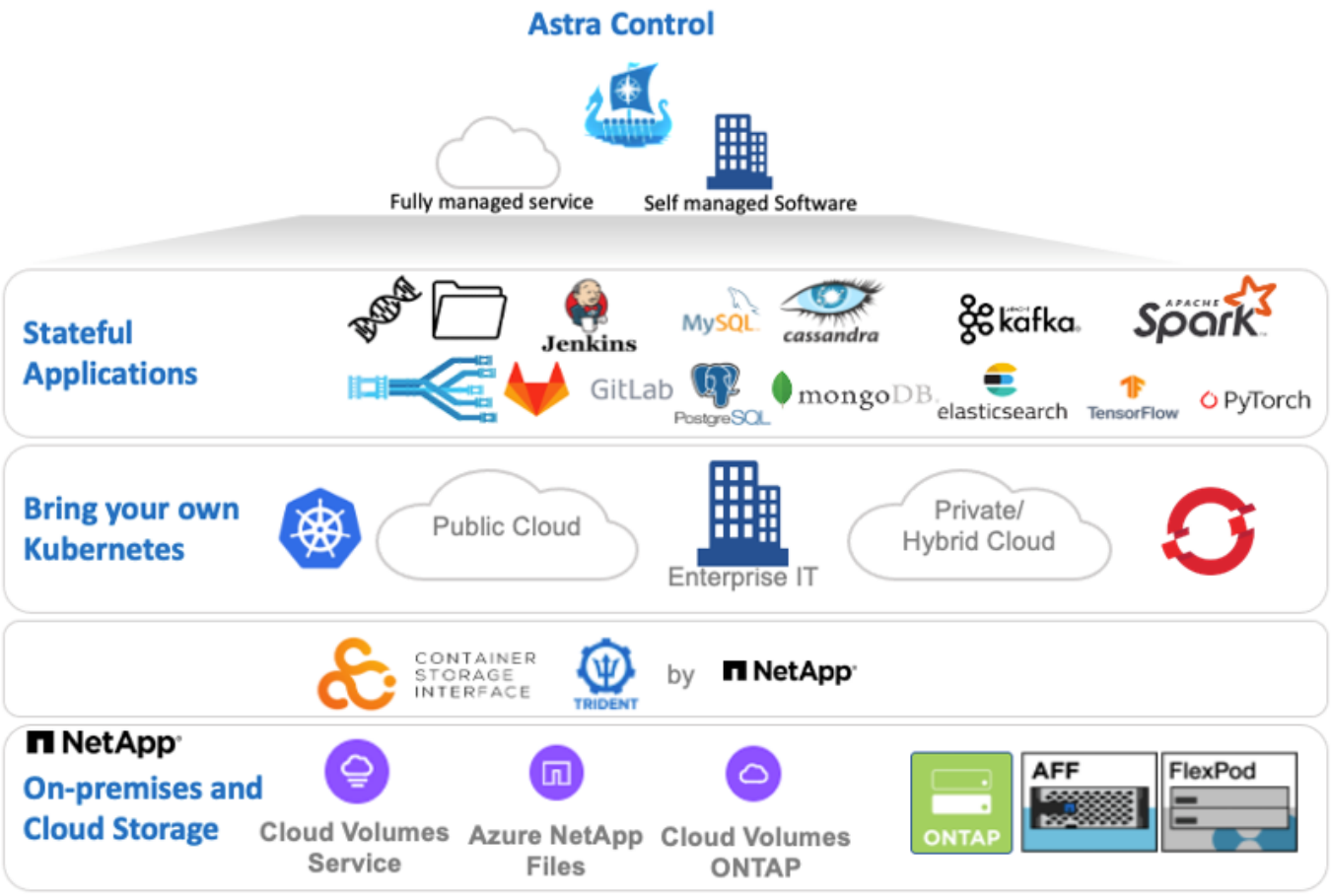
application and persistent storage management in the Red Hat OpenShift with NetApp solution:

- NetApp Astra Control Center
- NetApp Astra Trident

Next: NetApp Astra Control Center Overview

NetApp Astra Control Center overview

NetApp Astra Control Center offers a rich set of storage and application-aware data management services for stateful Kubernetes workloads deployed in an on-premises environment and powered by NetApp data protection technology.



NetApp Astra Control Center can be installed on a Red Hat OpenShift cluster that has the Astra Trident storage orchestrator deployed and configured with storage classes and storage backends to NetApp ONTAP storage systems.

For the installation and configuration of Astra Trident to support Astra Control Center, see this document here.

In a cloud-connected environment, Astra Control Center uses Cloud Insights to provide advanced monitoring and telemetry. In the absence of a Cloud Insights connection, limited monitoring and telemetry (7-days worth of metrics) is available and exported to Kubernetes native monitoring tools (Prometheus and Grafana) through open metrics endpoints.

Astra Control Center is fully integrated into the NetApp AutoSupport and Active IQ ecosystem to provide support for users, provide assistance with troubleshooting, and display usage statistics.

In addition to the paid version of Astra Control Center, a 90-day evaluation license is available. The evaluation version is supported through the email and community (Slack channel). Customers have access to these and other knowledge-base articles and the documentation available from the in-product support dashboard.

To get started with NetApp Astra Control Center, visit the Astra website.

Astra Control Center installation prerequisites

1. One or more Red Hat OpenShift clusters. Versions 4.6 EUS and 4.7 are currently supported.
2. Astra Trident must already be installed and configured on each Red Hat OpenShift cluster.
3. One or more NetApp ONTAP storage systems running ONTAP 9.5 or greater.

   It's best practice for each OpenShift install at a site to have a dedicated SVM for persistent storage. Multi-site deployments require additional storage systems.

4. A Trident storage backend must be configured on each OpenShift cluster with an SVM backed by an ONTAP cluster.
5. A default StorageClass configured on each OpenShift cluster with Astra Trident as the storage provisioner.
6. A load balancer must be installed and configured on each OpenShift cluster for load balancing and exposing OpenShift Services.

   See the link here for information about load balancers that have been validated for this purpose.

7. A private image registry must be configured to host the NetApp Astra Control Center images.

   See the link here to install and configure an OpenShift private registry for this purpose.

8. You must have Cluster Admin access to the Red Hat OpenShift cluster.
9. You must have Admin access to NetApp ONTAP clusters.
10. An admin workstation with docker or podman, tridentctl, and oc or kubectl tools installed and added to your $PATH.

    Docker installations must have docker version greater than 20.10 and Podman installations must have podman version greater than 3.0.

Install Astra Control Center

1. Log into the NetApp Support Site and download the latest version of NetApp Astra Control Center. TO do so requires a license attached to your NetApp account. After you download the tarball, transfer it to the admin workstation.

   To get started with a trial license for Astra Control, visit the Astra registration site.

2. Unpack the tar ball and change the working directory to the resulting folder.

```
[netapp-user@rhel7 ~]$ tar -vxzf astra-control-center-21.08.65.tar.gz
[netapp-user@rhel7 ~]$ cd astra-control-center-21.08.65
```

3. Before starting the installation, push the Astra Control Center images to an image registry.

You can choose to do this with either Docker or Podman; instructions for both are provided in this step.

podman

a. Export the registry FQDN with the organization/namespace/project name as a environment variable 'registry'.

```
[netapp-user@rhel7 ~]$ export registry=astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra
```

b. Log into the registry.

```
[netapp-user@rhel7 ~]$ podman login -u ocp-user -p password --tls
-verify=false astra-registry.apps.ocp-vmw.cie.netapp.com
```

If you are using `kubeadmin` user to log into the private registry, then use token instead of password - `podman login -u ocp-user -p token --tls-verify=false astra-registry.apps.ocp-vmw.cie.netapp.com`.

Alternatively, you can create a service account, assign registry-editor and/or registry-viewer role (based on whether you require push/pull access) and log into the registry using service account's token.

c. Create a shell script file and paste the following content in it.

```
[netapp-user@rhel7 ~]$ vi push-images-to-registry.sh

for astraImageFile in $(ls images/*.tar); do
   astraImage=$(podman load --input ${astraImageFile} | sed
's/Loaded image(s): //')
   podman tag $astraImage $registry/$(echo $astraImage | sed
's/^[^\/]\+\///')
   podman push $registry/$(echo $astraImage | sed 's/^[^\/]\+\///')
done
```

If you are using untrusted certificates for your registry, edit the shell script and use `--tls-verify=false` for the podman push command `podman push $registry/$(echo $astraImage | sed 's/[^\/]\+\///') --tls -verify=false`.

d. Make the file executable.

```
[netapp-user@rhel7 ~]$ chmod +x push-images-to-registry.sh
```

e. Execute the shell script.

```
[netapp-user@rhel7 ~]$ ./push-images-to-registry.sh
```

docker

a. Export the registry FQDN with the organization/namespace/project name as a environment variable ‘registry’.

```
[netapp-user@rhel7 ~]$ export registry=astra-registry.apps.ocp-vmw.cie.netapp.com/netapp-astra
```

b. Log into the registry.

```
[netapp-user@rhel7 ~]$ docker login -u ocp-user -p password astra-registry.apps.ocp-vmw.cie.netapp.com
```

If you are using `kubeadmin` user to log into the private registry, then use token instead of password - `docker login -u ocp-user -p token astra-registry.apps.ocp-vmw.cie.netapp.com`.

Alternatively, you can create a service account, assign registry-editor and/or registry-viewer role (based on whether you require push/pull access) and log into the registry using service account’s token.

c. Create a shell script file and paste the following content in it.

```
[netapp-user@rhel7 ~]$ vi push-images-to-registry.sh

for astraImageFile in $(ls images/*.tar); do
   astraImage=$(docker load --input ${astraImageFile} | sed 's/Loaded image: //')
   docker tag $astraImage $registry/$(echo $astraImage | sed 's/^[^\/]\+\///')
   docker push $registry/$(echo $astraImage | sed 's/^[^\/]\+\///')
done
```

d. Make the file executable.

```
[netapp-user@rhel7 ~]$ chmod +x push-images-to-registry.sh
```

e. Execute the shell script.

```
[netapp-user@rhel7 ~]$ ./push-images-to-registry.sh
```

4. Next, upload the image registry TLS certificates to the OpenShift nodes. To do so, create a configmap in the openshift-config namespace using the TLS certificates and patch it to the cluster image config to make the certificate trusted.

```
[netapp-user@rhel7 ~]$ oc create configmap default-ingress-ca -n
openshift-config --from-file=astra-registry.apps.ocp
-vmw.cie.netapp.com=tls.crt

[netapp-user@rhel7 ~]$ oc patch image.config.openshift.io/cluster
--patch '{"spec":{"additionalTrustedCA":{"name":"default-ingress-ca"}}}'
--type=merge
```

> If you are using an OpenShift internal registry with default TLS certificates from the ingress operator with a route, you still need to follow the previous step to patch the certificates to the route hostname. To extract the certificates from ingress operator, you can use the command `oc extract secret/router-ca --keys=tls.crt -n openshift-ingress-operator`.

5. Create a namespace `netapp-acc-operator` for installing the Astra Control Center Operator.

```
[netapp-user@rhel7 ~]$ oc create ns netapp-acc-operator
```

6. Create a secret with credentials to log into the image registry in `netapp-acc-operator` namespace.

```
[netapp-user@rhel7 ~]$ oc create secret docker-registry astra-registry-
cred --docker-server=astra-registry.apps.ocp-vmw.cie.netapp.com --docker
-username=ocp-user --docker-password=password -n netapp-acc-operator
secret/astra-registry-cred created
```

7. Edit the Astra Control Center Operator CR `astra_control_center_operator_deploy.yaml`, which is a set of all resources Astra Control Center deploys. In the operator CR, find the deployment definition for `acc-operator-controller-manager` and enter the FQDN for your registry along with the organization name as it was given while pushing the images to registry (in this example, `astra-registry.apps.ocp-vmw.cie.netapp.com/netapp-astra`) by replacing the text `ASTRA_IMAGE_REGISTRY` and provide the name of the secret we just created in `imagePullSecrets` section. Verify other details of the operator, save, and close.

```
[netapp-user@rhel7 ~]$ vim astra_control_center_operator_deploy.yaml

apiVersion: apps/v1
```

```
kind: Deployment
metadata:
  labels:
    control-plane: controller-manager
  name: acc-operator-controller-manager
  namespace: netapp-acc-operator
spec:
  replicas: 1
  selector:
    matchLabels:
      control-plane: controller-manager
  template:
    metadata:
      labels:
        control-plane: controller-manager
    spec:
      containers:
      - args:
        - --secure-listen-address=0.0.0.0:8443
        - --upstream=http://127.0.0.1:8080/
        - --logtostderr=true
        - --v=10
        image: ASTRA_IMAGE_REGISTRY/kube-rbac-proxy:v0.5.0
        name: kube-rbac-proxy
        ports:
        - containerPort: 8443
          name: https
      - args:
        - --health-probe-bind-address=:8081
        - --metrics-bind-address=127.0.0.1:8080
        - --leader-elect
        command:
        - /manager
        env:
        - name: ACCOP_LOG_LEVEL
          value: "2"
        image: astra-registry.apps.ocp-vmw.cie.netapp.com/netapp-
astra/acc-operator:21.08.7
        imagePullPolicy: IfNotPresent
        livenessProbe:
          httpGet:
            path: /healthz
            port: 8081
          initialDelaySeconds: 15
          periodSeconds: 20
        name: manager
```

```
        readinessProbe:
          httpGet:
            path: /readyz
            port: 8081
          initialDelaySeconds: 5
          periodSeconds: 10
        resources:
          limits:
            cpu: 300m
            memory: 750Mi
          requests:
            cpu: 100m
            memory: 75Mi
        securityContext:
          allowPrivilegeEscalation: false
      imagePullSecrets: [name: astra-registry-cred]
      securityContext:
        runAsUser: 65532
      terminationGracePeriodSeconds: 10
```

8. Create the operator by running the following command.

```
[netapp-user@rhel7 ~]$ oc create -f
astra_control_center_operator_deploy.yaml
```

9. Create a dedicated namespace for installing all the Astra Control Center resources.

```
[netapp-user@rhel7 ~]$ oc create ns netapp-astra-cc
namespace/netapp-astra-cc created
```

10. Create the secret for accessing the image registry in that namespace.

```
[netapp-user@rhel7 ~]$ oc create secret docker-registry astra-registry-
cred --docker-server=astra-registry.apps.ocp-vmw.cie.netapp.com --docker
-username=ocp-user --docker-password=password -n netapp-astra-cc

secret/astra-registry-cred created
```

11. Edit the Astra Control Center CRD file `astra_control_center_min.yaml` and enter the FQDN, image registry details, administrator email address, and other details.

```
[netapp-user@rhel7 ~]$ vim astra_control_center_min.yaml

apiVersion: astra.netapp.io/v1
kind: AstraControlCenter
metadata:
  name: astra
spec:
  accountName: "NetApp HCG Solutions"
  astraVersion: "21.08.65"
  astraAddress: "astra-control-center.cie.netapp.com"
  autoSupport:
    enrolled: true
  email: "solutions_tme@netapp.com"
  firstName: "NetApp HCG"
  lastName: "Admin"
  imageRegistry:
    name: "astra-registry.apps.ocp-vmw.cie.netapp.com/netapp-astra"
# use your registry
    secret: "astra-registry-cred"             # comment out if not
needed
```

12. Create the Astra Control Center CRD in the namespace created for it.

```
[netapp-user@rhel7 ~]$ oc apply -f astra_control_center_min.yaml -n
netapp-astra-cc
astracontrolcenter.astra.netapp.io/astra created
```

> The previous file `astra_control_center_min.yaml` is the minimum version of the Astra Control Center CRD. If you want to create the CRD with more control, such as defining a storageclass other than the default for creating PVCs or providing SMTP details for mail notifications, you can edit the file `astra_control_center.yaml`, enter then needed details, and use it to create the CRD.

Installation verification

1. It might take several minutes for the installation to complete. Verify that all the pods and services in the `netapp-astra-cc` namespace are up and running.

```
[netapp-user@rhel7 ~]$ oc get all -n netapp-astra-cc
```

2. Check the `acc-operator-controller-manager` logs to ensure that the installation is completed.

```
[netapp-user@rhel7 ~]$ oc logs deploy/acc-operator-controller-manager -n
netapp-acc-operator -c manager -f
```

The following message indicates the successful installation of Astra Control Center.

```
{"level":"info","ts":1624054318.029971,"logger":"controllers.AstraContro
lCenter","msg":"Successfully Reconciled AstraControlCenter in
[seconds]s","AstraControlCenter":"netapp-astra-
cc/astra","ae.Version":"[21.08.65]"}
```

3. The username for logging into Astra Control Center is the email address of the administrator provided in the CRD file and the password is a string `ACC-` appended to the Astra Control Center UUID. Run the following command:

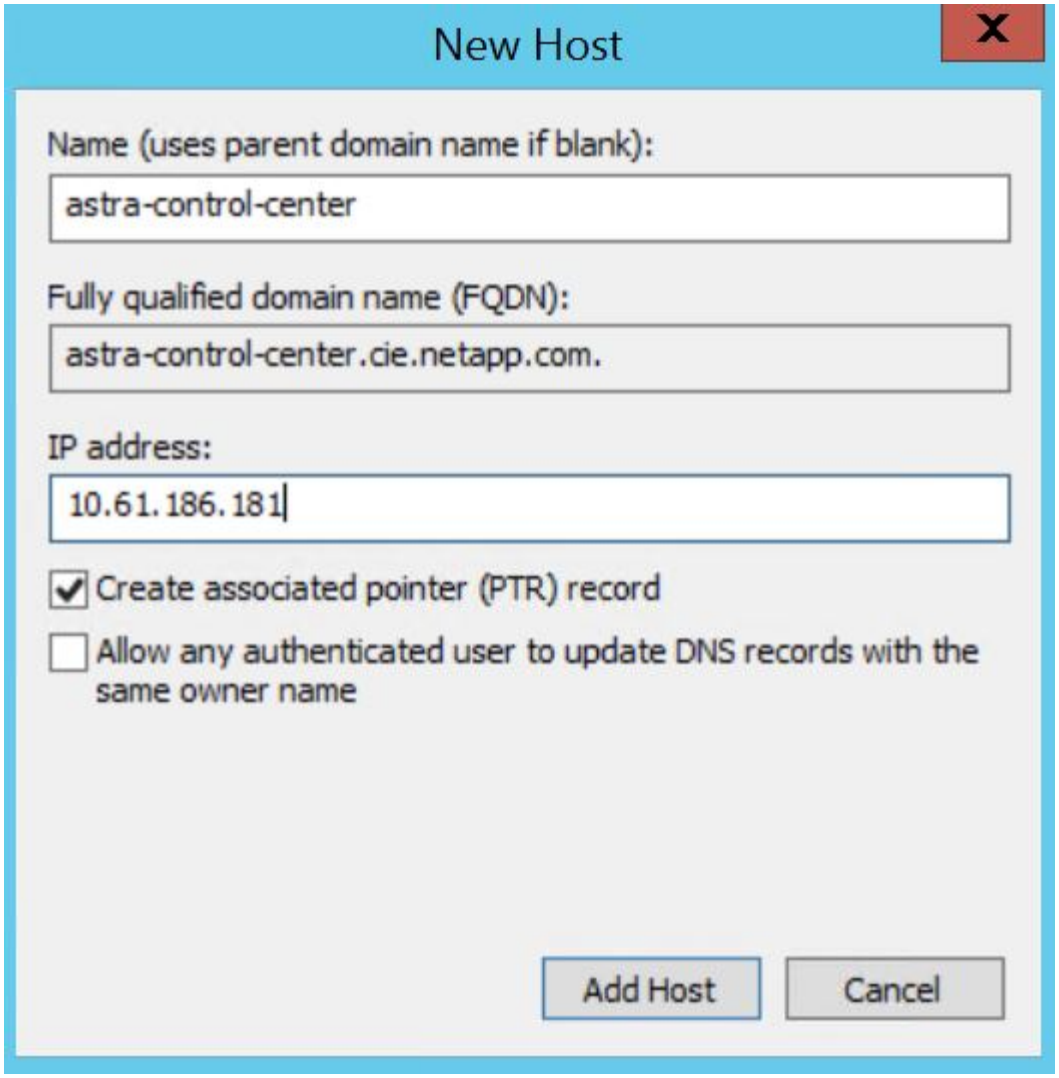
```
[netapp-user@rhel7 ~]$ oc get astracontrolcenters -n netapp-astra-cc
NAME    UUID
astra   345c55a5-bf2e-21f0-84b8-b6f2bce5e95f
```

In this example, the password is `ACC-345c55a5-bf2e-21f0-84b8-b6f2bce5e95f`.

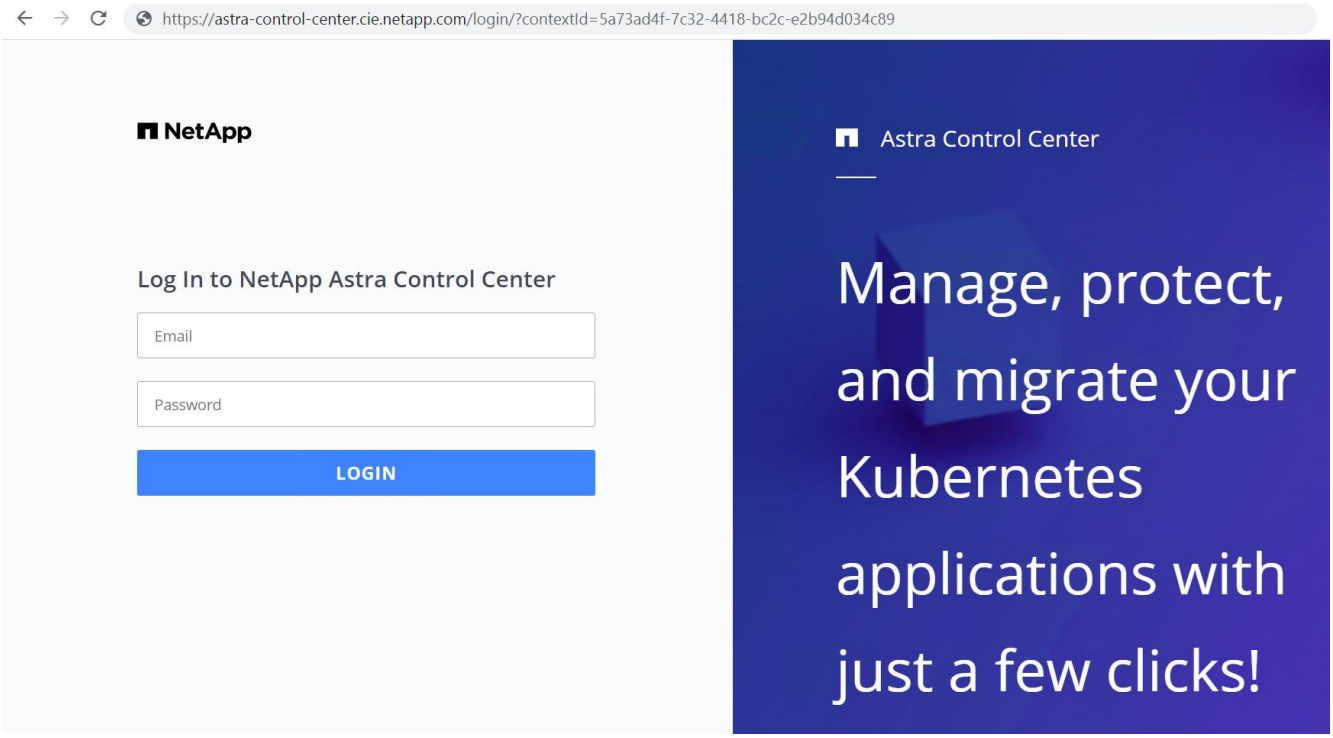
4. Get the traefik service load balancer IP.

```
[netapp-user@rhel7 ~]$ oc get svc -n netapp-astra-cc | egrep
'EXTERNAL|traefik'

NAME                                      TYPE           CLUSTER-IP
EXTERNAL-IP     PORT(S)
AGE
traefik                                   LoadBalancer   172.30.99.142
10.61.186.181   80:30343/TCP,443:30060/TCP
16m
```

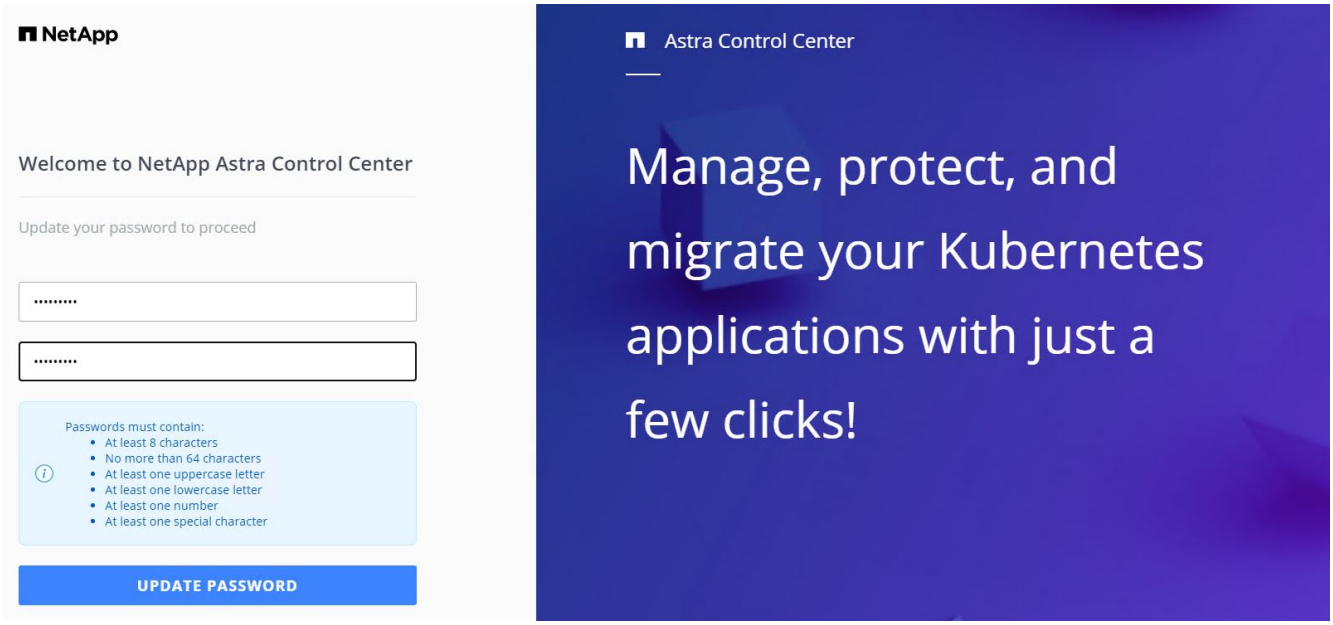
5. Add an entry in the DNS server pointing the FQDN provided in the Astra Control Center CRD file to the `EXTERNAL-IP` of the traefik service.



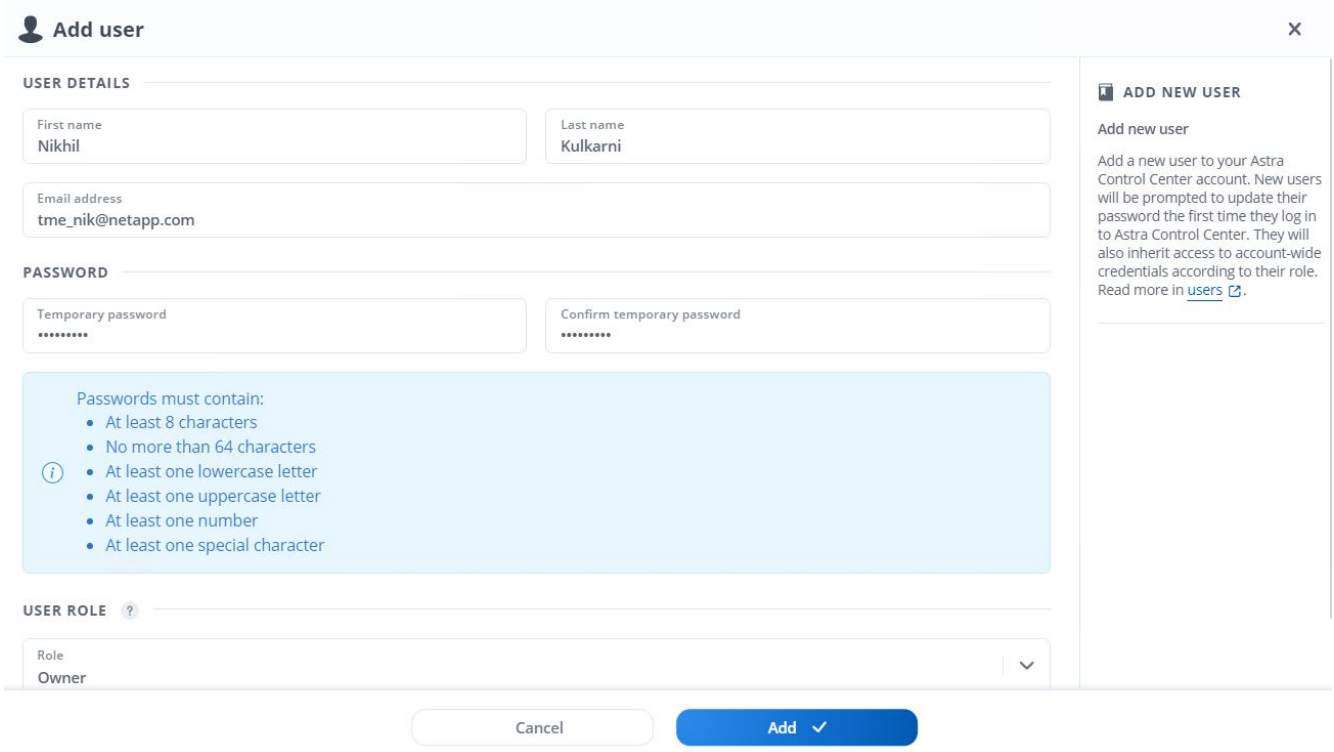
6. Log into the Astra Control Center GUI by browsing its FQDN.



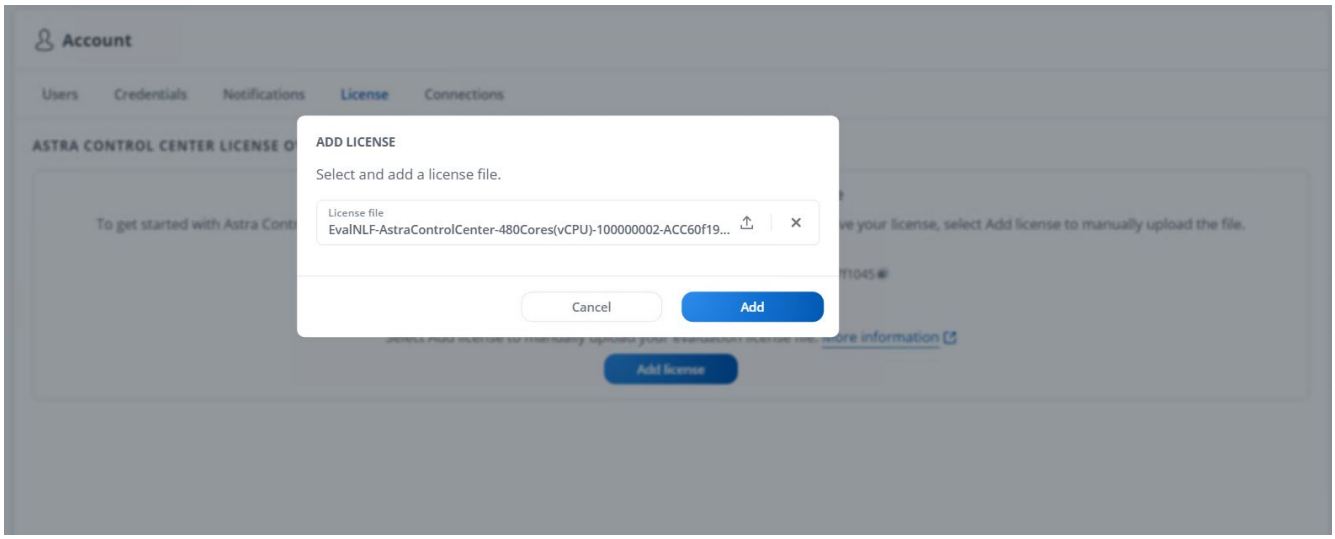
7. When you log into Astra Control Center GUI for the first time using the admin email address provided in CRD, you need to change the password.



8. If you wish to add a user to Astra Control Center, navigate to Account > Users, click Add, enter the details of the user, and click Add.



9. Astra Control Center requires a license for all of it's functionalities to work. To add a license, navigate to Account > License, click Add License, and upload the license file.



If you encounter issues with the install or configuration of NetApp Astra Control Center, the knowledge base of known issues is available here.

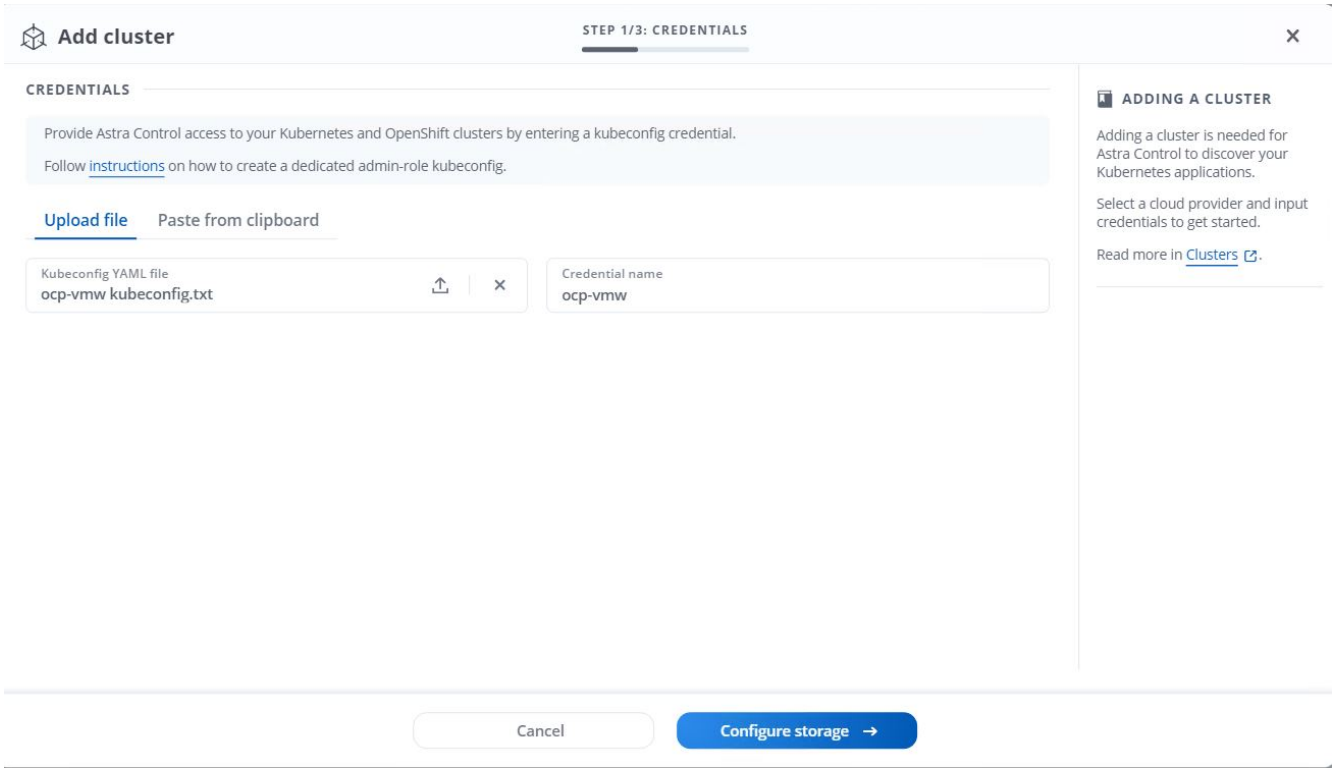
Next: Register your Red Hat OpenShift Clusters: Red Hat OpenShift with NetApp.

Register your Red Hat OpenShift Clusters with the Astra Control Center

To enable the Astra Control Center to manage your workloads, you must first register your Red Hat OpenShift cluster.

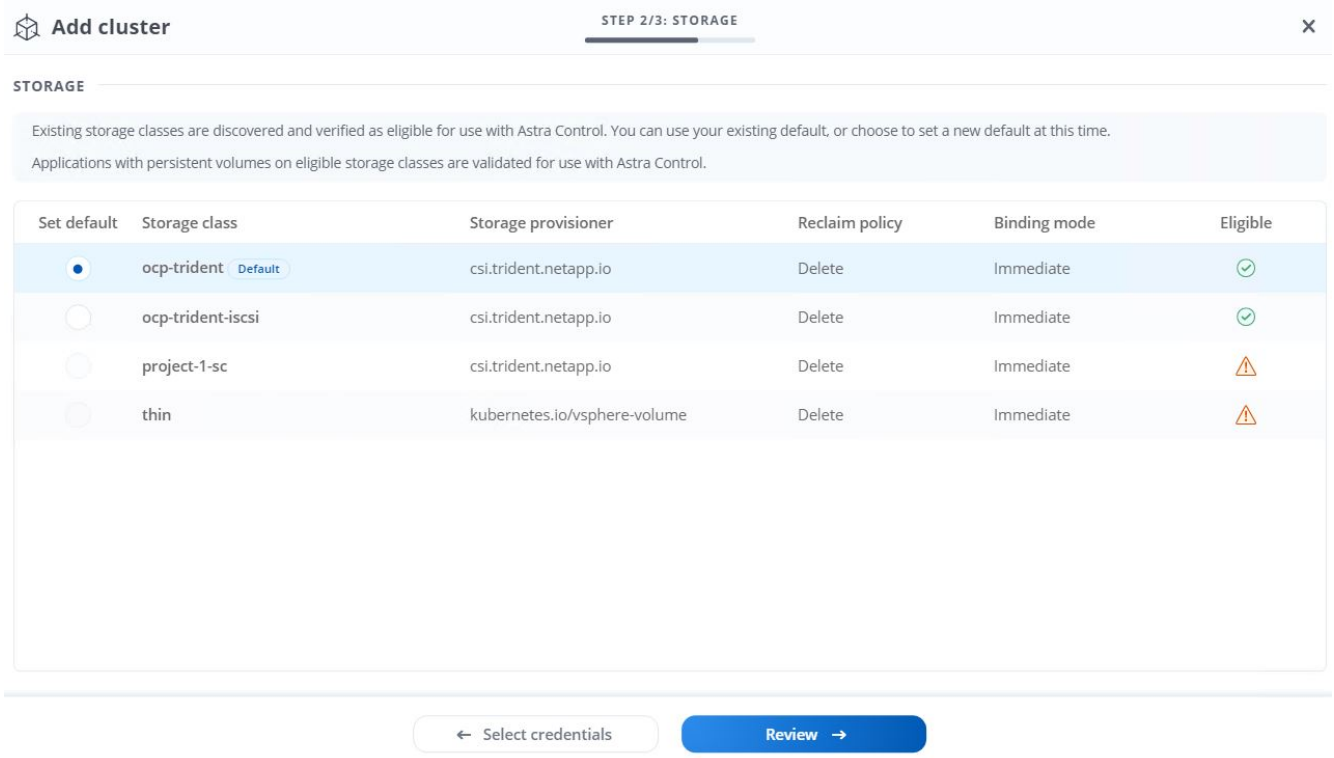
Register Red Hat OpenShift clusters

1. The first step is to add the OpenShift clusters to the Astra Control Center and manage them. Go to Clusters and click Add a Cluster, upload the kubeconfig file for the OpenShift cluster, and click Select Storage.

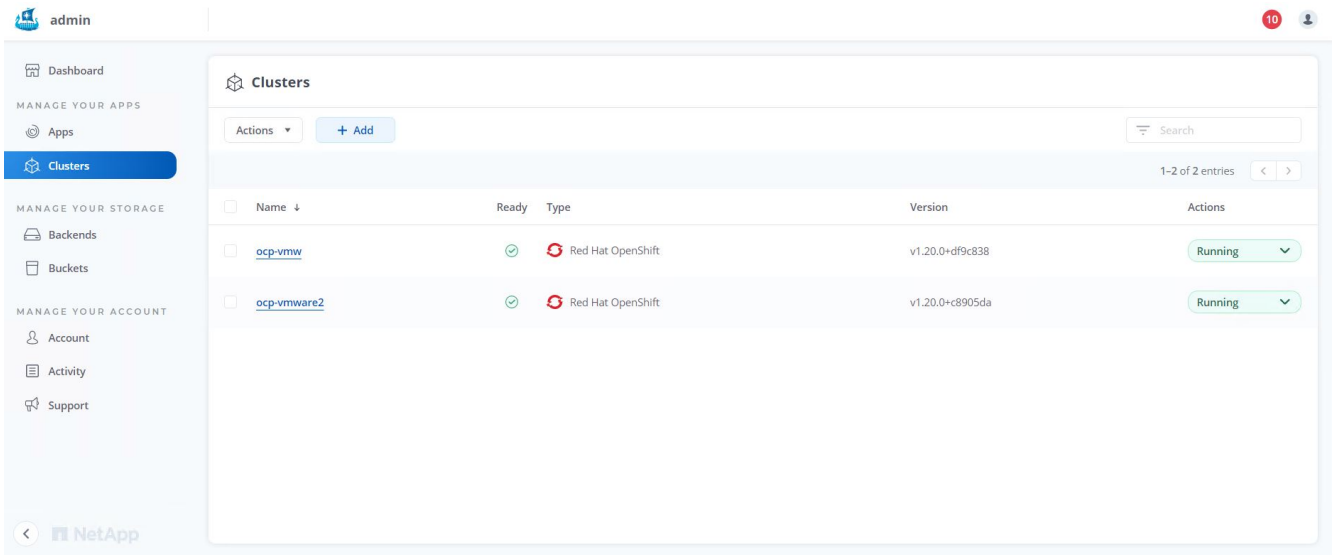


> The kubeconfig file can be generated to authenticate with a username and password or a token. Tokens expire after a limited amount of time and might leave the registered cluster unreachable. NetApp recommends using a kubeconfig file with a username and password to register your OpenShift clusters to Astra Control Center.

2. Astra Control Center detects the eligible storage classes. Now select the way that storageclass provisions volumes using Trident backed by an SVM on NetApp ONTAP and click Review. In the next pane, verify the details and click Add Cluster.

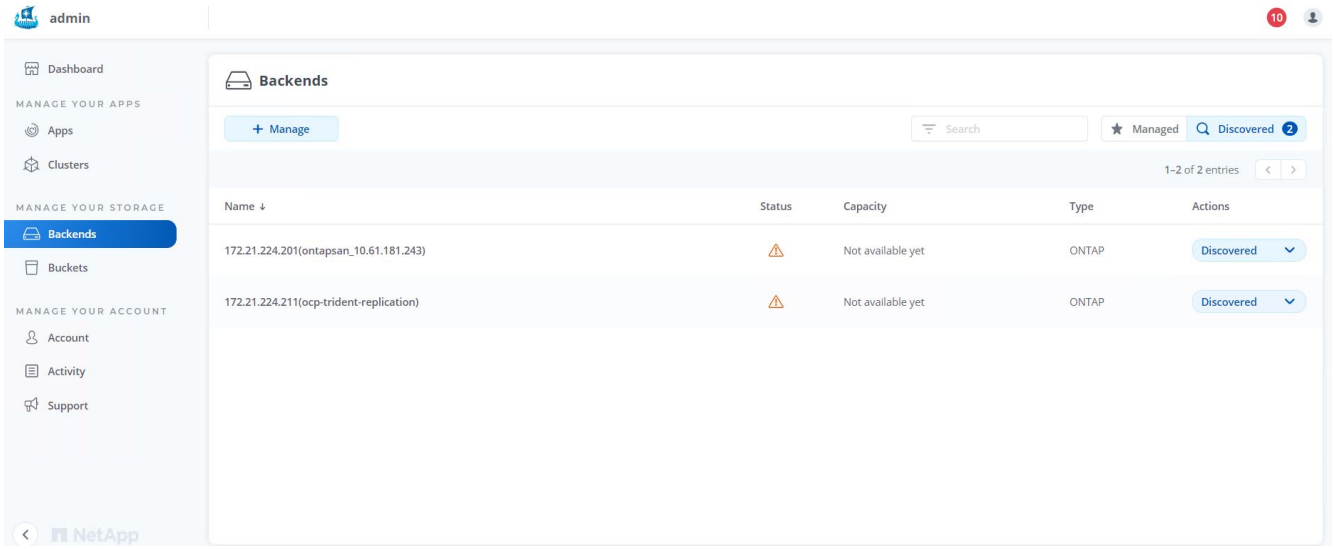


3. Register both OpenShift clusters as described in step 1. When added, the clusters move to the Discovering status while Astra Control Center inspects them and installs the necessary agents. Cluster status changes to Running after they are successfully registered.

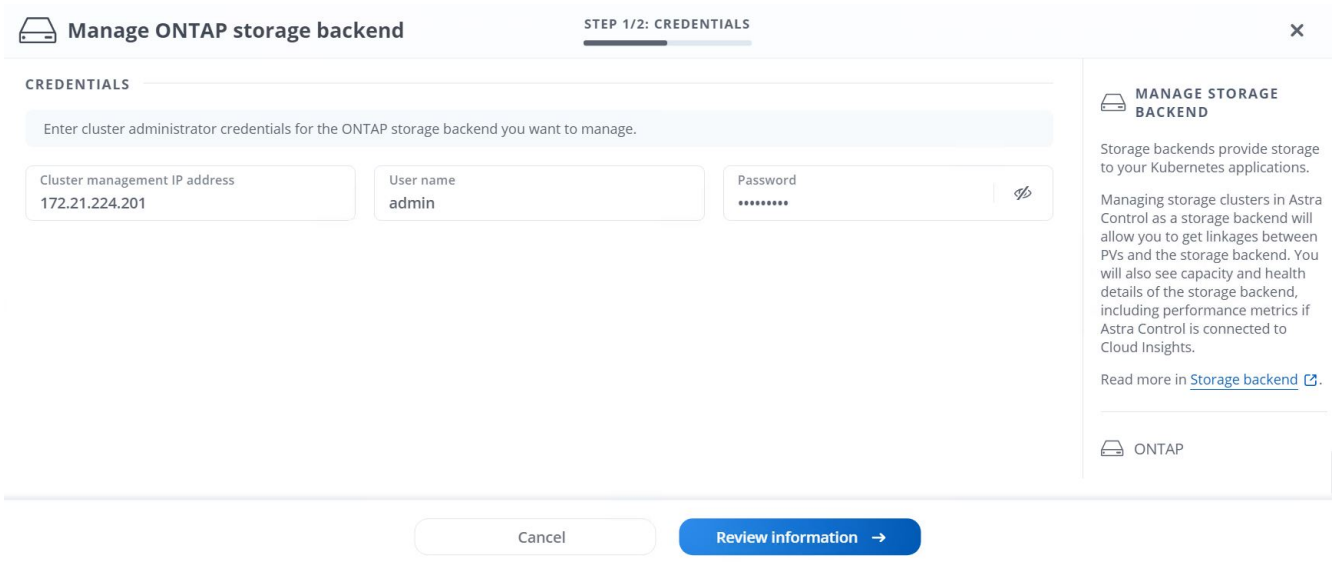


All Red Hat OpenShift clusters to be managed by Astra Control Center should have access to the image registry that was used for its installation as the agents installed on the managed clusters pull the images from that registry.

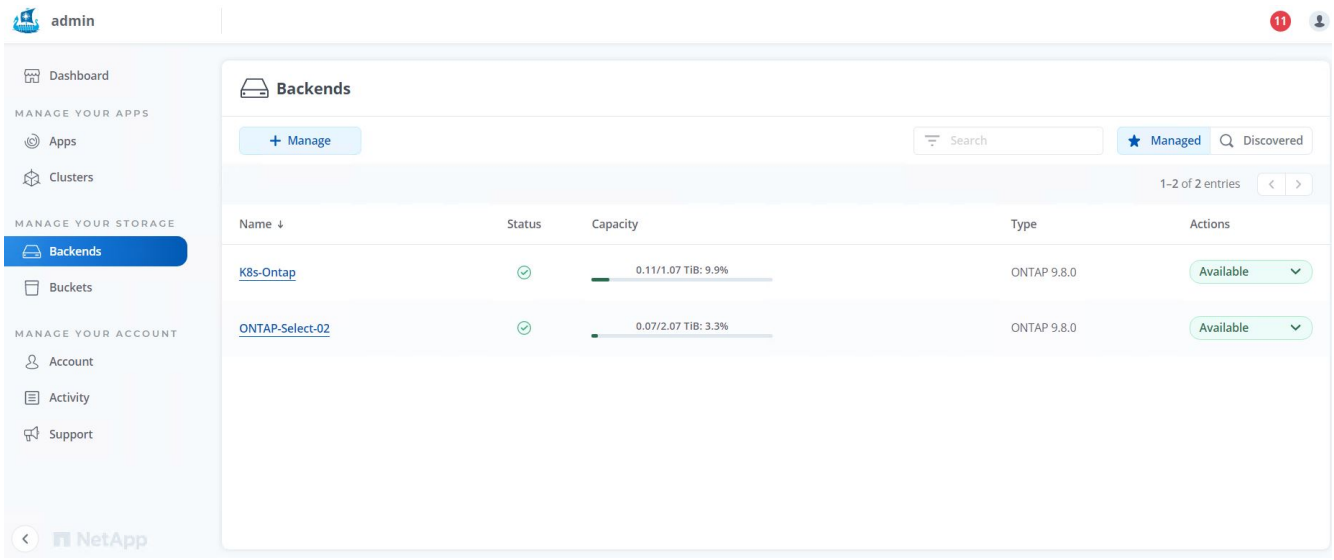
4. Import ONTAP clusters as storage resources to be managed as backends by Astra Control Center. When OpenShift clusters are added to Astra and a storageclass is configured, it automatically discovers and inspects the ONTAP cluster backing the storageclass but does not import it into the Astra Control Center to be managed.



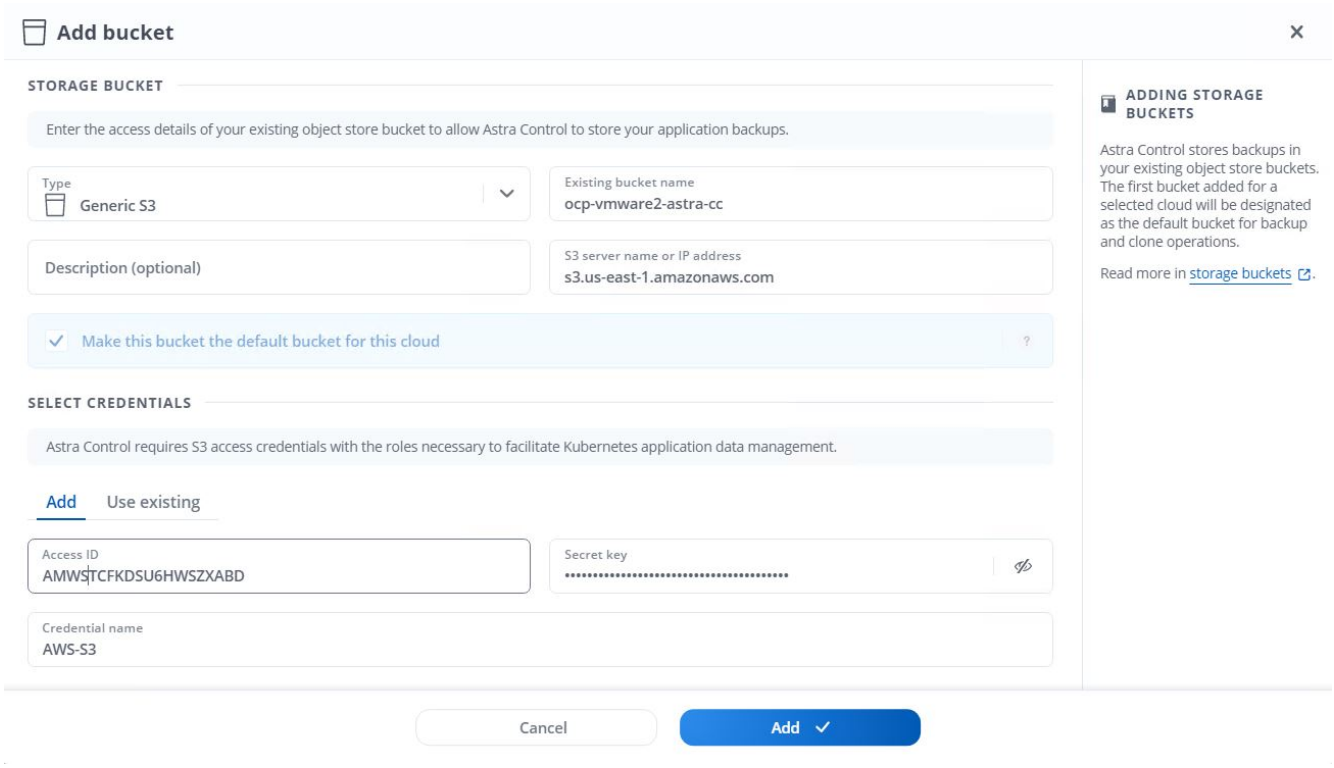
5. To import the ONTAP clusters, go to Backends, click the dropdown, and select Manage next to the ONTAP cluster to be managed. Enter the ONTAP cluster credentials, click Review Information, and then click Import Storage Backend.



6. After the backends are added, the status changes to Available. These backends now have the information about the persistent volumes in the OpenShift cluster and the corresponding volumes on the ONTAP system.



7. For backup and restore across OpenShift clusters using Astra Control Center, you must provision an object storage bucket that supports the S3 protocol. Currently supported options are ONTAP S3, StorageGRID, and AWS S3. For the purpose of this installation, we are going to configure an AWS S3 bucket. Go to Buckets, click Add bucket, and select Generic S3. Enter the details about the S3 bucket and credentials to access it, click the checkbox "Make this bucket the default bucket for the cloud," and then click Add.



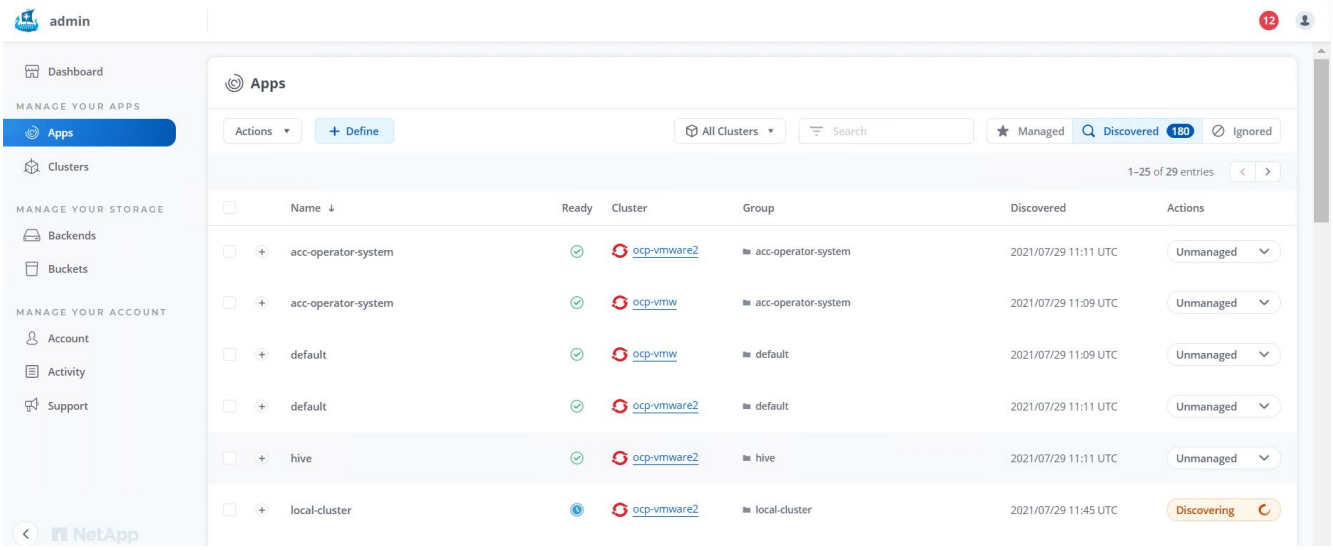
Next: Choose the Applications To Protect.

Choose the applications to protect

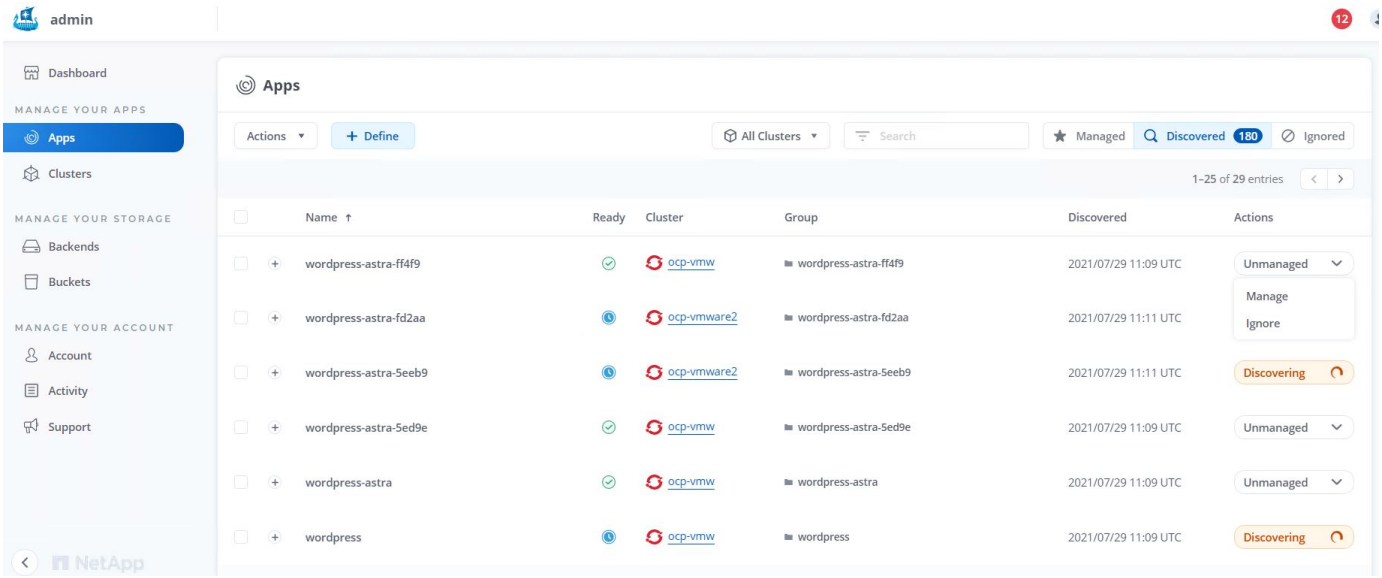
After you have registered your Red Hat OpenShift clusters, you can discover the applications that are deployed and manage them via the Astra Control Center.

Manage applications

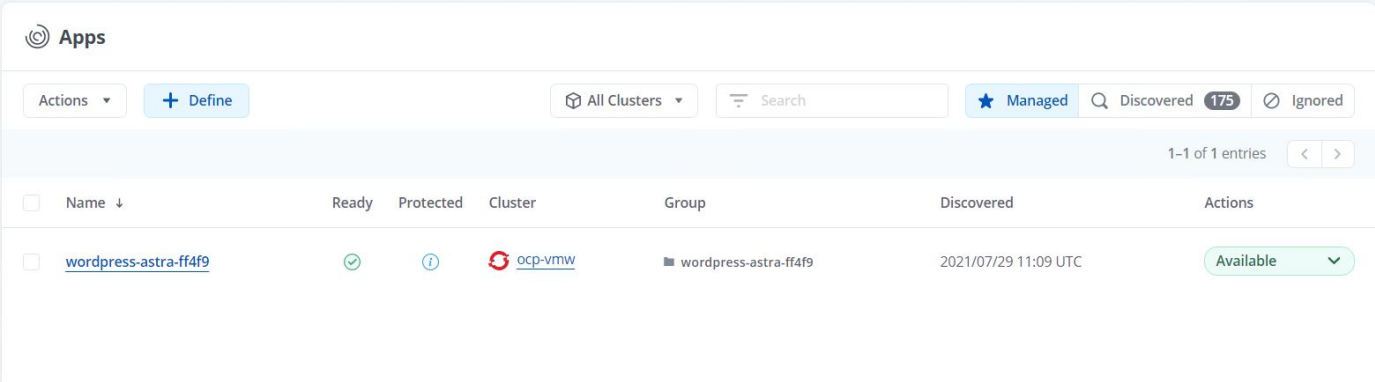
1. After the OpenShift clusters and ONTAP backends are registered with the Astra Control Center, the control center automatically starts discovering the applications in all the namespaces that are using the storageclass configured with the specified ONTAP backend.



2. Navigate to Apps > Discovered and click the dropdown menu next to the application you would like to manage using Astra. Then click Manage.



1. The application enters the Available state and can be viewed under the Managed tab in the Apps section.



Next: Protect Your applications.

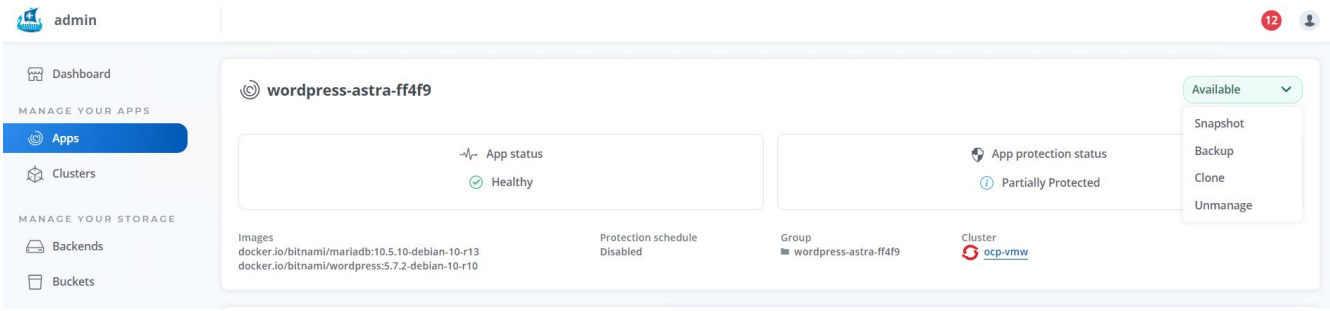
Protect your applications

After application workloads are managed by Astra Control Center, you can configure the protection settings for those workloads.

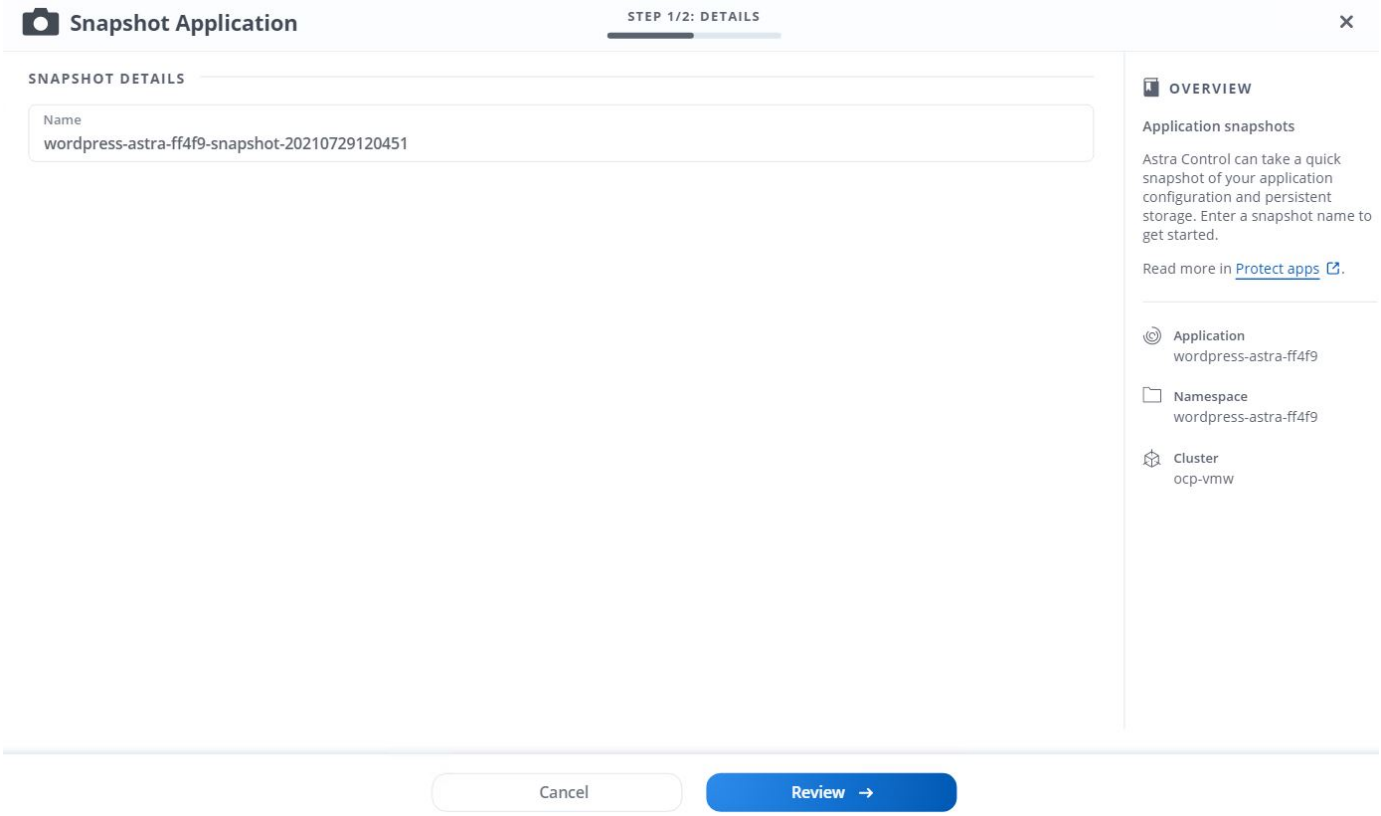
Creating an application snapshot

A snapshot of an application creates an ONTAP Snapshot copy that can be used to restore or clone the application to a specific point in time based on that Snapshot copy.

1. To take a snapshot of the application, navigate to the Apps > Managed tab and click the application you would like to make a Snapshot copy of. Click the dropdown menu next to the application name and click Snapshot.



2. Enter the snapshot details, click Review, and then click Snapshot. It takes about a minute to create the snapshot, and the status becomes Available after the snapshot is successfully created.



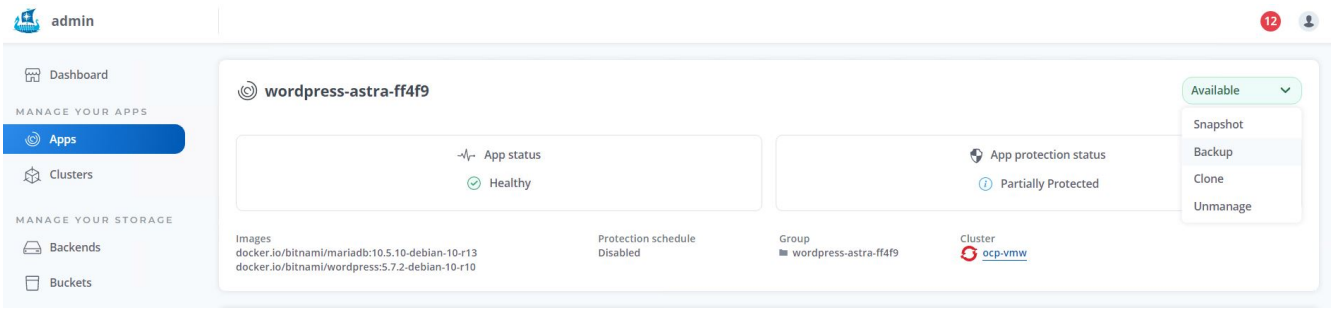
Creating an application backup

A backup of an application captures the active state of the application and the configuration of it's resources, coverts them into files, and stores them in a remote object storage bucket.

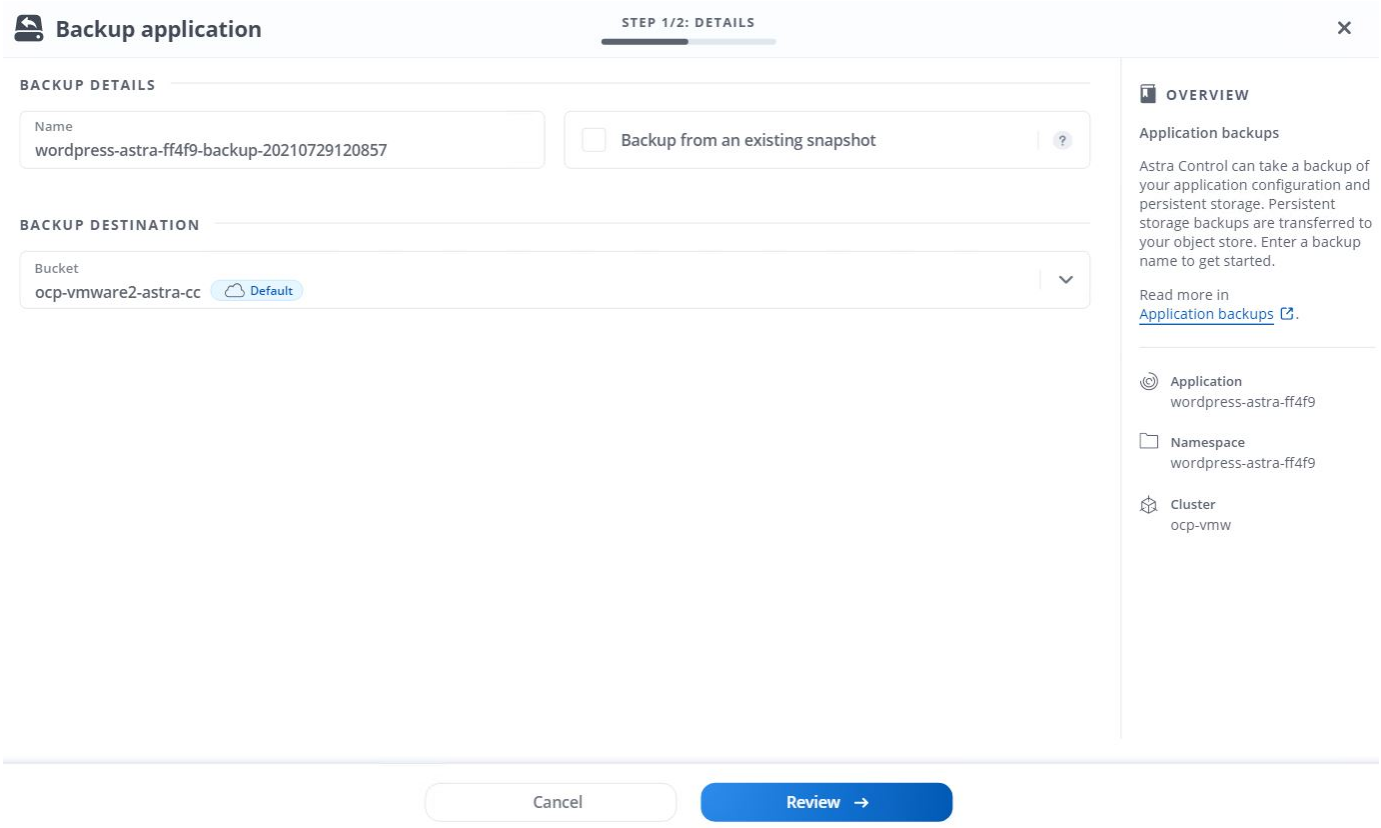
For the backup and restore of managed applications in the Astra Control Center, you must configure superuser settings for the backing ONTAP systems as a prerequisite. To do so, enter the following commands.

```
ONTAP::> export-policy rule modify -vserver ocp-trident -policyname
default -ruleindex 1 -superuser sys
ONTAP::> export-policy rule modify -policyname default -ruleindex 1 -anon
65534 -vserver ocp-trident
```

1. To create a backup of the managed application in the Astra Control Center, navigate to the Apps > Managed tab and click the application that you want to take a backup of. Click the dropdown menu next to the application name and click Backup.



2. Enter the backup details, select the object storage bucket to hold the backup files, click Review, and, after reviewing the details, click Backup. Depending on the size of the application and data, the backup can take several minutes, and the status of the backup becomes Available after the backup is completed successfully.

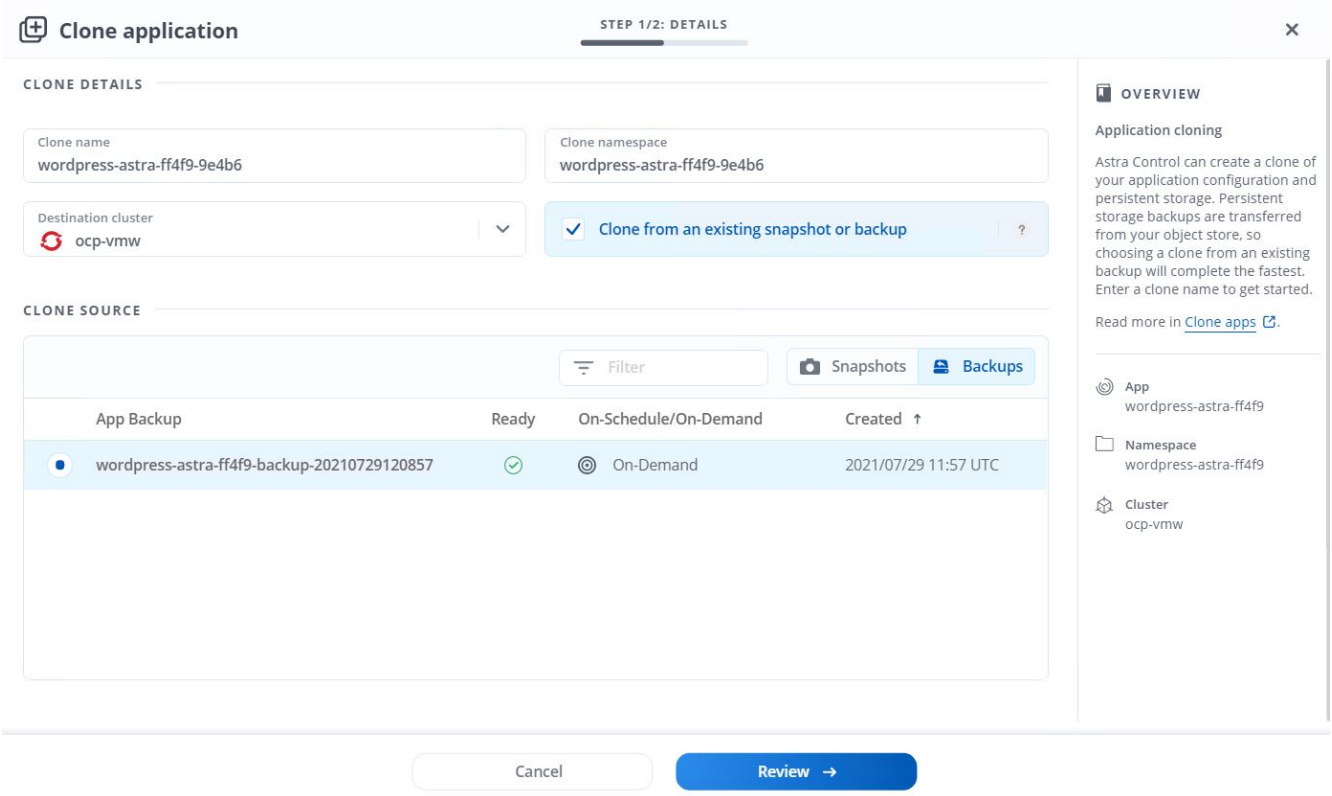


Restoring or cloning an application

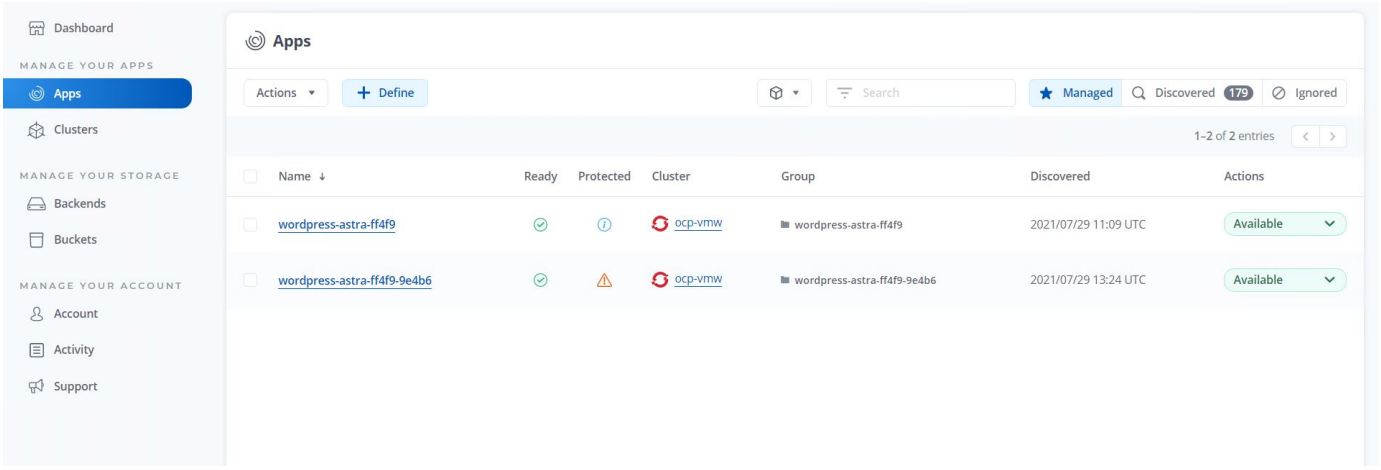
At the push of a button, you can restore an application to the originating cluster or clone it to a remote cluster for dev/test or application protection and disaster recovery purposes.

1. To restore or clone an application, navigate to the Apps > Managed tab and click the app in question. Click the dropdown menu next to the application name and click Clone.

2. Enter the details of the new namespace, select the cluster you want to restore or clone it to, and choose if you want to restore or clone it from an existing snapshot or from a backup of the current state of the application. Then click Review and click Clone after you have reviewed the details.



3. The new application goes to the Discovering state while Astra Control Center creates the application on the selected cluster. After all the resources of the application are installed and detected by Astra, the application goes to the Available state.



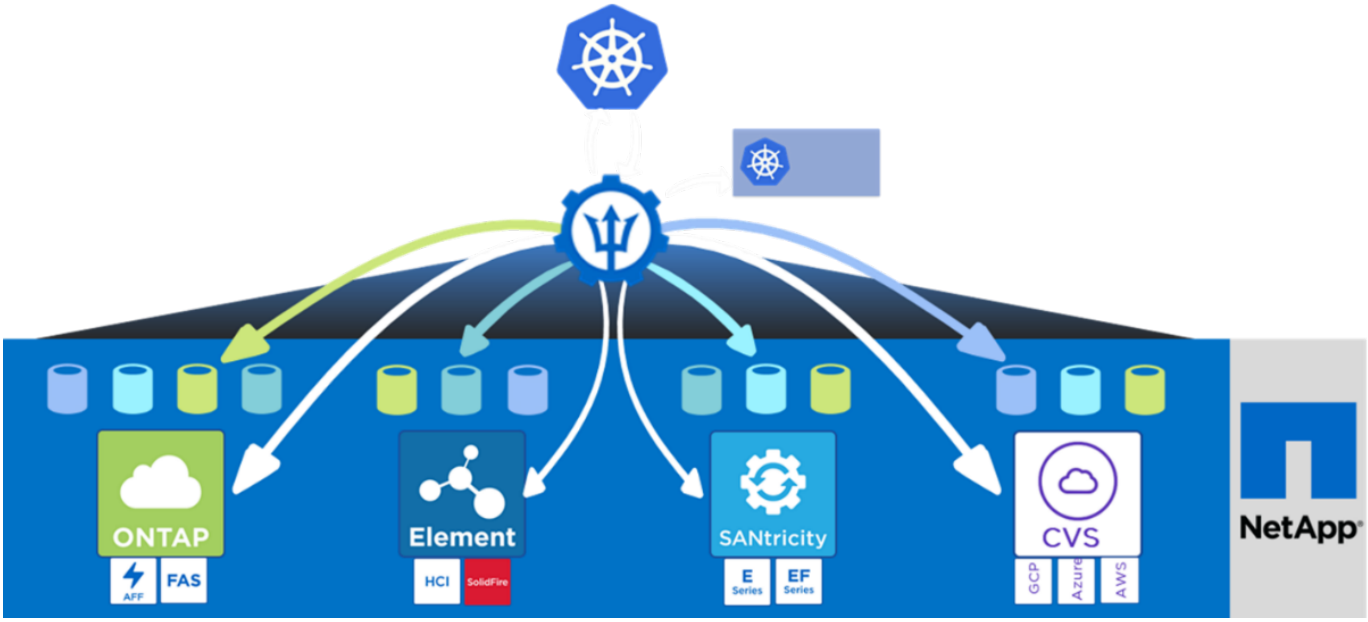
Next: Solution Validation/Use Cases.

Astra Trident Overview

Astra Trident is an open-source and fully supported storage orchestrator for containers and Kubernetes distributions, including Red Hat OpenShift. Trident works with the entire NetApp storage portfolio, including the NetApp ONTAP and Element storage systems, and it also supports NFS and iSCSI connections. Trident accelerates 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 enable advanced storage features, including compression, specific disk types, or QoS levels that

guarantee a certain level of performance. After they are defined, these backends can be used by developers in their projects to create persistent volume claims (PVCs) and to attach persistent storage to their containers on demand.



Astra Trident has a rapid development cycle, and just like Kubernetes, is released four times a year.

The latest version of Astra Trident is 21.07 released in July 2021. A support matrix for what version of Trident has been tested with which Kubernetes distribution can be found here.

Starting with the 20.04 release, Trident setup is performed by the Trident operator. The operator makes large scale deployments easier and provides additional support including self healing for pods that are deployed as a part of the Trident install.

With the 21.01 release, a Helm chart was made available to ease the installation of the Trident Operator.

Download Astra Trident

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

1. Download the installation archive to the admin workstation and extract the contents. The current version of Trident is 21.07, which can be downloaded here.

```
[netapp-user@rhel7 ~]$ wget
https://github.com/NetApp/trident/releases/download/v21.07.1/trident-
installer-21.07.1.tar.gz
--2021-05-06 15:17:30--
https://github.com/NetApp/trident/releases/download/v21.07.1/trident-
installer-21.07.1.tar.gz
Resolving github.com (github.com)... 140.82.114.3
Connecting to github.com (github.com)|140.82.114.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-
releases.githubusercontent.com/77179634/a4fa9f00-a9f2-11eb-9053-
```

```
98e8e573d4ae?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAIWNJYAX4CSVEH53A%2F20210506%2Fus-east-
1%2Fs3%2Faws4_request&X-Amz-Date=20210506T191643Z&X-Amz-Expires=300&X-
Amz-
Signature=8a49a2a1e08c147d1ddd8149ce45a5714f9853fee19bb1c507989b9543eb36
30&X-Amz-
SignedHeaders=host&actor_id=0&key_id=0&repo_id=77179634&response-
content-disposition=attachment%3B%20filename%3Dtrident-installer-
21.07.1.tar.gz&response-content-type=application%2Foctet-stream
[following]
--2021-05-06 15:17:30--  https://github-
releases.githubusercontent.com/77179634/a4fa9f00-a9f2-11eb-9053-
98e8e573d4ae?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-
Credential=AKIAIWNJYAX4CSVEH53A%2F20210506%2Fus-east-
1%2Fs3%2Faws4_request&X-Amz-Date=20210506T191643Z&X-Amz-Expires=300&X-
Amz-
Signature=8a49a2a1e08c147d1ddd8149ce45a5714f9853fee19bb1c507989b9543eb36
30&X-Amz-
SignedHeaders=host&actor_id=0&key_id=0&repo_id=77179634&response-
content-disposition=attachment%3B%20filename%3Dtrident-installer-
21.07.1.tar.gz&response-content-type=application%2Foctet-stream
Resolving github-releases.githubusercontent.com (github-
releases.githubusercontent.com)... 185.199.108.154, 185.199.109.154,
185.199.110.154, ...
Connecting to github-releases.githubusercontent.com (github-
releases.githubusercontent.com)|185.199.108.154|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 38349341 (37M) [application/octet-stream]
Saving to: 'trident-installer-21.07.1.tar.gz'

100%[=====================================================================
==========================================>] 38,349,341  88.5MB/s
in 0.4s

2021-05-06 15:17:30 (88.5 MB/s) - 'trident-installer-21.07.1.tar.gz'
saved [38349341/38349341]
```

2. Extract the Trident install from the downloaded bundle.

```
[netapp-user@rhel7 ~]$ tar -xzf trident-installer-21.07.1.tar.gz
[netapp-user@rhel7 ~]$ cd trident-installer/
[netapp-user@rhel7 trident-installer]$
```

Install the Trident Operator with Helm

1. 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.

```
[netapp-user@rhel7 trident-installer]$ export KUBECONFIG=~/ocp-
install/auth/kubeconfig
```

2. Run the Helm command to install the Trident operator from the tarball in the helm directory while creating the trident namespace in your user cluster.

```
[netapp-user@rhel7 trident-installer]$ helm install trident
helm/trident-operator-21.07.1.tgz --create-namespace --namespace trident
NAME: trident
LAST DEPLOYED: Fri May  7 12:54:25 2021
NAMESPACE: trident
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Thank you for installing trident-operator, which will deploy and manage
NetApp's Trident CSI
storage provisioner for Kubernetes.

Your release is named 'trident' and is installed into the 'trident'
namespace.
Please note that there must be only one instance of Trident (and
trident-operator) in a Kubernetes cluster.

To configure Trident to manage storage resources, you will need a copy
of tridentctl, which is
available in pre-packaged Trident releases.  You may find all Trident
releases and source code
online at https://github.com/NetApp/trident.

To learn more about the release, try:

  $ helm status trident
  $ helm get all trident
```

3. 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.

```
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-5z45l                   1/2     Running   2          30s
trident-csi-696b685cf8-htdb2        6/6     Running   0          30s
trident-csi-b74p2                   2/2     Running   0          30s
trident-csi-lrw4n                   2/2     Running   0          30s
trident-operator-7c748d957-gr2gw    1/1     Running   0          36s

[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.07.1        | 21.07.1        |
+----------------+----------------+
```

In some cases, customer environments might require the customization of the Trident deployment. In these cases, it is also possible to manually install the Trident operator and update the included manifests to customize the deployment.

Manually install the Trident Operator

1. 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.

```
[netapp-user@rhel7 trident-installer]$ export KUBECONFIG=~/ocp-
install/auth/kubeconfig
```

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

```
[netapp-user@rhel7 trident-installer]$ oc create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
customresourcedefinition.apiextensions.k8s.io/tridentorchestrators.tride
nt.netapp.io created
```

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

```
[netapp-user@rhel7 trident-installer]$ oc apply -f deploy/namespace.yaml
namespace/trident created
```

4. 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.

```
[netapp-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
```

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

```
[netapp-user@rhel7 trident-installer]$ oc get deployment -n trident
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator   1/1     1            1           23s
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-operator-66f48895cc-lzczk   1/1     Running   0          41s
```

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

```
[netapp-user@rhel7 trident-installer]$ oc create -f
deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created
[netapp-user@rhel7 trident-installer]$ oc describe torc trident
Name:         trident
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  trident.netapp.io/v1
Kind:         TridentOrchestrator
Metadata:
  Creation Timestamp:  2021-05-07T17:00:28Z
  Generation:          1
  Managed Fields:
    API Version:  trident.netapp.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      f:spec:
        .:
        f:debug:
        f:namespace:
    Manager:      kubectl-create
    Operation:    Update
    Time:         2021-05-07T17:00:28Z
    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:            2021-05-07T17:00:28Z
  Resource Version:  931421
  Self Link:
/apis/trident.netapp.io/v1/tridentorchestrators/trident
  UID:               8a26a7a6-dde8-4d55-9b66-a7126754d81f
Spec:
  Debug:      true
  Namespace:  trident
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:21.01
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Enable Node Prep:          false
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:           30
```

```
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident Image:        netapp/trident:21.07.1
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v21.07.1
Events:
  Type    Reason      Age   From                           Message
  ----    ------      ----  ----                           -------
  Normal  Installing  80s   trident-operator.netapp.io  Installing
Trident
  Normal  Installed   68s   trident-operator.netapp.io  Trident
installed
```

7. 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.

```
[netapp-user@rhel7 trident-installer]$ oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-bb64c6cb4-lmd6h         6/6     Running   0          82s
trident-csi-gn59q                   2/2     Running   0          82s
trident-csi-m4szj                   2/2     Running   0          82s
trident-csi-sb9k9                   2/2     Running   0          82s
trident-operator-66f48895cc-lzczk   1/1     Running   0          2m39s

[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.07.1          | 21.07.1          |
+----------------+----------------+
```

Prepare worker nodes for storage

Most Kubernetes distributions come with the packages and utilities to mount NFS backends installed by default, including Red Hat OpenShift.

To prepare worker nodes to allow for the mapping of block storage volumes through the iSCSI protocol, you must install the necessary packages to support that functionality.

In Red Hat OpenShift, this is handled by applying an MCO (Machine Config Operator) to your cluster after it is deployed.

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

1. Log into the OCP web console and navigate to Compute > Machine Configs. Click Create Machine Config. Copy and paste the YAML file and click Create.

When not using multipathing:

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: worker
  name: 99-worker-element-iscsi
spec:
  config:
    ignition:
      version: 3.2.0
    systemd:
      units:
        - name: iscsid.service
          enabled: true
          state: started
  osImageURL: ""
```

When using multipathing:

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 99-worker-ontap-iscsi
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-
8;base64,ZGVmYXVsdHMgewogICAgICAgIHVzZXJfZnJpZW5kbHlfbmFtZXMgbm8KICAgICAgI
CBmaW5kX211bHRpcGF0aHMgbm8KfQoKYmxhY2tsaXN0X2V4Y2VwdGlvbnMgewogICAgICAgIHB
yb3BlcnR5ICIoU0NTSV9JREVOVF98SURfV1dOKSIKfQoKYmxhY2tsaXN0IHsKfQoK
          verification: {}
        filesystem: root
        mode: 400
        path: /etc/multipath.conf
    systemd:
      units:
        - name: iscsid.service
          enabled: true
          state: started
        - name: multipathd.service
          enabled: true
          state: started
  osImageURL: ""
```

2. After the configuration is created, it takes approximately 20 to 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 into the worker nodes to confirm that the iscsid service is running (and the multipathd service is running if using multipathing).

```
[netapp-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

[netapp-user@rhel7 openshift-deploy]$ ssh core@10.61.181.22 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 2021-05-26 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

[netapp-user@rhel7 openshift-deploy]$ ssh core@10.61.181.22 sudo
systemctl status multipathd
 ● multipathd.service - Device-Mapper Multipath Device Controller
   Loaded: loaded (/usr/lib/systemd/system/multipathd.service; enabled;
vendor preset: enabled)
   Active: active (running) since Tue 2021-05-26 13:36:22 UTC; 3 min ago
  Main PID: 918 (multipathd)
    Status: "up"
    Tasks: 7
    Memory: 13.7M
    CPU: 57ms
    CGroup: /system.slice/multipathd.service
            └─918 /sbin/multipathd -d -s
```

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.

Create storage-system backends

After completing the Astra Trident Operator install, you must configure the backend for the specific NetApp storage platform you are using. Follow the links below in order to continue the setup and configuration of Astra

Trident.

- NetApp ONTAP NFS
- NetApp ONTAP iSCSI
- NetApp Element iSCSI

Next: Solution Validation/Use Cases: Red Hat OpenShift with NetApp.

NetApp ONTAP NFS configuration

To enable Trident integration with the NetApp ONTAP storage system, you must create a backend that enables communication with the storage system.

1. There are sample backend files available in the downloaded installation archive in the `sample-input` folder hierarchy. For NetApp ONTAP systems serving NFS, copy the `backend-ontap-nas.json` file to your working directory and edit the file.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/ontap-nas/backend-ontap-nas.json ./
[netapp-user@rhel7 trident-installer]$ vi backend-ontap-nas.json
```

2. Edit the backendName, managementLIF, dataLIF, svm, username, and password values in this file.

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "ontap-nas+10.61.181.221",
    "managementLIF": "172.21.224.201",
    "dataLIF": "10.61.181.221",
    "svm": "trident_svm",
    "username": "cluster-admin",
    "password": "password"
}
```

It is a best practice to define the custom backendName value as a combination of the storageDriverName and the dataLIF that is serving NFS for easy identification.

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

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-ontap-nas.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas+10.61.181.221 | ontap-nas      | be7a619d-c81d-445c-b80c-
5c87a73c5b1e | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

4. 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.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

5. 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, which 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: "ontap-san"
```

There is an optional field called `fsType` that is defined in this file. This line can be deleted in NFS backends.

6. Run the `oc` command to create the storage class.

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

7. 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.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-
basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

8. The only edit that must be made to this file is ensuring that the `storageClassName` field matches the one just created. The PVC definition can be further customized as required by the workload to be provisioned.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

9. Create the PVC by issuing the `oc` 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.

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-b4370d37-0fa4-4c17-bd86-94f96c94b42d   1Gi
RWO            basic-csi      7s
```

Next: Solution validation/use cases.

NetApp ONTAP iSCSI configuration

To enable Trident integration with the NetApp ONTAP storage system, you must create a backend that enables communication with the storage system.

1. There are sample backend files available in the downloaded installation archive in the `sample-input` folder hierarchy. For NetApp ONTAP systems serving iSCSI, copy the `backend-ontap-san.json` file to your working directory and edit the file.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/ontap-san/backend-ontap-san.json ./
[netapp-user@rhel7 trident-installer]$ vi backend-ontap-san.json
```

2. Edit the managementLIF, dataLIF, svm, username, and password values in this file.

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "172.21.224.201",
  "dataLIF": "10.61.181.240",
  "svm": "trident_svm",
  "username": "admin",
  "password": "password"
}
```

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

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-ontap-san.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| ontapsan_10.61.181.241 | ontap-san      | 6788533c-7fea-4a35-b797-
fb9bb3322b91 | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

4. 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.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

5. 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, which 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: "ontap-san"
```

There is an optional field called `fsType` that is defined in this file. In iSCSI backends, this value can be set to a specific Linux filesystem type (XFS, ext4, etc) or can be deleted to allow OpenShift to decide what filesystem to use.

6. Run the `oc` command to create the storage class.

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

7. 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.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-
basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

8. The only edit that must be made to this file is ensuring that the `storageClassName` field matches the one just created. The PVC definition can be further customized as required by the workload to be provisioned.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

9. Create the PVC by issuing the `oc` 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.

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-7ceac1ba-0189-43c7-8f98-094719f7956c   1Gi
RWO            basic-csi      3s
```

Next: Solution validation/use cases.

NetApp Element iSCSI configuration

To enable Trident integration with the NetApp Element storage system, you must create a backend that enables communication with the storage system using the iSCSI protocol.

1. There are sample backend files available in the downloaded installation archive in the `sample-input` folder hierarchy. For NetApp Element systems serving iSCSI, copy the `backend-solidfire.json` file to your working directory and edit the file.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/backends-
samples/solidfire/backend-solidfire.json ./
[netapp-user@rhel7 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": "10.61.180.200:3260",
    "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}}]
}
```

2. With this back-end file in place, run the following command to create your first backend.

```
[netapp-user@rhel7 trident-installer]$ ./tridentctl -n trident create
backend -f backend-solidfire.json
+------------------------+----------------
+--------------------------------------+--------+---------+
|          NAME          | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+------------------------+----------------
+--------------------------------------+--------+---------+
| solidfire_10.61.180.200 | solidfire-san  | b90783ee-e0c9-49af-8d26-
3ea87ce2efdf | online |       0 |
+------------------------+----------------
+--------------------------------------+--------+---------+
```

3. 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.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/storage-class-
samples/storage-class-csi.yaml.templ ./storage-class-basic.yaml
[netapp-user@rhel7 trident-installer]$ vi storage-class-basic.yaml
```

4. 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, which 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"
```

There is an optional field called `fsType` that is defined in this file. In iSCSI backends, this value can be set to a specific Linux filesystem type (XFS, ext4, and so on), or it can be deleted to allow OpenShift to decide what filesystem to use.

5. Run the `oc` command to create the storage class.

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

6. 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.

```
[netapp-user@rhel7 trident-installer]$ cp sample-input/pvc-samples/pvc-
basic.yaml ./
[netapp-user@rhel7 trident-installer]$ vi pvc-basic.yaml
```

7. The only edit that must be made to this file is ensuring that the `storageClassName` field matches the one just created. The PVC definition can be further customized as required by the workload to be provisioned.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

8. Create the PVC by issuing the `oc` 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.

```
[netapp-user@rhel7 trident-installer]$ oc create -f pvc-basic.yaml
persistentvolumeclaim/basic created

[netapp-user@rhel7 trident-installer]$ oc get pvc
NAME    STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
basic   Bound    pvc-3445b5cc-df24-453d-a1e6-b484e874349d   1Gi
RWO            basic-csi      5s
```

Next: Solution validation/use cases.

Advanced Configuration Options For OpenShift

Exploring load balancer options: Red Hat OpenShift with NetApp

In most cases, Red Hat OpenShift makes applications available to the outside world through routes. A service is exposed by giving it an externally reachable hostname. The defined route and the endpoints identified by its service can be consumed by an OpenShift router to provide this named connectivity to external clients.

However in some cases, applications require the deployment and configuration of customized load balancers

to expose the appropriate services. One example of this is NetApp Astra Control Center. To meet this need, we have evaluated a number of custom load balancer options. Their installation and configuration are described in this section.

The following pages have additional information about load balancer options validated in the Red Hat OpenShift with NetApp solution:

- MetalLB
- F5 BIG-IP

Next: Solution validation/use cases: Red Hat OpenShift with NetApp.

Installing MetalLB load balancers: Red Hat OpenShift with NetApp

This page lists the installation and configuration instructions for the MetalLB load balancer.

MetalLB is a self-hosted network load balancer installed on your OpenShift cluster that allows the creation of OpenShift services of type load balancer in clusters that do not run on a cloud provider. The two main features of MetalLB that work together to support LoadBalancer services are address allocation and external announcement.

MetalLB configuration options

Based on how MetalLB announces the IP address assigned to LoadBalancer services outside of the OpenShift cluster, it operates in two modes:

- **Layer 2 mode.** In this mode, one node in the OpenShift cluster takes ownership of the service and responds to ARP requests for that IP to make it reachable outside of the OpenShift cluster. Because only the node advertises the IP, it has a bandwidth bottleneck and slow failover limitations. For more information, see the documentation here.
- **BGP mode.** In this mode, all nodes in the OpenShift cluster establish BGP peering sessions with a router and advertise the routes to forward traffic to the service IPs. The prerequisite for this is to integrate MetalLB with a router in that network. Owing to the hashing mechanism in BGP, it has certain limitation when IP-to-Node mapping for a service changes. For more information, refer to the documentation here.

For the purpose of this document, we are configuring MetalLB in layer-2 mode.

Installing The MetalLB Load Balancer

1. Download the MetalLB resources.

```
[netapp-user@rhel7 ~]$ wget
https://raw.githubusercontent.com/metallb/metallb/v0.10.2/manifests/name
space.yaml
[netapp-user@rhel7 ~]$ wget
https://raw.githubusercontent.com/metallb/metallb/v0.10.2/manifests/meta
llb.yaml
```

2. Edit file `metallb.yaml` and remove `spec.template.spec.securityContext` from controller Deployment and the speaker DaemonSet.

Lines to be deleted:

```
securityContext:
  runAsNonRoot: true
  runAsUser: 65534
```

3. Create the `metallb-system` namespace.

```
[netapp-user@rhel7 ~]$ oc create -f namespace.yaml
namespace/metallb-system created
```

4. Create the MetalLB CR.

```
[netapp-user@rhel7 ~]$ oc create -f metallb.yaml
podsecuritypolicy.policy/controller created
podsecuritypolicy.policy/speaker created
serviceaccount/controller created
serviceaccount/speaker created
clusterrole.rbac.authorization.k8s.io/metallb-system:controller created
clusterrole.rbac.authorization.k8s.io/metallb-system:speaker created
role.rbac.authorization.k8s.io/config-watcher created
role.rbac.authorization.k8s.io/pod-lister created
role.rbac.authorization.k8s.io/controller created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:controller
created
clusterrolebinding.rbac.authorization.k8s.io/metallb-system:speaker
created
rolebinding.rbac.authorization.k8s.io/config-watcher created
rolebinding.rbac.authorization.k8s.io/pod-lister created
rolebinding.rbac.authorization.k8s.io/controller created
daemonset.apps/speaker created
deployment.apps/controller created
```

5. Before configuring the MetalLB speaker, grant the speaker DaemonSet elevated privileges so that it can perform the networking configuration required to make the load balancers work.

```
[netapp-user@rhel7 ~]$ oc adm policy add-scc-to-user privileged -n
metallb-system -z speaker
clusterrole.rbac.authorization.k8s.io/system:openshift:scc:privileged
added: "speaker"
```

6. Configure MetalLB by creating a `ConfigMap` in the `metallb-system` namespace.

```
[netapp-user@rhel7 ~]$ vim metallb-config.yaml

apiVersion: v1
kind: ConfigMap
metadata:
  namespace: metallb-system
  name: config
data:
  config: |
    address-pools:
    - name: default
      protocol: layer2
      addresses:
      - 10.63.17.10-10.63.17.200

[netapp-user@rhel7 ~]$ oc create -f metallb-config.yaml
configmap/config created
```

7. Now when loadbalancer services are created, MetalLB assigns an externalIP to the services and advertises the IP address by responding to ARP requests.

> If you wish to configure MetalLB in BGP mode, skip step 6 above and follow the procedure in the MetalLB documentation here.

Next: Solution validation/use cases: Red Hat OpenShift with NetApp.

Installing F5 BIG-IP Load Balancers

F5 BIG-IP is an Application Delivery Controller (ADC) that offers a broad set of advanced production-grade traffic management and security services like L4-L7 load balancing, SSL/TLS offload, DNS, firewall and many more. These services drastically increase the availability, security and performance of your applications.

F5 BIG-IP can be deployed and consumed in various ways, on dedicated hardware, in the cloud, or as a virtual appliance on-premises. Refer to the documentation here to explore and deploy F5 BIG-IP as per requirement.

For efficient integration of F5 BIG-IP services with Red Hat OpenShift, F5 offers the BIG-IP Container Ingress Service (CIS). CIS is installed as a controller pod that watches OpenShift API for certain Custom Resource Definitions (CRDs) and manages the F5 BIG-IP system configuration. F5 BIG-IP CIS can be configured to control service types LoadBalancers and Routes in OpenShift.

Further, for automatic IP address allocation to service the type LoadBalancer, you can utilize the F5 IPAM controller. The F5 IPAM controller is installed as a controller pod that watches OpenShift API for LoadBalancer services with an ipamLabel annotation to allocate the IP address from a preconfigured pool.

This page lists the installation and configuration instructions for F5 BIG-IP CIS and IPAM controller. As a prerequisite, you must have an F5 BIG-IP system deployed and licensed. It must also be licensed for SDN services, which are included by default with the BIG-IP VE base license.

F5 BIG-IP can be deployed in standalone or cluster mode. For the purpose of this validation, F5 BIG-IP was deployed in standalone mode, but, for production purposes, it is preferred to have a cluster of BIG-IPs to avoid a single point of failure.

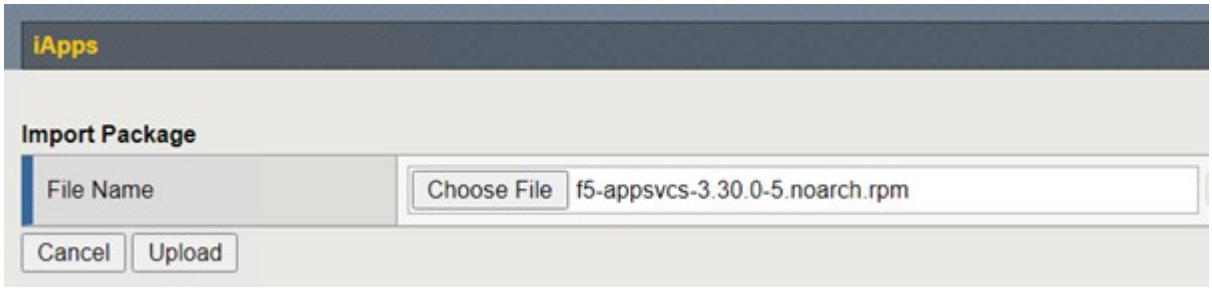
An F5 BIG-IP system can be deployed on dedicated hardware, in the cloud, or as a virtual appliance on-premises with versions greater than 12.x for it to be integrated with F5 CIS. For the purpose of this document, the F5 BIG-IP system was validated as a virtual appliance, for example using the BIG-IP VE edition.

Validated releases

Technology	Software version
Red Hat OpenShift	4.6 EUS, 4.7
F5 BIG-IP VE edition	16.1.0
F5 Container Ingress Service	2.5.1
F5 IPAM Controller	0.1.4
F5 AS3	3.30.0

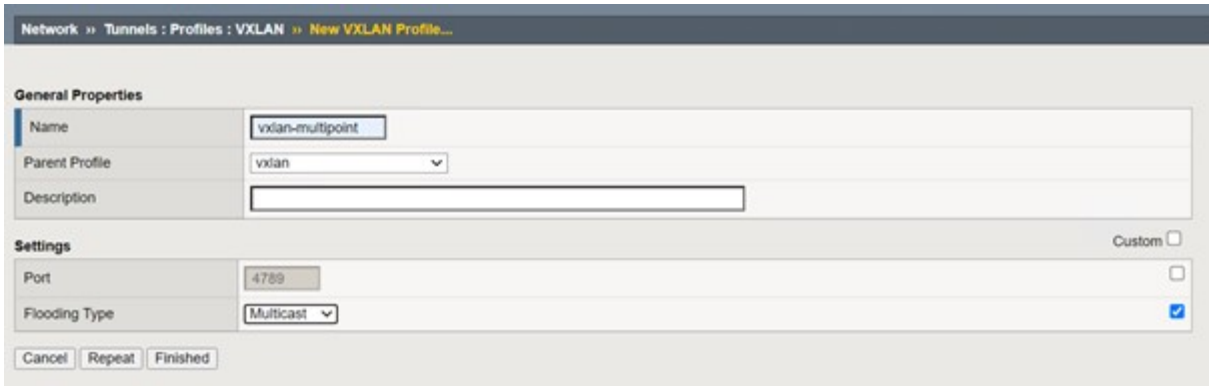
Installation

1. Install the F5 Application Services 3 extension to allow BIG-IP systems to accept configurations in JSON instead of imperative commands. Go to F5 AS3 GitHub repository, and download the latest RPM file.
2. Log into F5 BIG-IP system, navigate to iApps > Package Management LX and click Import.
3. Click Choose File and select the downloaded AS3 RPM file, click OK, and then click Upload.

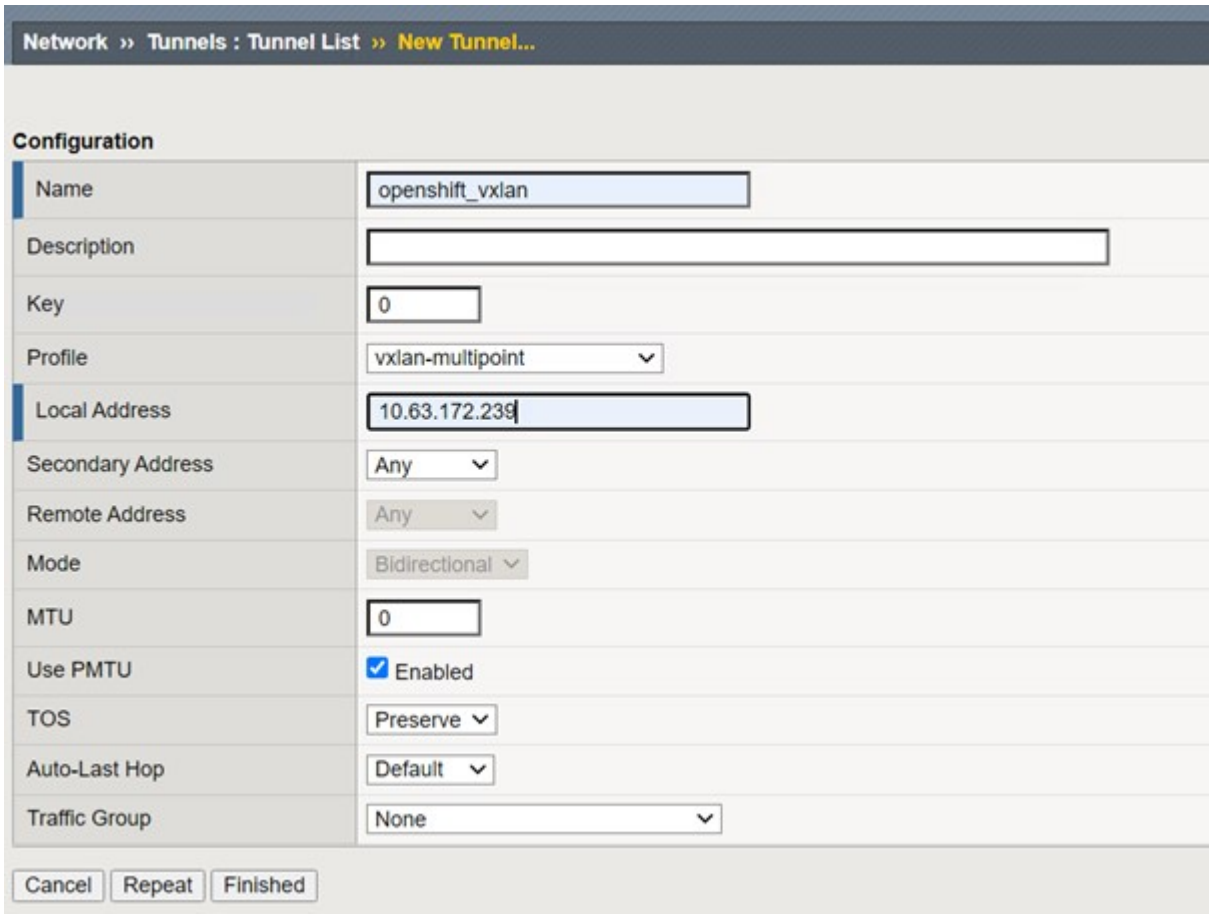


4. Confirm that the AS3 extension is installed successfully.

5. Next configure the resources required for communication between OpenShift and BIG-IP systems. First create a tunnel between OpenShift and the BIG-IP server by creating a VXLAN tunnel interface on the BIG-IP system for OpenShift SDN. Navigate to Network > Tunnels > Profiles, click Create, and set the Parent Profile to vxlan and the Flooding Type to Multicast. Enter a name for the profile and click Finished.



6. Navigate to Network > Tunnels > Tunnel List, click Create, and enter the name and local IP address for the tunnel. Select the tunnel profile that was created in the previous step and click Finished.



7. Log into the Red Hat OpenShift cluster with cluster-admin privileges.
8. Create a hostsubnet on OpenShift for the F5 BIG-IP server, which extends the subnet from the OpenShift cluster to the F5 BIG-IP server. Download the host subnet YAML definition.

```
wget https://github.com/F5Networks/k8s-bigip-
ctlr/blob/master/docs/config_examples/openshift/f5-kctlr-openshift-
hostsubnet.yaml
```

9. Edit the host subnet file and add the BIG-IP VTEP (VXLAN tunnel) IP for the OpenShift SDN.

```
apiVersion: v1
kind: HostSubnet
metadata:
  name: f5-server
  annotations:
    pod.network.openshift.io/fixed-vnid-host: "0"
    pod.network.openshift.io/assign-subnet: "true"
# provide a name for the node that will serve as BIG-IP's entry into the
cluster
host: f5-server
# The hostIP address will be the BIG-IP interface address routable to
the
# OpenShift Origin nodes.
# This address is the BIG-IP VTEP in the SDN's VXLAN.
hostIP: 10.63.172.239
```

Change the hostIP and other details as applicable to your environment.

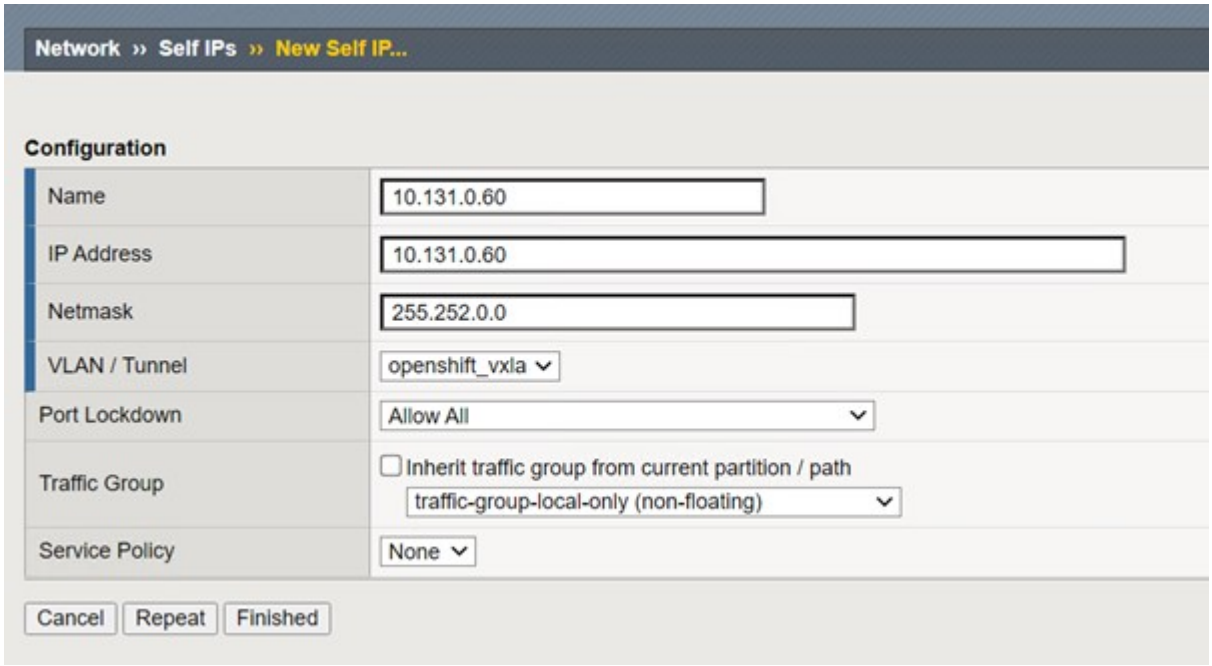
10. Create the HostSubnet resource.

```
[admin@rhel-7 ~]$ oc create -f f5-kctlr-openshift-hostsubnet.yaml

hostsubnet.network.openshift.io/f5-server created
```

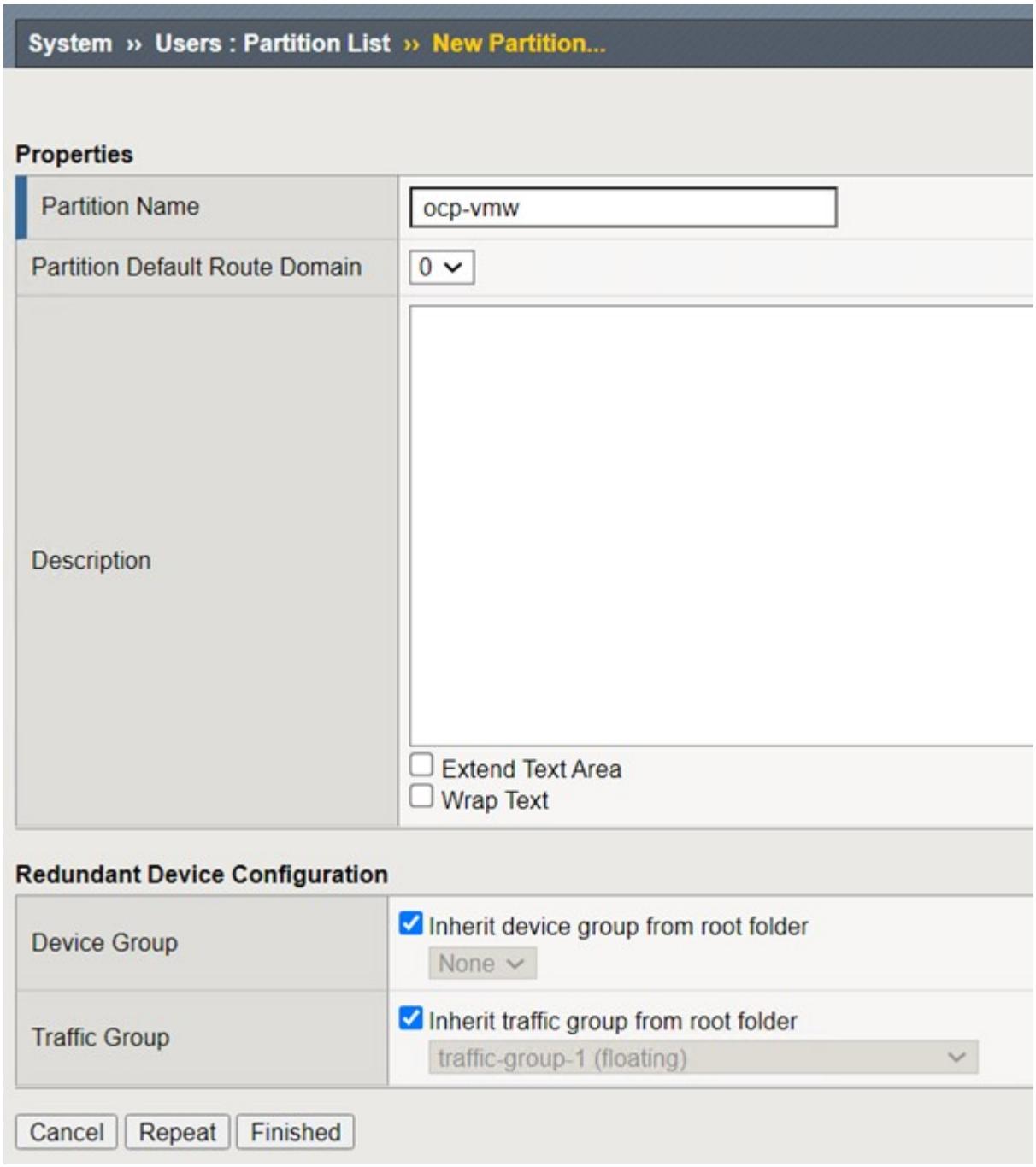
11. Get the cluster IP subnet range for the host subnet created for the F5 BIG-IP server.

```
[admin@rhel-7 ~]$ oc get hostsubnet

NAME                          HOST                          HOST IP
SUBNET          EGRESS CIDRS   EGRESS IPS
f5-server                     f5-server                     10.63.172.239
10.131.0.0/23
ocp-vmw-nszws-master-0        ocp-vmw-nszws-master-0        10.63.172.44
10.128.0.0/23
ocp-vmw-nszws-master-1        ocp-vmw-nszws-master-1        10.63.172.47
10.130.0.0/23
ocp-vmw-nszws-master-2        ocp-vmw-nszws-master-2        10.63.172.48
10.129.0.0/23
ocp-vmw-nszws-worker-r8fh4    ocp-vmw-nszws-worker-r8fh4    10.63.172.7
10.130.2.0/23
ocp-vmw-nszws-worker-tvr46    ocp-vmw-nszws-worker-tvr46    10.63.172.11
10.129.2.0/23
ocp-vmw-nszws-worker-wdxhg    ocp-vmw-nszws-worker-wdxhg    10.63.172.24
10.128.2.0/23
ocp-vmw-nszws-worker-wg8r4    ocp-vmw-nszws-worker-wg8r4    10.63.172.15
10.131.2.0/23
ocp-vmw-nszws-worker-wtgfw    ocp-vmw-nszws-worker-wtgfw    10.63.172.17
10.128.4.0/23
```

12. Create a self IP on OpenShift VXLAN with an IP in OpenShift's host subnet range corresponding to the F5 BIG-IP server. Log into the F5 BIG-IP system, navigate to Network > Self IPs and click Create. Enter an IP from the cluster IP subnet created for F5 BIG-IP host subnet, select the VXLAN tunnel, and enter the other details. Then click Finished.



13. Create a partition in the F5 BIG-IP system to be configured and used with CIS. Navigate to System > Users > Partition List, click Create, and enter the details. Then click Finished.



F5 recommends that no manual configuration be done on the partition that is managed by CIS.

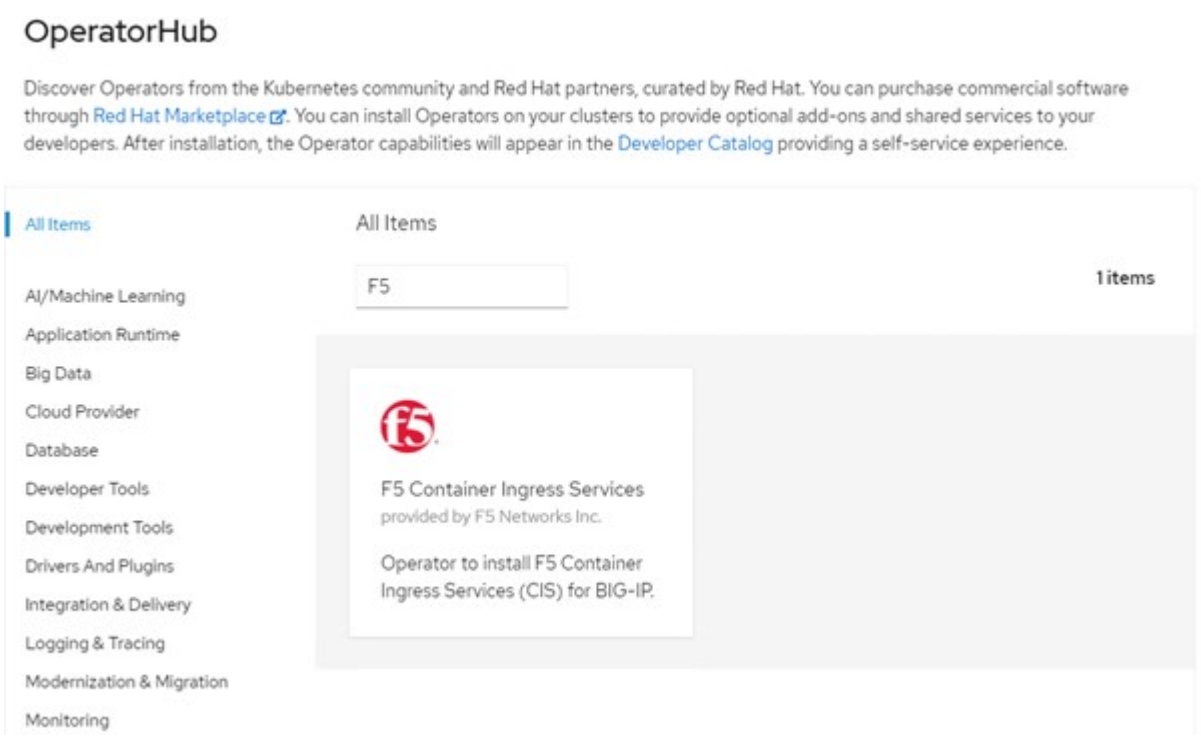
14. Install the F5 BIG-IP CIS using the operator from OperatorHub. Log into the Red Hat OpenShift cluster with cluster-admin privileges and create a secret with F5 BIG-IP system login credentials, which is a prerequisite for the operator.

```
[admin@rhel-7 ~]$ oc create secret generic bigip-login -n kube-system
--from-literal=username=admin --from-literal=password=admin

secret/bigip-login created
```

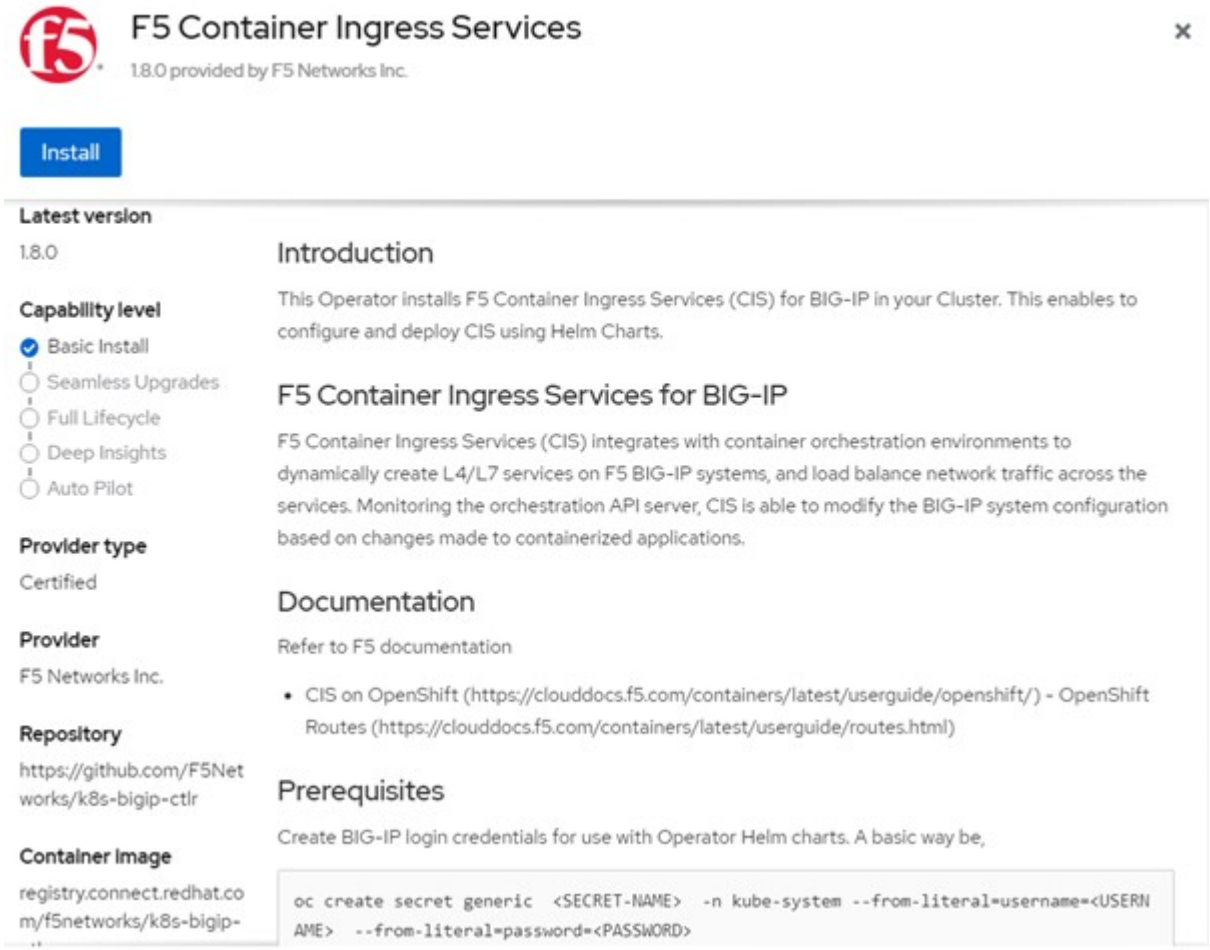
15. Install the F5 CIS CRDs.

```
[admin@rhel-7 ~]$ oc apply -f
https://raw.githubusercontent.com/F5Networks/k8s-bigip-
ctlr/master/docs/config_examples/crd/Install/customresourcedefinitions.y
ml

customresourcedefinition.apiextensions.k8s.io/virtualservers.cis.f5.com
created
customresourcedefinition.apiextensions.k8s.io/tlsprofiles.cis.f5.com
created
customresourcedefinition.apiextensions.k8s.io/transportservers.cis.f5.co
m created
customresourcedefinition.apiextensions.k8s.io/externaldnss.cis.f5.com
created
customresourcedefinition.apiextensions.k8s.io/ingresslinks.cis.f5.com
created
```

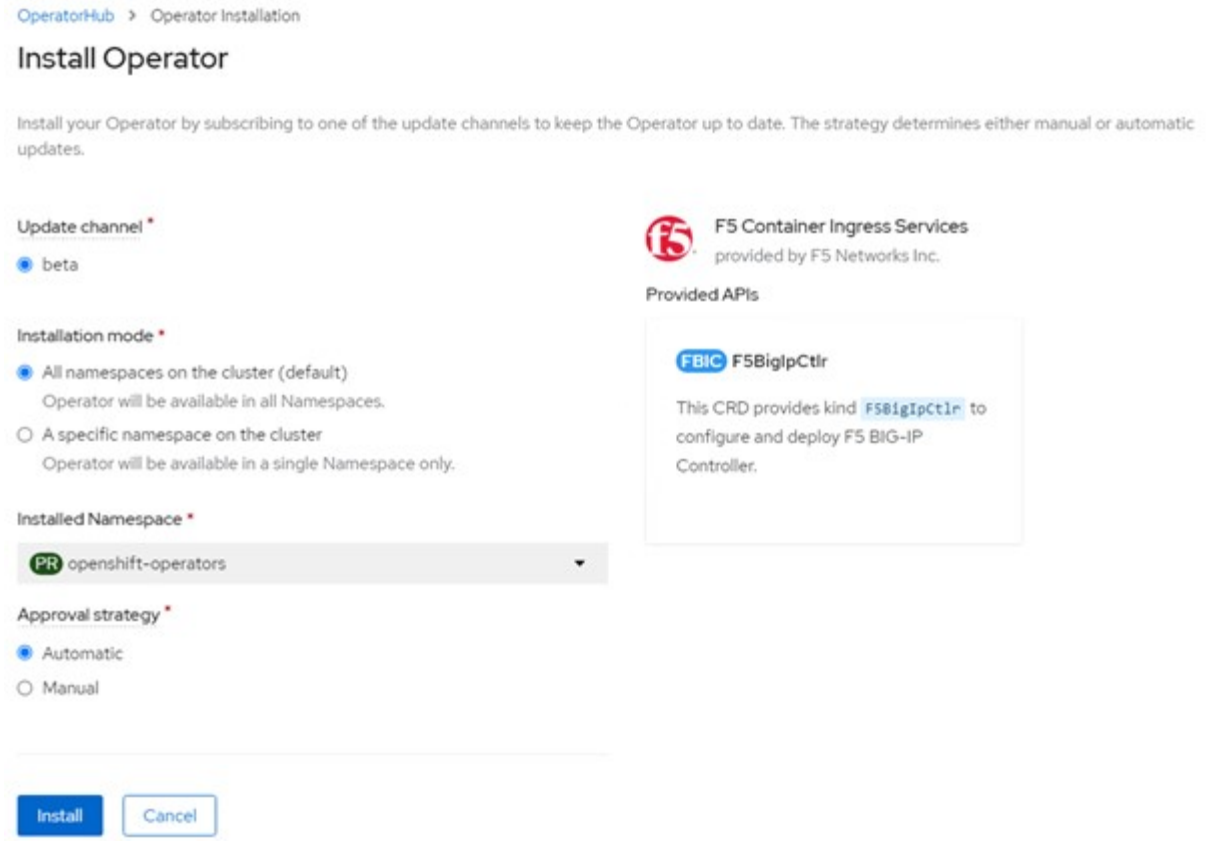
16. Navigate to Operators > OperatorHub, search for the keyword F5, and click the F5 Container Ingress Service tile.



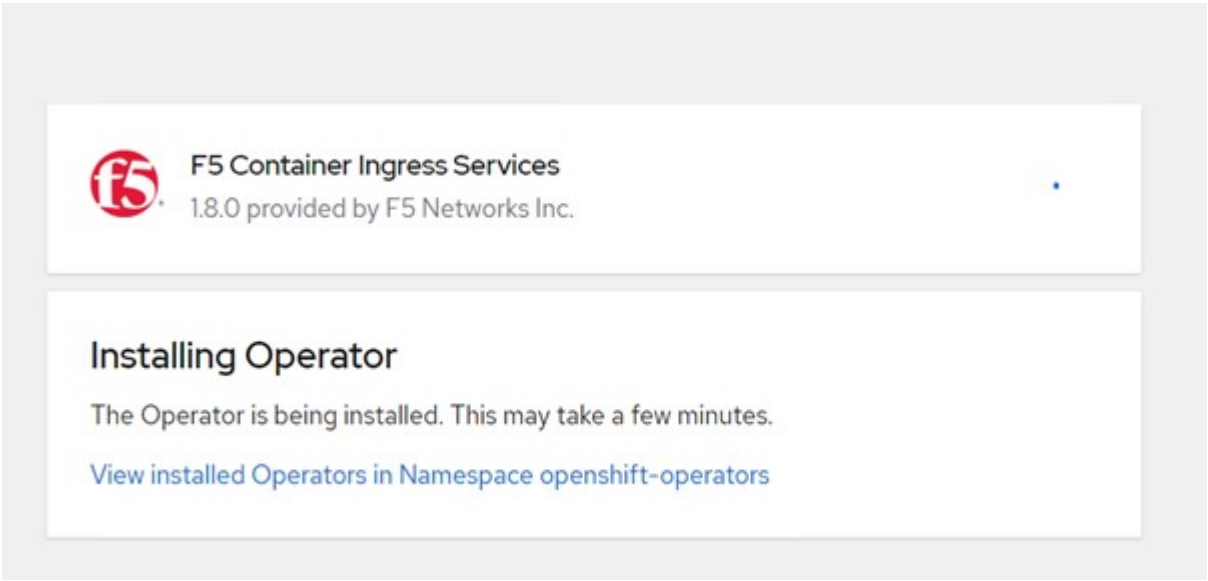
17. Read the operator information and click Install.



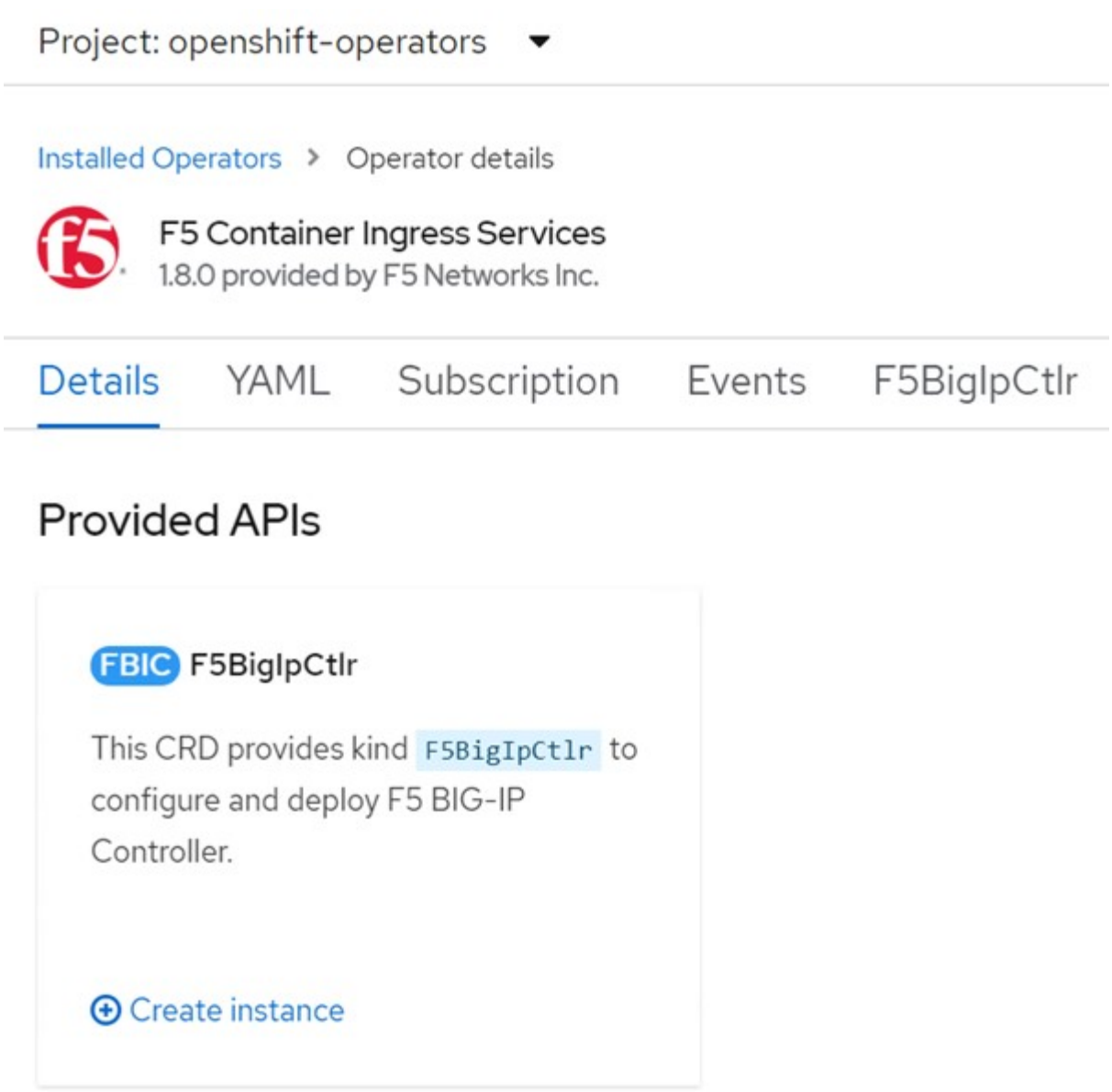
18. On the Install operator screen, leave all default parameters, and click Install.



19. It takes a while to install the operator.



20. After the operator is installed, the Installation Successful message is displayed.
21. Navigate to Operators > Installed Operators, click F5 Container Ingress Service, and then click Create Instance under the F5BigIpCtlr tile.

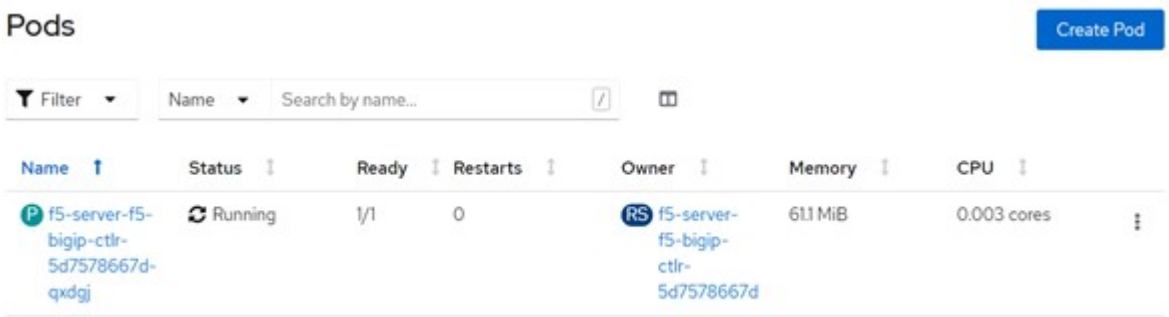


22. Click YAML View and paste the following content after updating the necessary parameters.

> Update the parameters `bigip_partition`, ` openshift_sdn_name`, `bigip_url` and `bigip_login_secret` below to reflect the values for your setup before copying the content.

```
apiVersion: cis.f5.com/v1
kind: F5BigIpCtlr
metadata:
  name: f5-server
  namespace: openshift-operators
spec:
  args:
    log_as3_response: true
    agent: as3
    log_level: DEBUG
    bigip_partition: ocp-vmw
    openshift_sdn_name: /Common/openshift_vxlan
    bigip_url: 10.61.181.19
    insecure: true
    pool-member-type: cluster
    custom_resource_mode: true
    as3_validation: true
    ipam: true
    manage_configmaps: true
  bigip_login_secret: bigip-login
  image:
    pullPolicy: Always
    repo: f5networks/cntr-ingress-svcs
    user: registry.connect.redhat.com
  namespace: kube-system
  rbac:
    create: true
  resources: {}
  serviceAccount:
    create: true
  version: latest
```

23. After pasting this content, click Create. This installs the CIS pods in the kube-system namespace.



Red Hat OpenShift, by default, provides a way to expose the services via Routes for L7 load balancing. An inbuilt OpenShift router is responsible for advertising and handling traffic for these routes. However, you can also configure the F5 CIS to support the Routes through an external F5 BIG-IP system, which can run either as an auxiliary router or a replacement to the self-hosted OpenShift router. CIS creates a virtual server in the BIG-IP system that acts as a router for the OpenShift routes, and BIG-IP handles the advertisement and traffic routing. Refer to the documentation here for information on parameters to enable this feature. Note that these parameters are defined for OpenShift Deployment resource in the apps/v1 API. Therefore, when using these with the F5BigIpCtlr resource cis.f5.com/v1 API, replace the hyphens (-) with underscores (_) for the parameter names.

24. The arguments that are passed to the creation of CIS resources include `ipam: true` and `custom_resource_mode: true`. These parameters are required for enabling CIS integration with an IPAM controller. Verify that the CIS has enabled IPAM integration by creating the F5 IPAM resource.

```
[admin@rhel-7 ~]$ oc get f5ipam -n kube-system

NAMESPACE   NAME                              AGE
kube-system   ipam.10.61.181.19.ocp-vmw       43s
```

25. Create the service account, role and rolebinding required for the F5 IPAM controller. Create a YAML file and paste the following content.

```
[admin@rhel-7 ~]$ vi f5-ipam-rbac.yaml

kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: ipam-ctlr-clusterrole
rules:
  - apiGroups: ["fic.f5.com"]
    resources: ["ipams","ipams/status"]
    verbs: ["get", "list", "watch", "update", "patch"]
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: ipam-ctlr-clusterrole-binding
  namespace: kube-system
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: ipam-ctlr-clusterrole
subjects:
  - apiGroup: ""
    kind: ServiceAccount
    name: ipam-ctlr
    namespace: kube-system
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: ipam-ctlr
  namespace: kube-system
```

26. Create the resources.

```
[admin@rhel-7 ~]$ oc create -f f5-ipam-rbac.yaml

clusterrole.rbac.authorization.k8s.io/ipam-ctlr-clusterrole created
clusterrolebinding.rbac.authorization.k8s.io/ipam-ctlr-clusterrole-
binding created
serviceaccount/ipam-ctlr created
```

27. Create a YAML file and paste the F5 IPAM deployment definition provided below.

Update the ip-range parameter in spec.template.spec.containers[0].args below to reflect the ipamLabels and IP address ranges corresponding to your setup.

ipamLabels [`range1` and `range2` in below example] are required to be annotated for the services of type LoadBalancer for the IPAM controller to detect and assign an IP address from the defined range.

```
[admin@rhel-7 ~]$ vi f5-ipam-deployment.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    name: f5-ipam-controller
  name: f5-ipam-controller
  namespace: kube-system
spec:
  replicas: 1
  selector:
    matchLabels:
      app: f5-ipam-controller
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: f5-ipam-controller
    spec:
      containers:
      - args:
        - --orchestration=openshift
        - --ip-range='{"range1":"10.63.172.242-10.63.172.249",
"range2":"10.63.170.111-10.63.170.129"}'
        - --log-level=DEBUG
        command:
        - /app/bin/f5-ipam-controller
        image: registry.connect.redhat.com/f5networks/f5-ipam-
controller:latest
        imagePullPolicy: IfNotPresent
        name: f5-ipam-controller
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      serviceAccount: ipam-ctlr
      serviceAccountName: ipam-ctlr
```

28. Create the F5 IPAM controller deployment.

```
[admin@rhel-7 ~]$ oc create -f f5-ipam-deployment.yaml

deployment/f5-ipam-controller created
```

29. Verify the F5 IPAM controller pods are running.

```
[admin@rhel-7 ~]$ oc get pods -n kube-system

NAME                                      READY   STATUS    RESTARTS
AGE
f5-ipam-controller-5986cff5bd-2bvn6       1/1     Running   0
30s
f5-server-f5-bigip-ctlr-5d7578667d-qxdgj  1/1     Running   0
14m
```

30. Create the F5 IPAM schema.

```
[admin@rhel-7 ~]$ oc create -f
https://raw.githubusercontent.com/F5Networks/f5-ipam-
controller/main/docs/_static/schemas/ipam_schema.yaml

customresourcedefinition.apiextensions.k8s.io/ipams.fic.f5.com
```

Verification

1. Create a service of type LoadBalancer

```
[admin@rhel-7 ~]$ vi example_svc.yaml

apiVersion: v1
kind: Service
metadata:
  annotations:
    cis.f5.com/ipamLabel: range1
  labels:
    app: f5-demo-test
  name: f5-demo-test
  namespace: default
spec:
  ports:
  - name: f5-demo-test
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: f5-demo-test
  sessionAffinity: None
  type: LoadBalancer
```

```
[admin@rhel-7 ~]$ oc create -f example_svc.yaml

service/f5-demo-test created
```

2. Check if the IPAM controller assigns an external IP to it.

```
[admin@rhel-7 ~]$ oc get svc

NAME           TYPE           CLUSTER-IP       EXTERNAL-IP
PORT(S)        AGE
f5-demo-test   LoadBalancer   172.30.210.108   10.63.172.242
80:32605/TCP   27s
```

3. Create a deployment and use the LoadBalancer service that was created.

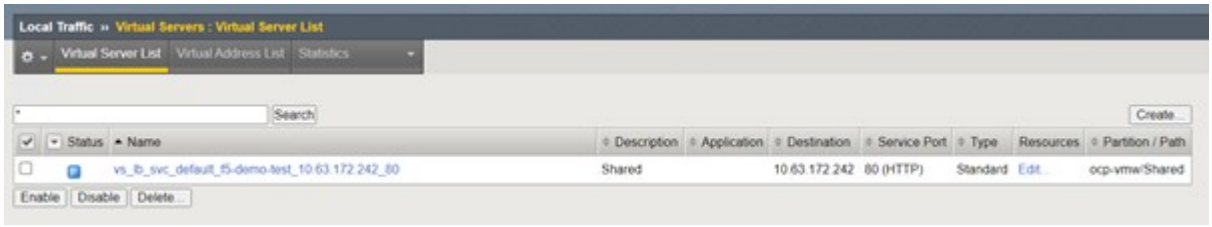
```
[admin@rhel-7 ~]$ vi example_deployment.yaml

apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: f5-demo-test
  name: f5-demo-test
spec:
  replicas: 2
  selector:
    matchLabels:
      app: f5-demo-test
  template:
    metadata:
      labels:
        app: f5-demo-test
    spec:
      containers:
      - env:
        - name: service_name
          value: f5-demo-test
        image: nginx
        imagePullPolicy: Always
        name: f5-demo-test
        ports:
        - containerPort: 80
          protocol: TCP
```

```
[admin@rhel-7 ~]$ oc create -f example_deployment.yaml

deployment/f5-demo-test created
```

4. Check if the pods are running.

```
[admin@rhel-7 ~]$ oc get pods

NAME                           READY   STATUS    RESTARTS   AGE
f5-demo-test-57c46f6f98-47wwp   1/1     Running   0          27s
f5-demo-test-57c46f6f98-cl2m8   1/1     Running   0          27s
```

5. Check if the corresponding virtual server is created in the BIG-IP system for the service of type LoadBalancer in OpenShift. Navigate to Local Traffic > Virtual Servers > Virtual Server List.



Next: Solution Validation/Use Cases: Red Hat OpenShift with NetApp.

Creating Private Image Registries

For most deployments of Red Hat OpenShift, using a public registry like Quay.io or DockerHub meets most customer's needs. However there are times when a customer may want to host their own private or customized images.

This procedure documents creating a private image registry which is backed by a persistent volume provided by Astra Trident and NetApp ONTAP.

> Astra Control Center requires a registry to host the images the Astra containers require. The following section describes the steps to setup a private registry on Red Hat OpenShift cluster and pushing the images required to support the installation of Astra Control Center.

Creating A private image registry

1. Remove the default annotation from the current default storage class and annotate the Trident-backed storage class as default for the OpenShift cluster.

```
[netapp-user@rhel7 ~]$ oc patch storageclass thin -p '{"metadata":
{"annotations": {"storageclass.kubernetes.io/is-default-class":
"false"}}}'
storageclass.storage.k8s.io/thin patched

[netapp-user@rhel7 ~]$ oc patch storageclass ocp-trident -p
'{"metadata": {"annotations": {"storageclass.kubernetes.io/is-default-
class": "true"}}}'
storageclass.storage.k8s.io/ocp-trident patched
```

2. Edit the imageregistry operator by entering the following storage parameters in the `spec` section.

```
[netapp-user@rhel7 ~]$ oc edit
configs.imageregistry.operator.openshift.io

storage:
  pvc:
    claim:
```

3. Enter the following parameters in the `spec` section for creating a OpenShift route with a custom hostname.

Save and exit.

```
routes:
  - hostname: astra-registry.apps.ocp-vmw.cie.netapp.com
    name: netapp-astra-route
```

The above route config is used when you want a custom hostname for your route. If you want OpenShift to create a route with a default hostname, you can add the following parameters to the `spec` section: `defaultRoute: true`.

Custom TLS certificates

When you are using a custom hostname for the route, by default, it uses the default TLS configuration of the OpenShift Ingress operator. However, you can add a custom TLS configuration to the route. To do so, complete the following steps.

a. Create a secret with the route's TLS certificates and key.

```
[netapp-user@rhel7 ~]$ oc create secret tls astra-route-tls -n
openshift-image-registry –cert/home/admin/netapp-astra/tls.crt
--key=/home/admin/netapp-astra/tls.key
```

b. Edit the imageregistry operator and add the following parameters to the `spec` section.

```
[netapp-user@rhel7 ~]$ oc edit
configs.imageregistry.operator.openshift.io

routes:
  - hostname: astra-registry.apps.ocp-vmw.cie.netapp.com
    name: netapp-astra-route
    secretName: astra-route-tls
```

4. Edit the imageregistry operator again and change the management state of the operator to the `Managed` state. Save and exit.

```
oc edit configs.imageregistry/cluster

managementState: Managed
```

5. If all the prerequisites are satisfied, PVCs, pods, and services are created for the private image registry. In a few minutes, the registry should be up.

```
[netapp-user@rhel7 ~]$oc get all -n openshift-image-registry

NAME                                                  READY   STATUS
RESTARTS   AGE
pod/cluster-image-registry-operator-74f6d954b6-rb7zr   1/1     Running
3          90d
pod/image-pruner-1627257600-f5cpj                     0/1     Completed
0          2d9h
pod/image-pruner-1627344000-swqx9                     0/1     Completed
0          33h
pod/image-pruner-1627430400-rv5nt                     0/1     Completed
0          9h
pod/image-registry-6758b547f-6pnj8                    1/1     Running
0          76m
pod/node-ca-bwb5r                                     1/1     Running
0          90d
pod/node-ca-f8w54                                     1/1     Running
0          90d
pod/node-ca-gjx7h                                     1/1     Running
0          90d
pod/node-ca-lcx4k                                     1/1     Running
0          33d
pod/node-ca-v7zmx                                     1/1     Running
0          7d21h
pod/node-ca-xpppp                                     1/1     Running
0          89d

NAME                              TYPE        CLUSTER-IP       EXTERNAL-
IP   PORT(S)     AGE
service/image-registry            ClusterIP   172.30.196.167   <none>
5000/TCP    15h
service/image-registry-operator   ClusterIP   None             <none>
60000/TCP   90d

NAME                      DESIRED   CURRENT   READY   UP-TO-DATE
AVAILABLE   NODE SELECTOR            AGE
daemonset.apps/node-ca   6         6         6       6            6
kubernetes.io/os=linux   90d

NAME                                          READY   UP-TO-DATE
AVAILABLE   AGE
deployment.apps/cluster-image-registry-operator   1/1     1            1
90d
deployment.apps/image-registry                    1/1     1            1
15h
```

```
NAME                                                                DESIRED
CURRENT   READY   AGE
replicaset.apps/cluster-image-registry-operator-74f6d954b6   1         1
1       90d
replicaset.apps/image-registry-6758b547f                    1         1
1       76m
replicaset.apps/image-registry-78bfbd7f59                   0         0
0       15h
replicaset.apps/image-registry-7fcc8d6cc8                   0         0
0       80m
replicaset.apps/image-registry-864f88f5b                    0         0
0       15h
replicaset.apps/image-registry-cb47fffb                     0         0
0       10h

NAME                                COMPLETIONS   DURATION   AGE
job.batch/image-pruner-1627257600   1/1           10s        2d9h
job.batch/image-pruner-1627344000   1/1           6s         33h
job.batch/image-pruner-1627430400   1/1           5s         9h

NAME                          SCHEDULE    SUSPEND   ACTIVE   LAST
SCHEDULE   AGE
cronjob.batch/image-pruner   0 0 * * *   False     0        9h
90d

NAME                                       HOST/PORT
PATH   SERVICES         PORT    TERMINATION   WILDCARD
route.route.openshift.io/public-routes   astra-registry.apps.ocp-
vmw.cie.netapp.com          image-registry   <all>   reencrypt     None
```

6. If you are using the default TLS certificates for the ingress operator OpenShift registry route, you can fetch the TLS certificates using the following command.

```
[netapp-user@rhel7 ~]$ oc extract secret/router-ca --keys=tls.crt -n
openshift-ingress-operator
```

7. To allow OpenShift nodes to access and pull the images from the registry, add the certificates to the docker client on the OpenShift nodes. Create a configmap in the `openshift-config` namespace using the TLS certificates and patch it to the cluster image config to make the certificate trusted.

```
[netapp-user@rhel7 ~]$ oc create configmap astra-ca -n openshift-config
--from-file=astra-registry.apps.ocp-vmw.cie.netapp.com=tls.crt

[netapp-user@rhel7 ~]$ oc patch image.config.openshift.io/cluster
--patch '{"spec":{"additionalTrustedCA":{"name":"astra-ca"}}}'
--type=merge
```

8. The OpenShift internal registry is controlled by authentication. All the OpenShift users can access the OpenShift registry, but the operations that the logged in user can perform depends on the user permissions.
   a. To allow a user or a group of users to pull images from the registry, the user(s) must have the registry-viewer role assigned.

```
[netapp-user@rhel7 ~]$ oc policy add-role-to-user registry-viewer
ocp-user

[netapp-user@rhel7 ~]$ oc policy add-role-to-group registry-viewer
ocp-user-group
```

   b. To allow a user or group of users to write or push images, the user(s) must have the registry-editor role assigned.

```
[netapp-user@rhel7 ~]$ oc policy add-role-to-user registry-editor
ocp-user

[netapp-user@rhel7 ~]$ oc policy add-role-to-group registry-editor
ocp-user-group
```

9. For OpenShift nodes to access the registry and push or pull the images, you need to configure a pull secret.

```
[netapp-user@rhel7 ~]$ oc create secret docker-registry astra-registry-
credentials --docker-server=astra-registry.apps.ocp-vmw.cie.netapp.com
--docker-username=ocp-user --docker-password=password
```

10. This pull secret can then be patched to serviceaccounts or be referenced in the corresponding pod definition.
    a. To patch it to service accounts, run the following command.

```
[netapp-user@rhel7 ~]$ oc secrets link <service_account_name> astra-
registry-credentials --for=pull
```

b. To reference the pull secret in the pod definition, add the following parameter to the `spec` section.

```
imagePullSecrets:
  - name: astra-registry-credentials
```

11. To push or pull an image from workstations apart from OpenShift node, complete the following steps.
    a. Add the TLS certificates to the docker client.

```
[netapp-user@rhel7 ~]$ sudo mkdir /etc/docker/certs.d/astra-
registry.apps.ocp-vmw.cie.netapp.com

[netapp-user@rhel7 ~]$ sudo cp /path/to/tls.crt
/etc/docker/certs.d/astra-registry.apps.ocp-vmw.cie.netapp.com
```

    b. Log into OpenShift using the oc login command.

```
[netapp-user@rhel7 ~]$ oc login --token=sha256~D49SpB_lesSrJYwrM0LIO
-VRcjWHu0a27vKa0 --server=https://api.ocp-vmw.cie.netapp.com:6443
```

    c. Log into the registry using OpenShift user credentials with the podman/docker command.

podman

```
[netapp-user@rhel7 ~]$ podman login astra-registry.apps.ocp-
vmw.cie.netapp.com -u kubeadmin -p $(oc whoami -t) --tls
-verify=false
```

+
NOTE: If you are using `kubeadmin` user to log into the private registry, then use token instead of password.

docker

```
[netapp-user@rhel7 ~]$ docker login astra-registry.apps.ocp-
vmw.cie.netapp.com -u kubeadmin -p $(oc whoami -t)
```

+
NOTE: If you are using `kubeadmin` user to log into the private registry, then use token instead of password.

    d. Push or pull the images.

podman

```
[netapp-user@rhel7 ~]$ podman push astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
[netapp-user@rhel7 ~]$ podman pull astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
```

docker

```
[netapp-user@rhel7 ~]$ docker push astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
[netapp-user@rhel7 ~]$ docker pull astra-registry.apps.ocp-
vmw.cie.netapp.com/netapp-astra/vault-controller:latest
```

Next: Solution Validation/Use Cases: Red Hat OpenShift with NetApp.

Solution Validation and Use Cases: Red Hat OpenShift with NetApp

The examples provided on this page are solution validations and use cases for Red Hat OpenShift with NetApp.

- Deploy a Jenkins CI/CD Pipeline with Persistent Storage
- Configure Multitenancy on Red Hat OpenShift with NetApp
- Red Hat OpenShift Virtualization with NetApp ONTAP
- Advanced Cluster Management for Kubernetes on Red Hat OpenShift with NetApp

Next: Videos and Demos.

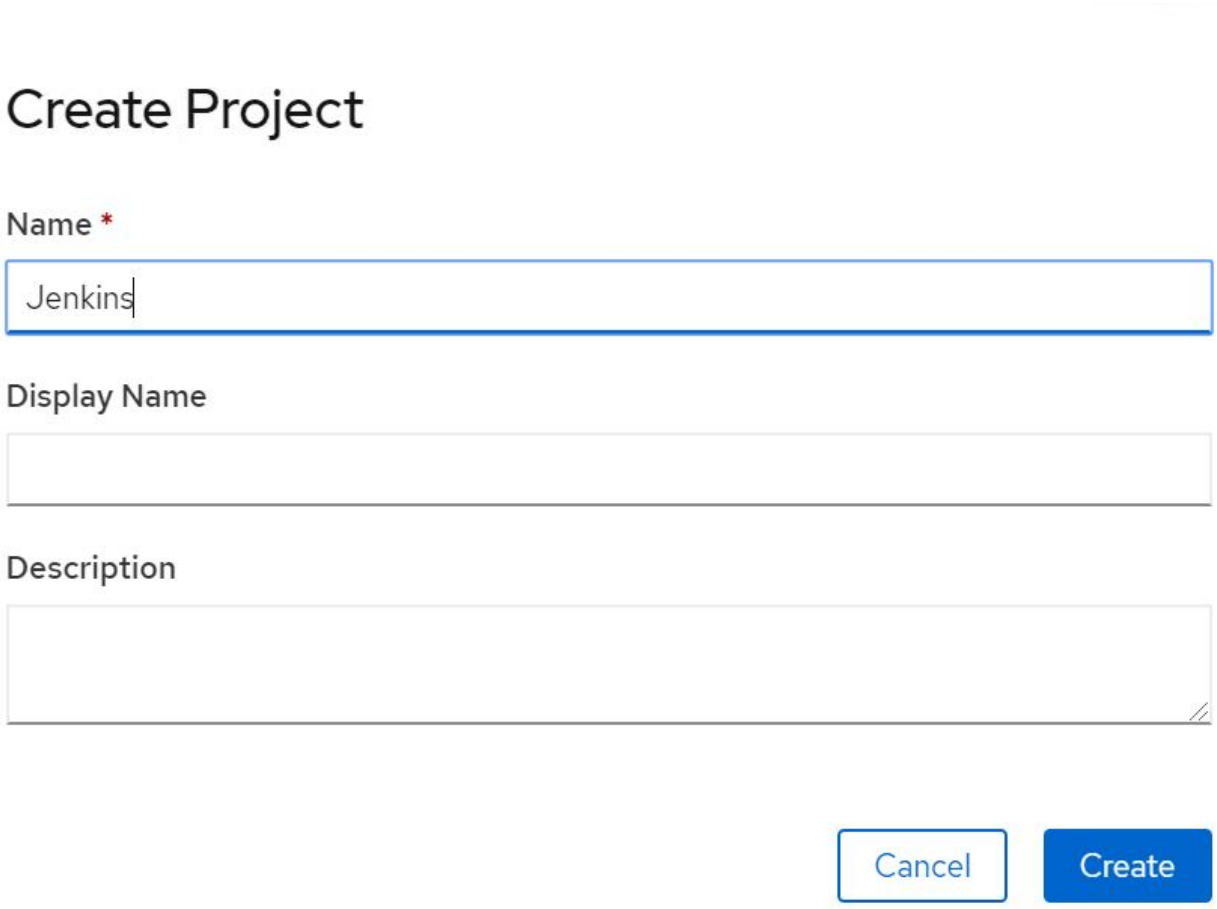
Deploy a Jenkins CI/CD Pipeline with Persistent Storage: Red Hat OpenShift with NetApp

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

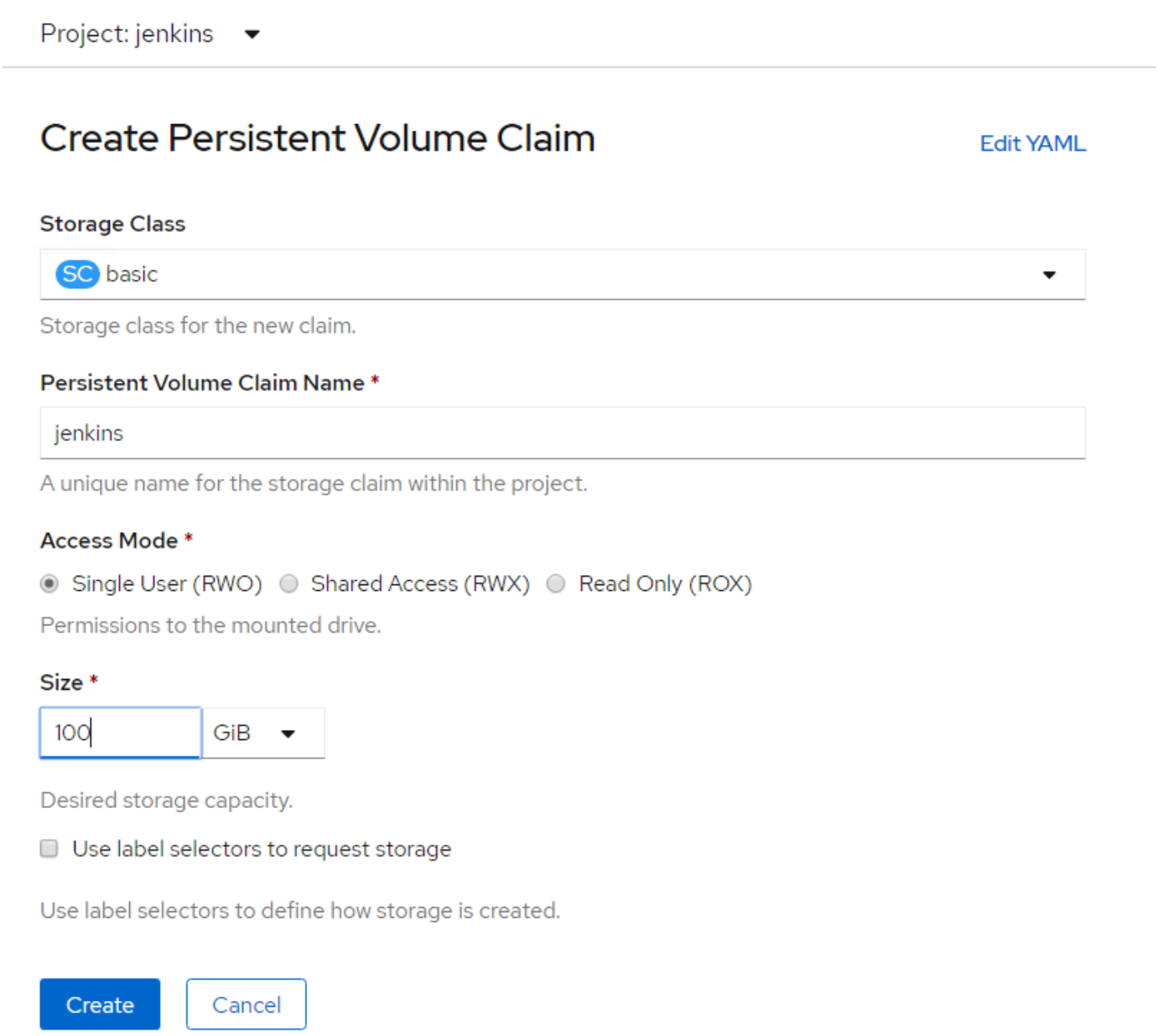
Create the resources required for Jenkins deployment

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

1. Create a new project named Jenkins.



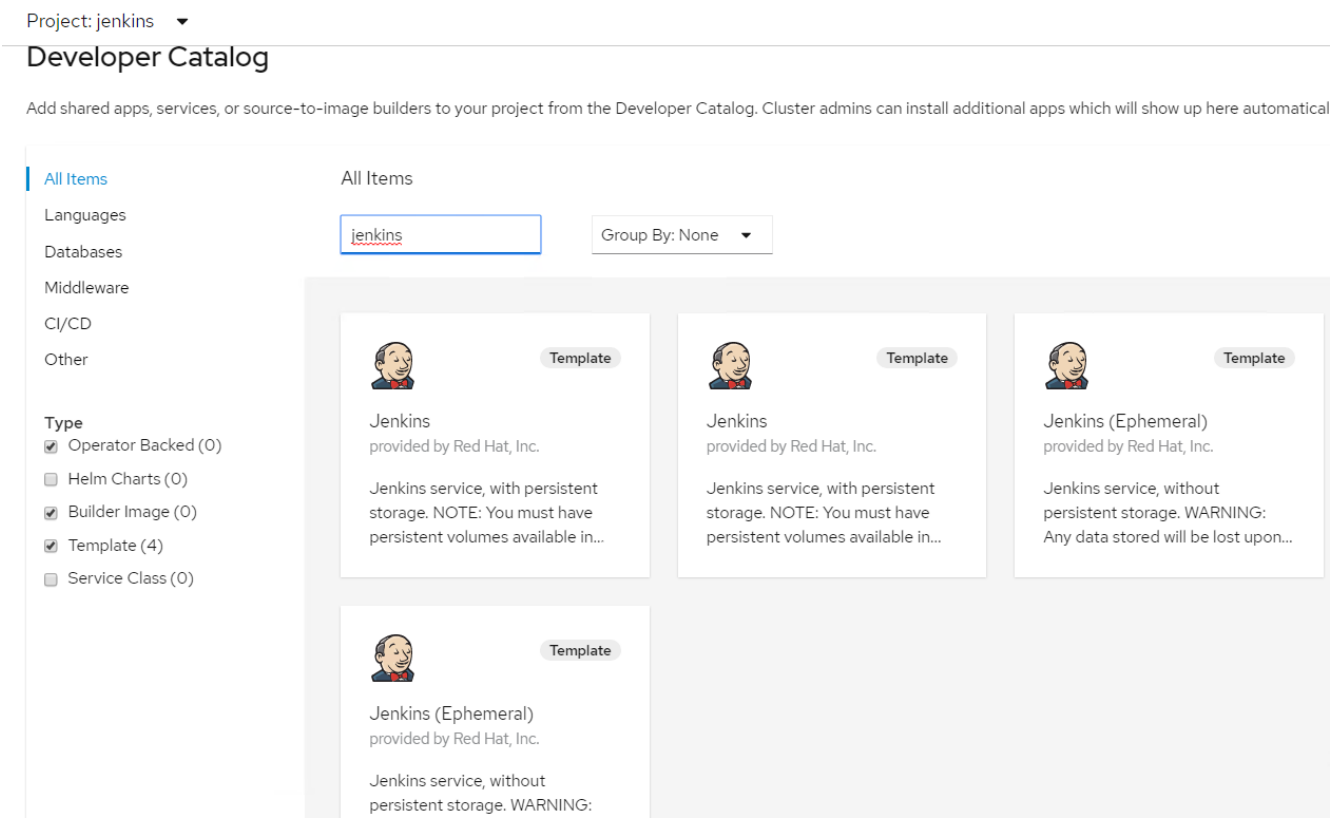
2. In this example, we deployed Jenkins with persistent storage. To support the Jenkins build, create the PVC. Navigate to Storage > Persistent Volume Claims and click Create Persistent Volume Claim. Select the storage class that was created, make sure that the Persistent Volume Claim Name is jenkins,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



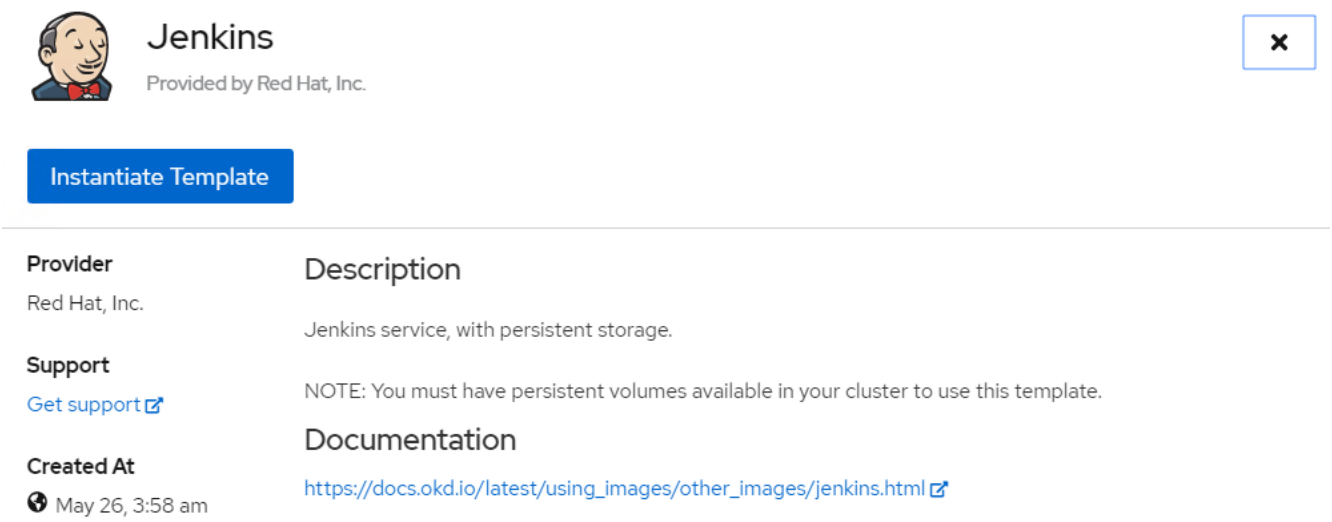
Deploy Jenkins with Persistent Storage

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 for jenkins. Select Jenkins Service with Persistent Storage.

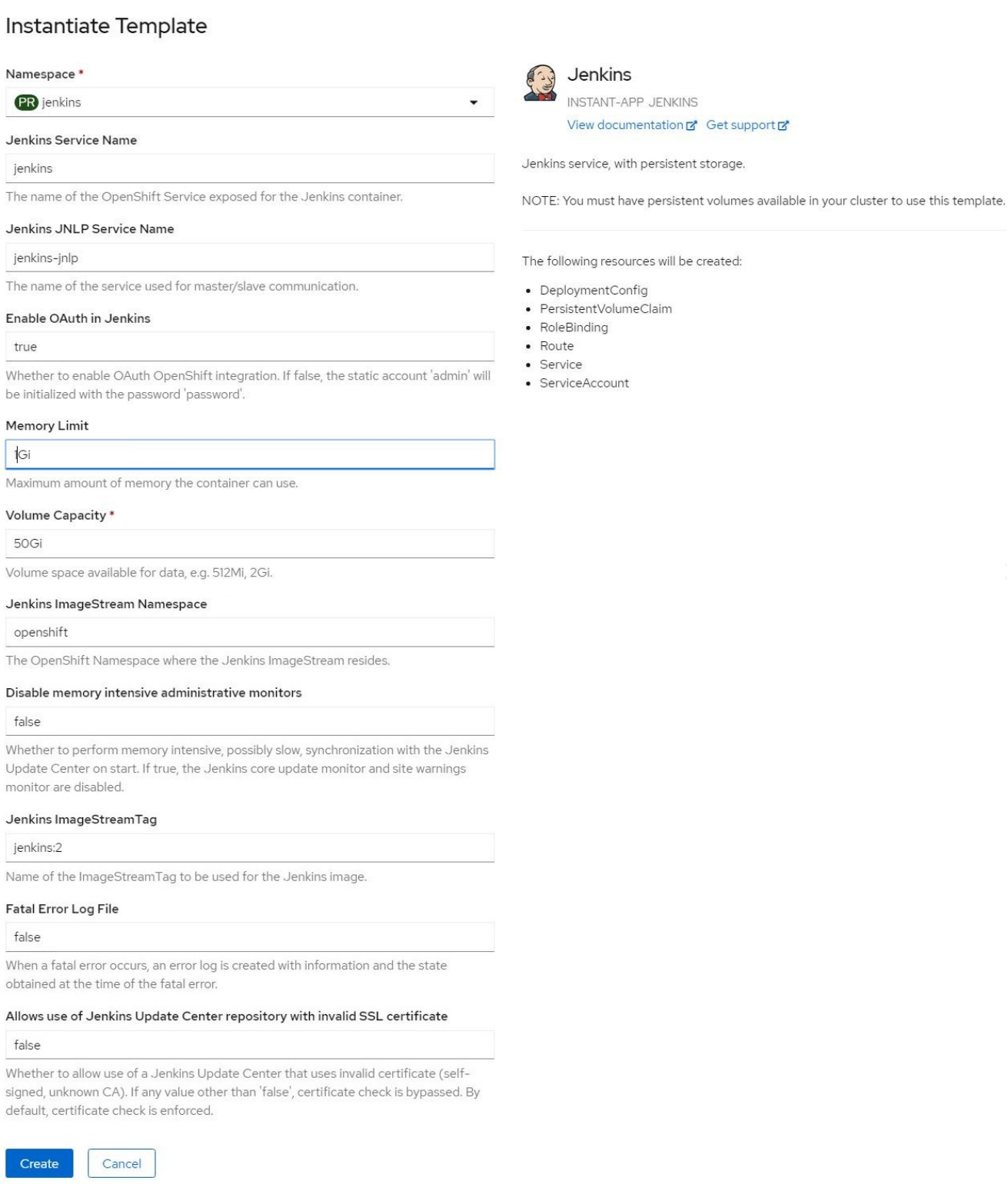


2. Click `Instantiate Template`.

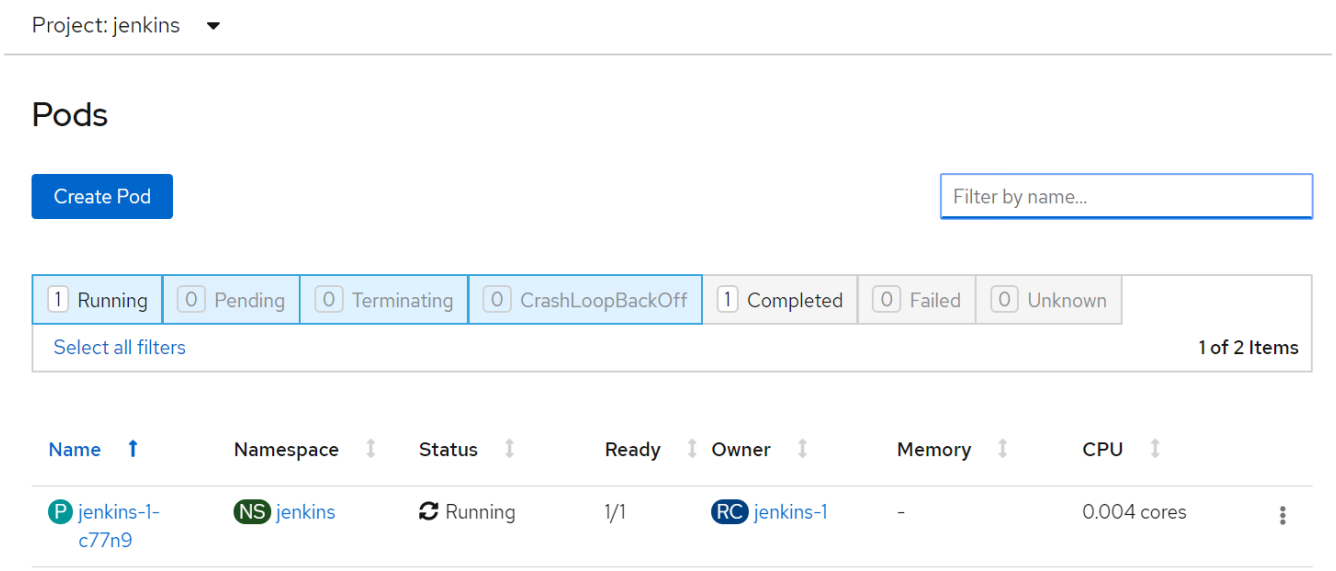


3. By default, the details for the Jenkins application are populated. Based on your requirements, modify the parameters and click Create. This process creates all the required resources for supporting Jenkins on

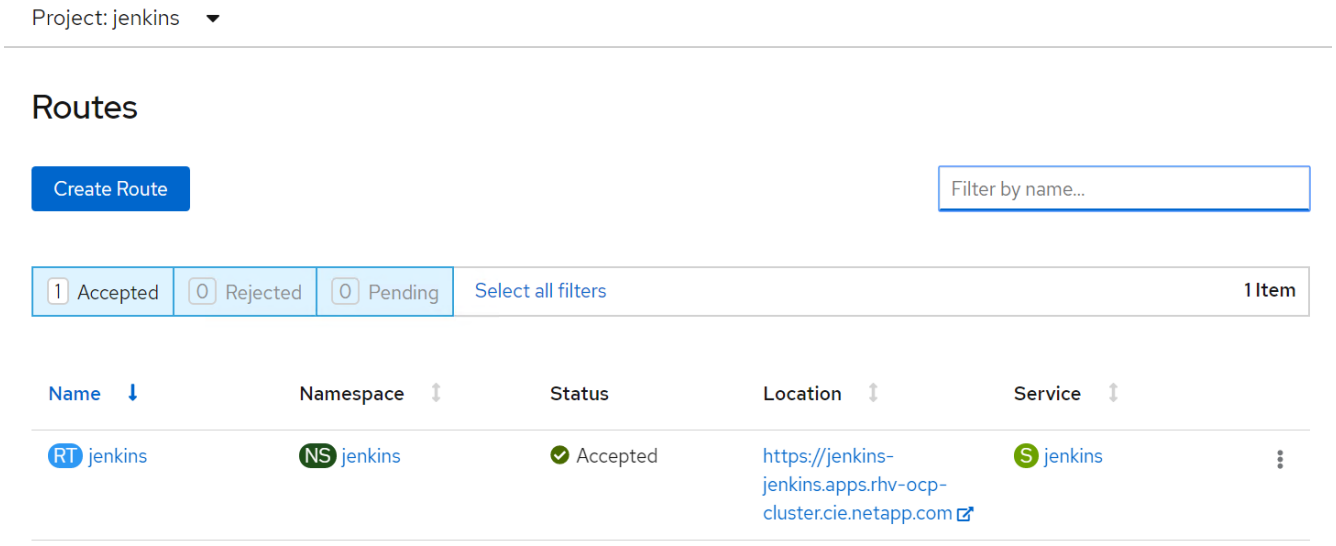
OpenShift.



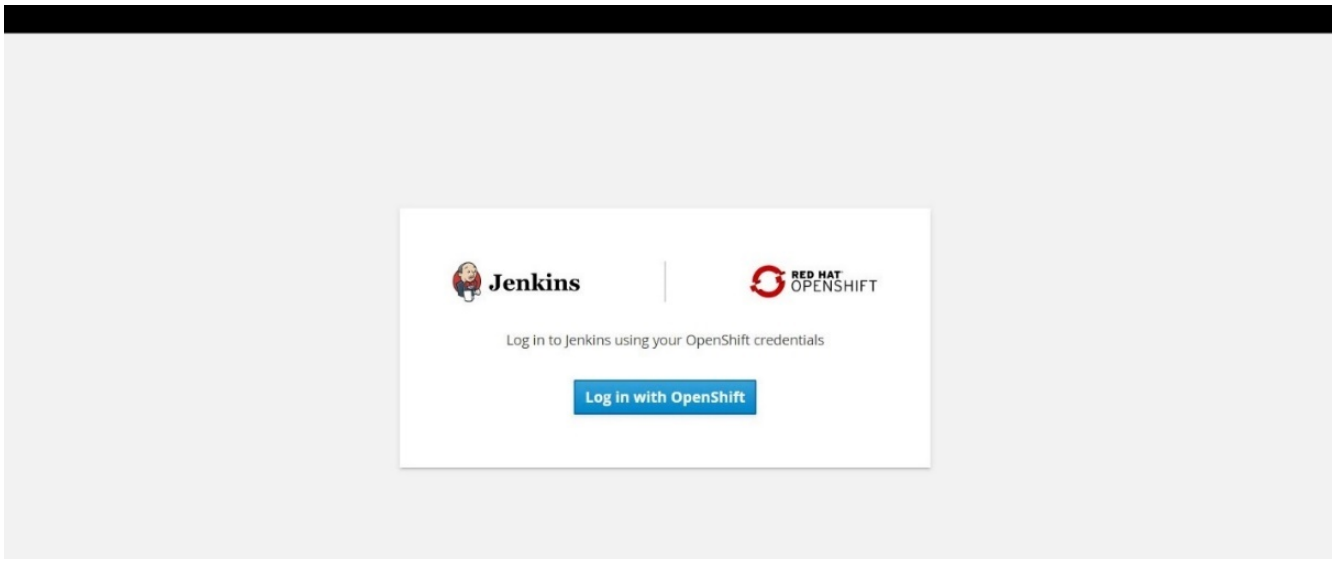
4. The Jenkins pods take approximately 10 to 12 minutes to enter the Ready state.



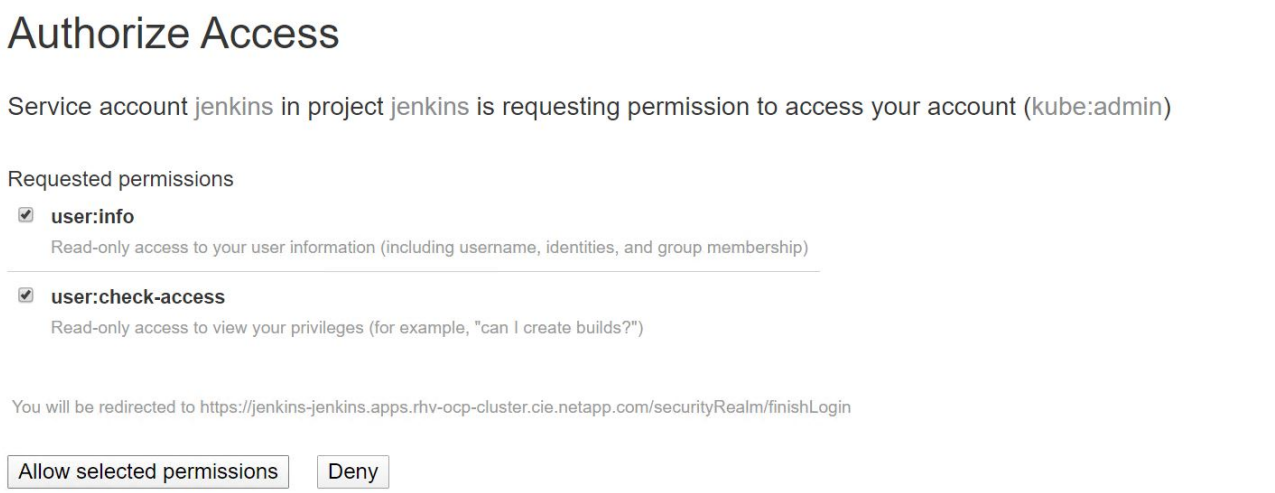
5. After the pods are instantiated, navigate to Networking > Routes. To open the Jenkins webpage, click the URL provided for the jenkins route.



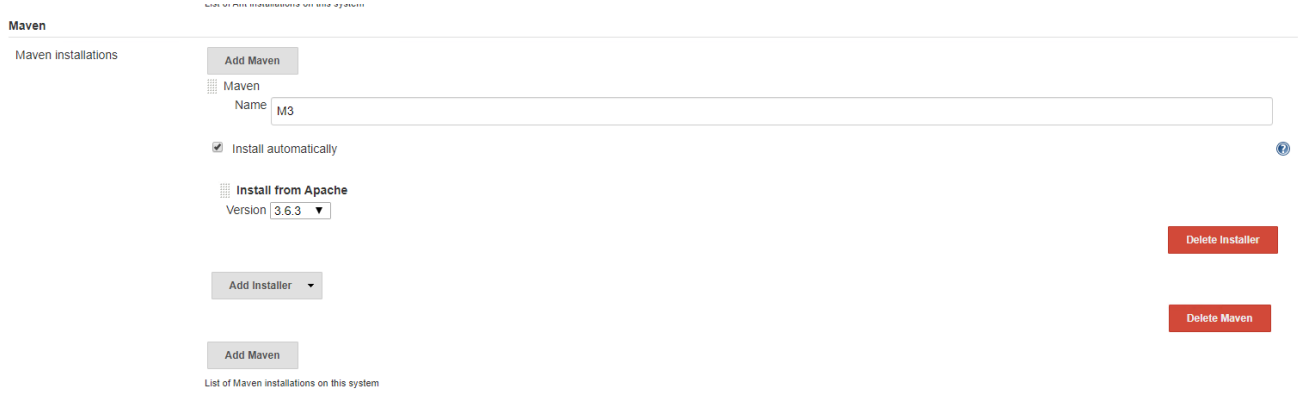
6. Because OpenShift OAuth was used while creating the Jenkins app, click Log in with OpenShift.



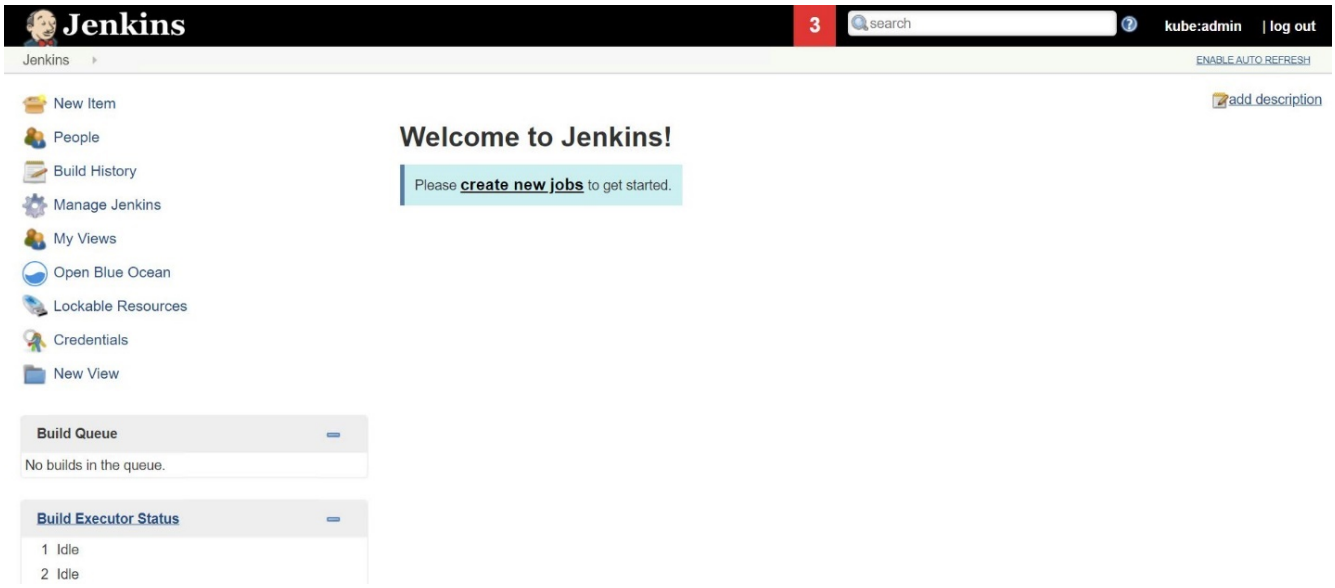
7. Authorize the Jenkins service account to access the OpenShift users.



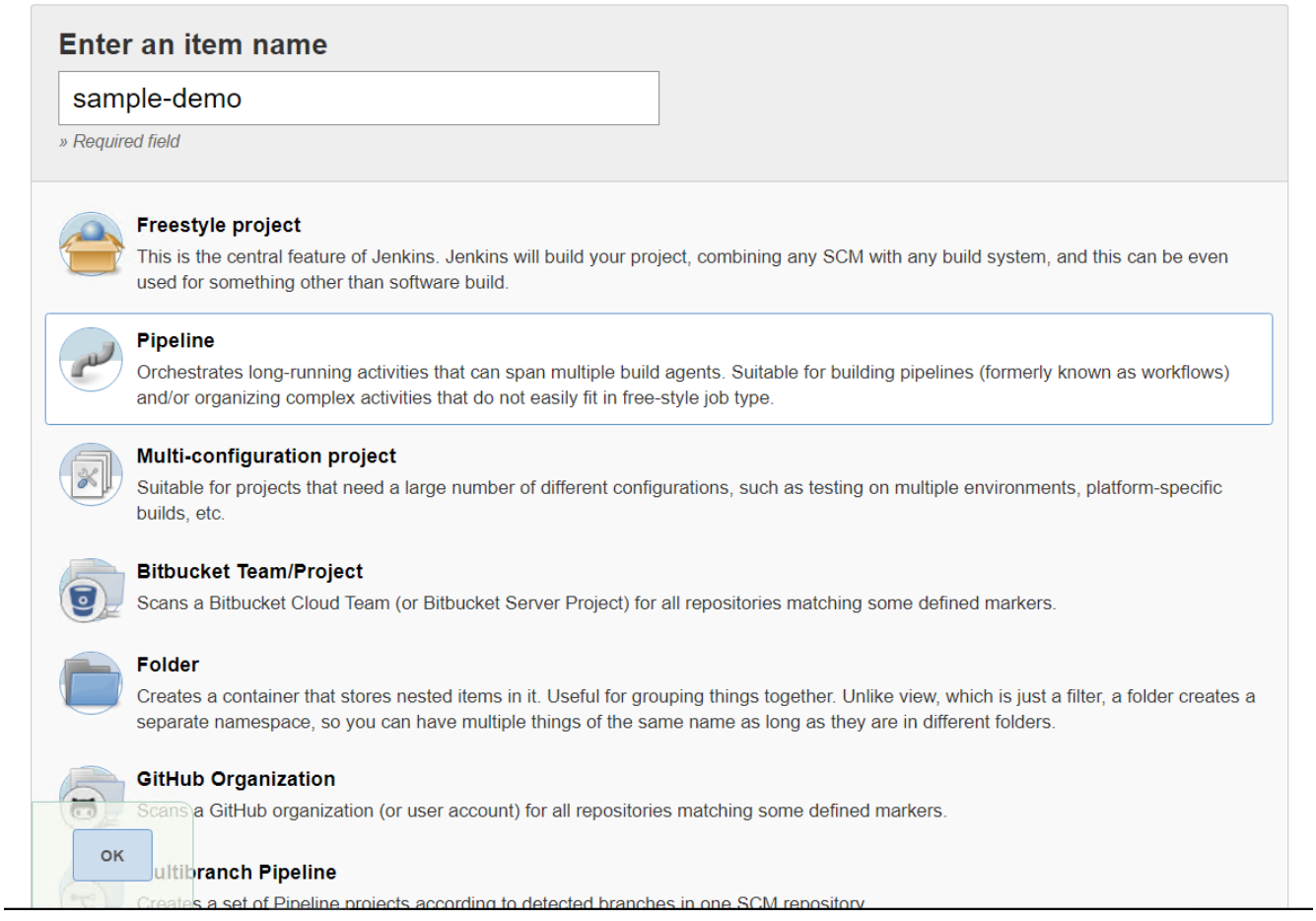
8. The Jenkins welcome page is displayed. Because we are using a Maven build, complete the Maven installation first. Navigate to Manage Jenkins > Global Tool Configuration, and then, in the Maven subhead, click Add Maven. Enter the name of your choice and make sure that the Install Automatically option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



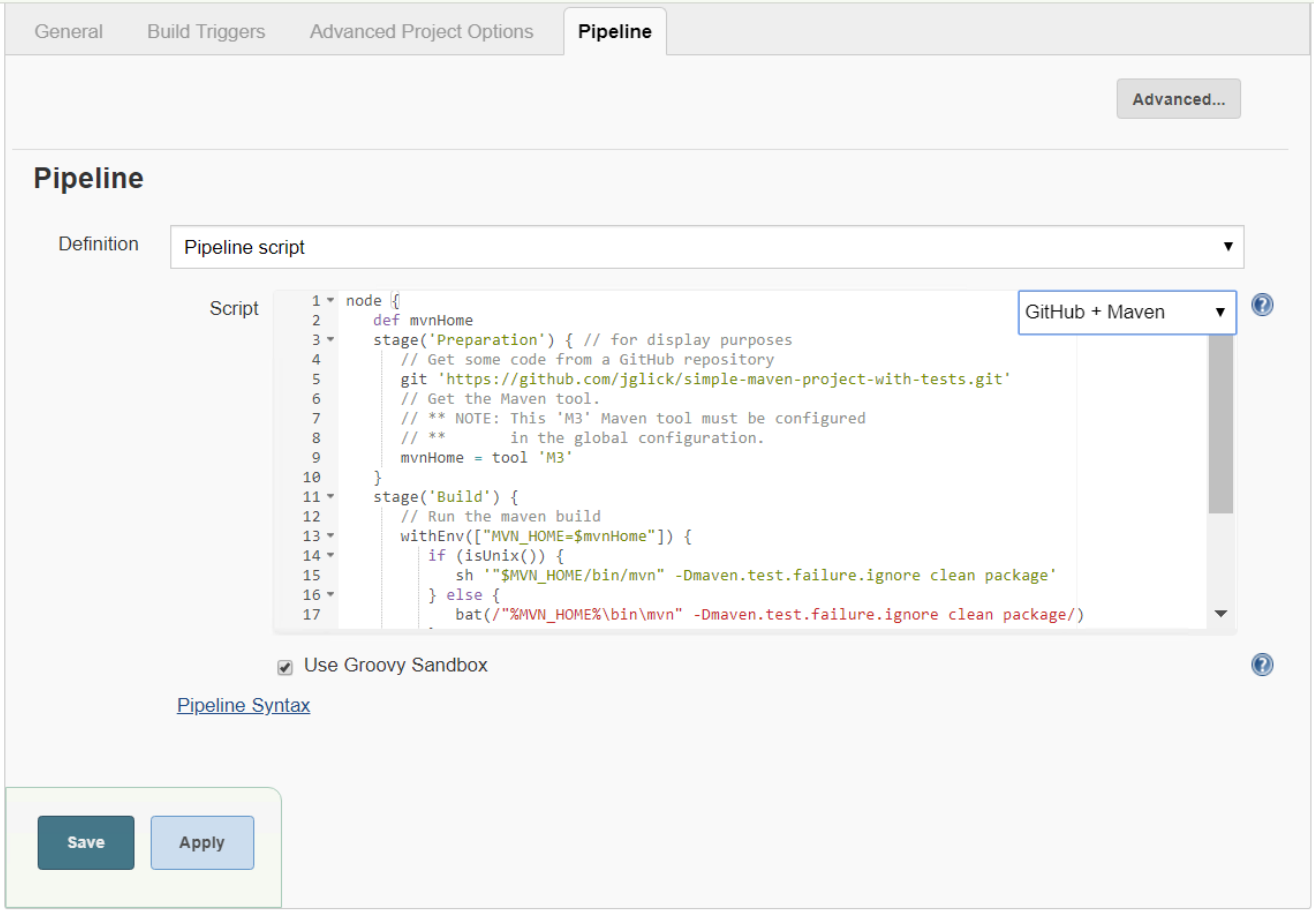
9. You can now create a pipeline to demonstrate the CI/CD workflow. On the home page, click Create New Jobs or New Item from the left-hand menu.



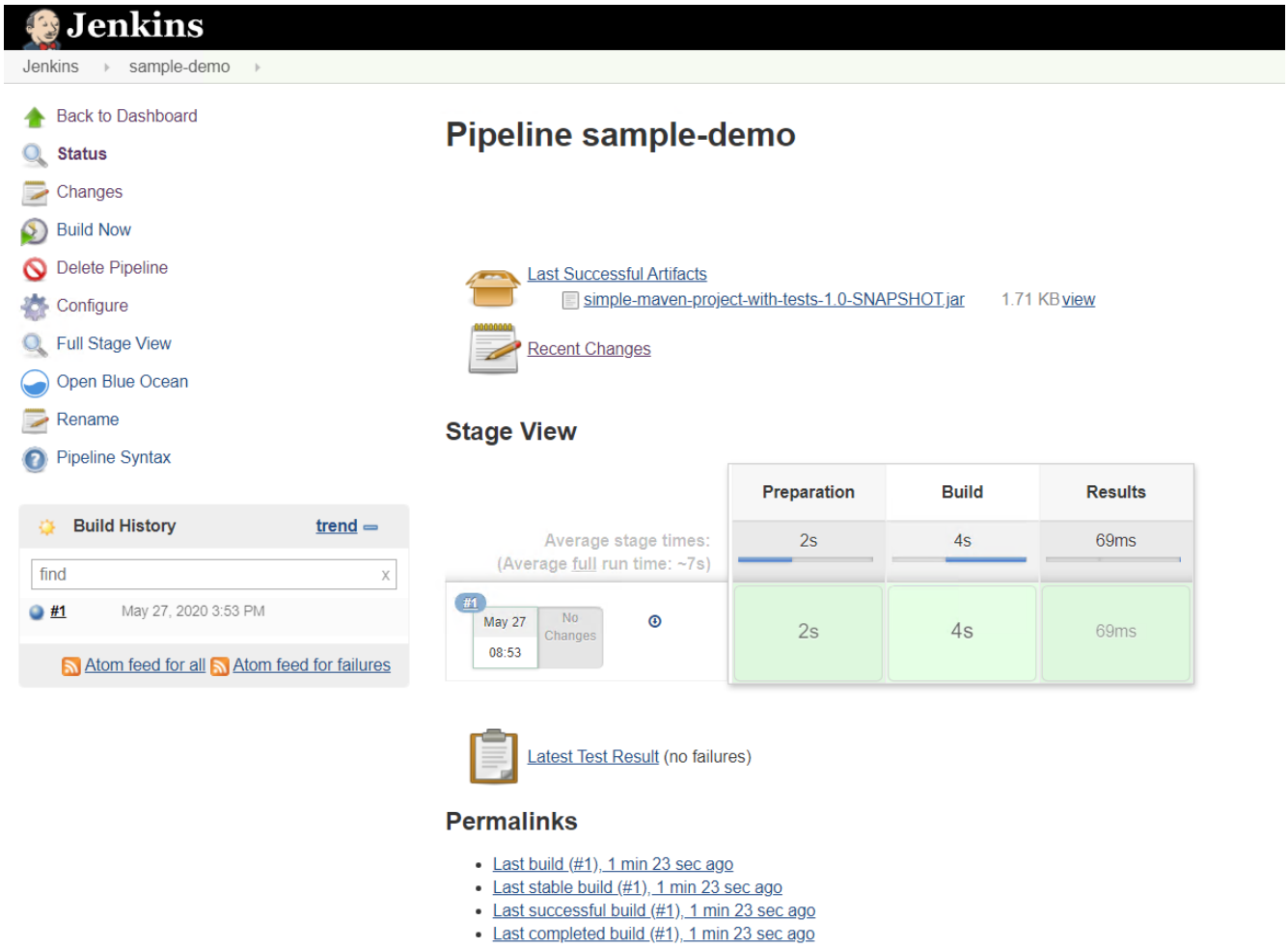
10. On the Create Item page, enter the name of your choice, select Pipeline, and click Ok.



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.

Next: Videos and Demos.

Configure Multi-tenancy on Red Hat OpenShift with NetApp ONTAP

Configuring multitenancy on Red Hat OpenShift with NetApp

Many organizations that run multiple applications or workloads on containers tend to deploy one Red Hat OpenShift cluster per application or workload. This allows them to implement strict isolation for the application or workload, optimize performance, and reduce security vulnerabilities. However, deploying a separate Red Hat OpenShift cluster for each application poses its own set of problems. It increases operational overhead having to monitor and manage each cluster on its own, increases cost owing to dedicated resources for different applications, and hinders efficient scalability.

To overcome these problems, one can consider running all the applications or workloads in a single Red Hat OpenShift cluster. But in such an architecture, resource isolation and application security vulnerabilities are one of the major challenges. Any security vulnerability in one workload could naturally spill over into another workload, thus increasing the impact zone. In addition, any abrupt uncontrolled resource utilization by one application can affect the performance of another application, because there is no resource allocation policy by default.

Therefore, organizations look out for solutions that pick up the best in both worlds, for example, by allowing them to run all their workloads in a single cluster and yet offering the benefits of a dedicated cluster for each workload.

One such effective solution is to configure multitenancy on Red Hat OpenShift. Multitenancy is an architecture that allows multiple tenants to coexist on the same cluster with proper isolation of resources, security, and so on. In this context, a tenant can be viewed as a subset of the cluster resources that are configured to be used by a particular group of users for an exclusive purpose. Configuring multitenancy on a Red Hat OpenShift cluster provides the following advantages:

- A reduction in CapEx and OpEx by allowing cluster resources to be shared
- Lower operational and management overhead
- Securing the workloads from cross-contamination of security breaches
- Protection of workloads from unexpected performance degradation due to resource contention

For a fully realized multitenant OpenShift cluster, quotas and restrictions must be configured for cluster resources belonging to different resource buckets: compute, storage, networking, security, and so on. Although we cover certain aspects of all the resource buckets in this solution, we focus on best practices for isolating and securing the data served or consumed by multiple workloads on the same Red Hat OpenShift cluster by configuring multitenancy on storage resources that are dynamically allocated by Astra Trident backed by NetApp ONTAP.

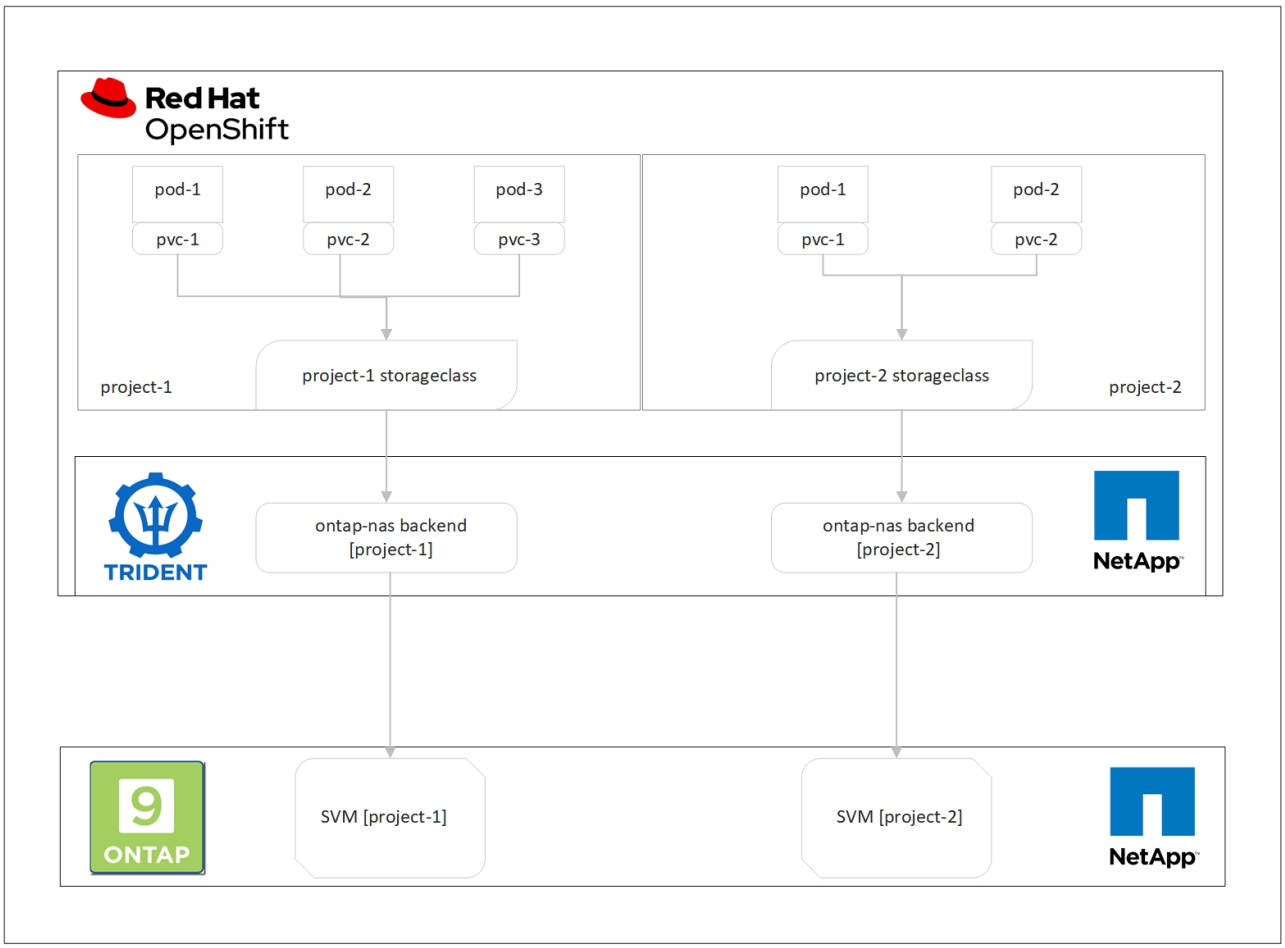
Next: Architecture.

Architecture

Although Red Hat OpenShift and Astra Trident backed by NetApp ONTAP do not provide isolation between workloads by default, they offer a wide range of features that can be used to configure multitenancy. To better understand designing a multitenant solution on a Red Hat OpenShift cluster with Astra Trident backed by NetApp ONTAP, let us consider an example with a set of requirements and outline the configuration around it.

Let us assume that an organization runs two of its workloads on a Red Hat OpenShift cluster as part of two projects that two different teams are working on. The data for these workloads reside on PVCs that are dynamically provisioned by Astra Trident on a NetApp ONTAP NAS backend. The organization has a requirement to design a multitenant solution for these two workloads and isolate the resources used for these projects to make sure that security and performance is maintained, primarily focused on the data that serves those applications.

The following figure depicts the multitenant solution on a Red Hat OpenShift cluster with Astra Trident backed by NetApp ONTAP.



Technology requirements

1. NetApp ONTAP storage cluster
2. Red Hat OpenShift cluster
3. Astra Trident

Red Hat OpenShift – Cluster resources

From the Red Hat OpenShift cluster point of view, the top-level resource to start with is the project. An OpenShift project can be viewed as a cluster resource that divides the whole OpenShift cluster into multiple virtual clusters. Therefore, isolation at project level provides a base for configuring multitenancy.

Next up is to configure RBAC in the cluster. The best practice is to have all the developers working on a single project or workload configured into a single user group in the Identity Provider (IdP). Red Hat OpenShift allows IdP integration and user group synchronization thus allowing the users and groups from the IdP to be imported into the cluster. This helps the cluster administrators to segregate access of the cluster resources dedicated to a project to a user group or groups working on that project, thereby restricting unauthorized access to any cluster resources. To learn more about IdP integration with Red Hat OpenShift, see the documentation here.

NetApp ONTAP

It is important to isolate the shared storage serving as a persistent storage provider for a Red Hat OpenShift cluster to make sure that the volumes created on the storage for each project appear to the hosts as if they are

created on separate storage. To do this, create as many SVMs (storage virtual machines) on NetApp ONTAP as there are projects or workloads, and dedicate each SVM to a workload.

Astra Trident

After you have different SVMs for different projects created on NetApp ONTAP, you must map each SVM to a different Trident backend. The backend configuration on Trident drives the allocation of persistent storage to OpenShift cluster resources, and it requires the details of the SVM to be mapped to. This should be the protocol driver for the backend at the minimum. Optionally, it allows you to define how the volumes are provisioned on the storage and to set limits for the size of volumes or usage of aggregates and so on. Details concerning the definition of the Trident backend for NetApp ONTAP can be found here.

Red Hat OpenShift – storage resources

After configuring the Trident backends, the next step is to configure StorageClasses. Configure as many storage classes as there are backends, providing each storage class access to spin up volumes only on one backend. We can map the StorageClass to a particular Trident backend by using the storagePools parameter while defining the storage class. The details to define a storage class can be found here. Thus, there is a one-to-one mapping from StorageClass to Trident backend which points back to one SVM. This ensures that all storage claims via the StorageClass assigned to that project are served by the SVM dedicated to that project only.

Because storage classes are not namespaced resources, how do we ensure that storage claims to storage class of one project by pods in another namespace or project gets rejected? The answer is to use ResourceQuotas. ResourceQuotas are objects that control the total usage of resources per project. It can limit the number as well as the total amount of resources that can be consumed by objects in the project. Almost all the resources of a project can be limited using ResourceQuotas and using this efficiently can help organizations cut cost and outages due to overprovisioning or overconsumption of resources. Refer to the documentation here for more information.

For this use case, we need to limit the pods in a particular project from claiming storage from storage classes that are not dedicated to their project. To do that, we need to limit the persistent volume claims for other storage classes by setting `<storage-class-name>.storageclass.storage.k8s.io/persistentvolumeclaims` to 0. In addition, a cluster administrator must ensure that the developers in a project should not have access to modify the ResourceQuotas.

Next: Configuration.

Configuration

For any multitenant solution, no user can have access to more cluster resources than is required. So, the entire set of resources that are to be configured as part of the multitenancy configuration is divided between cluster-admin, storage-admin, and developers working on each project.

The following table outlines the different tasks to be performed by different users:

Role	Tasks
Cluster-admin	Create projects for different applications or workloads
	Create ClusterRoles and RoleBindings for storage-admin
	Create Roles and RoleBindings for developers assigning access to specific projects
	[Optional] Configure projects to schedule pods on specific nodes
Storage-admin	Create SVMs on NetApp ONTAP
	Create Trident backends
	Create StorageClasses
	Create storage ResourceQuotas
Developers	Validate access to create or patch PVCs or pods in assigned project
	Validate access to create or patch PVCs or pods in another project
	Validate access to view or edit Projects, ResourceQuotas, and StorageClasses

Next: Prerequisites.

Configuration

Prerequisites

- NetApp ONTAP cluster
- Red Hat OpenShift cluster
- Trident installed on the cluster
- Admin workstation with tridentctl and oc tools installed and added to $PATH
- Admin access to ONTAP
- Cluster-admin access to OpenShift cluster
- Cluster is integrated with Identity Provider
- Identity provider is configured to efficiently distinguish between users in different teams

Next: Cluster Administrator Tasks.

Configuration: cluster-admin tasks

The following tasks are performed by the Red Hat OpenShift cluster-admin:

1. Log into Red Hat OpenShift cluster as the cluster-admin.
2. Create two projects corresponding to different projects.

```
oc create namespace project-1
oc create namespace project-2
```

3. Create the developer role for project-1.

```
cat << EOF | oc create -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: project-1
  name: developer-project-1
rules:
  - verbs:
      - '*'
    apiGroups:
      - apps
      - batch
      - autoscaling
      - extensions
      - networking.k8s.io
      - policy
      - apps.openshift.io
      - build.openshift.io
      - image.openshift.io
      - ingress.operator.openshift.io
      - route.openshift.io
      - snapshot.storage.k8s.io
      - template.openshift.io
    resources:
      - '*'
  - verbs:
      - '*'
    apiGroups:
      - ''
    resources:
      - bindings
      - configmaps
      - endpoints
      - events
      - persistentvolumeclaims
      - pods
      - pods/log
      - pods/attach
      - podtemplates
      - replicationcontrollers
```

```
      - services
      - limitranges
      - namespaces
      - componentstatuses
      - nodes
  - verbs:
      - '*'
    apiGroups:
      - trident.netapp.io
    resources:
      - tridentsnapshots
EOF
```

The role definition provided in this section is just an example. Developer roles must be defined based on end-user requirements.

4. Similarly, create developer roles for project-2.
5. All OpenShift and NetApp storage resources are usually managed by a storage admin. Access for storage administrators is controlled by the trident operator role that is created when Trident is installed. In addition to this, the storage admin also requires access to ResourceQuotas to control how storage is consumed.
6. Create a role for managing ResourceQuotas in all projects in the cluster to attach it to storage admin.

```
cat << EOF | oc create -f -
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: resource-quotas-role
rules:
  - verbs:
      - '*'
    apiGroups:
      - ''
    resources:
      - resourcequotas
  - verbs:
      - '*'
    apiGroups:
      - quota.openshift.io
    resources:
      - '*'
EOF
```

7. Make sure that the cluster is integrated with the organization's identity provider and that user groups are synchronized with cluster groups. The following example shows that the identity provider has been integrated with the cluster and synchronized with the user groups.

```
$ oc get groups
NAME                          USERS
ocp-netapp-storage-admins     ocp-netapp-storage-admin
ocp-project-1                 ocp-project-1-user
ocp-project-2                 ocp-project-2-user
```

8. Configure ClusterRoleBindings for storage admins.

```
cat << EOF | oc create -f -
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: netapp-storage-admin-trident-operator
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-netapp-storage-admins
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-operator
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: netapp-storage-admin-resource-quotas-cr
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-netapp-storage-admins
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: resource-quotas-role
EOF
```

For storage admins, two roles must be bound: trident-operator and resource-quotas.

9. Create RoleBindings for developers binding the developer-project-1 role to the corresponding group (ocp-project-1) in project-1.

```
cat << EOF | oc create -f -
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: project-1-developer
  namespace: project-1
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-project-1
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: developer-project-1
EOF
```

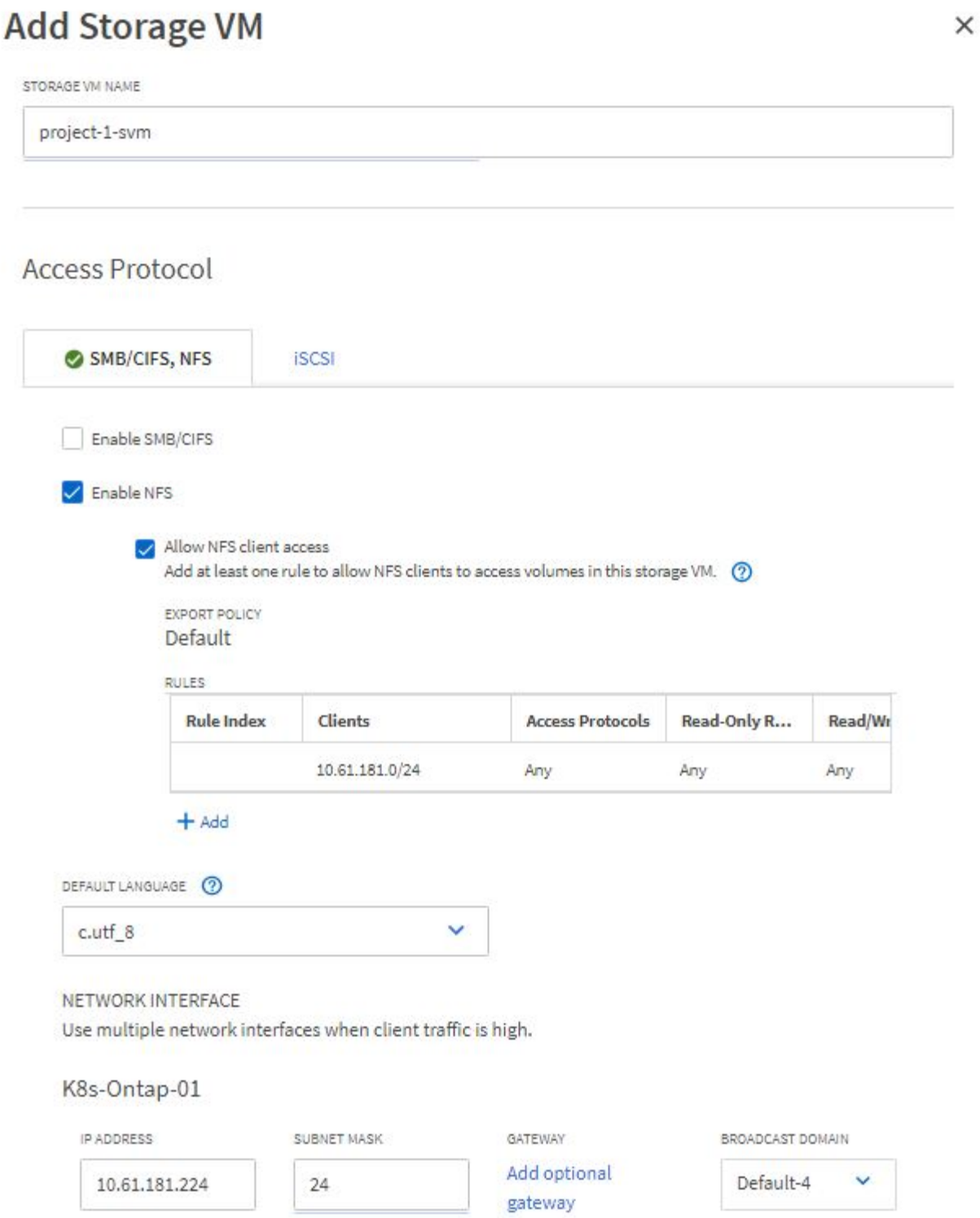
10. Similarly, create RoleBindings for developers binding the developer roles to the corresponding user group in project-2.

Next: Storage Administrator Tasks.

Configuration: Storage-admin tasks

The following resources must be configured by a storage administrator:

1. Log into the NetApp ONTAP cluster as admin.
2. Navigate to Storage > Storage VMs and click Add. Create two SVMs, one for project-1 and the other for project-2, by providing the required details. Also create a vsadmin account to manage the SVM and its resources.



3. Log into the Red Hat OpenShift cluster as the storage administrator.
4. Create the backend for project-1 and map it to the SVM dedicated to the project. NetApp recommends using the SVM's vsadmin account to connect the backend to SVM instead of using the ONTAP cluster administrator.

```
cat << EOF | tridentctl -n trident create backend -f
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "nfs_project_1",
    "managementLIF": "172.21.224.210",
    "dataLIF": "10.61.181.224",
    "svm": "project-1-svm",
    "username": "vsadmin",
    "password": "NetApp123"
}
EOF
```

We are using the ontap-nas driver for this example. Use the appropriate driver when creating the backend based on the use case.

We assume that Trident is installed in the trident project.

5. Similarly create the Trident backend for project-2 and map it to the SVM dedicated to project-2.
6. Next, create the storage classes. Create the storage class for project-1 and configure it to use the storage pools from backend dedicated to project-1 by setting the storagePools parameter.

```
cat << EOF | oc create -f -
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: project-1-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
  storagePools: "nfs_project_1:.*"
EOF
```

7. Likewise, create a storage class for project-2 and configure it to use the storage pools from backend dedicated to project-2.
8. Create a ResourceQuota to restrict resources in project-1 requesting storage from storageclasses dedicated to other projects.

```
cat << EOF | oc create -f -
kind: ResourceQuota
apiVersion: v1
metadata:
  name: project-1-sc-rq
  namespace: project-1
spec:
  hard:
    project-2-sc.storageclass.storage.k8s.io/persistentvolumeclaims: 0
EOF
```

9. Similarly, create a ResourceQuota to restrict resources in project-2 requesting storage from storageclasses dedicated to other projects.

Next: Validation.

Validation

To validate the multitenant architecture that was configured in the previous steps, complete the following steps:

Validate access to create PVCs or pods in assigned project

1. Log in as ocp-project-1-user, developer in project-1.
2. Check access to create a new project.

```
oc create ns sub-project-1
```

3. Create a PVC in project-1 using the storageclass that is assigned to project-1.

```
cat << EOF | oc create -f -
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: test-pvc-project-1
  namespace: project-1
  annotations:
    trident.netapp.io/reclaimPolicy: Retain
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: project-1-sc
EOF
```

4. Check the PV associated with the PVC.

```
oc get pv
```

5. Validate that the PV and its volume is created in an SVM dedicated to project-1 on NetApp ONTAP.

```
volume show -vserver project-1-svm
```

6. Create a pod in project-1 and mount the PVC created in previous step.

```
cat << EOF | oc create -f -
kind: Pod
apiVersion: v1
metadata:
  name: test-pvc-pod
  namespace: project-1
spec:
  volumes:
    - name: test-pvc-project-1
      persistentVolumeClaim:
       claimName: test-pvc-project-1
  containers:
    - name: test-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: test-pvc-project-1
EOF
```

7. Check if the pod is running and whether it mounted the volume.

```
oc describe pods test-pvc-pod -n project-1
```

Validate access to create PVCs or pods in another project or use resources dedicated to another project

1. Log in as ocp-project-1-user, developer in project-1.
2. Create a PVC in project-1 using the storageclass that is assigned to project-2.

```
cat << EOF | oc create -f -
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: test-pvc-project-1-sc-2
  namespace: project-1
  annotations:
    trident.netapp.io/reclaimPolicy: Retain
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: project-2-sc
EOF
```

3. Create a PVC in project-2.

```
cat << EOF | oc create -f -
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: test-pvc-project-2-sc-1
  namespace: project-2
  annotations:
    trident.netapp.io/reclaimPolicy: Retain
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: project-1-sc
EOF
```

4. Make sure that PVCs `test-pvc-project-1-sc-2` and `test-pvc-project-2-sc-1` were not created.

```
oc get pvc -n project-1
oc get pvc -n project-2
```

5. Create a pod in project-2.

```
cat << EOF | oc create -f -
kind: Pod
apiVersion: v1
metadata:
  name: test-pvc-pod
  namespace: project-1
spec:
  containers:
    - name: test-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
EOF
```

Validate access to view and edit Projects, ResourceQuotas, and StorageClasses

1. Log in as ocp-project-1-user, developer in project-1.
2. Check access to create new projects.

```
oc create ns sub-project-1
```

3. Validate access to view projects.

```
oc get ns
```

4. Check if the user can view or edit ResourceQuotas in project-1.

```
oc get resourcequotas -n project-1
oc edit resourcequotas project-1-sc-rq -n project-1
```

5. Validate that the user has access to view the storageclasses.

```
oc get sc
```

6. Check access to describe the storageclasses.
7. Validate the user’s access to edit the storageclasses.

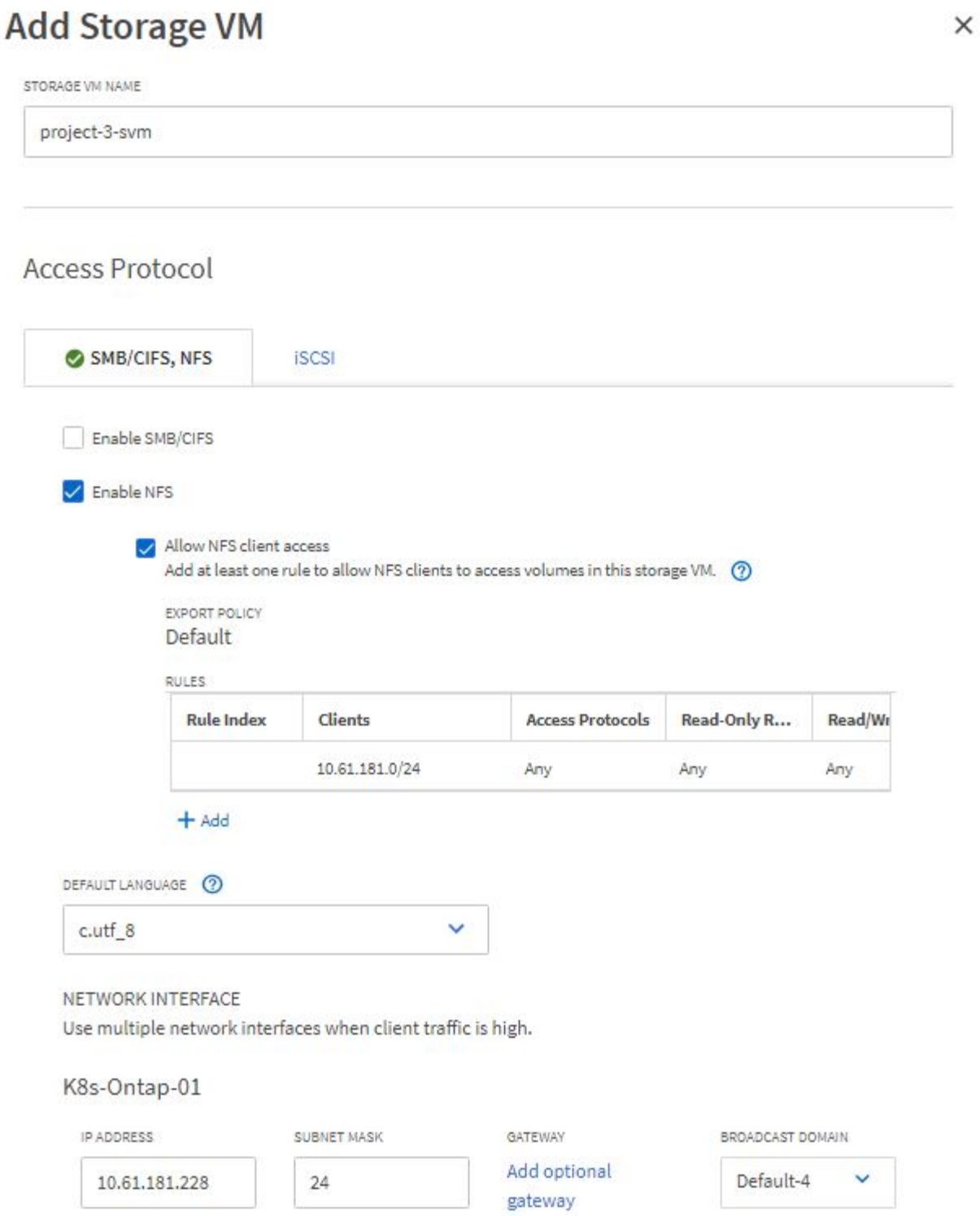
```
oc edit sc project-1-sc
```

Next: Scaling.

Scaling: Adding more projects

In a multitenant configuration, adding new projects with storage resources requires additional configuration to make sure that multitenancy is not violated. For adding more projects in a multitenant cluster, complete the following steps:

1. Log into the NetApp ONTAP cluster as a storage admin.
2. Navigate to `Storage → Storage VMs` and click `Add`. Create a new SVM dedicated to project-3. Also create a vsadmin account to manage the SVM and its resources.



3. Log into the Red Hat OpenShift cluster as cluster admin.
4. Create a new project.

```
oc create ns project-3
```

5. Make sure that the user group for project-3 is created on IdP and synchronized with the OpenShift cluster.

```
oc get groups
```

6. Create the developer role for project-3.

```
cat << EOF | oc create -f -
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: project-3
  name: developer-project-3
rules:
  - verbs:
      - '*'
    apiGroups:
      - apps
      - batch
      - autoscaling
      - extensions
      - networking.k8s.io
      - policy
      - apps.openshift.io
      - build.openshift.io
      - image.openshift.io
      - ingress.operator.openshift.io
      - route.openshift.io
      - snapshot.storage.k8s.io
      - template.openshift.io
    resources:
      - '*'
  - verbs:
      - '*'
    apiGroups:
      - ''
    resources:
      - bindings
      - configmaps
      - endpoints
      - events
      - persistentvolumeclaims
      - pods
      - pods/log
      - pods/attach
      - podtemplates
      - replicationcontrollers
      - services
```

```
      - limitranges
      - namespaces
      - componentstatuses
      - nodes
  - verbs:
      - '*'
    apiGroups:
      - trident.netapp.io
    resources:
      - tridentsnapshots
EOF
```

The role definition provided in this section is just an example. The developer role must be defined based on the end-user requirements.

7. Create RoleBinding for developers in project-3 binding the developer-project-3 role to the corresponding group (ocp-project-3) in project-3.

```
cat << EOF | oc create -f -
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: project-3-developer
  namespace: project-3
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-project-3
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: developer-project-3
EOF
```

8. Login to the Red Hat OpenShift cluster as storage admin
9. Create a Trident backend and map it to the SVM dedicated to project-3. NetApp recommends using the SVM's vsadmin account to connect the backend to the SVM instead of using the ONTAP cluster administrator.

```
cat << EOF | tridentctl -n trident create backend -f
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "nfs_project_3",
    "managementLIF": "172.21.224.210",
    "dataLIF": "10.61.181.228",
    "svm": "project-3-svm",
    "username": "vsadmin",
    "password": "NetApp!23"
}
EOF
```

We are using the ontap-nas driver for this example. Use the appropriate driver for creating the backend based on the use-case.

We assume that Trident is installed in the trident project.

10. Create the storage class for project-3 and configure it to use the storage pools from backend dedicated to project-3.

```
cat << EOF | oc create -f -
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: project-3-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
  storagePools: "nfs_project_3:.*"
EOF
```

11. Create a ResourceQuota to restrict resources in project-3 requesting storage from storageclasses dedicated to other projects.

```
cat << EOF | oc create -f -
kind: ResourceQuota
apiVersion: v1
metadata:
  name: project-3-sc-rq
  namespace: project-3
spec:
  hard:
    project-1-sc.storageclass.storage.k8s.io/persistentvolumeclaims: 0
    project-2-sc.storageclass.storage.k8s.io/persistentvolumeclaims: 0
EOF
```

12. Patch the ResourceQuotas in other projects to restrict resources in those projects from accessing storage from the storageclass dedicated to project-3.

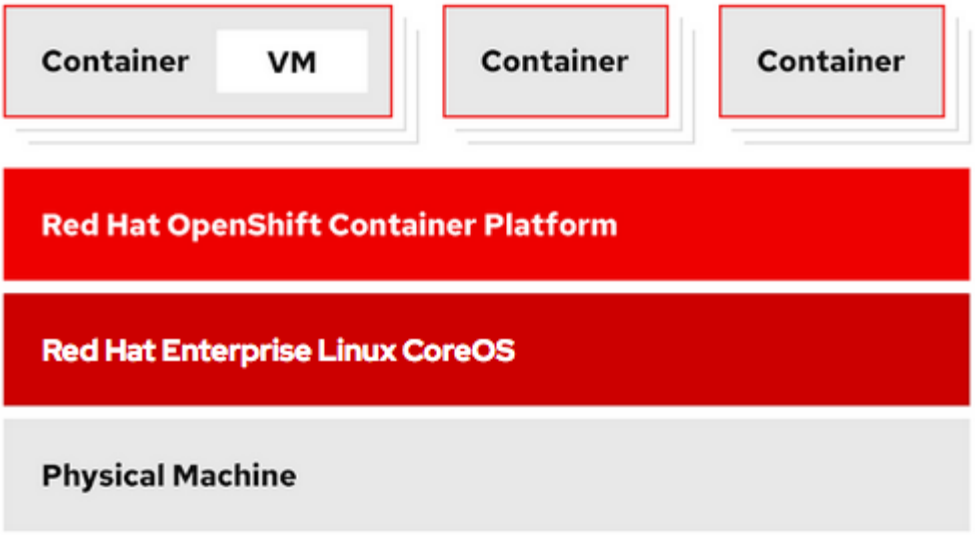
```
oc patch resourcequotas project-1-sc-rq -n project-1 --patch
'{"spec":{"hard":{ "project-3-
sc.storageclass.storage.k8s.io/persistentvolumeclaims": 0}}}'
oc patch resourcequotas project-2-sc-rq -n project-2 --patch
'{"spec":{"hard":{ "project-3-
sc.storageclass.storage.k8s.io/persistentvolumeclaims": 0}}}'
```

Red Hat OpenShift Virtualization with NetApp ONTAP

Red Hat OpenShift Virtualization with NetApp ONTAP

Depending on the specific use case, both containers and virtual machines (VMs) can serve as optimal platforms for different types of applications. Therefore, many organizations run some of their workloads on containers and some on VMs. Often, this leads organizations to face additional challenges by having to manage separate platforms: a hypervisor for VMs and a container orchestrator for applications.

To address this challenge, Red Hat introduced OpenShift Virtualization (formerly known as Container Native Virtualization) starting from OpenShift version 4.6. The OpenShift Virtualization feature enables you to run and manage virtual machines alongside containers on the same OpenShift Container Platform installation, providing hybrid management capability to automate deployment and management of VMs through operators. In addition to creating VMs in OpenShift, with OpenShift Virtualization, Red Hat also supports importing VMs from VMware vSphere, Red Hat Virtualization, and Red Hat OpenStack Platform deployments.



Certain features like live VM migration, VM disk cloning, VM snapshots and so on are also supported by OpenShift Virtualization with assistance from Astra Trident when backed by NetApp ONTAP. Examples of each of these workflows are discussed later in this document in their respective sections.

To learn more about Red Hat OpenShift Virtualization, see the documentation here.

Next: Deployment Prerequisites.

Deployment

Deploy Red Hat OpenShift Virtualization with NetApp ONTAP

Prerequisites

- A Red Hat OpenShift cluster (later than version 4.6) installed on bare-metal infrastructure with RHCOS worker nodes
- The OpenShift cluster must be installed via installer provisioned infrastructure (IPI)
- Deploy Machine Health Checks to maintain HA for VMs
- A NetApp ONTAP cluster
- Astra Trident installed on the OpenShift cluster
- A Trident backend configured with an SVM on ONTAP cluster
- A StorageClass configured on the OpenShift cluster with Astra Trident as the provisioner
- Cluster-admin access to Red Hat OpenShift cluster
- Admin access to NetApp ONTAP cluster
- An admin workstation with tridentctl and oc tools installed and added to $PATH

Because OpenShift Virtualization is managed by an operator installed on the OpenShift cluster, it imposes additional overhead on memory, CPU, and storage, which must be accounted for while planning the hardware requirements for the cluster. See the documentation here for more details.

Optionally, you can also specify a subset of the OpenShift cluster nodes to host the OpenShift Virtualization operators, controllers, and VMs by configuring node placement rules. To configure node placement rules for OpenShift Virtualization, follow the documentation here.

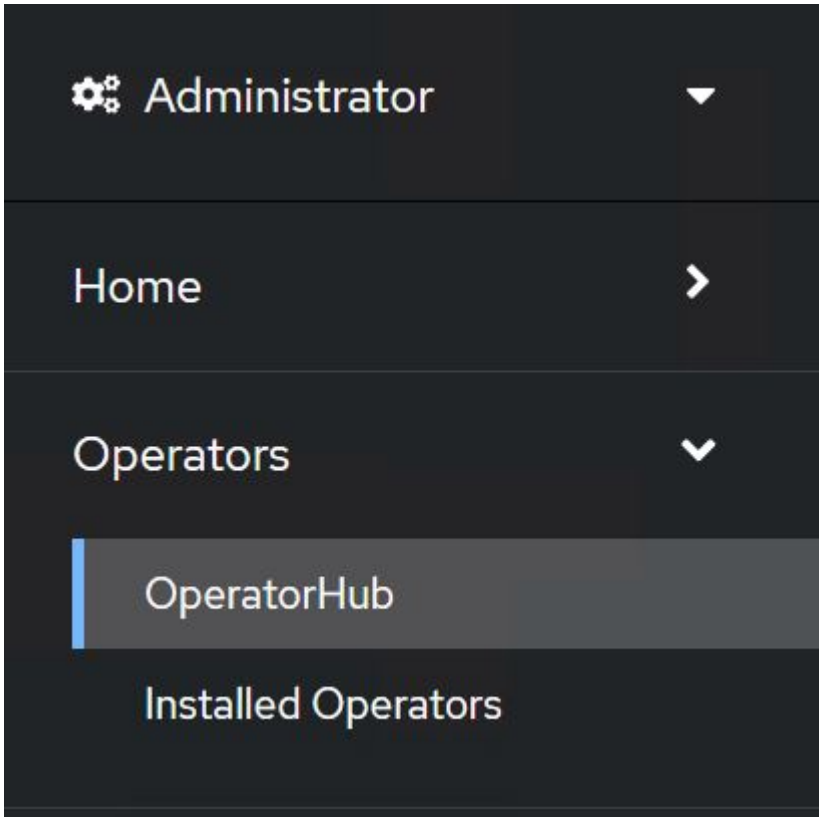
For the storage backing OpenShift Virtualization, NetApp recommends having a dedicated StorageClass that requests storage from a particular Trident backend, which in turn is backed by a dedicated SVM. This maintains a level of multitenancy with regard to the data being served for VM-based workloads on the OpenShift cluster.

Next: Deploy via operator.

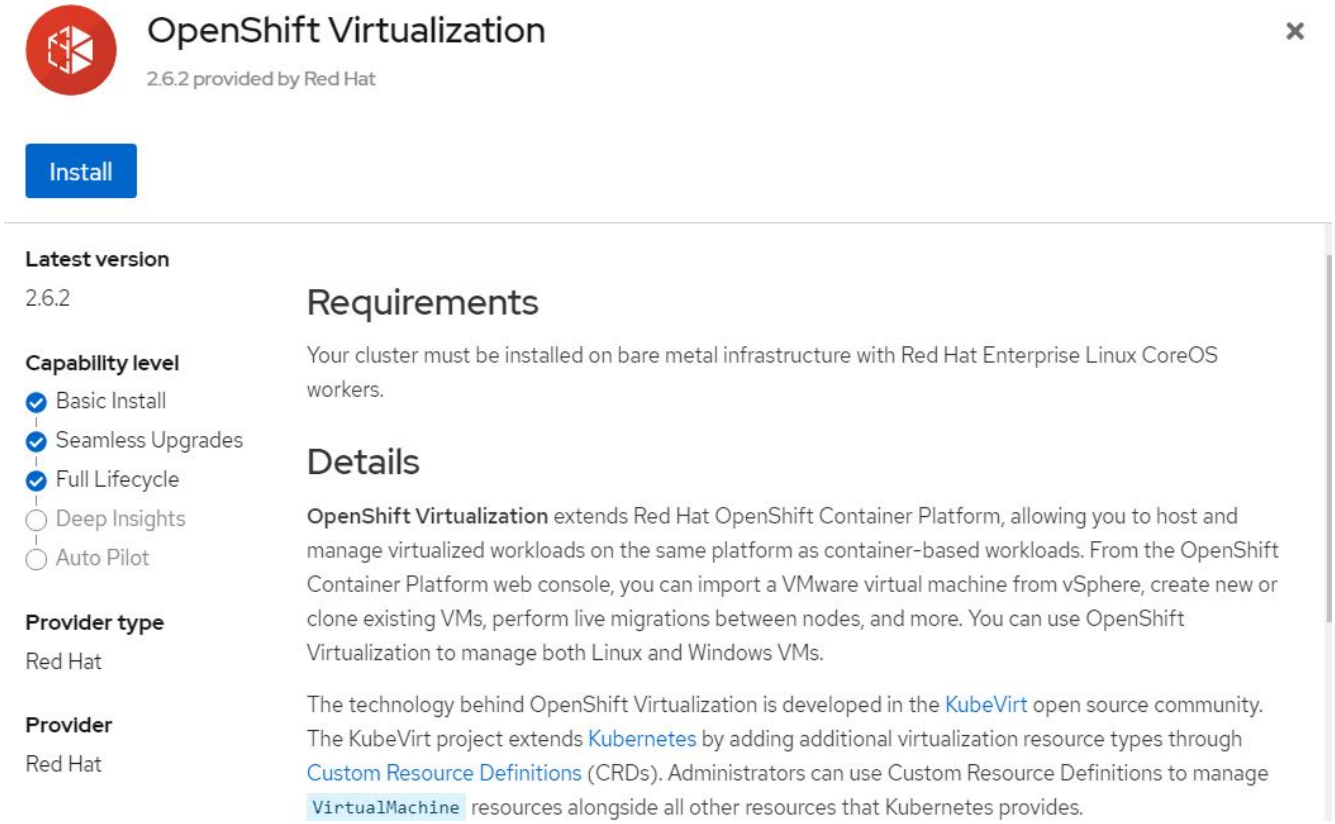
Deploy Red Hat OpenShift Virtualization with NetApp ONTAP

To install OpenShift Virtualization, complete the following steps:

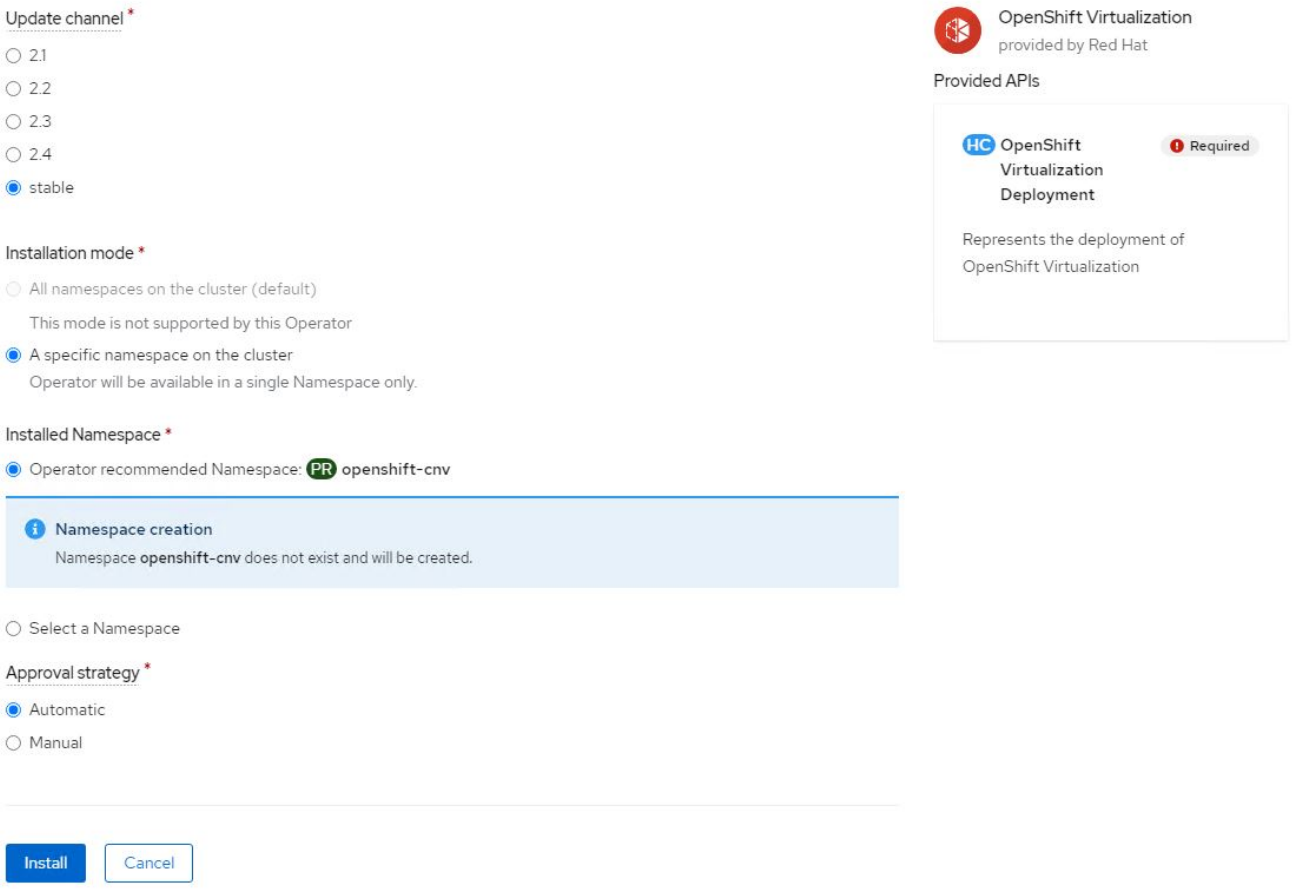
1. Log into the Red Hat OpenShift bare-metal cluster with cluster-admin access.
2. Select Administrator from the Perspective drop down.
3. Navigate to Operators > OperatorHub and search for OpenShift Virtualization.



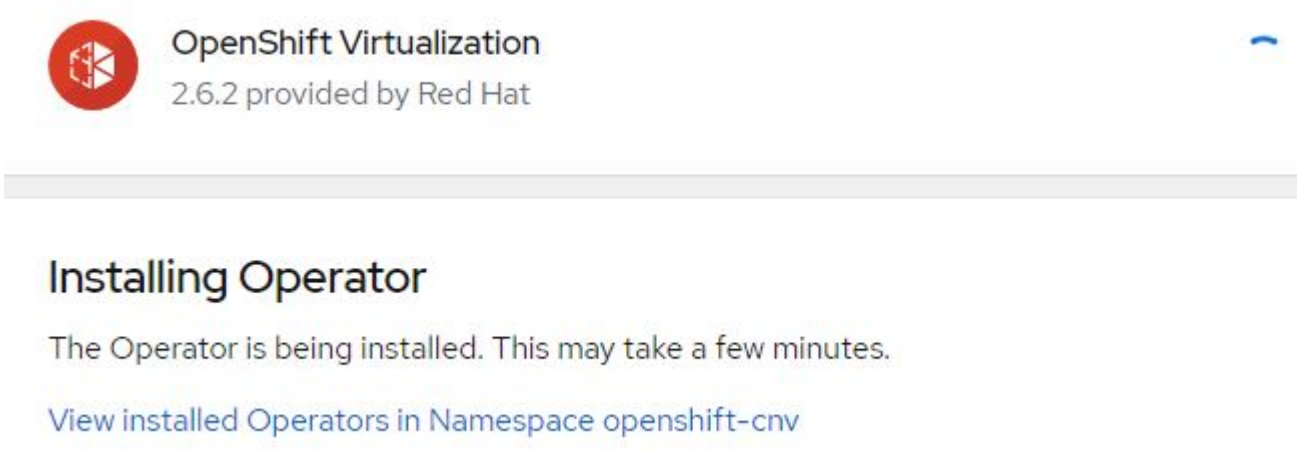
4. Select the OpenShift Virtualization tile and click Install.



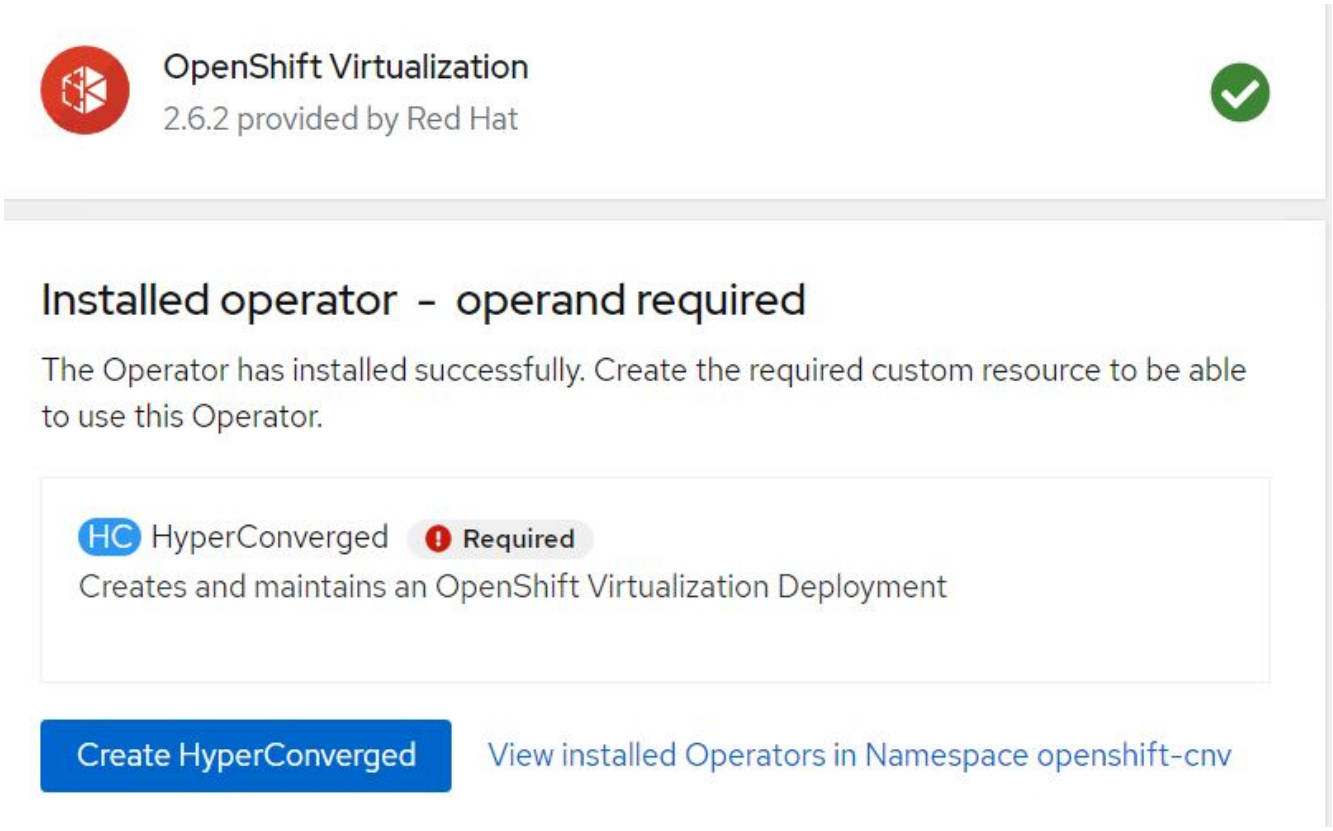
5. On the Install Operator screen, leave all default parameters and click Install.



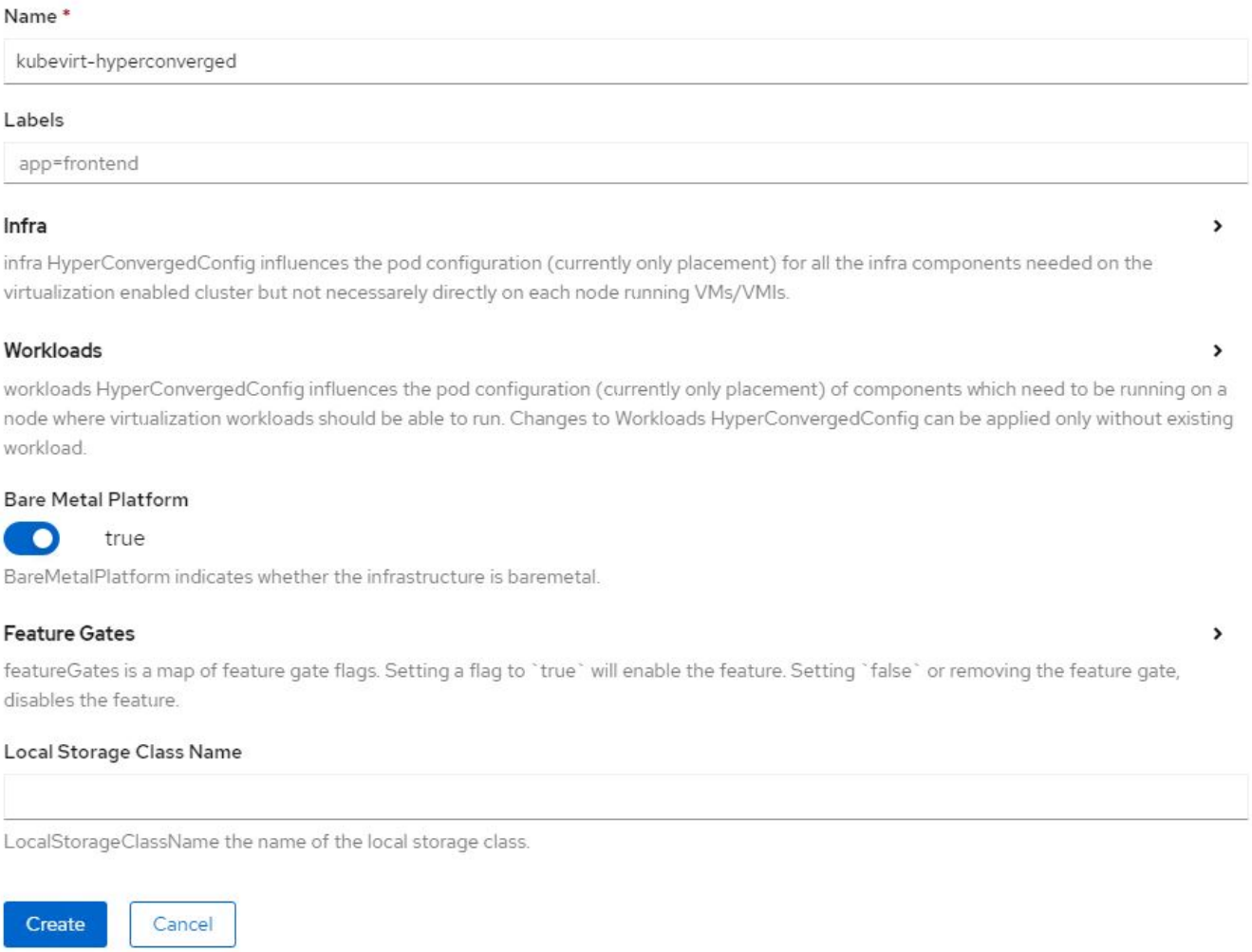
6. Wait for the operator installation to complete.



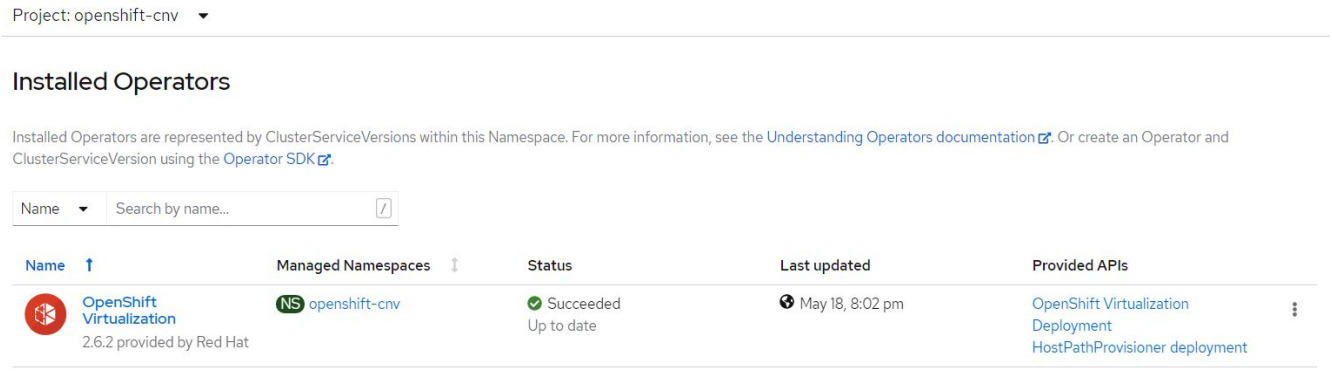
7. After the operator has installed, click Create HyperConverged.



8. On the Create HyperConverged screen, click Create, accepting all default parameters. This step starts the installation of OpenShift Virtualization.



9. After all the pods move to the Running state in the openshift-cnv namespace and the OpenShift Virtualization operator is in the Succeeded state, the operator is ready to use. VMs can now be created on the OpenShift cluster.



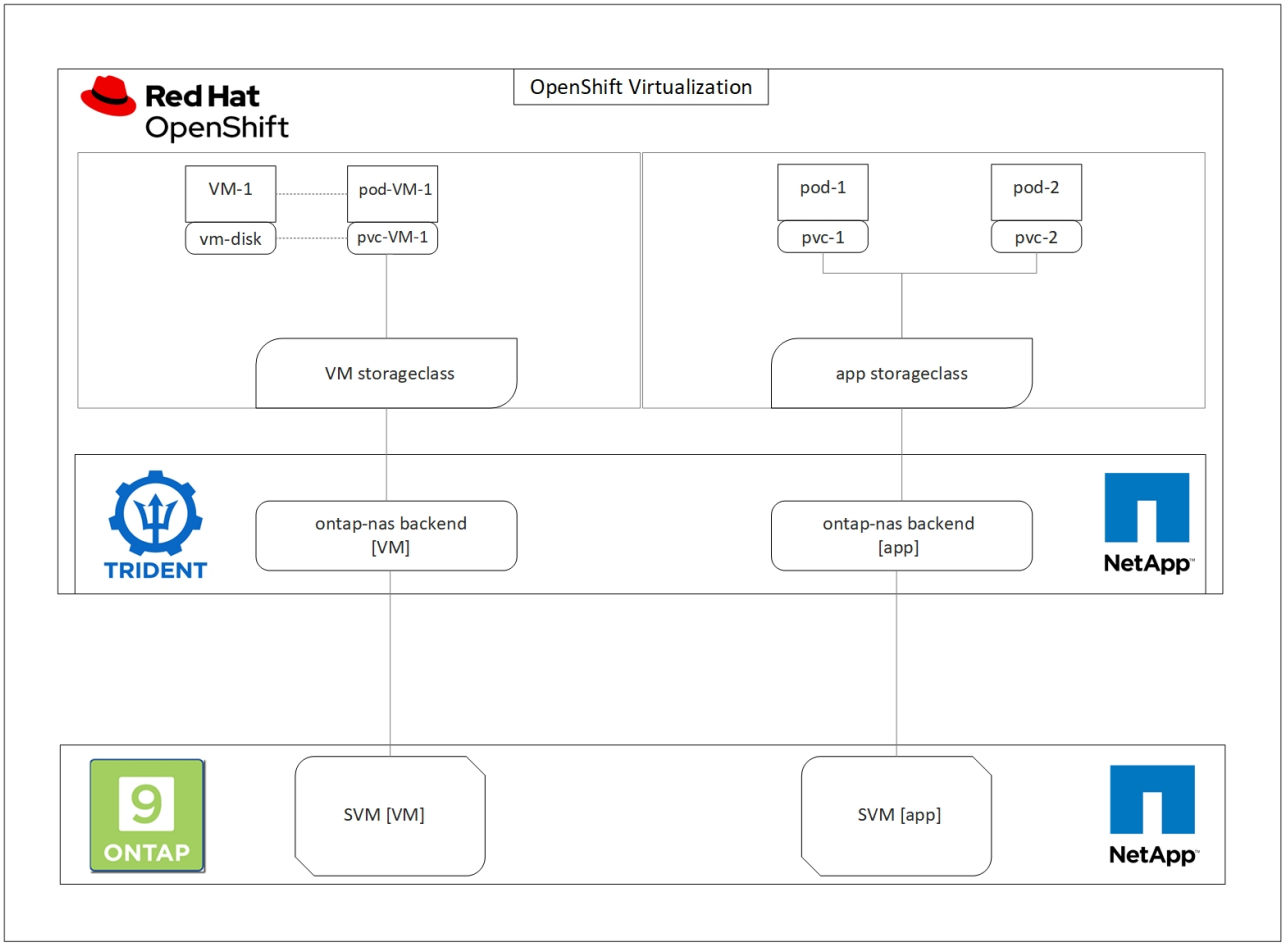
Next: Workflows: Create VM.

Workflows

Workflows: Red Hat OpenShift Virtualization with NetApp ONTAP

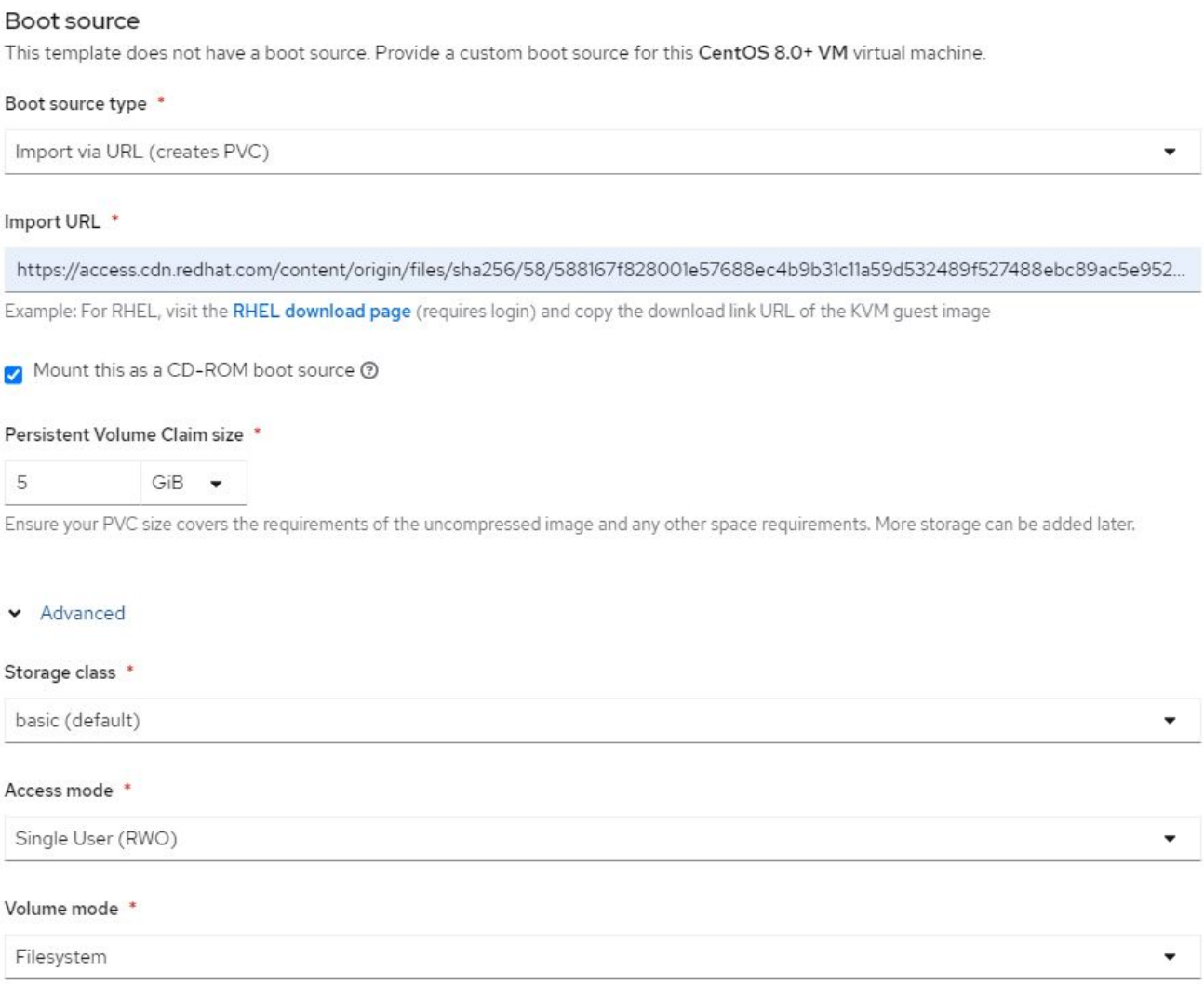
Create VM

VMs are stateful deployments that require volumes to host the operating system and data. With CNV, because the VMs are run as pods, the VMs are backed by PVs hosted on NetApp ONTAP through Trident. These volumes are attached as disks and store the entire filesystem including the boot source of the VM.

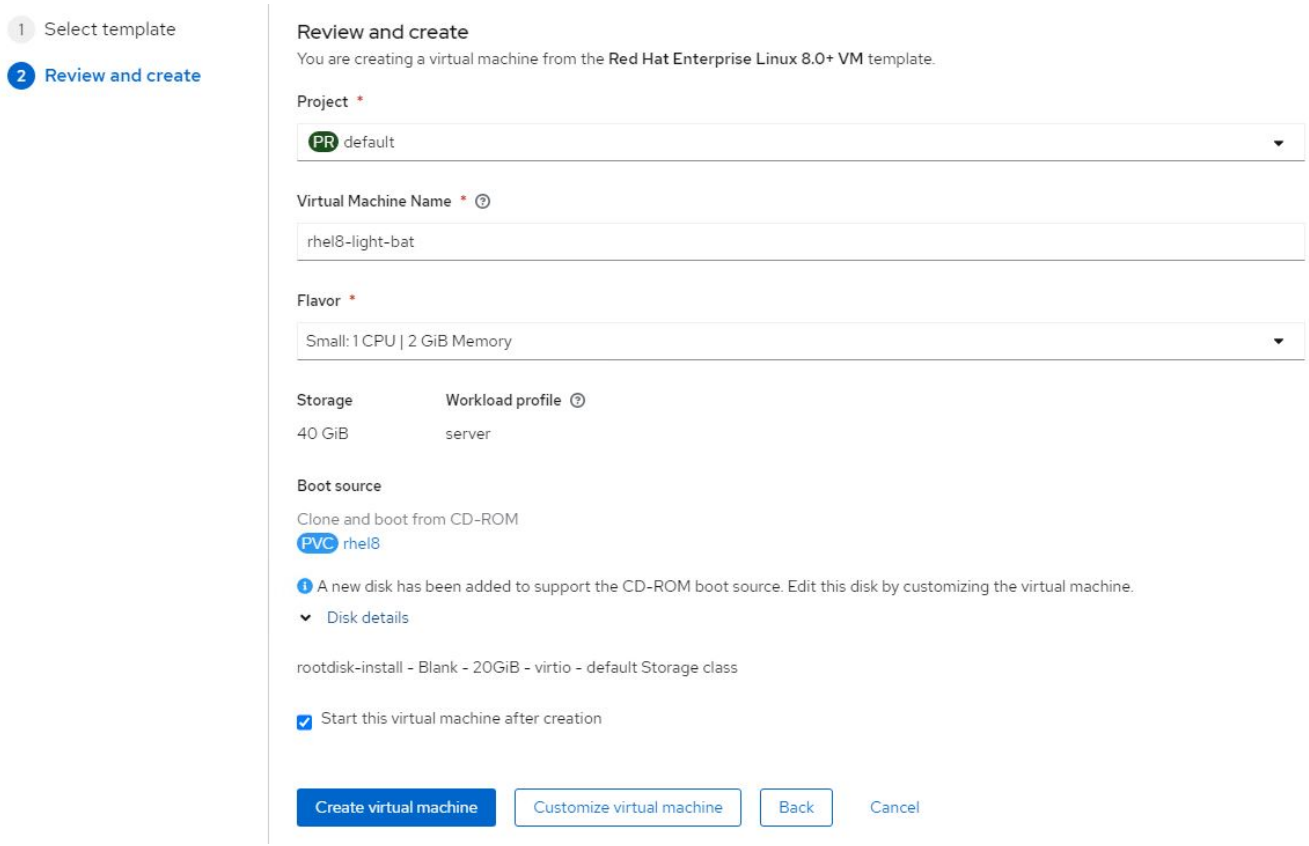


To create a virtual machine on the OpenShift cluster, complete the following steps:

1. Navigate to Workloads > Virtualization > Virtual Machines and click Create > With Wizard.
2. Select the desired the operating system and click Next.
3. If the selected operating system has no boot source configured, you must configure it. For Boot Source, select whether you want to import the OS image from an URL or from a registry and provide the corresponding details. Expand Advanced and select the Trident-backed StorageClass. Then click Next.



4. If the selected operating system already has a boot source configured, the previous step can be skipped.
5. In the Review and Create pane, select the project you want to create the VM in and furnish the VM details. Make sure that the boot source is selected to be Clone and boot from CD-ROM with the appropriate PVC assigned for the selected OS.



6. If you wish to customize the virtual machine, click Customize Virtual Machine and modify the required parameters.
7. Click Create Virtual Machine to create the virtual machine; this spins up a corresponding pod in the background.

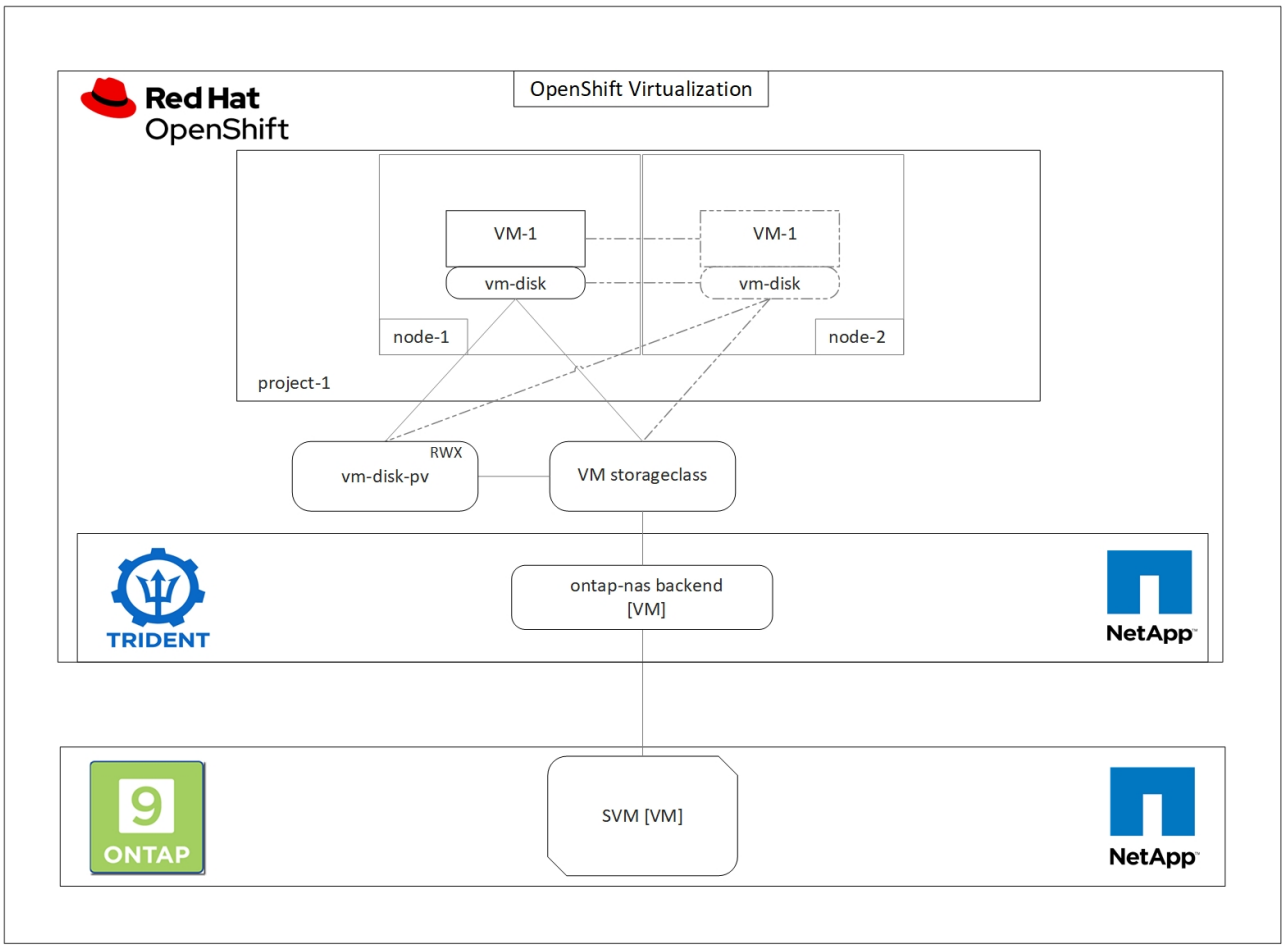
When a boot source is configured for a template or an operating system from an URL or from a registry, it creates a PVC in the `openshift-virtualization-os-images` project and downloads the KVM guest image to the PVC. You must make sure that template PVCs have enough provisioned space to accommodate the KVM guest image for the corresponding OS. These PVCs are then cloned and attached as rootdisks to virtual machines when they are created using the respective templates in any project.

Next: Workflows: VM Live Migration.

Workflows: Red Hat OpenShift Virtualization with NetApp ONTAP

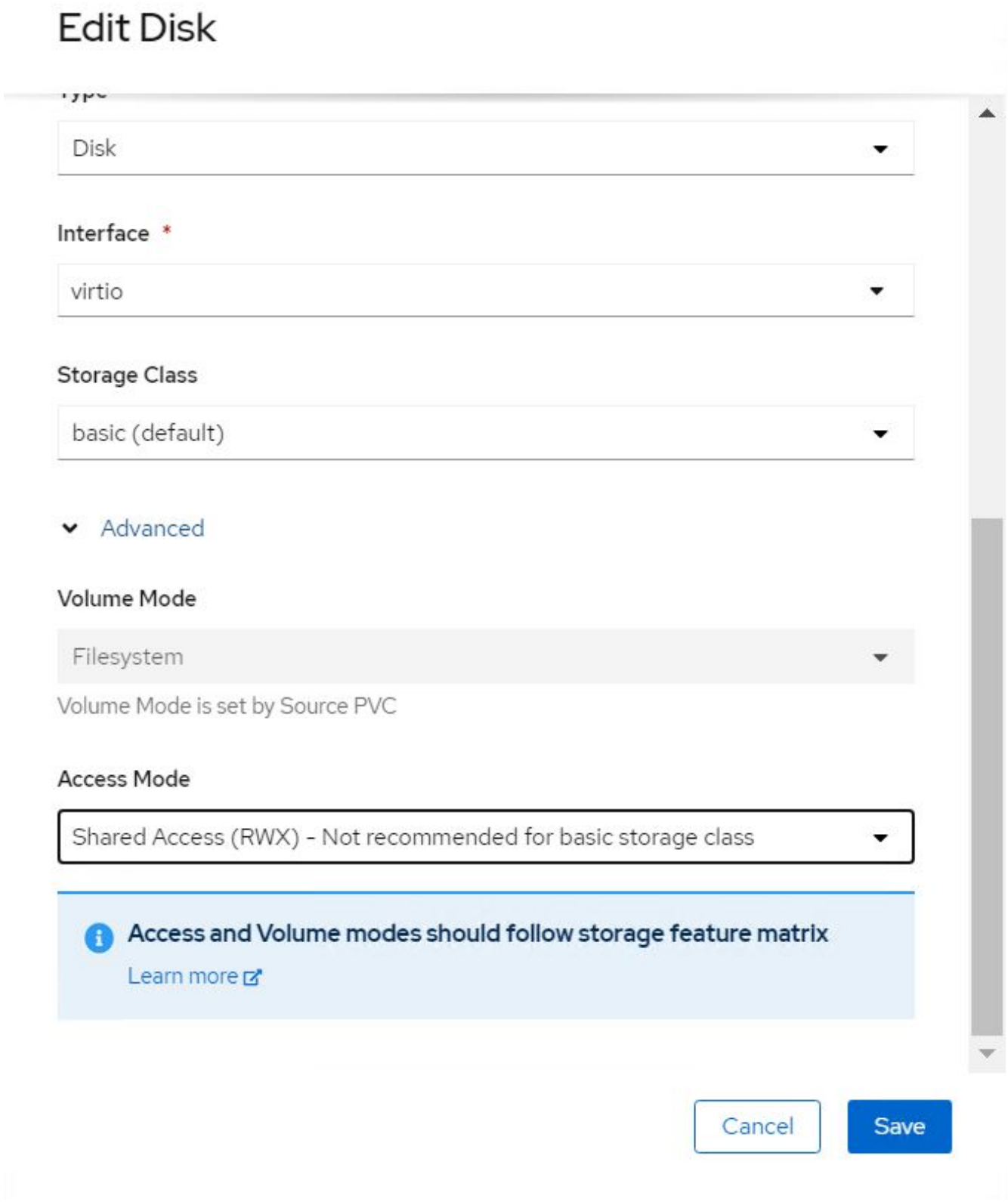
VM Live Migration

Live Migration is a process of migrating a VM instance from one node to another in an OpenShift cluster with no downtime. For live migration to work in an OpenShift cluster, VMs must be bound to PVCs with shared ReadWriteMany access mode. Astra Trident backend configured with an SVM on a NetApp ONTAP cluster that is enabled for NFS protocol supports shared ReadWriteMany access for PVCs. Therefore, the VMs with PVCs that are requested from StorageClasses provisioned by Trident from NFS-enabled SVM can be migrated with no downtime.



To create a VM bound to PVCs with shared ReadWriteMany access:

1. Navigate to Workloads > Virtualization > Virtual Machines and click Create > With Wizard.
2. Select the desired the operating system and click Next. Let us assume the selected OS already had a boot source configured with it.
3. In the Review and Create pane, select the project you want to create the VM in and furnish the VM details. Make sure that the boot source is selected to be Clone and boot from CD-ROM with the appropriate PVC assigned for the selected OS.
4. Click Customize Virtual Machine and then click Storage.
5. Click the ellipsis next to rootdisk, and make sure that the storageclass provisioned using Trident is selected. Expand Advanced and select Shared Access (RWX) for Access Mode. Then click Save.



6. Click Review and confirm and then click Create Virtual Machine.

To manually migrate a VM to another node in the OpenShift cluster, complete the following steps.

1. Navigate to Workloads > Virtualization > Virtual Machines.

2. For the VM you wish to migrate, click the ellipsis, and then click Migrate the Virtual Machine.
3. Click Migrate when the message pops up to confirm.

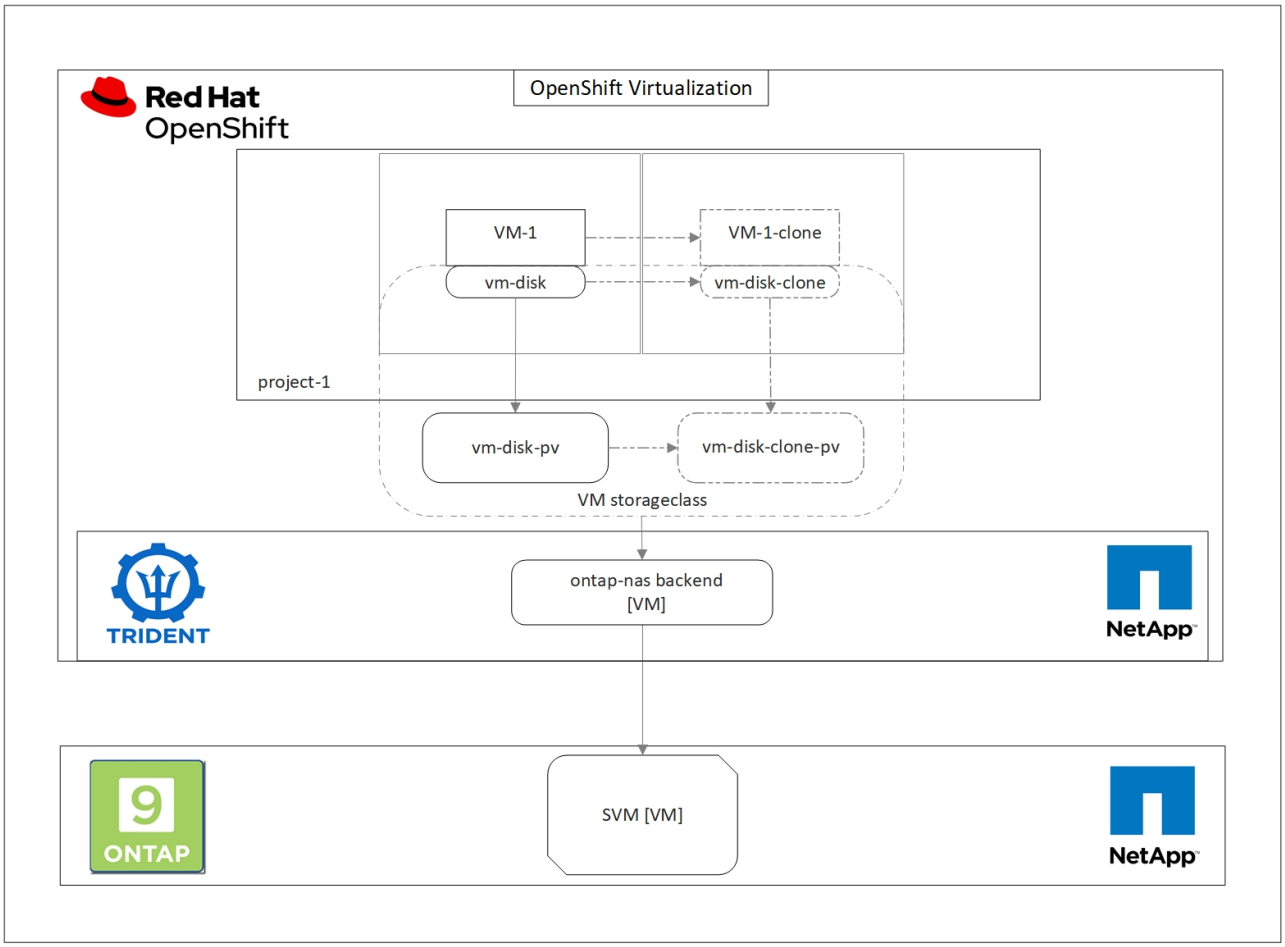
> A VM instance in an OpenShift cluster automatically migrates to another node when the original node is placed into maintenance mode if the evictionStrategy is set to LiveMigrate.

Next: Workflows: VM Cloning.

Workflows: Red Hat OpenShift Virtualization with NetApp ONTAP

VM cloning

Cloning an existing VM in OpenShift is achieved with the support of Astra Trident's Volume CSI cloning feature. CSI volume cloning allows for creation of a new PVC using an existing PVC as the data source by duplicating its PV. After the new PVC is created, it functions as a separate entity and without any link to or dependency on the source PVC.



There are certain restrictions with CSI volume cloning to consider:

1. Source PVC and destination PVC must be in the same project.
2. Cloning is supported within the same storage class.
3. Cloning can be performed only when source and destination volumes use the same VolumeMode setting;

for example, a block volume can only be cloned to another block volume.

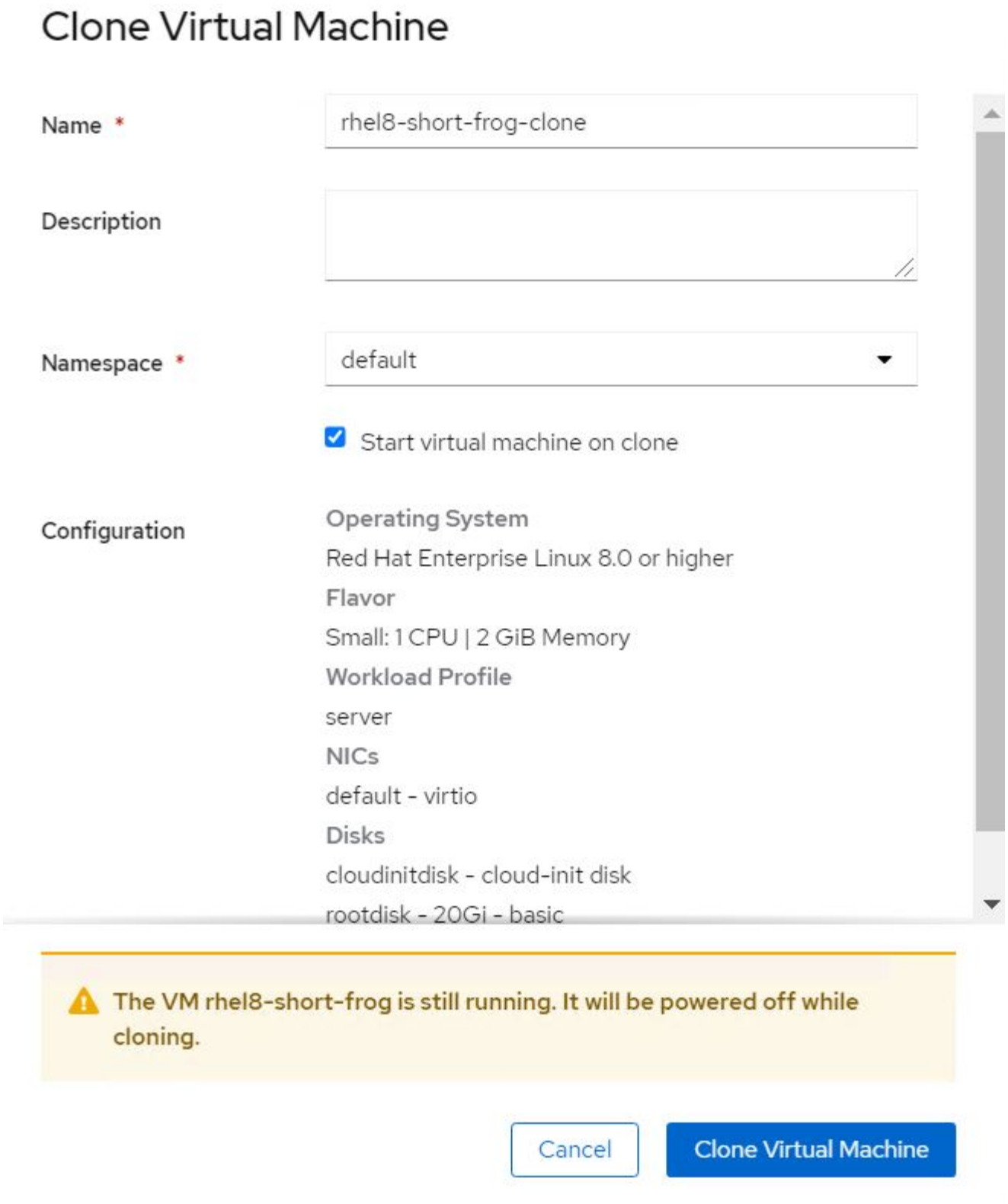
VMs in an OpenShift cluster can be cloned in two ways:

1. By shutting down the source VM
2. By keeping the source VM live

By Shutting down the source VM

Cloning an existing VM by shutting down the VM is a native OpenShift feature that is implemented with support from Astra Trident. Complete the following steps to clone a VM.

1. Navigate to Workloads > Virtualization > Virtual Machines and click the ellipsis next to the virtual machine you wish to clone.
2. Click Clone Virtual Machine and provide the details for the new VM.

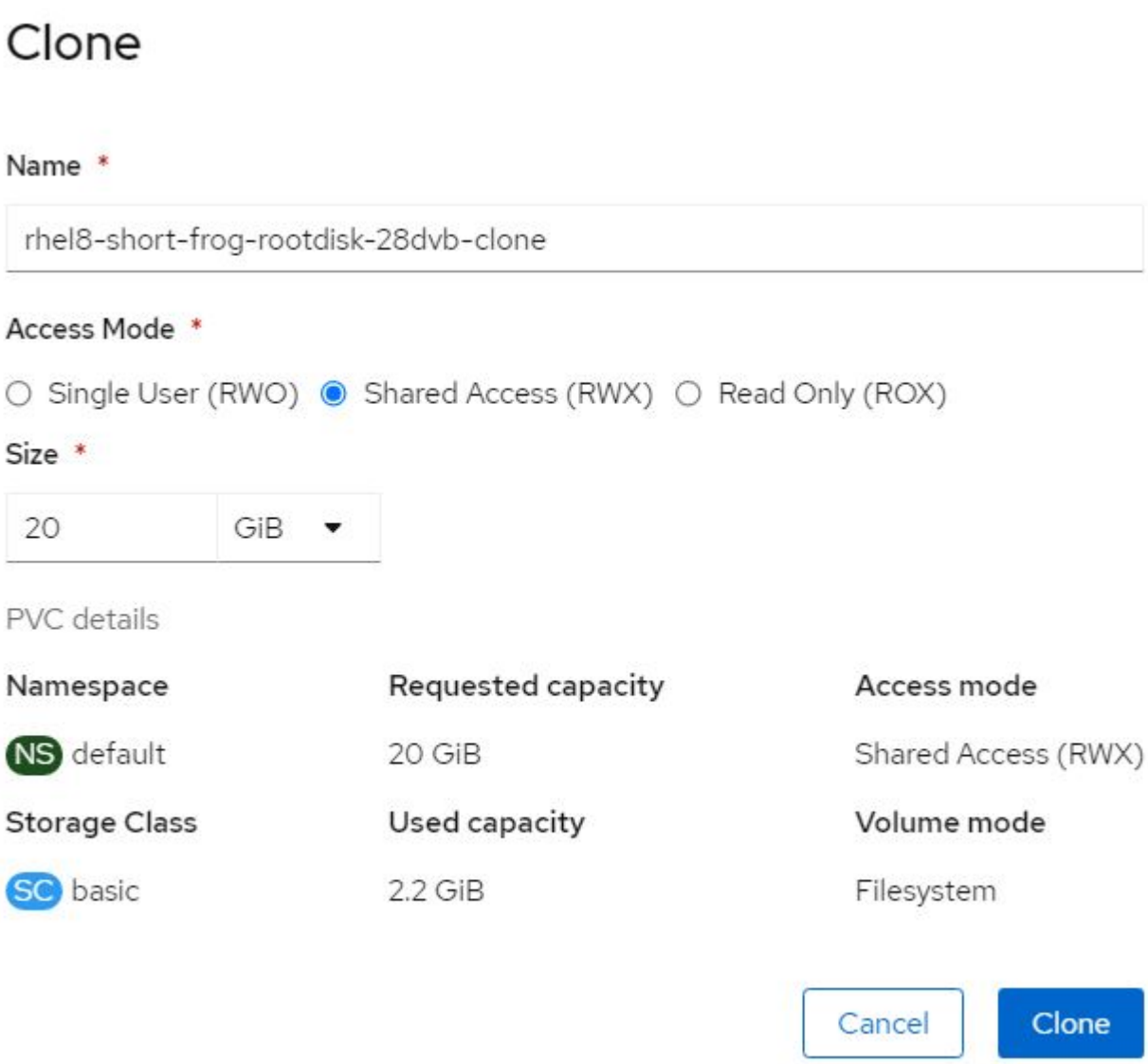


3. Click Clone Virtual Machine; this shuts down the source VM and initiates the creation of the clone VM.
4. After this step is completed, you can access and verify the content of the cloned VM.

By keeping the source VM live

An existing VM can also be cloned by cloning the existing PVC of the source VM and then creating a new VM using the cloned PVC. This method does not require you to shut down the source VM. Complete the following steps to clone a VM without shutting it down.

1. Navigate to Storage > PersistentVolumeClaims and click the ellipsis next to the PVC that is attached to the source VM.
2. Click Clone PVC and furnish the details for the new PVC.



3. Then click Clone. This creates a PVC for the new VM.
4. Navigate to Workloads > Virtualization > Virtual Machines and click Create > With YAML.
5. In the spec > template > spec > volumes section, attach the cloned PVC instead of the container disk. Provide all other details for the new VM according to your requirements.

```
- name: rootdisk
  persistentVolumeClaim:
    claimName: rhel8-short-frog-rootdisk-28dvb-clone
```

6. Click Create to create the new VM.
7. After the VM is created successfully, access and verify that the new VM is a clone of the source VM.

Next: Workflows: Create VM from a Snapshot.

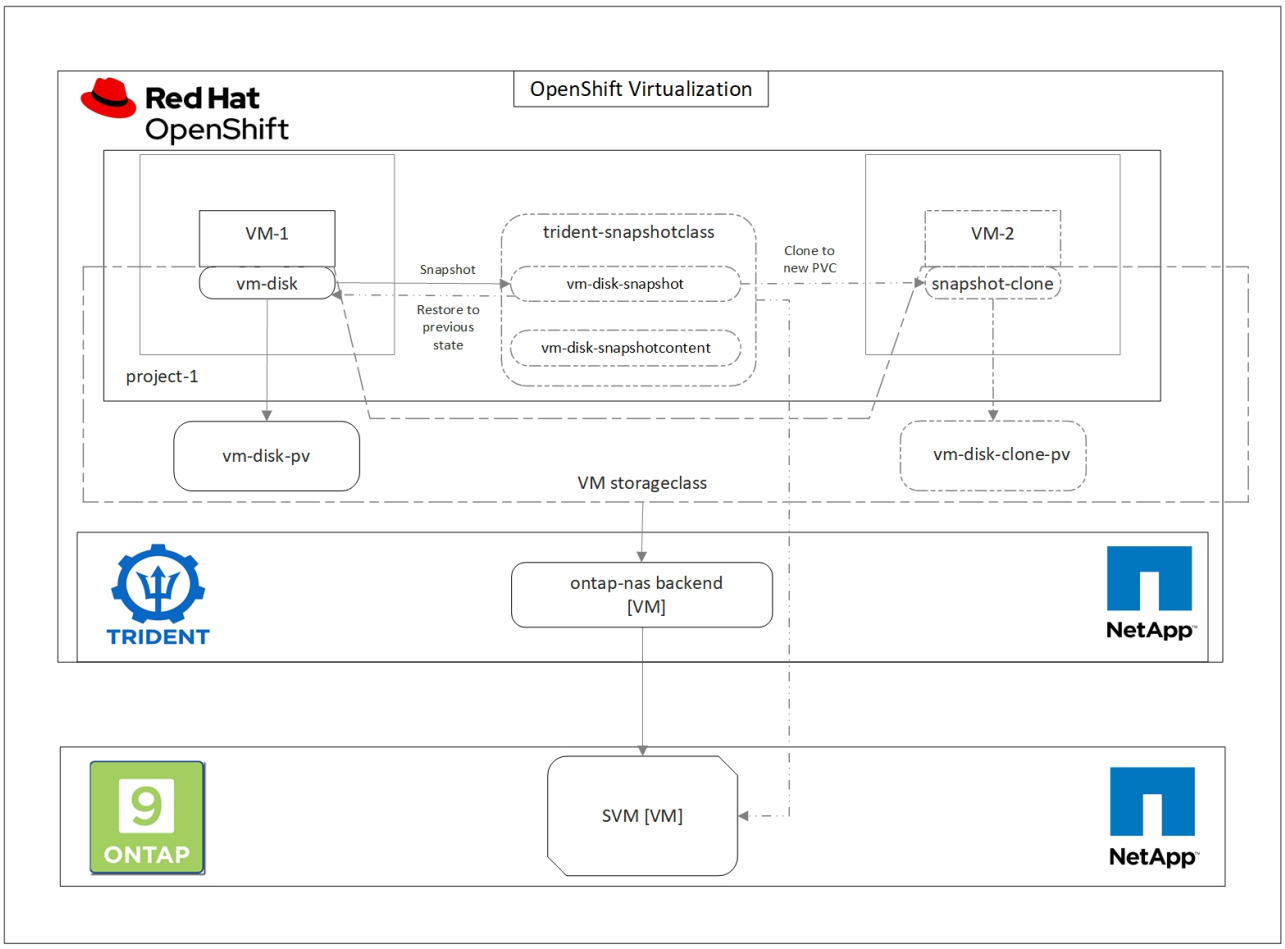
Workflows: Red Hat OpenShift Virtualization with NetApp ONTAP

Create VM from a Snapshot

With Astra Trident and Red Hat OpenShift, users can take a snapshot of a persistent volume on Storage Classes provisioned by it. With this feature, users can take a point-in-time copy of a volume and use it to create a new volume or restore the same volume back to a previous state. This enables or supports a variety of use-cases, from rollback to clones to data restore.

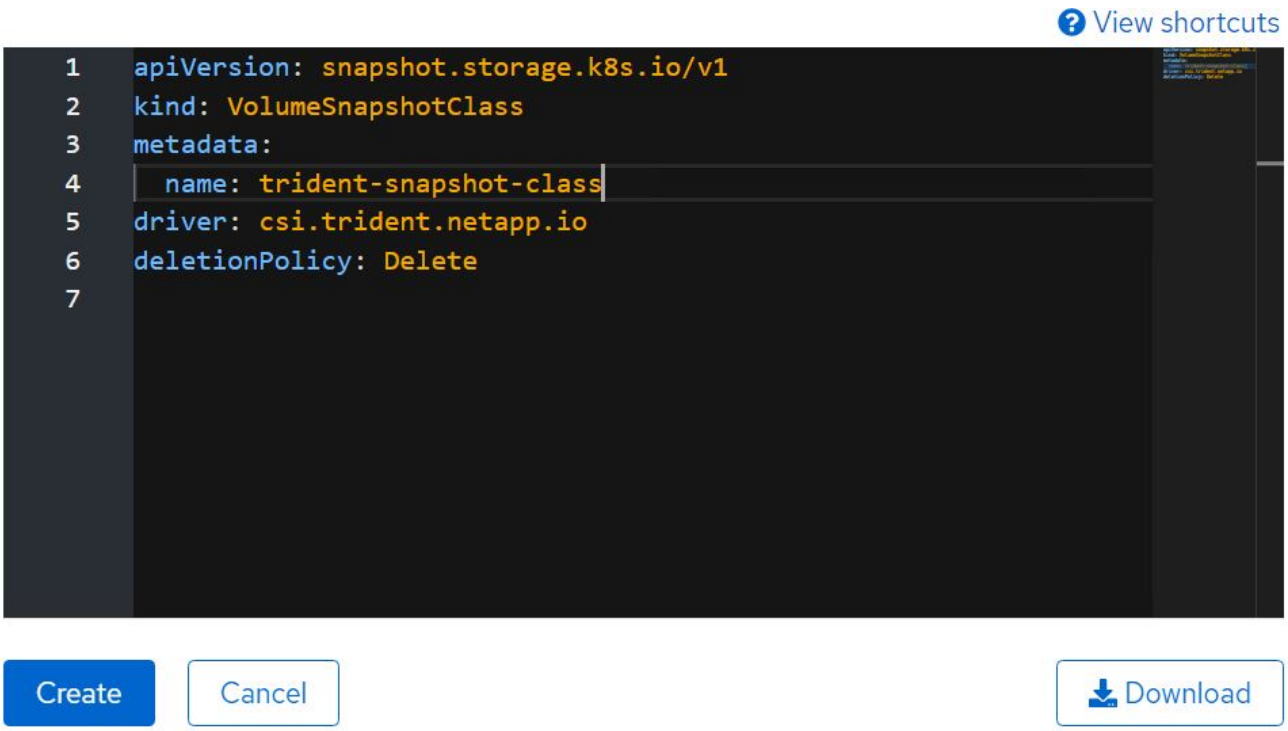
For Snapshot operations in OpenShift, the resources VolumeSnapshotClass, VolumeSnapshot, and VolumeSnapshotContent must be defined.

- A VolumeSnapshotContent is the actual snapshot taken from a volume in the cluster. It is cluster-wide resource analogous to PersistentVolume for storage.
- A VolumeSnapshot is a request for creating the snapshot of a volume. It is analogous to a PersistentVolumeClaim.
- VolumeSnapshotClass lets the administrator specify different attributes for a VolumeSnapshot. It allows you to have different attributes for different snapshots taken from the same volume.

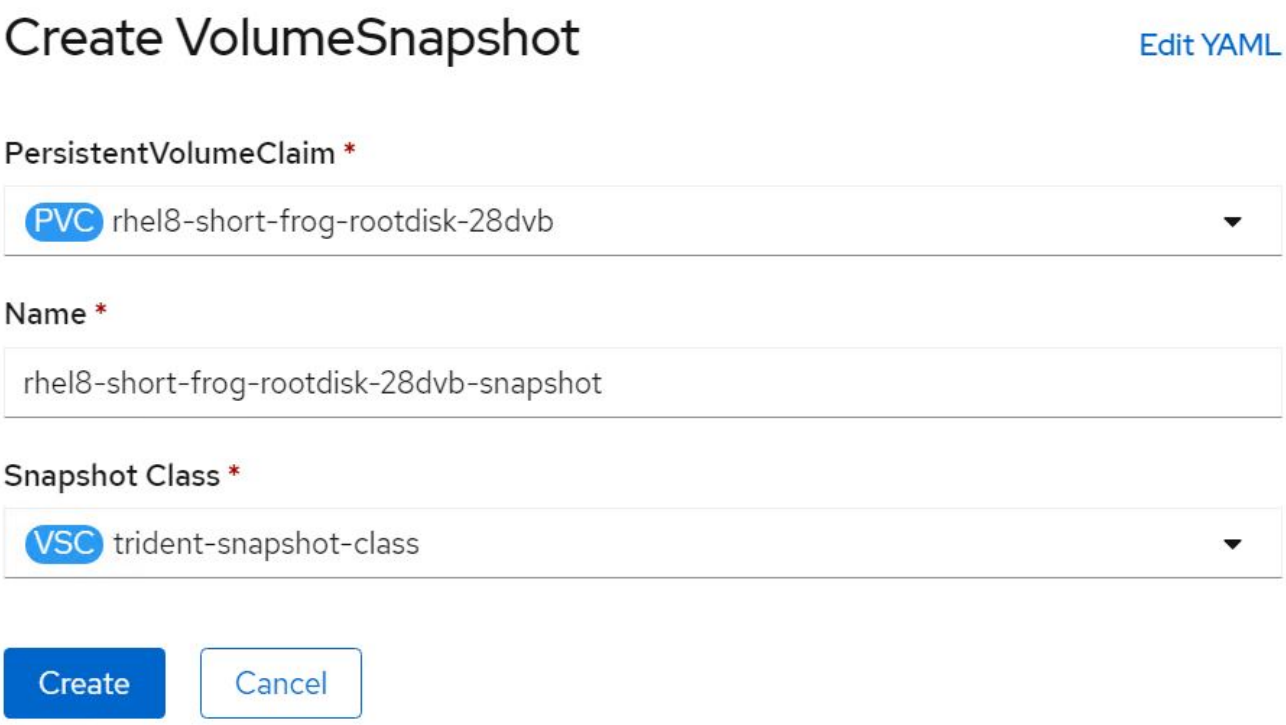


To create Snapshot of a VM, complete the following steps:

1. Create a VolumeSnapshotClass that can then be used to create a VolumeSnapshot. Navigate to Storage > VolumeSnapshotClasses and click Create VolumeSnapshotClass.
2. Enter the name of the Snapshot Class, enter csi.trident.netapp.io for the driver, and click Create.



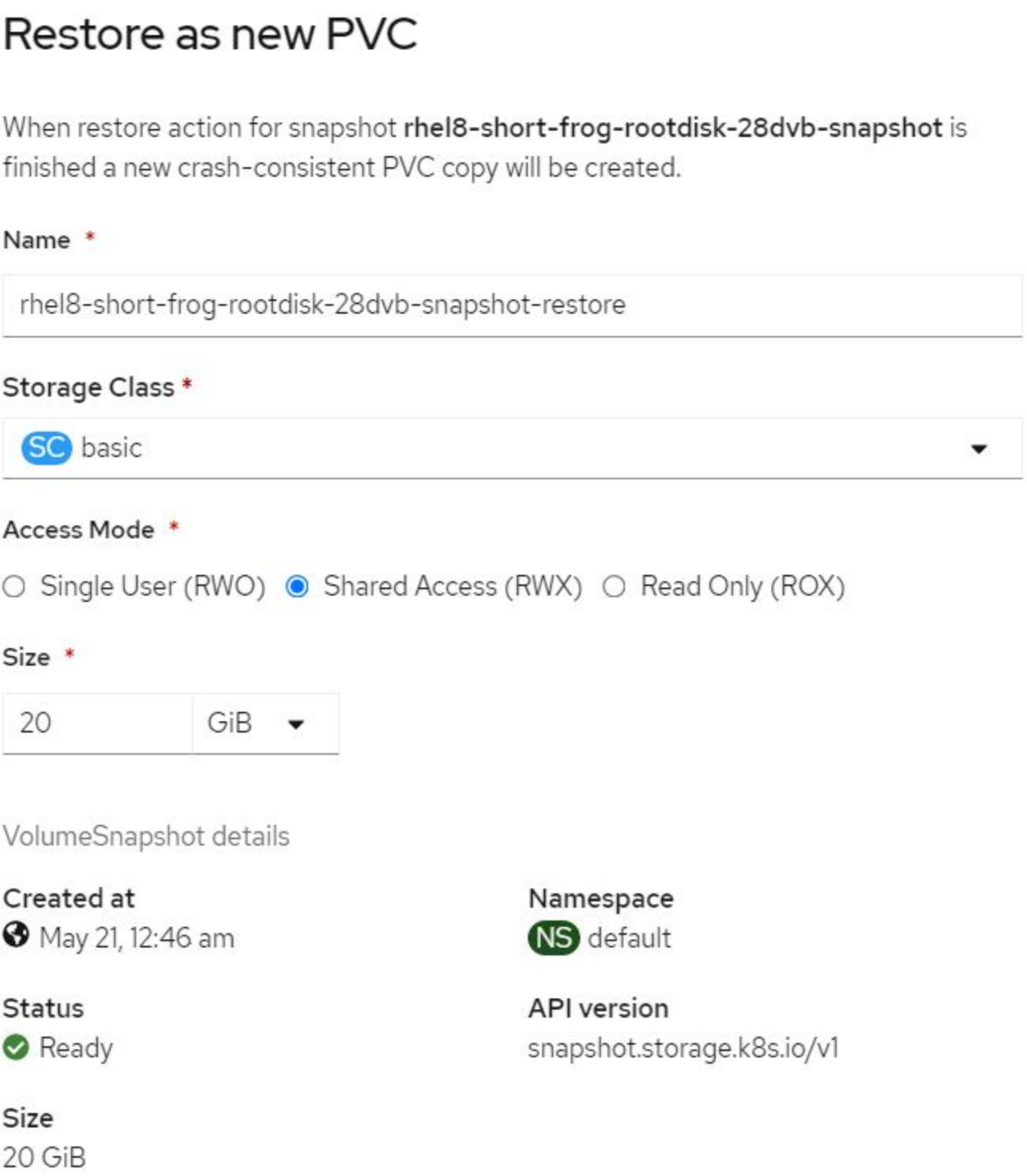
3. Identify the PVC that is attached to the source VM and then create a Snapshot of that PVC. Navigate to `Storage > VolumeSnapshots` and click Create VolumeSnapshots.
4. Select the PVC that you want to create the Snapshot for, enter the name of the Snapshot or accept the default, and select the appropriate VolumeSnapshotClass. Then click Create.



5. This creates the snapshot of the PVC at that point in time.

Create a new VM from the snapshot

1. First, restore the Snapshot into a new PVC. Navigate to Storage > VolumeSnapshots, click the ellipsis next to the Snapshot that you wish to restore, and click Restore as new PVC.
2. Enter the details of the new PVC and click Restore. This creates a new PVC.



3. Next, create a new VM from this PVC. Navigate to Workloads > Virtualization > Virtual Machines and click Create > With YAML.
4. In the spec > template > spec > volumes section, specify the new PVC created from Snapshot instead of

from the container disk. Provide all other details for the new VM according to your requirements.

```
- name: rootdisk
  persistentVolumeClaim:
    claimName: rhel8-short-frog-rootdisk-28dvb-snapshot-restore
```

5. Click Create to create the new VM.
6. After the VM is created successfully, access and verify that the new VM has the same state as that of the VM whose PVC was used to create the snapshot at the time when the snapshot was created.

Advanced Cluster Management for Kubernetes on Red Hat OpenShift with NetApp

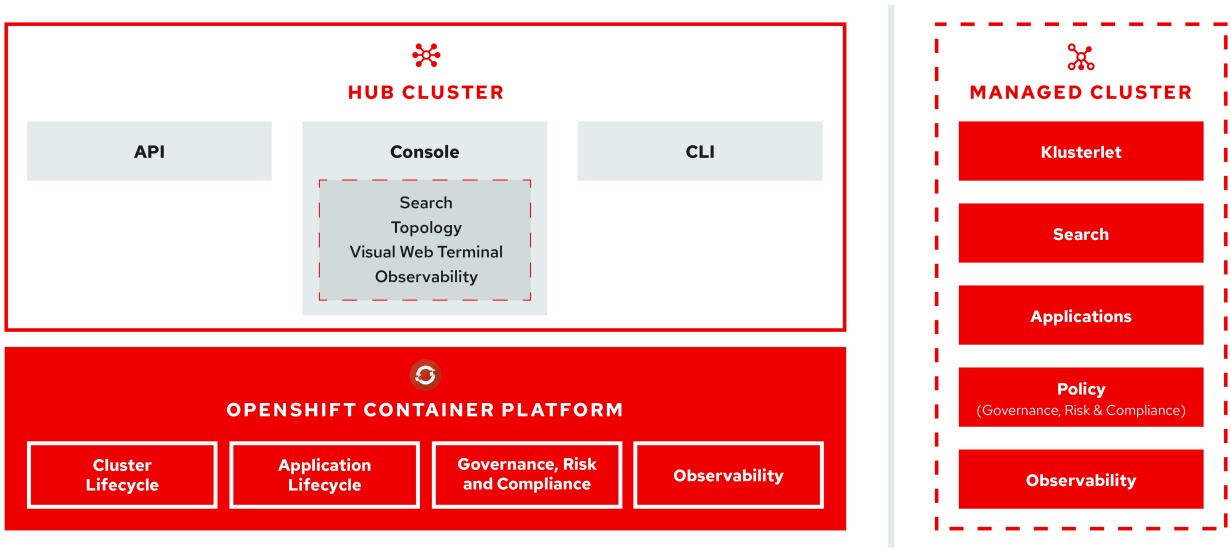
Advanced Cluster Management for Kubernetes: Red Hat OpenShift with NetApp

As a containerized application transitions from development to production, many organizations require multiple Red Hat OpenShift clusters to support the testing and deployment of that application. In conjunction with this, organizations usually host multiple applications or workloads on OpenShift clusters. Therefore, each organization ends up managing a set of clusters, and OpenShift administrators must thus face the added challenge of managing and maintaining multiple clusters across a range of environments that span multiple on-premises data centers and public clouds. To address these challenges, Red Hat introduced Advanced Cluster Management for Kubernetes.

Red Hat Advanced Cluster Management for Kubernetes enables you to perform the following tasks:

1. Create, import, and manage multiple clusters across data centers and public clouds
2. Deploy and manage applications or workloads on multiple clusters from a single console
3. Monitor and analyze health and status of different cluster resources
4. Monitor and enforce security compliance across multiple clusters

Red Hat Advanced Cluster Management for Kubernetes is installed as an add-on to a Red Hat OpenShift cluster, and it uses this cluster as a central controller for all its operations. This cluster is known as hub cluster, and it exposes a management plane for the users to connect to Advanced Cluster Management. All the other OpenShift clusters that are either imported or created via the Advanced Cluster Management console are managed by the hub cluster and are called managed clusters. It installs an agent called Klusterlet on the managed clusters to connect them to the hub cluster and serve the requests for different activities related to cluster lifecycle management, application lifecycle management, observability, and security compliance.



For more information, see the documentation here.

Next: Deployment Prerequisites.

Deployment

Deploy Advanced Cluster Management for Kubernetes

Prerequisites

1. A Red Hat OpenShift cluster (greater than version 4.5) for the hub cluster
2. Red Hat OpenShift clusters (greater than version 4.4.3) for managed clusters
3. Cluster-admin access to the Red Hat OpenShift cluster
4. A Red Hat subscription for Advanced Cluster Management for Kubernetes

Advanced Cluster Management is an add-on on for the OpenShift cluster, so there are certain requirements and restrictions on the hardware resources based on the features used across the hub and managed clusters. You need to take these issues into account when sizing the clusters. See the documentation here for more details.

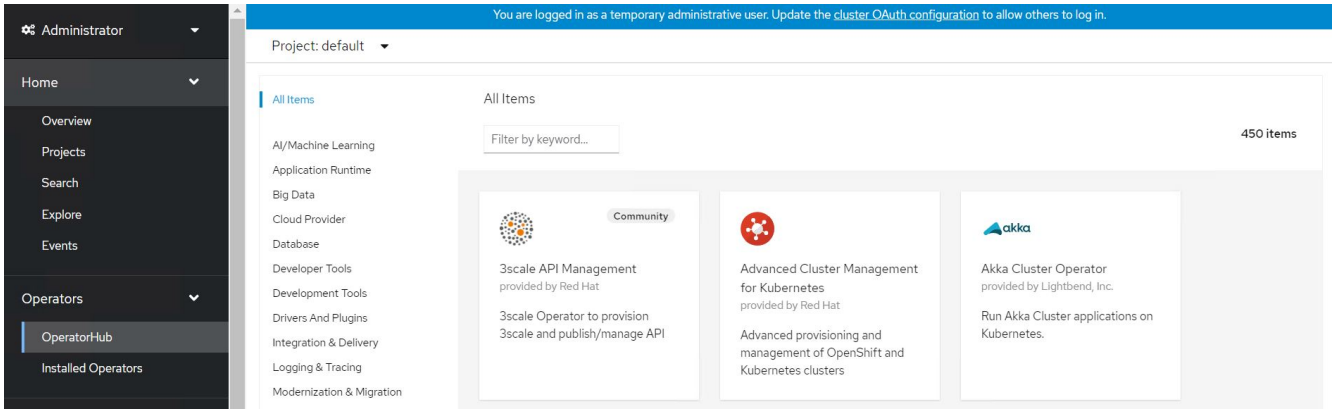
Optionally, if the hub cluster has dedicated nodes for hosting infrastructure components and you would like to install Advanced Cluster Management resources only on those nodes, you need to add tolerations and selectors to those nodes accordingly. For more details, see the documentation here.

Next: Installation.

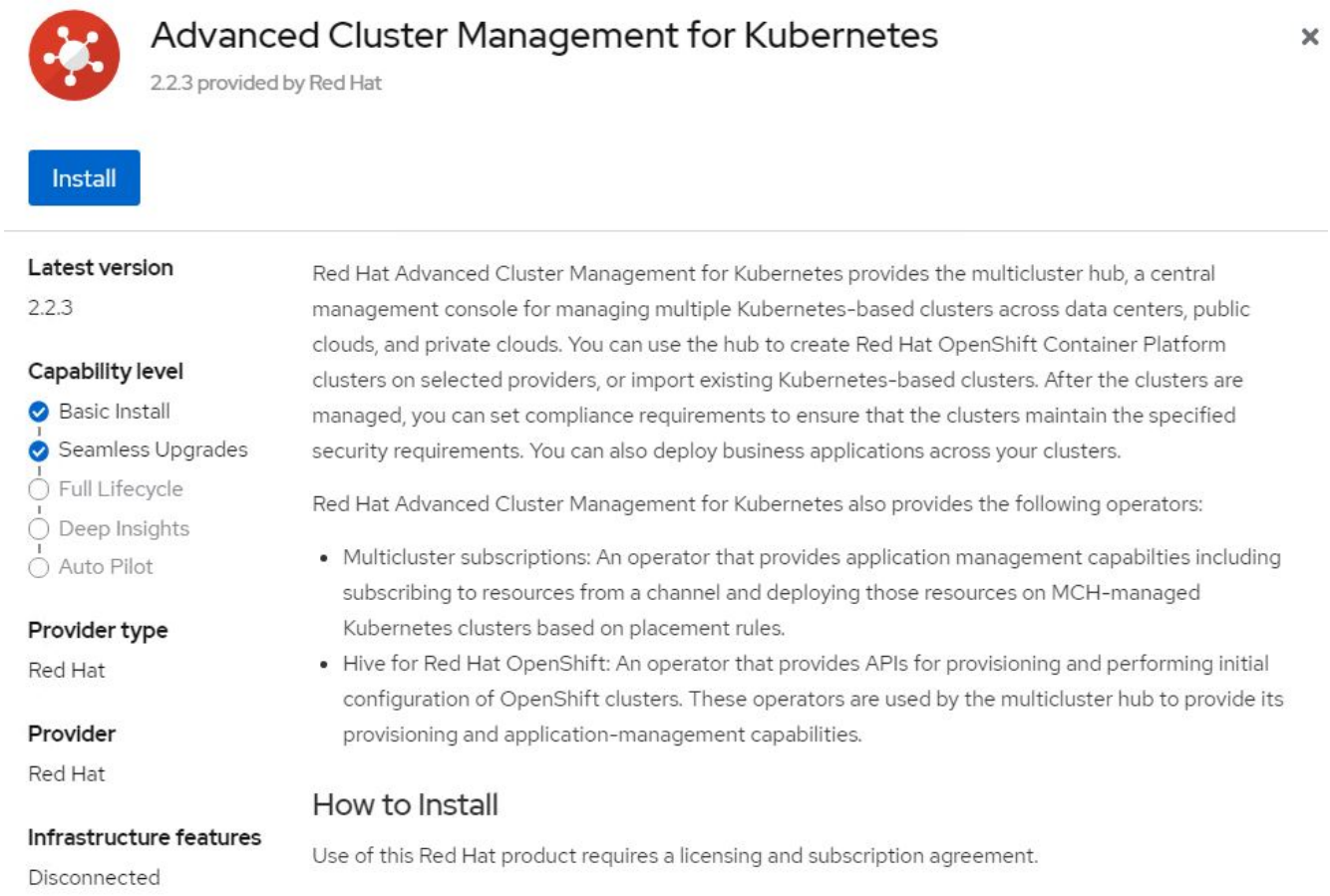
Deploy Advanced Cluster Management for Kubernetes

To install Advanced Cluster Management for Kubernetes on an OpenShift cluster, complete the following steps:

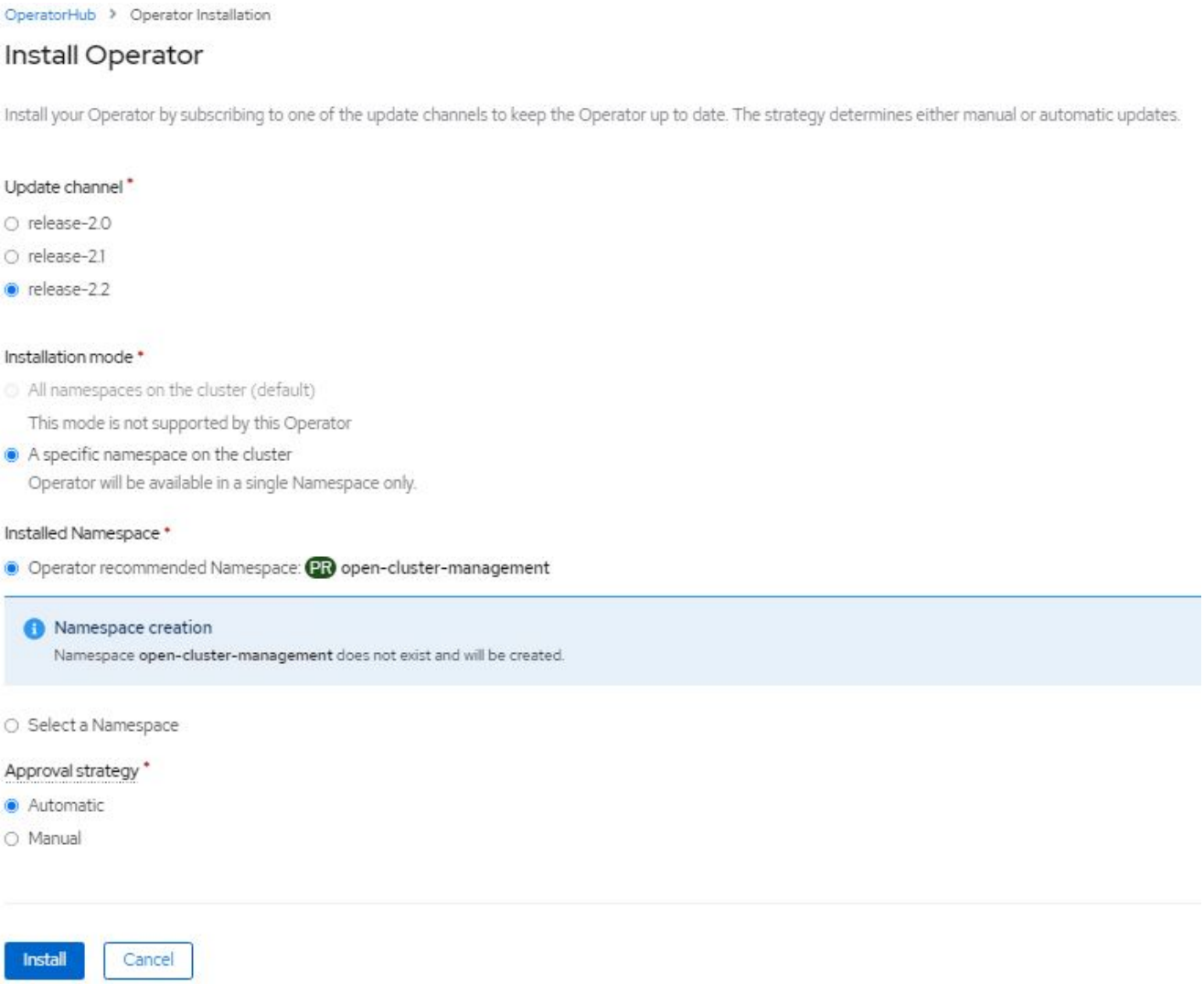
1. Choose an OpenShift cluster as the hub cluster and log into it with cluster-admin privileges.
2. Navigate to Operators > Operators Hub and search for Advanced Cluster Management for Kubernetes.



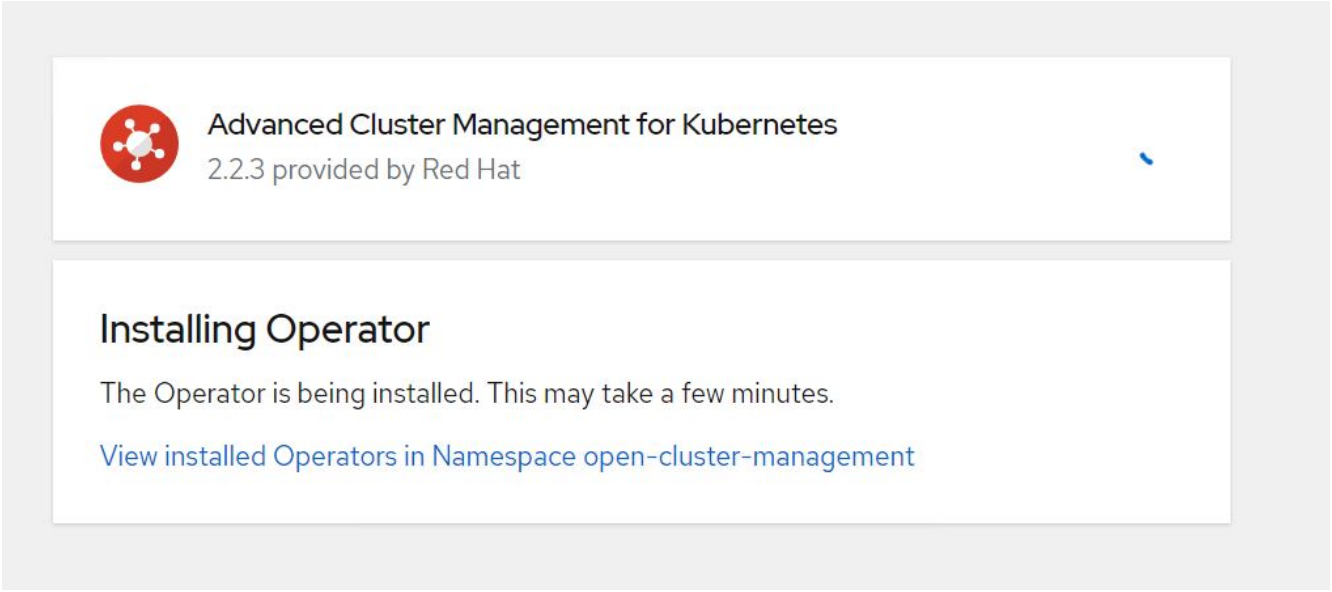
3. Select Advanced Cluster Management for Kubernetes and click Install.



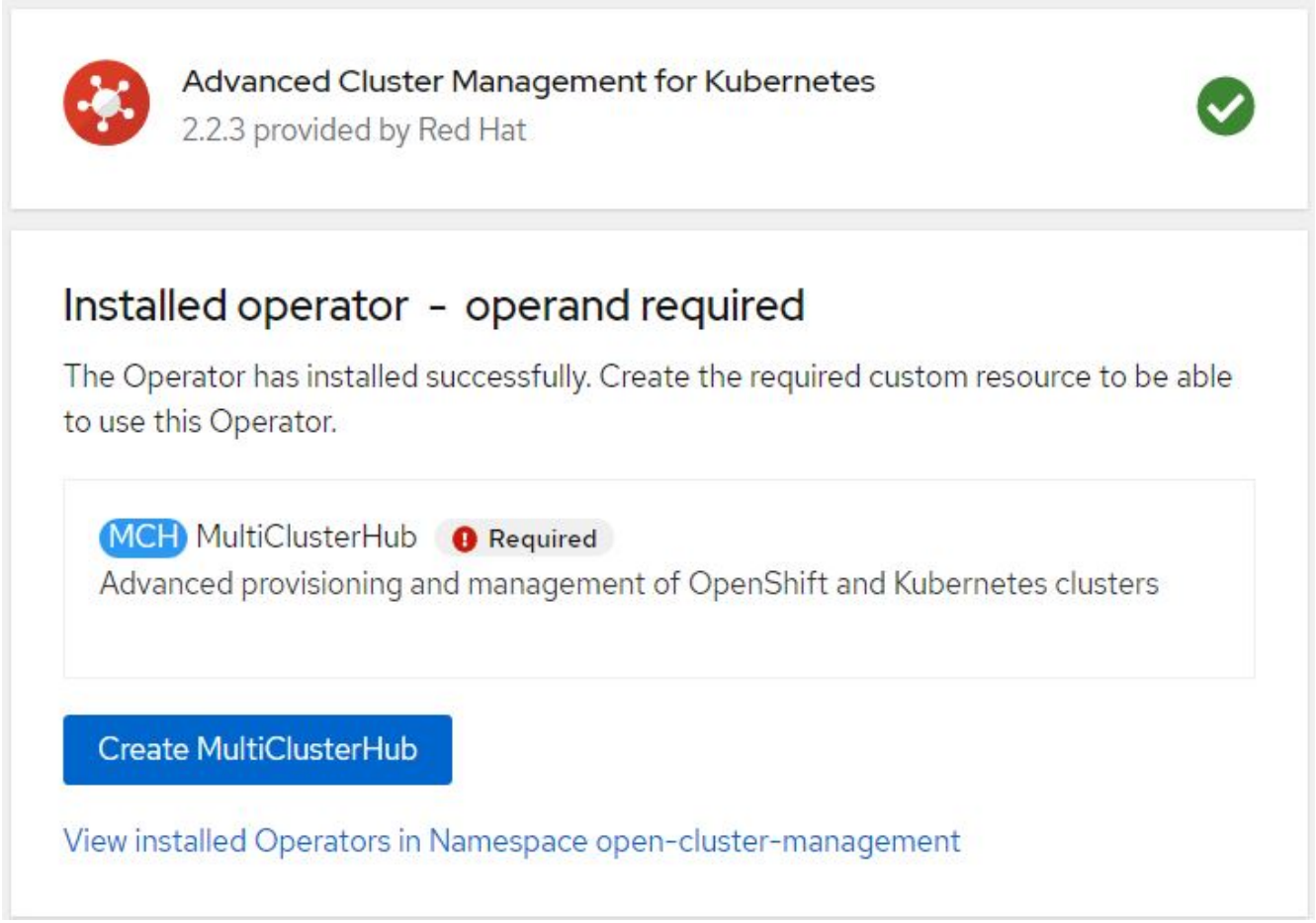
4. On the Install Operator screen, provide the necessary details (NetApp recommends retaining the default parameters) and click Install.



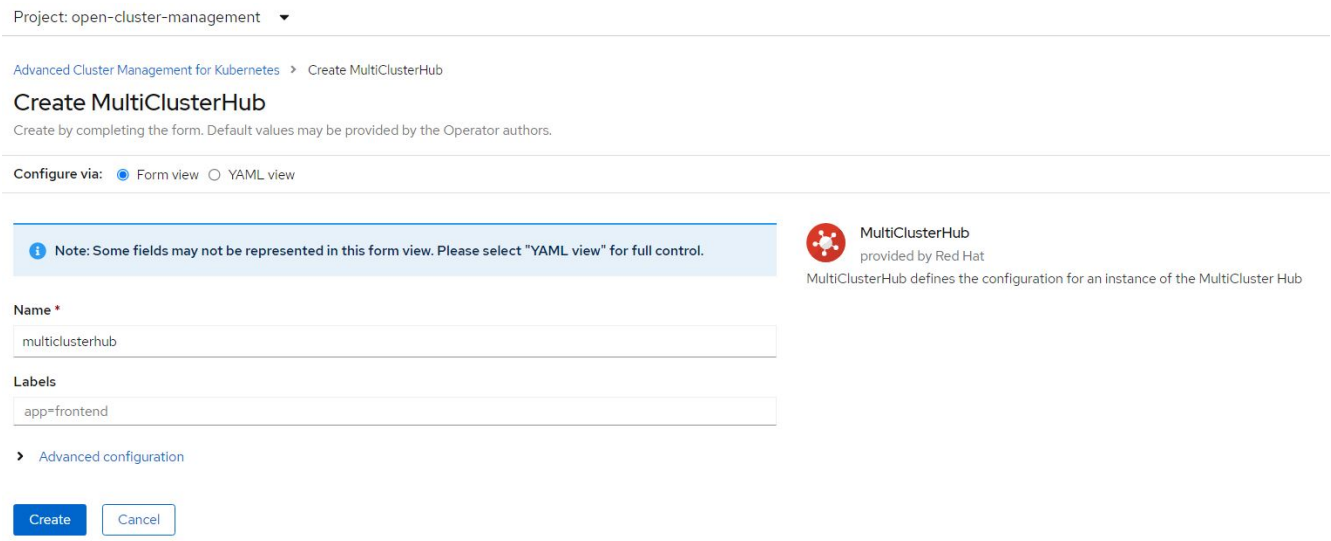
5. Wait for the operator installation to complete.



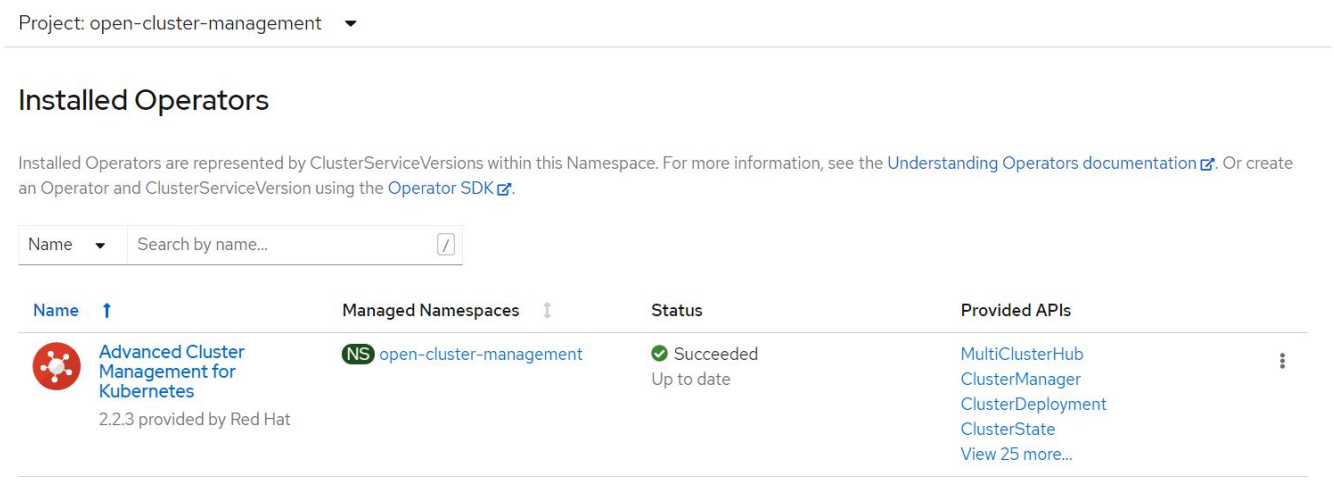
6. After the operator is installed, click Create MultiClusterHub.



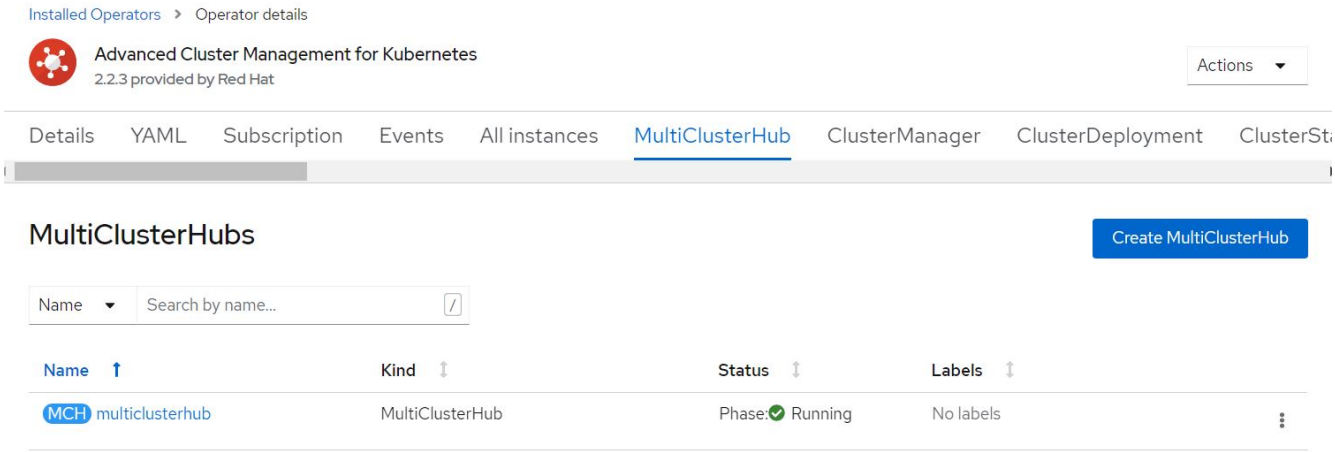
7. On the Create MultiClusterHub screen, click Create after furnishing the details. This initiates the installation of a multi-cluster hub.



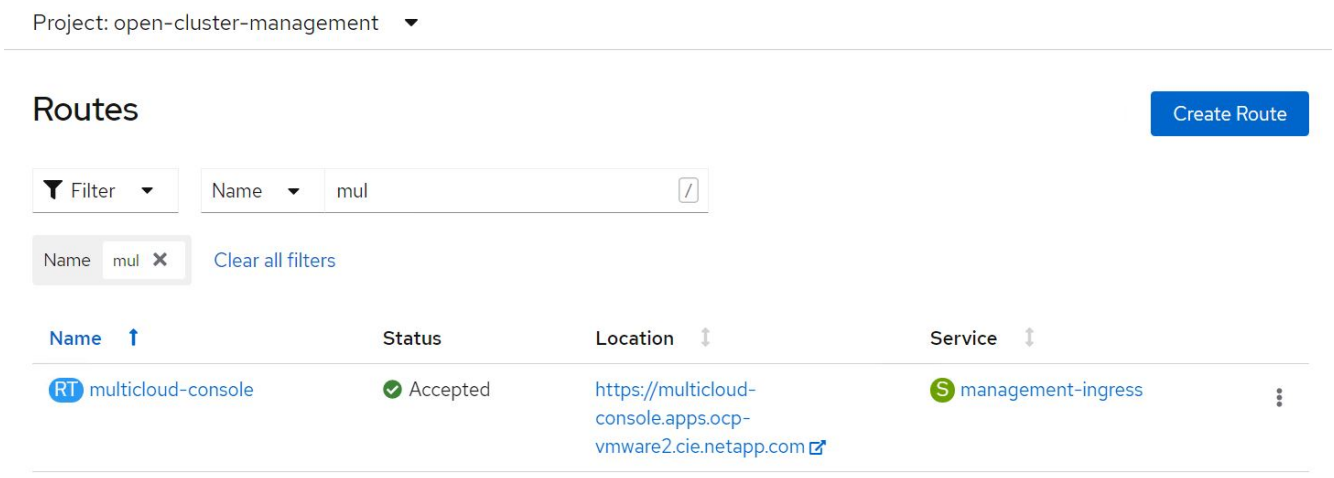
8. After all the pods move to the Running state in the open-cluster-management namespace and the operator moves to the Succeeded state, Advanced Cluster Management for Kubernetes is installed.



9. It takes some time to complete the hub installation, and, after it is done, the MultiCluster hub moves to Running state.



10. It creates a route in the open-cluster-management namespace. Connect to the URL in the route to access the Advanced Cluster Management console.



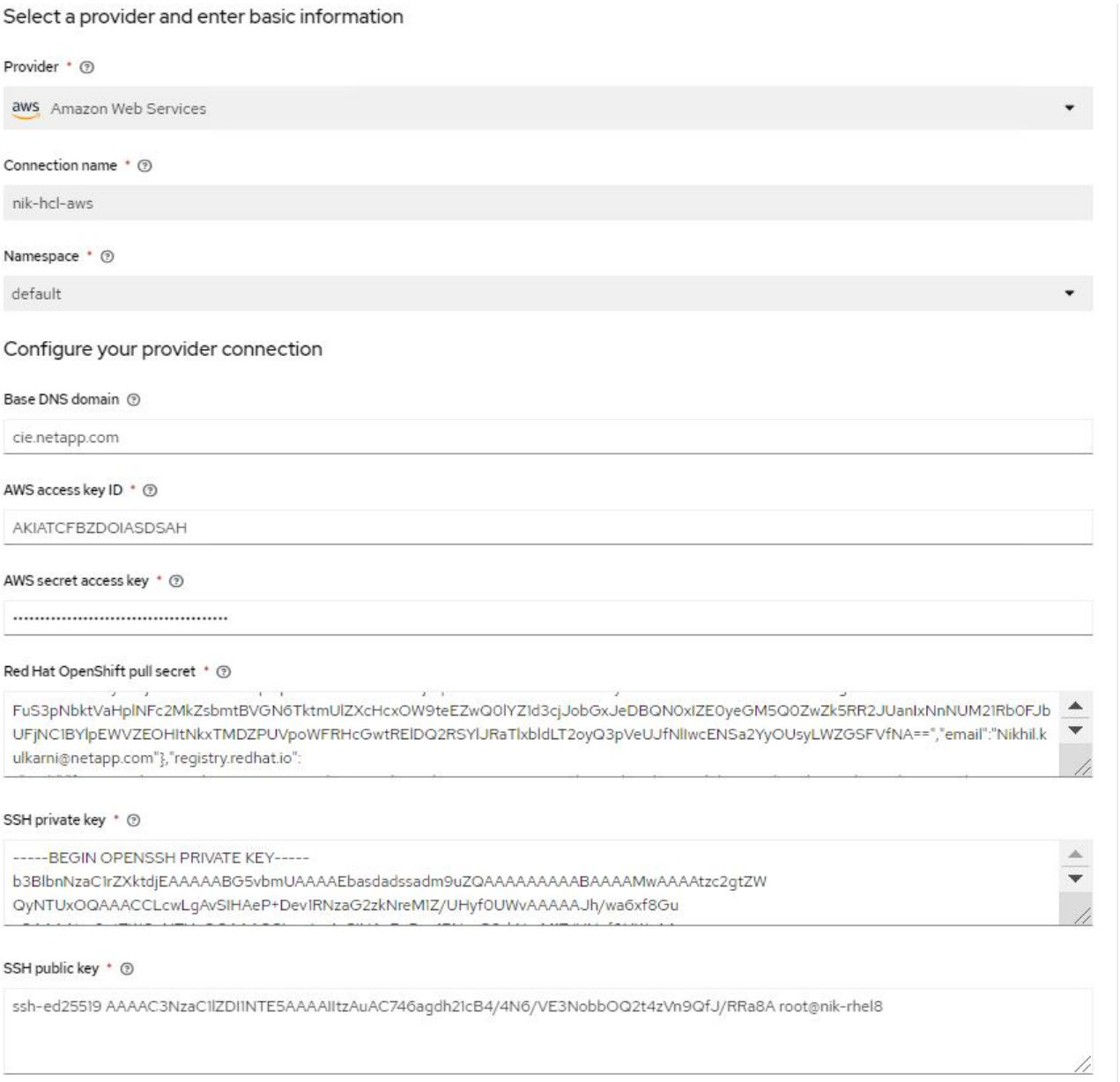
Next: Features - Cluster Lifecycle Management.

Features: Advanced Cluster Management for Kubernetes on Red Hat OpenShift with NetApp

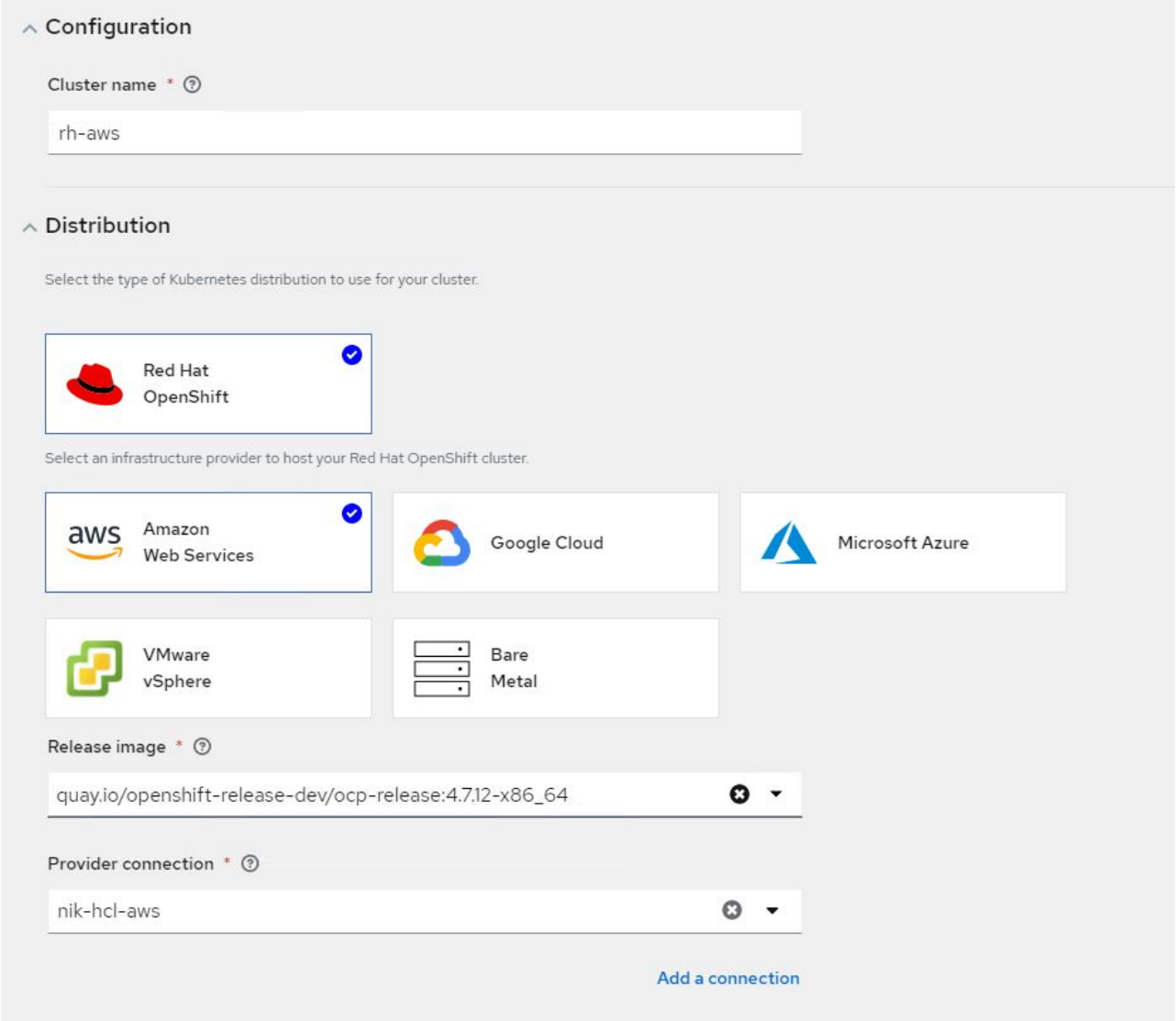
Cluster Lifecycle Management

To manage different OpenShift clusters, you can either create or import them into Advanced Cluster Management.

1. First navigate to Automate Infrastructures > Clusters.
2. To create a new OpenShift cluster, complete the following steps:
   a. Create a provider connection: Navigate to Provider Connections and click Add a Connection, provide all the details corresponding to the selected provider type and click Add.



   b. To create a new cluster, navigate to Clusters and click Add a Cluster > Create a Cluster. Provide the details for the cluster and the corresponding provider and click Create.



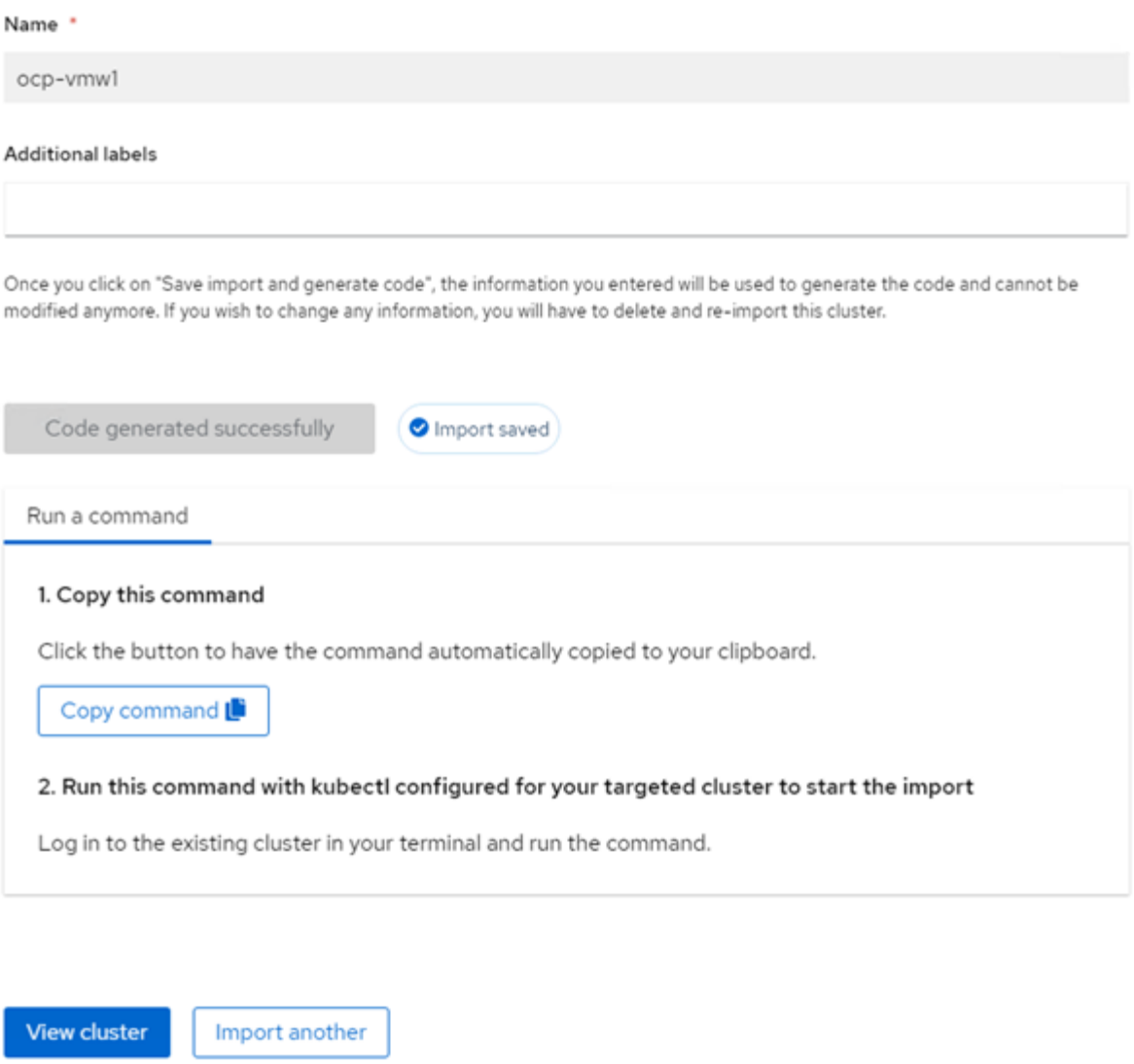
c. After the cluster is created, it appears in the cluster list with the status Ready.

3. To import an existing cluster, complete the following steps:

   a. Navigate to Clusters and click Add a Cluster > Import an Existing Cluster.

   b. Enter the name of the cluster and click Save Import and Generate Code. A command to add the existing cluster is displayed.

   c. Click Copy Command and run the command on the cluster to be added to the hub cluster. This initiates the installation of the necessary agents on the cluster, and, after this process is complete, the cluster appears in the cluster list with status Ready.



4. After you create and import multiple clusters, you can monitor and manage them from a single console.

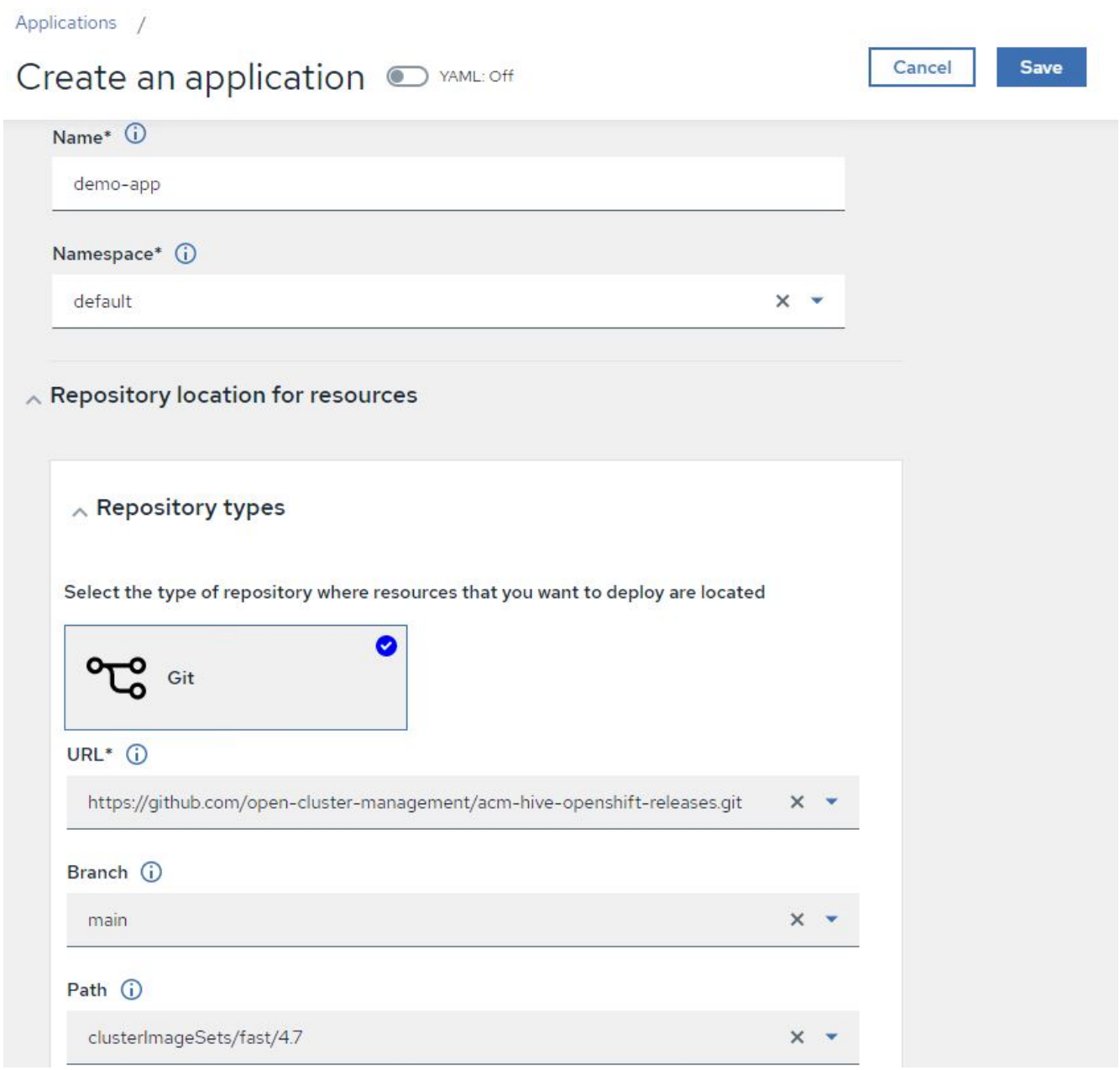
Next: Features - Application Lifecycle Management.

Features: Advanced Cluster Management for Kubernetes on Red Hat OpenShift with NetApp

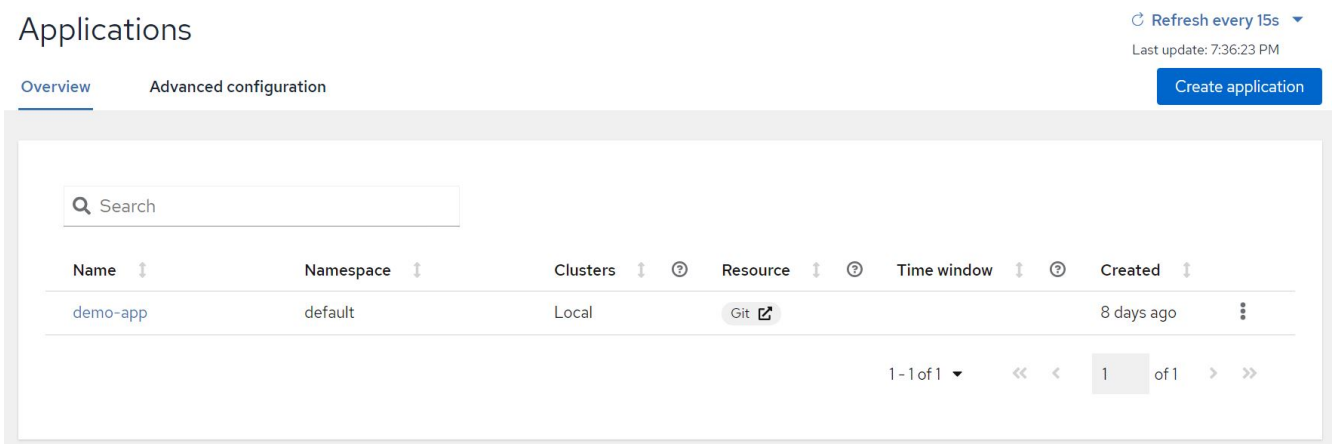
Application lifecycle management

To create an application and manage it across a set of clusters,

1. Navigate to Manage Applications from the sidebar and click Create Application. Provide the details of the application you would like to create and click Save.



2. After the application components are installed, the application appears in the list.



3. The application can now be monitored and managed from the console.

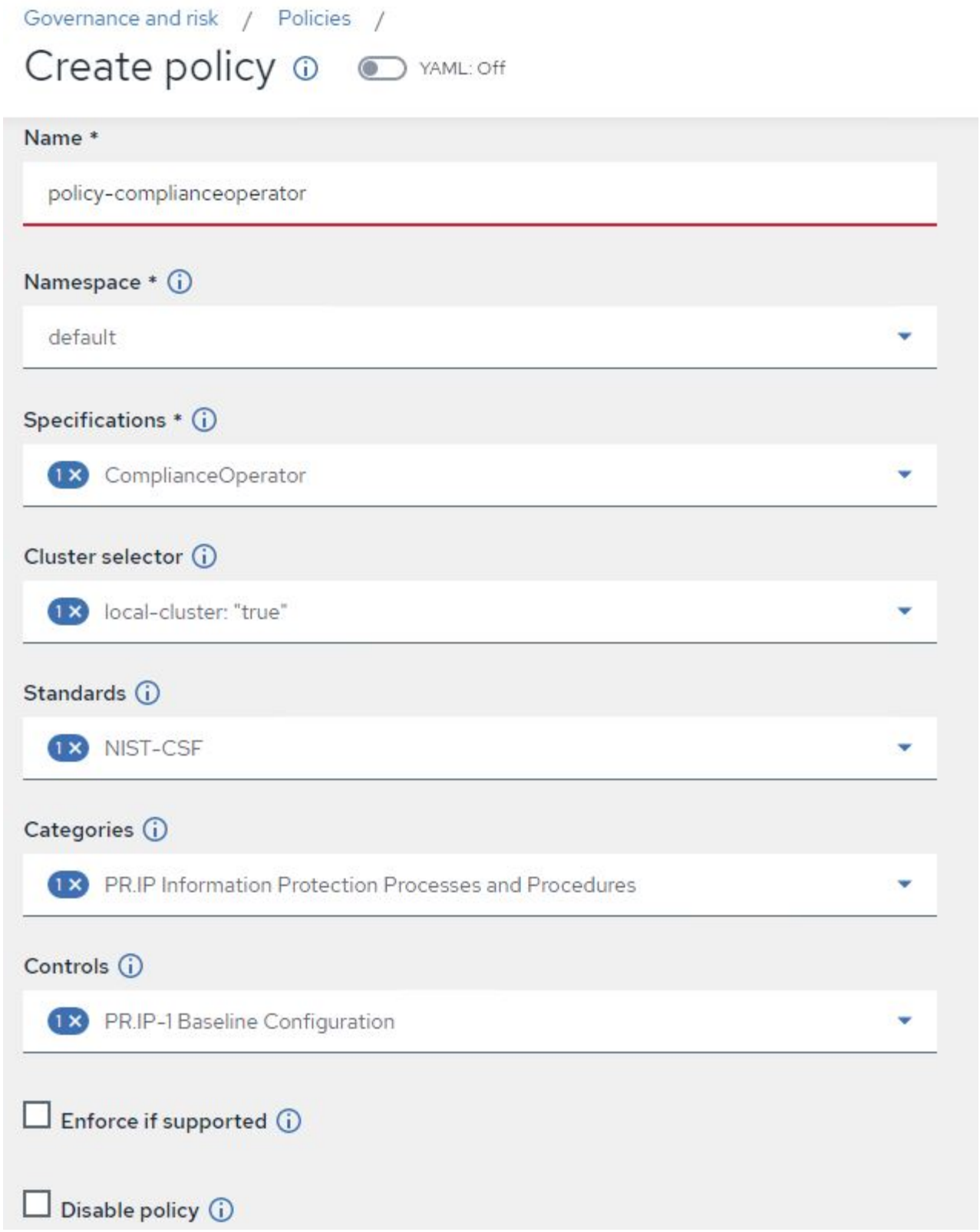
Next: Features - governance and risk.

Features: Advanced Cluster Management for Kubernetes on Red Hat OpenShift with NetApp

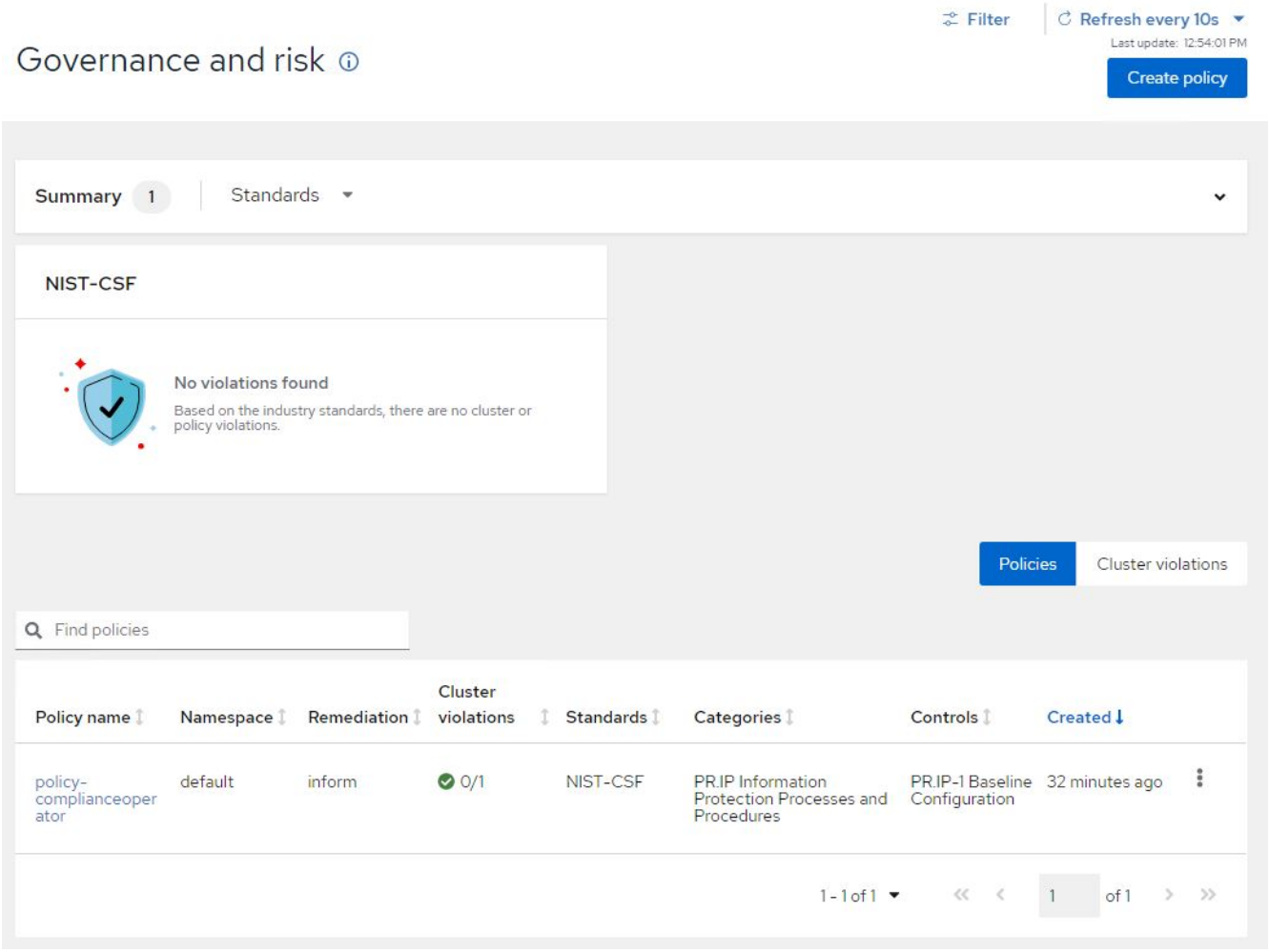
Governance and risk

This feature allows you to define the compliance policies for different clusters and make sure that the clusters adhere to it. You can configure the policies to either inform or remediate any deviations or violations of the rules.

1. Navigate to Governance and Risk from the sidebar.
2. To create compliance policies, click Create Policy, enter the details of the policy standards, and select the clusters that should adhere to this policy. If you want to automatically remediate the violations of this policy, select the checkbox Enforce if Supported and click Create.



3. After all the required policies are configured, any policy or cluster violations can be monitored and remediated from Advanced Cluster Management.



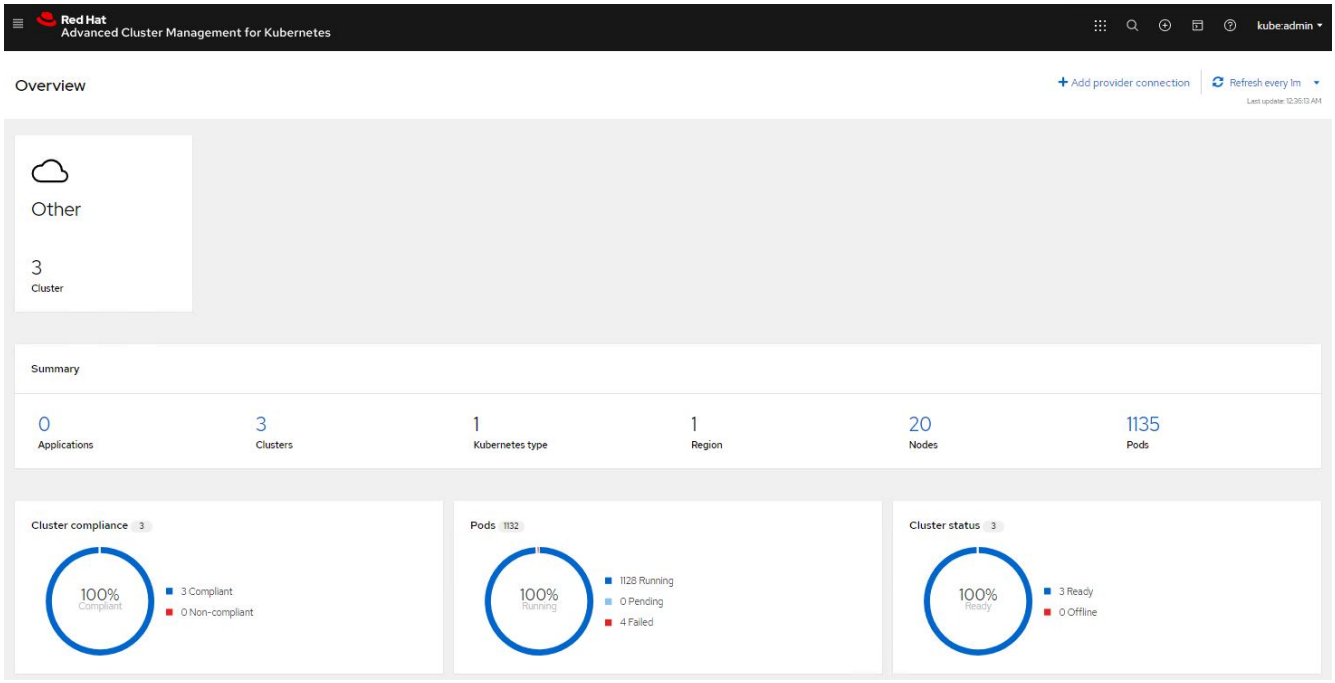
Next: Features - Observability.

Features: Advanced Cluster Management for Kubernetes on Red Hat OpenShift with NetApp

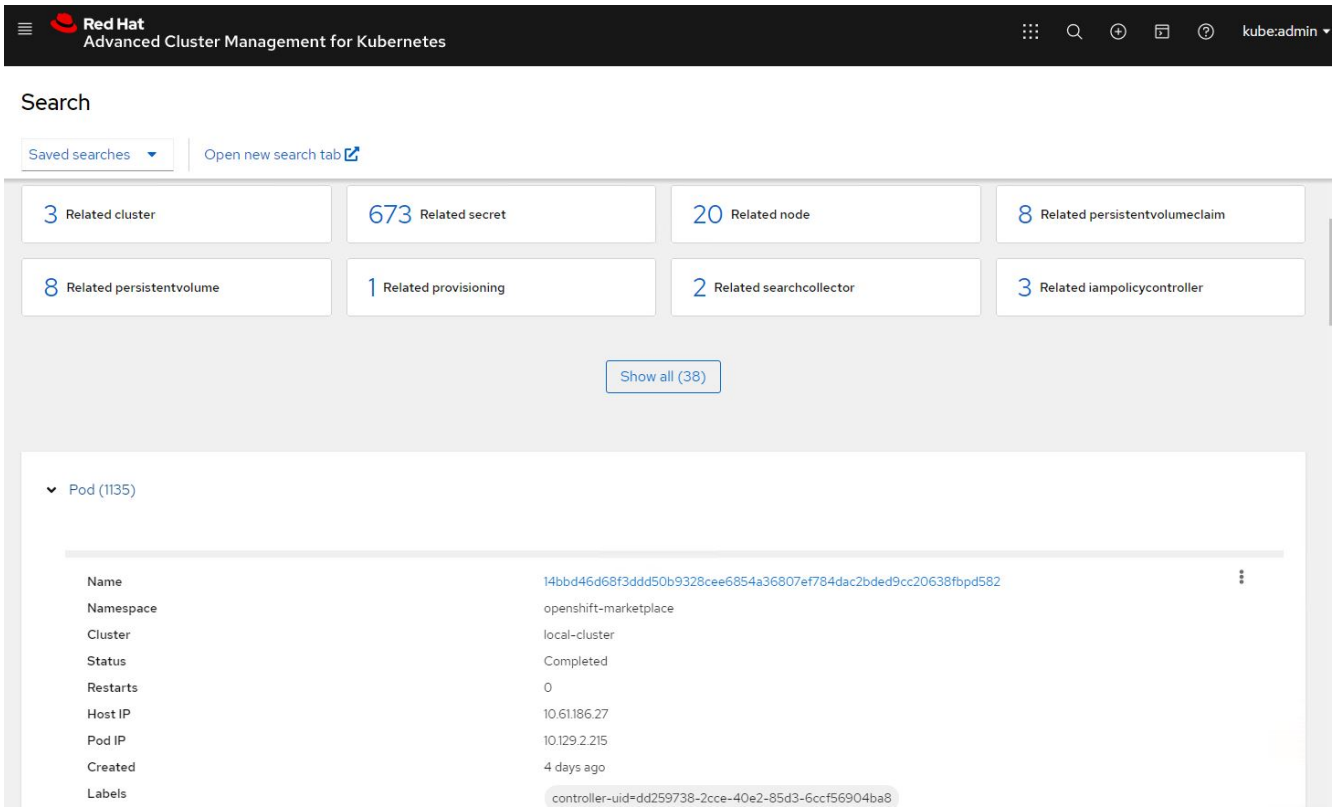
Observability

Advanced Cluster Management for Kubernetes provides a way to monitor the nodes, pods, and applications, and workloads across all the clusters.

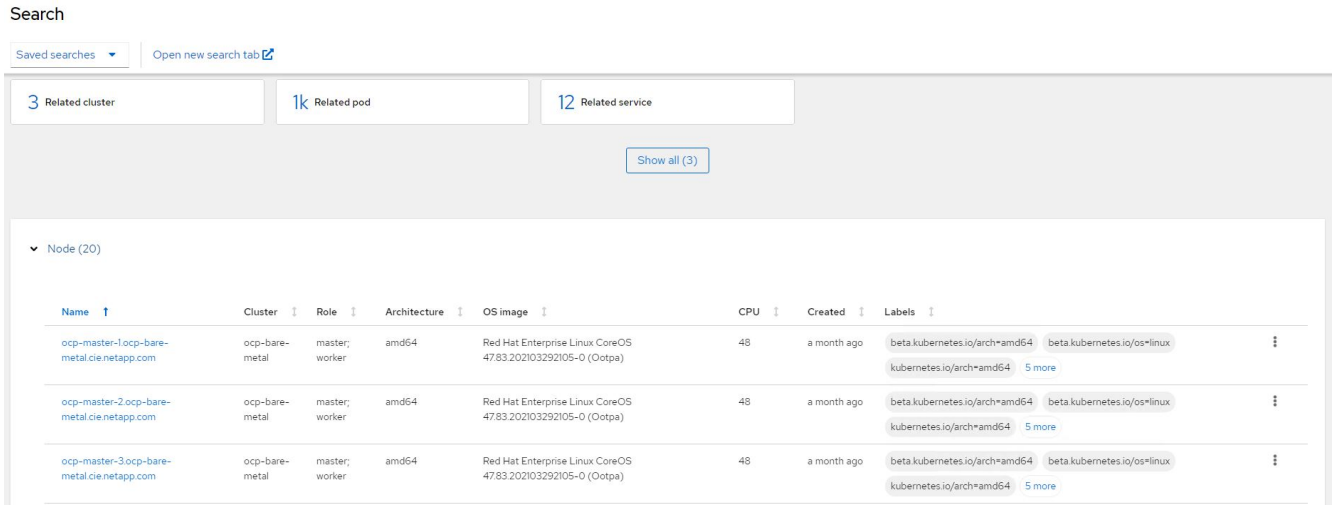
1. Navigate to Observe Environments > Overview.



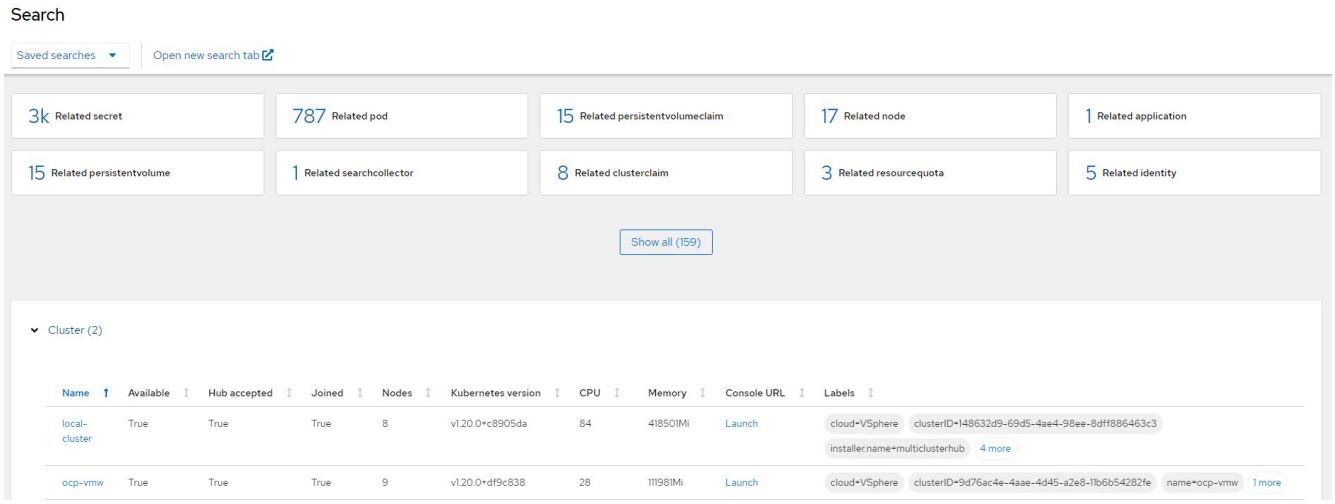
2. All pods and workloads across all clusters are monitored and sorted based on a variety of filters. Click Pods to view the corresponding data.



3. All nodes across the clusters are monitored and analyzed based on a variety of data points. Click Nodes to get more insight into the corresponding details.



4. All clusters are monitored and organized based on different cluster resources and parameters. Click Clusters to view cluster details.

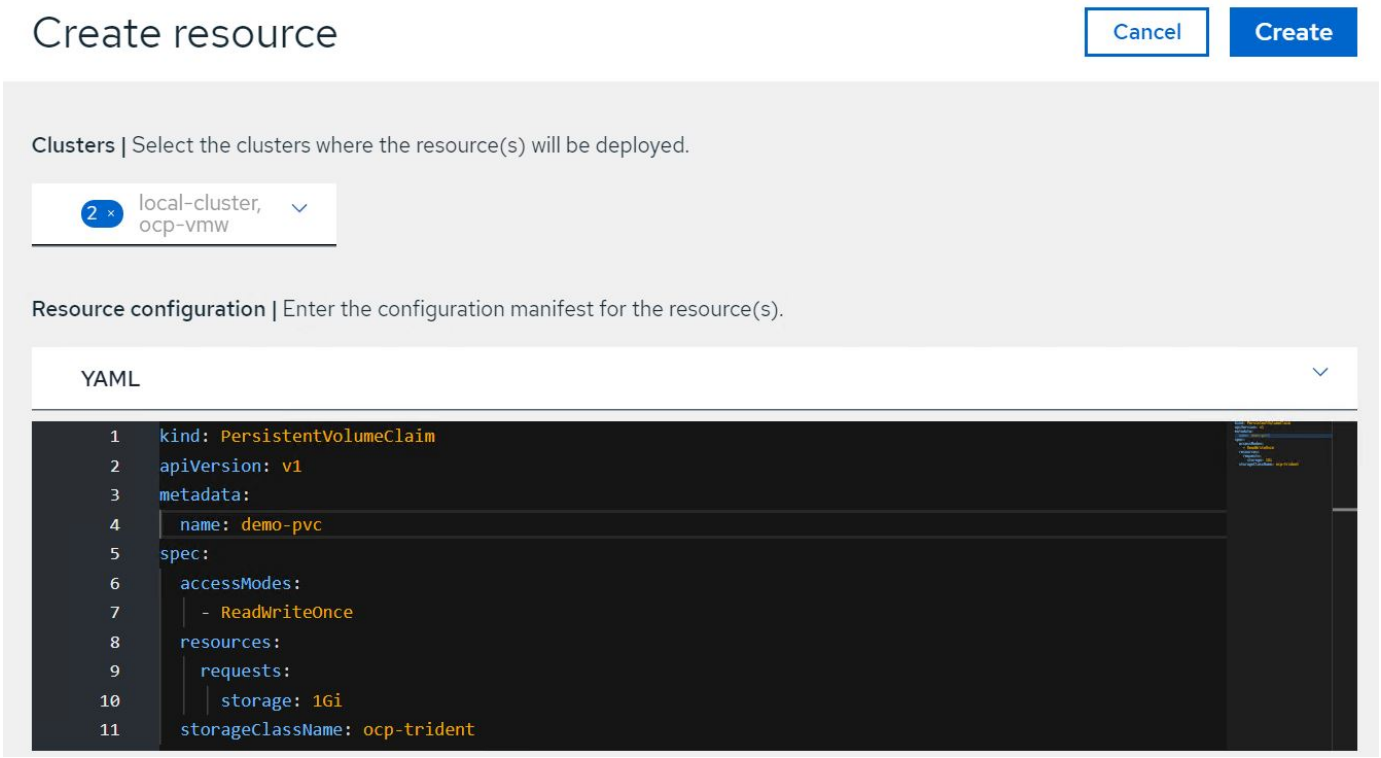


Next: Features - Create Resources.

Features: Advanced Cluster Management for Kubernetes on Red Hat OpenShift with NetApp

Create resources on multiple clusters

Advanced Cluster Management for Kubernetes allows users to create resources on one or more managed clusters simultaneously from the console. As an example, if you have OpenShift clusters at different sites backed with different NetApp ONTAP clusters and want to provision PVC's at both sites, you can click the (+) sign on the top bar. Then select the clusters on which you want to create the PVC, paste the resource YAML, and click Create.



Videos and Demos: Red Hat OpenShift with NetApp

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

- Video: Workload Migration using Astra Control Center - Red Hat OpenShift with NetApp
- Video: Workload Migration using Astra Trident and SnapMirror - Red Hat OpenShift with NetApp
- Video: Installing OpenShift Virtualization - Red Hat OpenShift with NetApp
- Video: Deploying a Virtual Machine with OpenShift Virtualization - Red Hat OpenShift with NetApp
- Video: NetApp HCI for Red Hat OpenShift on Red Hat Virtualization Deployment

Next: Additional Information: Red Hat OpenShift with NetApp.

Additional Information: Red Hat OpenShift with NetApp

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

- NetApp Documentation

  https://docs.netapp.com/

- Astra Trident Documentation

  https://netapp-trident.readthedocs.io/en/stable-v21.07/

- NetApp Astra Control Center Documentation

  https://docs.netapp.com/us-en/astra-control-center/

- Red Hat OpenShift Documentation

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.7/

- Red Hat OpenStack Platform Documentation

  https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/16.1/

- Red Hat Virtualization Documentation

  https://access.redhat.com/documentation/en-us/red_hat_virtualization/4.4/

- VMware vSphere Documentation

  https://docs.vmware.com/

Google Anthos

WP-7337: Anthos on Bare Metal

Alan Cowles and Nikhil M Kulkarni, NetApp

NetApp and Google Cloud have had a strong relationship for several years now, with NetApp first introducing cloud data services for Google Cloud with Cloud Volumes ONTAP and the Cloud Volumes Service. This relationship was then expanded by validating the NetApp HCI platform for use with Google Cloud Anthos on-premises, a hypervisor-based hybrid multi-cloud Kubernetes solution deployed on VMware vSphere. NetApp then passed Anthos Ready qualification for NetApp Trident, ONTAP, and the NFS protocol to provide dynamic persistent storage for containers.

Anthos can now be directly install on bare metal servers in a customer's environment, which adds an additional option for customers to extend Google Cloud into their local data centers without a hypervisor. Additionally, by leveraging the capabilities of NetApp ONTAP storage operating system and NetApp Trident, you can extend your platform's capabilities by integrating persistent storage for containers.

This combination allows you to realize the full potential of your servers, storage, and networking combined with the support, service levels, monthly billing, and on-demand flexibility that Google Cloud provides. Because you are using your own hardware, network, and storage, you have direct control over application scale, security, and network latency, as well as having the benefit of managed and containerized applications with Anthos on bare metal.

Next: Solution overview.

Solution overview

NetApp ONTAP on NetApp AFF/FAS

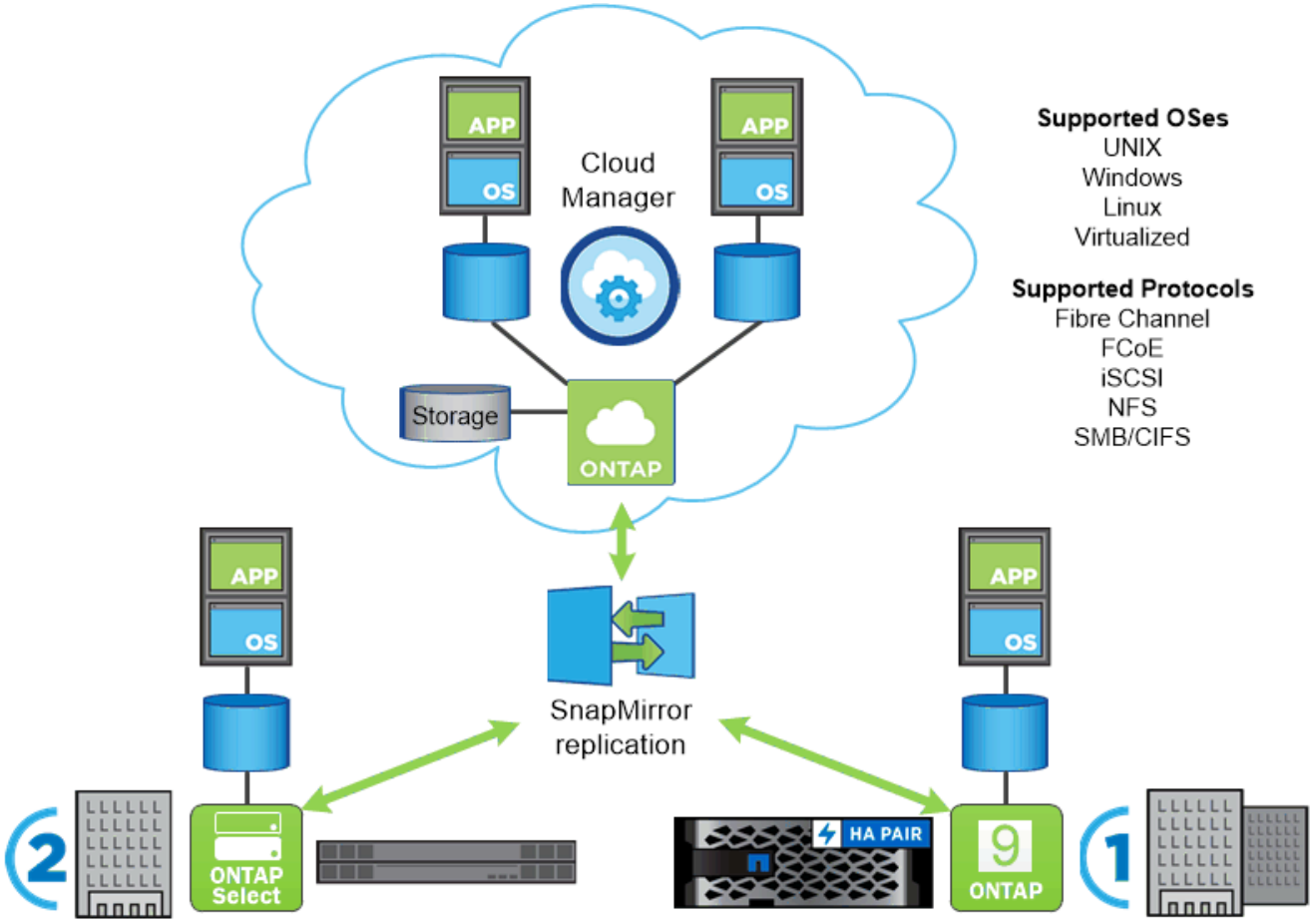
NetApp AFF is a robust all-flash storage platform that provides low-latency performance, integrated data protection, multiprotocol support, and nondisruptive operations. Powered by NetApp ONTAP data management software, NetApp AFF ensures nondisruptive operations, from maintenance to upgrades to complete replacement of your storage system.

NetApp ONTAP is a powerful storage-software tool with capabilities such as an intuitive GUI, REST APIs with automation integration, AI-informed predictive analytics and corrective action, nondisruptive hardware upgrades, and cross-storage import.

ONTAP provides the following features:

- A unified storage system with simultaneous data access and management of NFS, CIFS, iSCSI, FC, FCoE, and FC-NVMe protocols.
- Different deployment models include on-premises on all-flash, hybrid, and all-HDD hardware configurations; VM-based storage platforms on a supported hypervisor such as ONTAP Select; and in the cloud as Cloud Volumes ONTAP.
- Increased data storage efficiency on ONTAP systems with support for automatic data tiering, inline data compression, deduplication, and compaction.
- Workload-based, QoS-controlled storage.
- Seamless integration with a public cloud for tiering and protection of data. ONTAP also provides robust data protection capabilities that sets it apart in any environment:
  - **NetApp Snapshot copies.** A fast, point-in-time backup of data using a minimal amount of disk space with no additional performance overhead.
  - **NetApp SnapMirror.** Mirrors the Snapshot copies of data from one storage system to another. ONTAP supports mirroring data to other physical platforms and cloud-native services as well.
  - **NetApp SnapLock.** Efficiently administration of non-rewritable data by writing it to special volumes that cannot be overwritten or erased for a designated period.
  - **NetApp SnapVault.** Backs up data from multiple storage systems to a central Snapshot copy that serves as a backup to all designated systems.
  - **NetApp SyncMirror.** Provides real-time, RAID-level mirroring of data to two different plexes of disks that are connected physically to the same controller.
  - **NetApp SnapRestore.** Provides fast restoration of backed-up data on demand from Snapshot copies.
  - **NetApp FlexClone.** Provides instantaneous provisioning of a fully readable and writeable copy of a NetApp volume based on a Snapshot copy. For more information about ONTAP, see the ONTAP 9 Documentation Center.

NetApp ONTAP is available on-premises, virtualized, or in the cloud.



Across the NetApp data fabric, you can count on a common set of features and fast, efficient replication across platforms. You can use the same interface and the same data management tools.

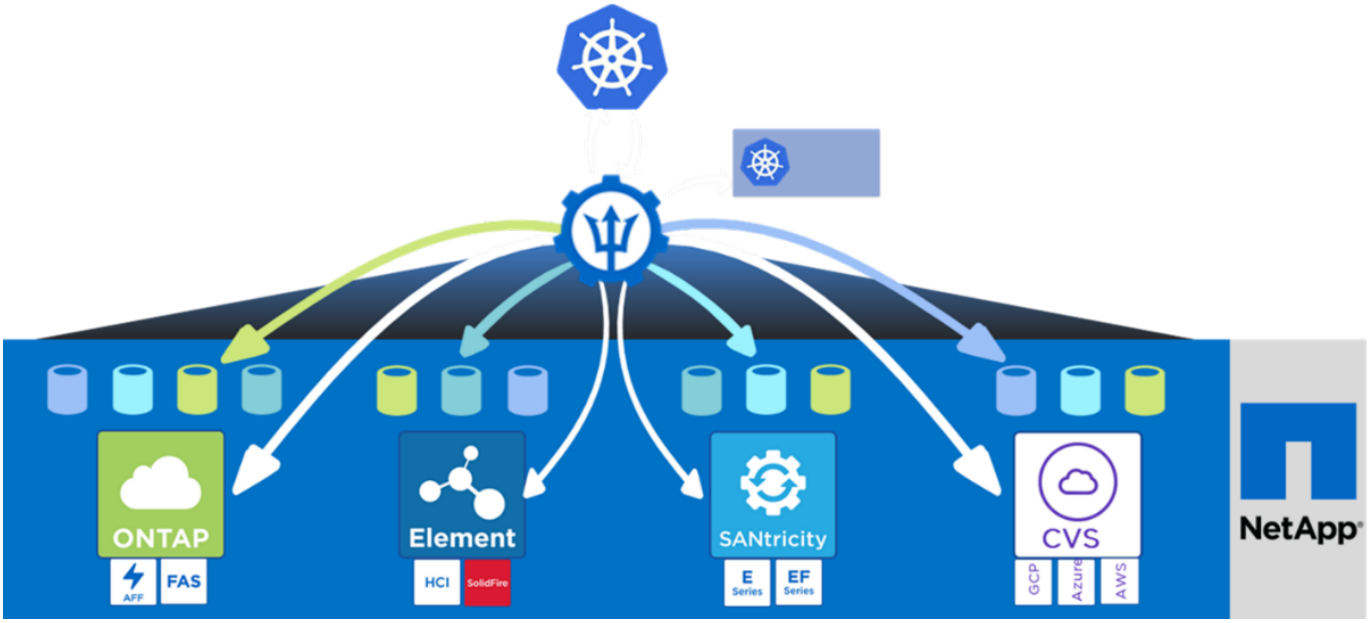
NetApp Trident

NetApp Trident is an open-source and fully supported storage orchestrator for containers and Kubernetes distributions, including Google Cloud Anthos. It works with the entire NetApp storage portfolio, including NetApp ONTAP software. Trident is fully CSI-compliant, and it accelerates the DevOps workflow by allowing you to provision and manage storage from your NetApp storage systems, without intervention from a storage administrator. Trident is deployed as an operator that communicates directly with the Kubernetes API endpoint to serve containers' storage requests in the form of persistent volume claims (PVCs) by creating and managing volumes on the NetApp storage system.

Persistent volumes (PVs) are provisioned based on storage classes defined in the Kubernetes environment. They use storage backends created by a storage administrator (which can be customized based on project needs) and storage system models to allow for any number of advanced storage features, such as compression, specific disk types, or QoS levels that guarantee performance.

For more information about NetApp Trident, see the Trident page.

Trident orchestrates storage from each system and service in the NetApp portfolio.



Google Cloud's Anthos

Google Cloud's Anthos is a cloud-based 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 bare metal extends the capability of Anthos to run on-premises directly on physical servers without a hypervisor layer and interoperate with Anthos GKE clusters in Google Cloud.

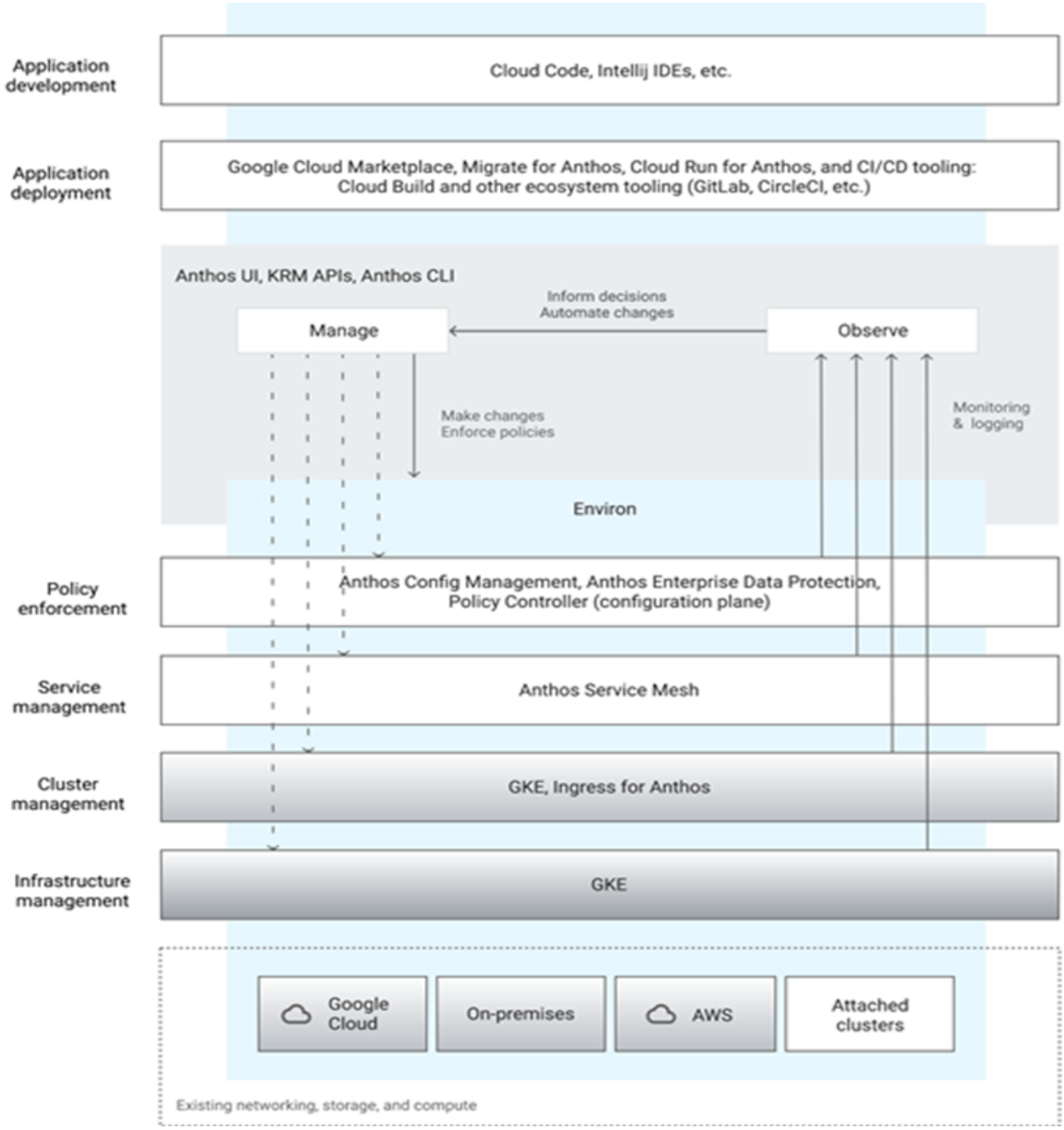
Adopting containers, service mesh, and other transformational technologies enables organizations to experience consistent application development cycles and production-ready workloads in local and cloud-based environments.

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. Figure 3 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.

The following figure presents Google Cloud's Anthos architecture.



Anthos on bare metal

Anthos on bare metal is an extension of GKE that is deployed in a customer's private data center. An organization can deploy the same applications designed to run in containers in Google Cloud in Anthos clusters on-premises. Anthos on bare metal runs directly on physical servers with the user's choice of underlying Linux operating system and provides customers with a full-fledged hybrid cloud environment with the capability to run at the core or edge of their data centers.

Anthos on bare metal offers the following benefits:

- **Hardware agnostic.** Customers can run Anthos on their choice of optimized hardware platform in their existing data centers.
- **Cost savings.** You can realize significant cost savings by using your own physical resources for application deployments instead of provisioning resources in the Google Cloud environment.
- **Develop then publish.** You can use on-premises deployments while applications are in development, which allows for the testing of applications in the privacy of your local data center before you make them publicly available in the cloud.

- **Better performance.** Intensive applications that demand low latency and the highest levels of performance can be run closer to the hardware.
- **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.
- **Management and operations.** Anthos on bare metal comes with a wide range of facilities that increase operational efficiency such as built-in networking, lifecycle management, diagnostics, health checks, logging, and monitoring.

Next: Solution requirements.

Solution requirements

Hardware requirements

Compute: bring your own server

The hardware-agnostic capabilities of Anthos on bare metal allow you to select a compute platform optimized for your use-case. Therefore, your can match your existing infrastructure and reduce capital expenditure.

The following table lists the minimum number of compute hardware components that are required to implement this solution, although the hardware models used can vary based on customer requirements.

Usage	Hardware and model	Quantity
Admin nodes	Cisco UCS B200	3
Worker nodes	HP Proliant DL360	4

Storage: NetApp ONTAP

The following table lists the minimum number of storage hardware components needed to implement the solution, although the hardware models used can vary based on customer requirements.

Hardware	Model	Quantity
NetApp AFF	NetApp AFF A300	2 (1 HA pair)

Software requirements

The software versions identified in the following table were used by NetApp and our partners to validate the solution with NetApp, although the software components used can vary based on customer requirements.

Software	Purpose	Version
Ubuntu	OS on 3 Admins	20.04
	OS on Worker4	20.04
	OS on Worker3	18.04
CentOS	OS on Worker2	8.2
Red Hat Enterprise Linux	OS on Worker1	8.1
Anthos on bare metal	Container Orchestration	1.6.0

Software	Purpose	Version
NetApp ONTAP	Storage OS	9.7P8
NetApp Trident	Container Storage Management	20.10

This multi-OS environment shows the interoperability with supported OS versions of the of Anthos on bare metal solution. We anticipate that customers will standardize on one or a subset of operating systems for deployment.

For Anthos on bare metal hardware and software requirements, see the Anthos on bare metal documentation page.

Next: Deployment summary.

Deployment summary

For the initial validation of this solution, NetApp partnered with World Wide Technology (WWT) to establish an environment at WWT's Advanced Technology Center (ATC). Anthos was deployed on a bare metal infrastructure using the bmctl tool provided by Google Cloud. The following section details the deployment used for validation purposes.

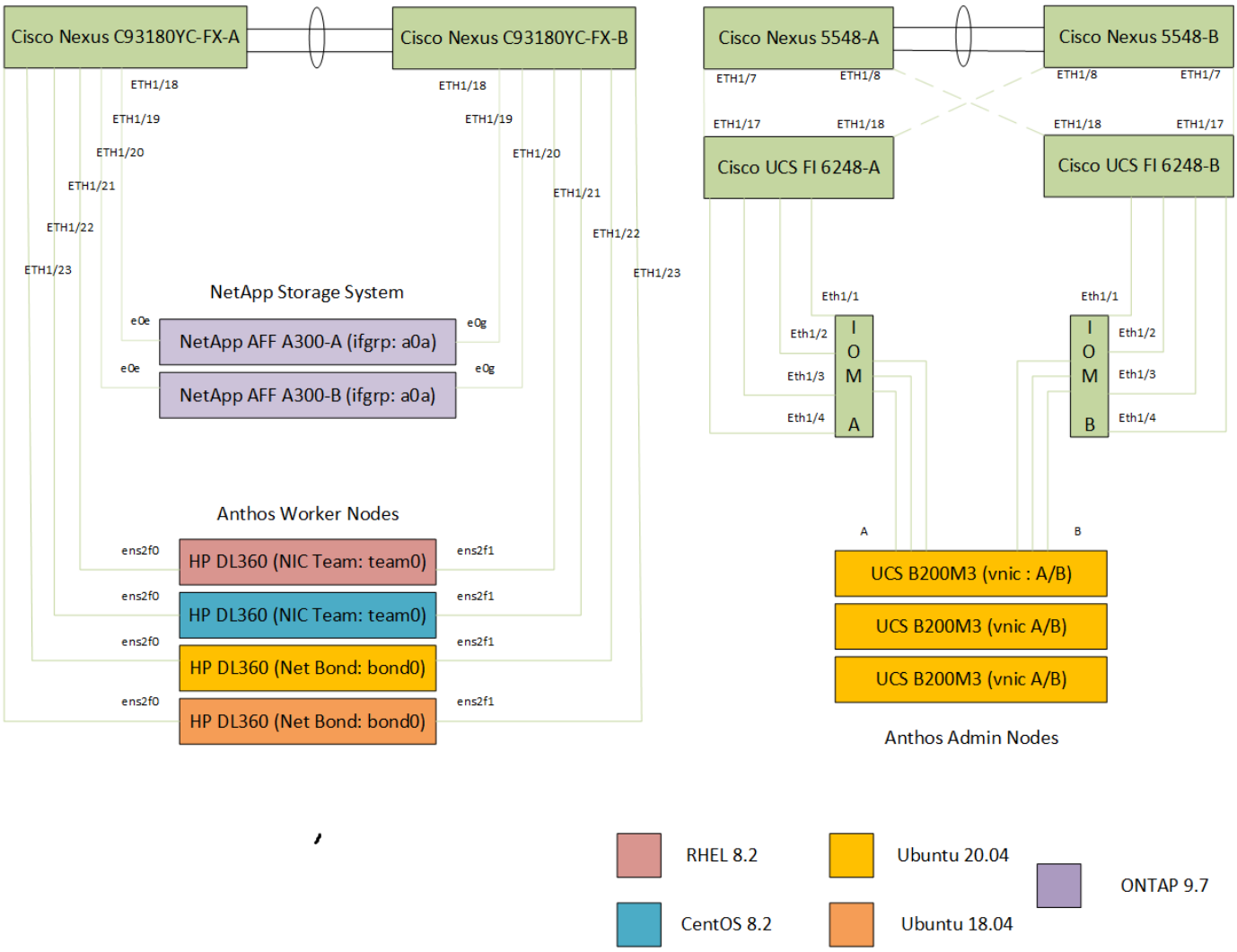
The Anthos on bare metal with NetApp solution was built as a highly available hybrid cluster with three Anthos control-plane nodes and four Anthos worker nodes.

The control-plane nodes used were Cisco UCS B200M3 blade servers hosted in a chassis and configured with a single virtual network interface card (vNIC) on each, which allowed for A/B failover at the Cisco UCS platform level for fault tolerance. The Cisco UCS chassis connected upstream to a pair of Cisco UCS 6248 fabric interconnects providing disparate paths for the separation of traffic along fabric A and fabric B. Those fabric interconnects connected upstream to a pair of Cisco Nexus 5548 data center switches that tied back to the core network at WWT.

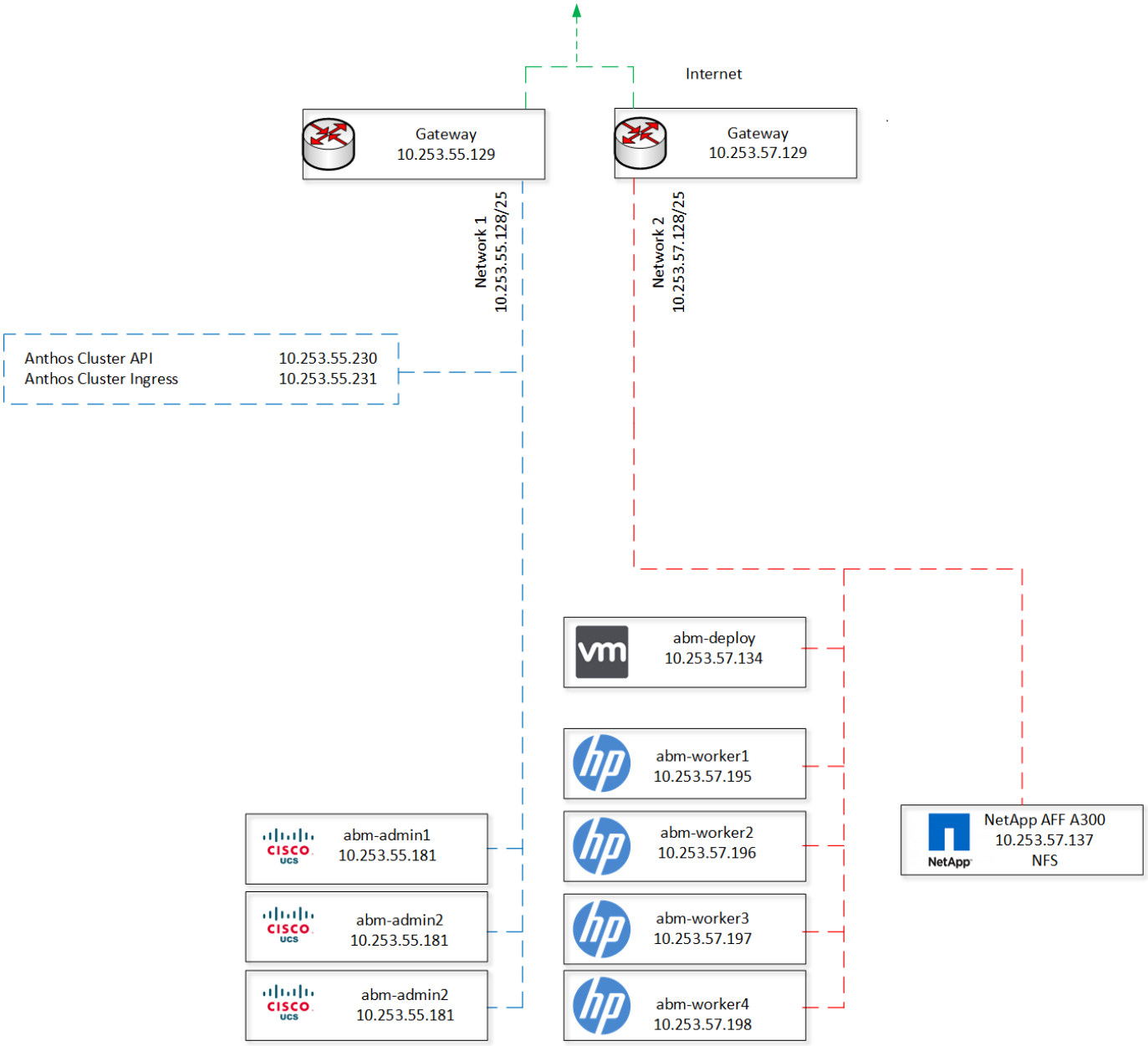
The worker nodes were HP Proliant DL360 nodes, each running one of the supported Linux distributions for Anthos on bare metal: Red Hat Enterprise Linux 8.2, CentOS 8.2, Ubuntu 20.04 LTS, or Ubuntu 18.04 LTS. The Red Hat Enterprise Linux 8 and CentOS 8 nodes were configured with NIC teams running in LACP mode and cabled to two Nexus 9k C93180YC-FX switches for fault tolerance. The Ubuntu servers were configured for network bonding in LACP mode and cabled to the same pair of Nexus 9k switches for fault tolerance.

The NetApp AFF A300 storage system running ONTAP 9.7 software was installed and connected physically to the same pair of Nexus 9k switches as the Anthos worker nodes. These network uplinks were aggregated into an interface group (a0a), and the appropriate data network VLAN was tagged to allow the worker nodes to interact with the storage system. A storage virtual machine (SVM) was created with data LIFs supporting the NFS protocol and dedicated to storage operations for Trident to provide persistent storage to the containers deployed in the Anthos on bare metal cluster. These persistent volumes were provided by NetApp Trident 20.10, the latest release of the fully supported NetApp open-source storage orchestrator for Kubernetes.

The following figure depicts a physical cabling diagram of the solution to the top of rack data center switches.



The next figure presents a logical view of the solution as deployed and validated on the hardware in the lab at the NetApp partner WWT.



Next: Solution validation.

Solution validation

The current deployment of this solution was put through two rigorous validation processes using tools provided by the Google Cloud team. These validations include a subset of the following tests:

- Partner validation of the Anthos-ready platform:
  - Confirm that all Anthos on bare metal platform services are installed and running.
  - Scale down the physical Anthos on bare metal cluster from four worker nodes to three and then back to four.
  - Create and delete a custom namespace.
  - Create a deployment of the Nginx web server, scaling that deployment by increasing the number of replicas.

    - Create an ingress for the Nginx application and verify connectivity by curling the index.html.
    - Successfully clean up all test suite activities and return the cluster to a pretest state.
- Partner validation of Anthos-ready storage:
    - Create a deployment with a persistent volume claim.
    - Use NetApp Trident to provision and attach the requested persistent volume from NetApp ONTAP.
    - Validate the detach and reattach capability of persistent volumes.
    - Validate multi-attach read-only access of persistent volumes from other pods on the node.
    - Validate the offline volume resize operation.
    - Verify that the persistent volume survives a cluster-scaling operation.

Next: Conclusion.

Conclusion

Anthos on bare metal with NetApp provides a robust platform to run container-based workloads efficiently by allowing for the customization of deployed infrastructure. Customers can use the server infrastructure and supported operating system of their choice or even deploy the solution within their existing infrastructure. The power and flexibility of these environments increases greatly through the integration of NetApp ONTAP and NetApp Trident, supporting stateful application workloads by efficiently provisioning and managing persistent storage for containers. By extending the potential of Google Cloud into their data center powered by NetApp, a customer can realize the benefits of a fully supported, highly available, easily scalable, and fully managed Kubernetes solution for development and production of their application workloads.

Next: Where to find additional information.

Where to find additional information

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

- NetApp ONTAP Documentation Center

  https://docs.netapp.com/ontap-9/index.jsp

- NetApp Trident

  https://netapp-trident.readthedocs.io/en/stable-v20.10/

- Google Cloud’s Anthos

  https://cloud.google.com/anthos

- Anthos on bare metal

  https://cloud.google.com/anthos/gke/docs/bare-metal

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.

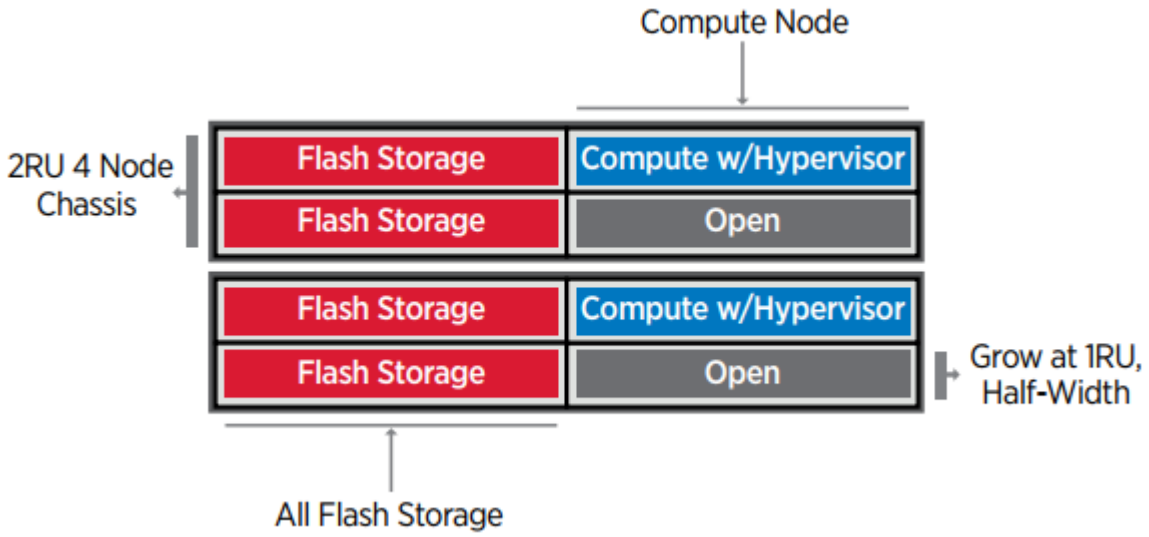
NetApp HCI

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:

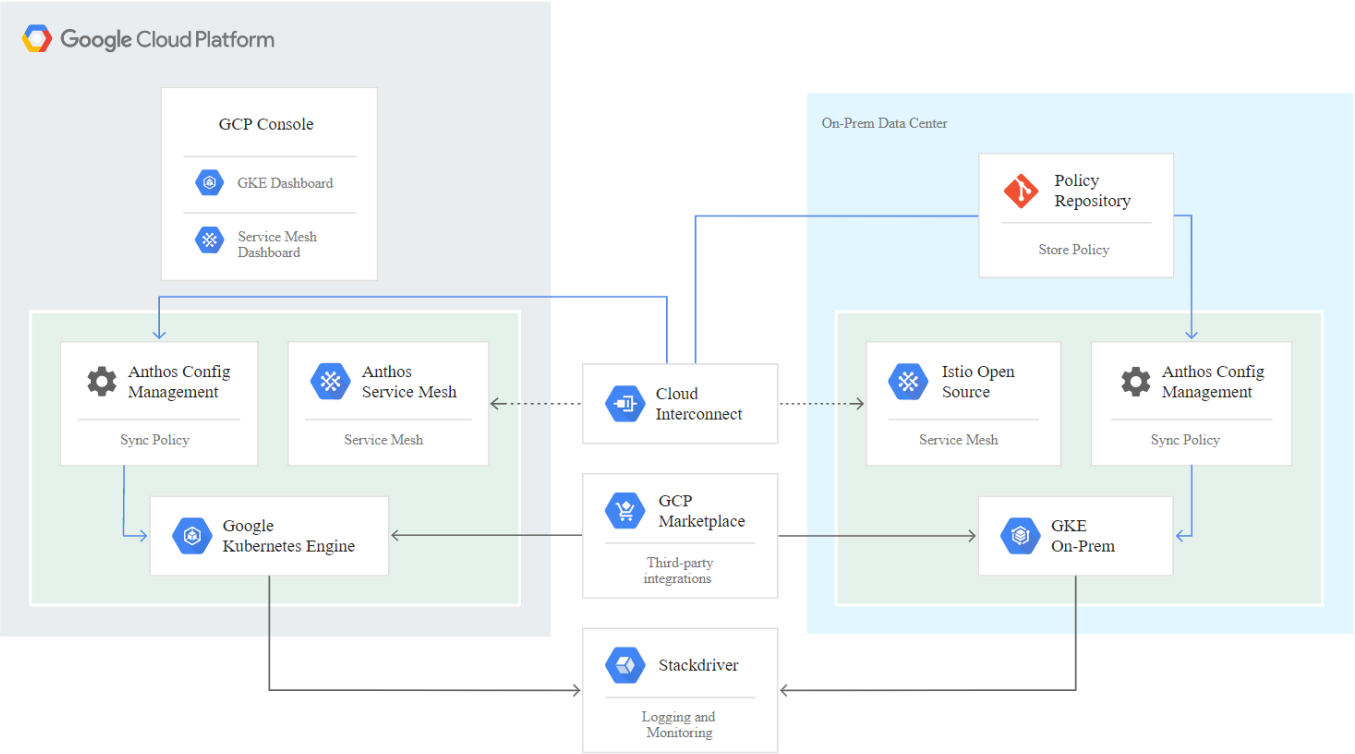
- **Centralized Management.** Through vSphere, individual hypervisors can be grouped into data centers and combined into clusters, allowing for advanced organization to ease the overall management of resources.
- **VMware HA.** This feature allows virtual guests to restart automatically if their host 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.

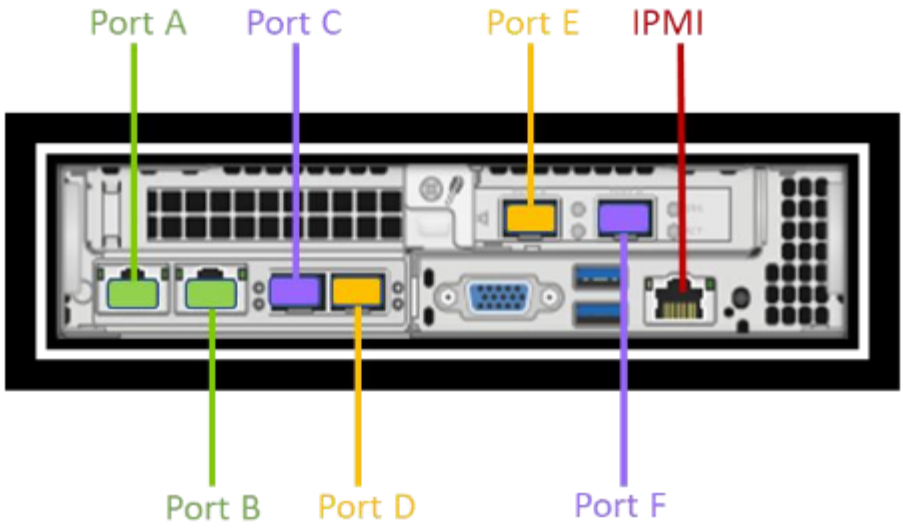
Next: Design Considerations

Design considerations

This section describes the design considerations necessary for the successful deployment of the NetApp HCI Anthos solution.

Port identification

NetApp HCI consists of NetApp H-Series nodes dedicated to either compute or storage. Both node configurations are available with two 1GbE ports (ports A and B) and two 10/25 GbE ports (ports C and D) on board. The compute nodes have additional 10/25GbE ports (ports E and F) available in the first mezzanine slot. Each node also has an additional out-of-band management port that supports Intelligent Platform Management Interface (IPMI) functionality. The following figure identifies each of these ports on the rear of an H410C node.



Network design

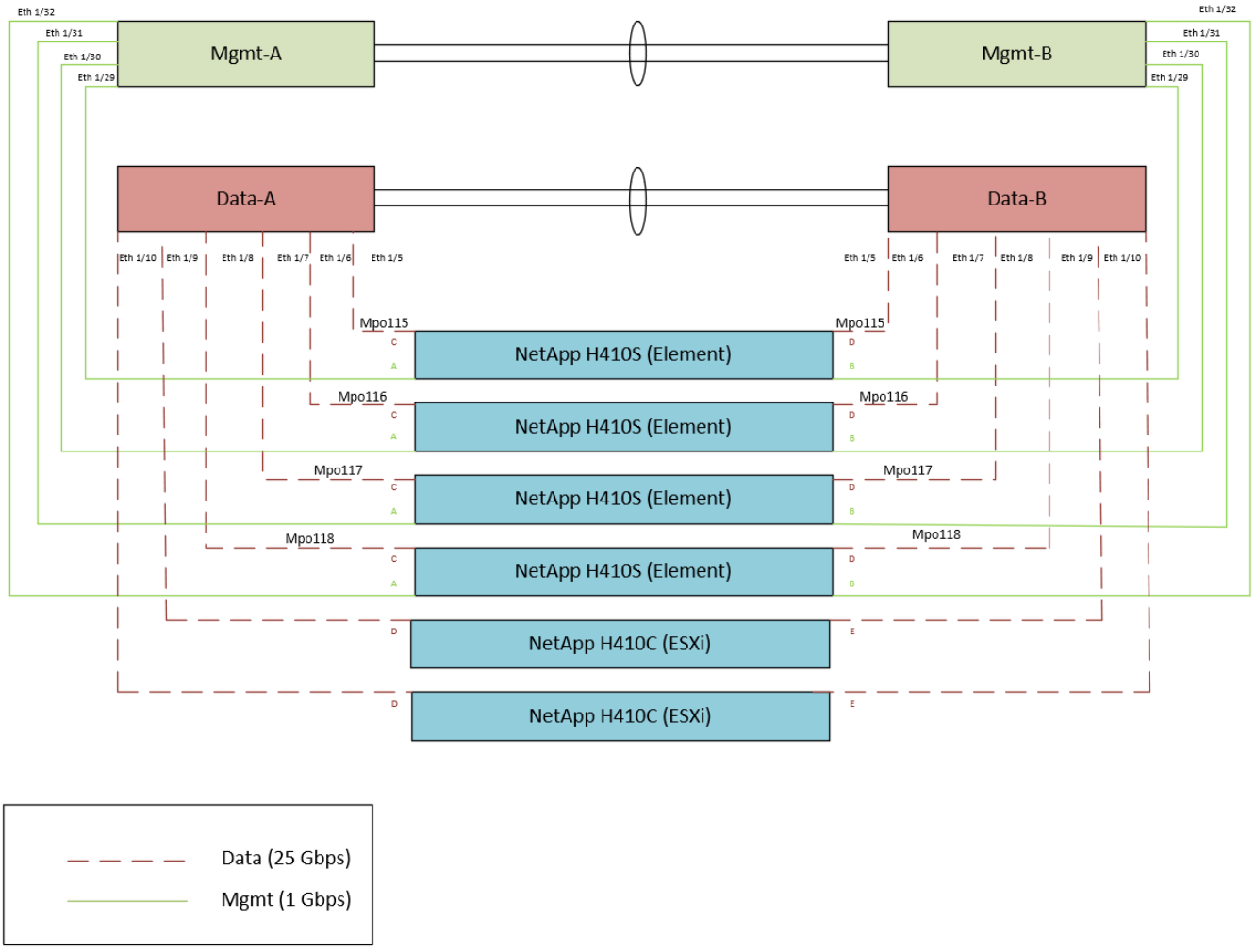
The NetApp HCI with Anthos solution uses two data switches to provide primary data connectivity at 25Gbps. It also uses two additional management switches that provide connectivity at 1Gbps for in-band management for the storage nodes and out-of-band management for IPMI functionality.

Cabling storage nodes

The management ports A and B must be active on each storage node to run NDE, configure the NetApp HCI cluster, and provide management accessibility to Element after the solution is deployed. The two 25Gbps ports (C and D) should be connected, one to each data switch, to provide physical fault tolerance. The switch ports should be configured for multi-chassis link aggregation (MLAG) and the data ports on the node should be configured for LACP with jumbo-frames support enabled. The IPMI ports on each node can be used to remotely manage the node after it is installed in a data center. With IPMI, the node can be accessed with a web-browser-based console to run the initial installation, run diagnostics, and reboot or shut down the node if necessary.

Cabling compute nodes

The 25Gbps ports on the compute nodes are cabled with one onboard port © cabled to one data switch, and an additional port from the PCI slot (E) cabled to the second switch to provide physical fault tolerance. These ports should be configured to support jumbo frames. Connectivity for the node is managed by the vDS after VMware vSphere is deployed in the environment. The IPMI ports can also be used to remotely manage the node after it is installed in a data center. With IPMI, the node can be accessed via a web-browser-based console to run diagnostics and to be rebooted or shut down if necessary. The following figure provides a reference for network cabling.



VLAN requirements

The 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 validated architecture deployment.

VLANs	Purpose	VLAN used
Out-of-band management	Management for HCI nodes	16
In-band management	Management for HCI nodes and infrastructure virtual guests	3480
Storage network	Storage network for NetApp Element	3481
vMotion network	Network for VMware vMotion	3482
VM network	Network for virtual guests	1172

Network infrastructure support resources

The following infrastructure should be in place prior to the deployment of the Anthos on NetApp HCI solution:

- A DHCP server providing addresses for both the in-band management network and the VM network. The DHCP pool must be large enough to support at least 10 VMs for an initial deployment and should be scaled as necessary.
- At least one DNS server providing 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.
- Outbound internet connectivity for both the in-band management network and the VM network.

Best practices

The details in this document describe a deployment of Anthos on VMware that meets the minimum requirements for deployment. Prior to deploying the solution in a production environment, you should use the information presented in this Best Practice section.

Install a second SeeSaw load balancer

In a production environment, it is a best practice to avoid single points of failure in your environment. For this validation, a single Seesaw bundled load balancer was allocated to the admin and each user cluster deployed. While this works fine for a simple validation, loss of communication with the control plane VIP for a cluster can make a cluster inaccessible or unable to be managed from the admin workstation or the Google Cloud console. By deploying HA Seesaw load balancers, it is possible to make sure disruptions do not happen. The setup procedures and additional requirements to enable this function are not described in detail in this document, however full instructions can be found here.

Install a second F5 Big-IP Virtual Edition appliance

In a production environment, it is a best practice to avoid single points of failure in your environment. For this validation, a single F5 BIG-IP Virtual Edition Load Balancer appliance was used to validate connectivity to the control plane and the ingress VIP addresses for the Anthos on VMware clusters. Although this works fine for a simple validation, loss of communication with the control plane VIP for a cluster can make a cluster inaccessible or unable to be managed from the admin workstation or the Google Cloud console. F5 BIG-IP Virtual Edition supports application-based HA to make sure disruptions do not happen. Although this issue is mentioned briefly, setup procedures for this functionality are not described in detail in this document. However, NetApp recommends investigating this feature further before deploying the NetApp HCI for Anthos solution into production.

Enable VMware vSphere DRS and configure anti-affinity rules

VMware vSphere provides a feature that makes sure that no single node in the cluster runs low on physical resources available to virtual guests. The Distributed Resource Scheduler (DRS) can be configured on vSphere clusters consisting of at least three ESXi nodes. The NetApp HCI minimum configuration described in this deployment guide consists of two compute nodes and is unable to make use of this feature. As a result of this limitation, we were also forced to disable anti-affinity rules for the Anthos on VMware clusters that we deployed.

Anti-affinity rules ensure that all masters or all workers for a specific user cluster run on different nodes so that a single node failure cannot disable an entire user cluster or the pods that it is hosting. The NetApp HCI system is both easily and rapidly scalable and the minimum deployment described in this validation has two open chassis slots for immediate expansion of HCI 410C nodes. Therefore, NetApp suggests adding additional compute nodes into the empty chassis slots prior to deploying the solution into production and enabling DRS with anti-affinity rules.

Use SnapMirror to copy data remotely for disaster recovery

NetApp Element storage systems can use NetApp SnapMirror technology to replicate storage volumes to systems running the NetApp ONTAP system, including AFF, FAS, and Cloud Volumes ONTAP. You can set up regularly scheduled SnapMirror operations to back up the VMware datastores and restore from a remote site in the event of a disaster. It is also possible to use SnapMirror to back up or migrate the persistent volumes provisioned by Trident and reattach them to Kubernetes clusters deployed in other environments and in the cloud.

Next: Hardware and Software Requirements

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
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
```

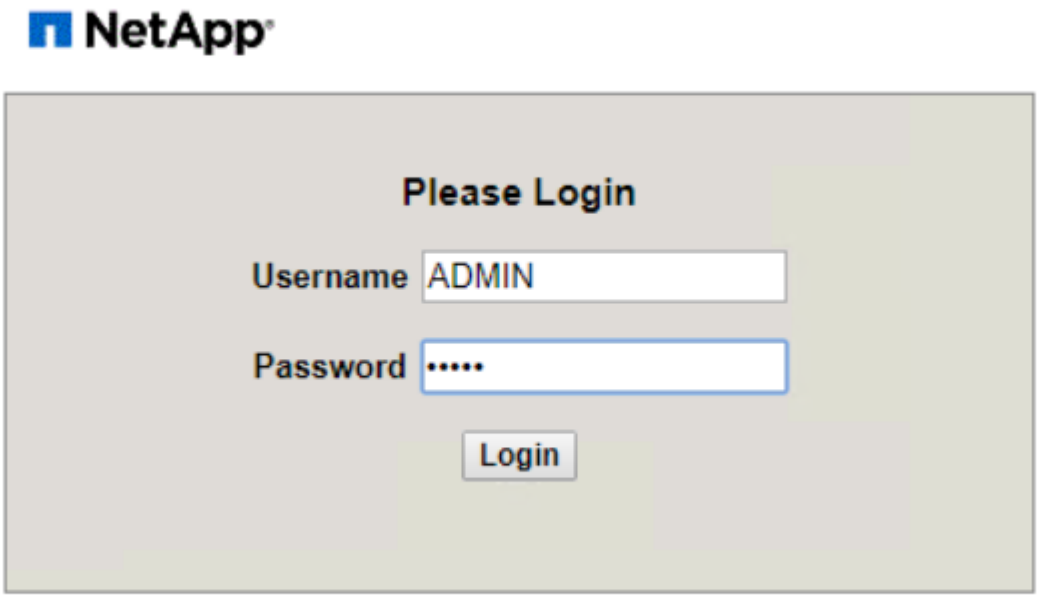
Next: Deploy NetApp HCI with the NetApp Deployment Engine

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:

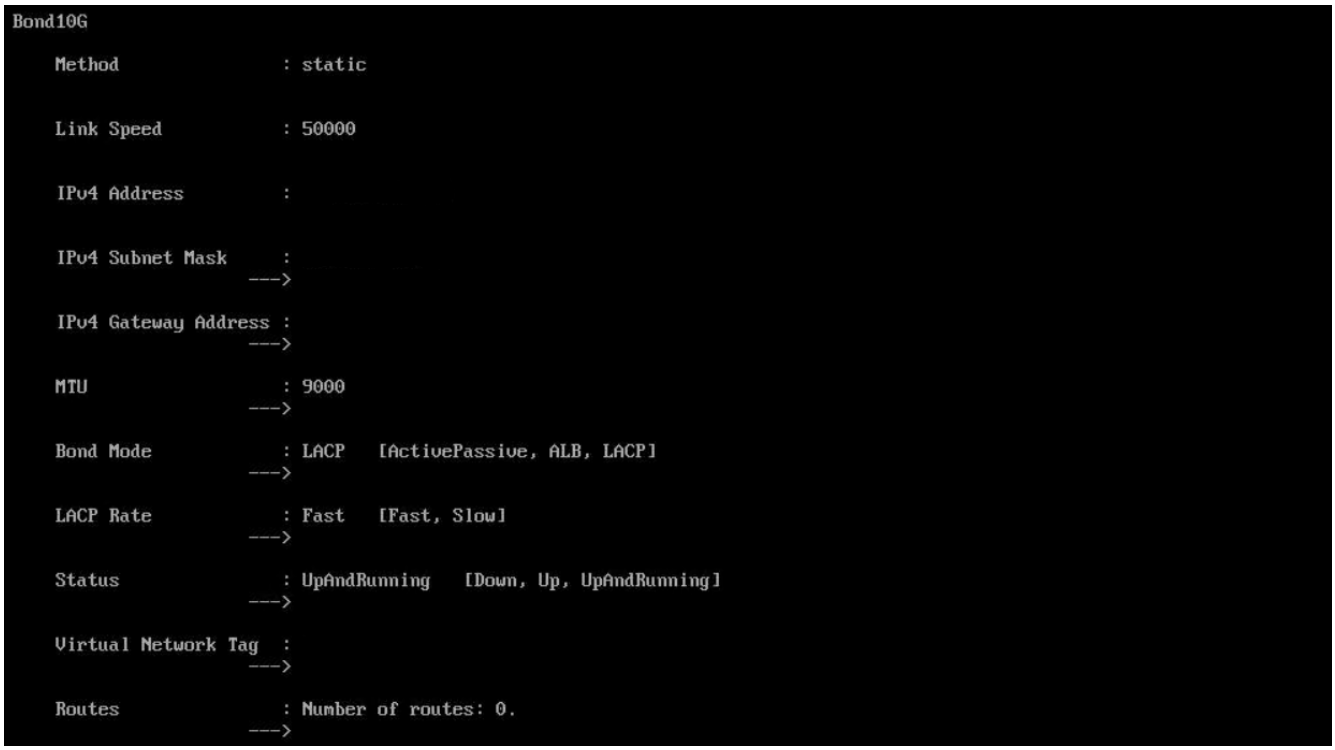
1. Access the out-of-band management console for one of the storage nodes in the cluster and log in with the default credentials ADMIN/ADMIN.



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.

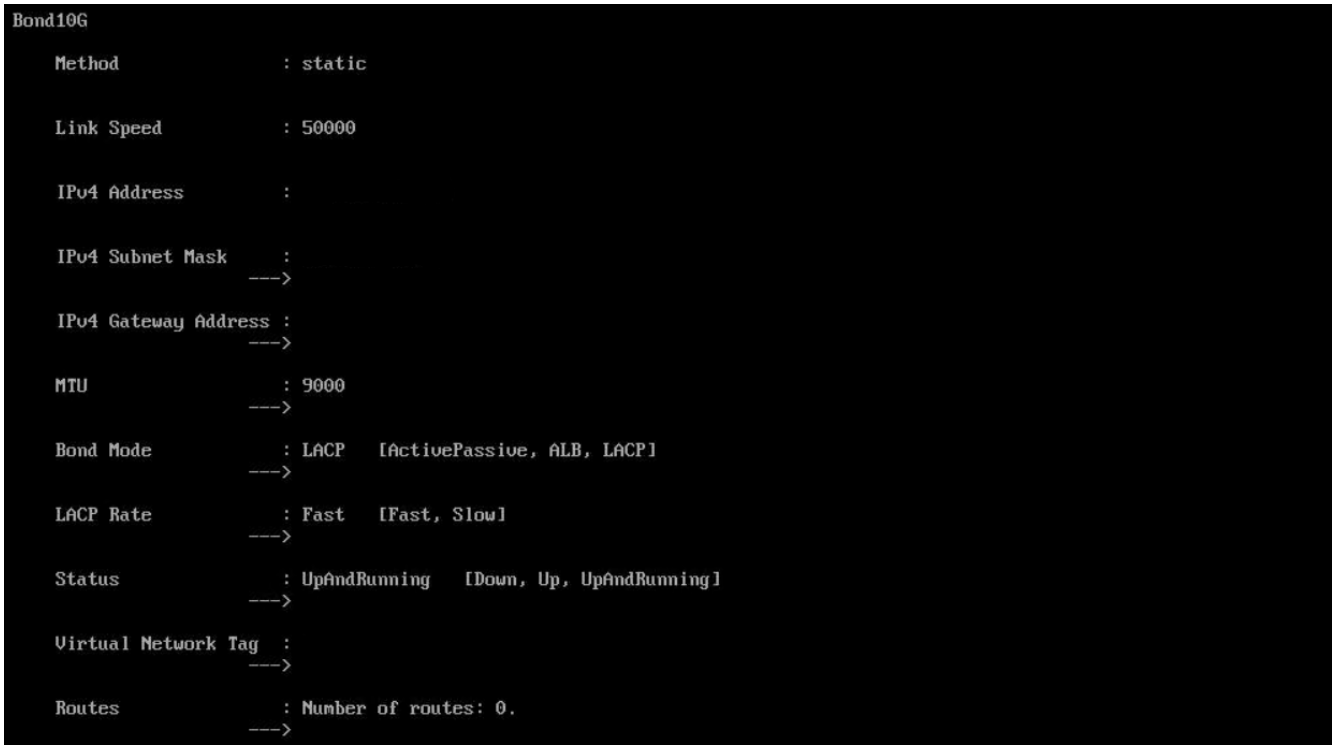
4. The Bond1G interface must have an IP, a netmask, and a gateway set statically; its VLAN set to 3480; and DNS servers defined for the environment.



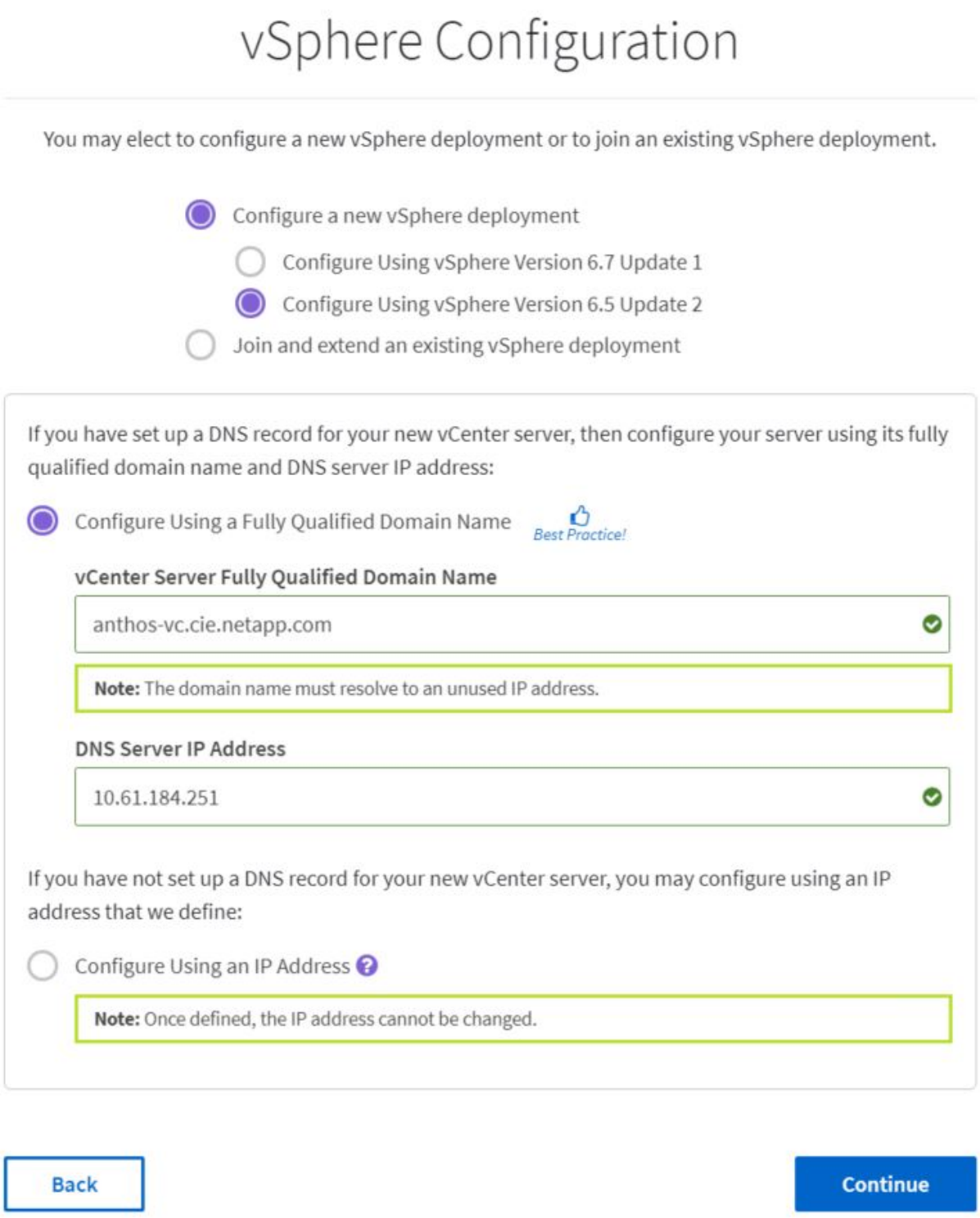
Select an IP that is within the subnet you intend to use for in-band management but not an IP you would like to use in production. NDE reconfigures the node with a production IP after initial access.

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.

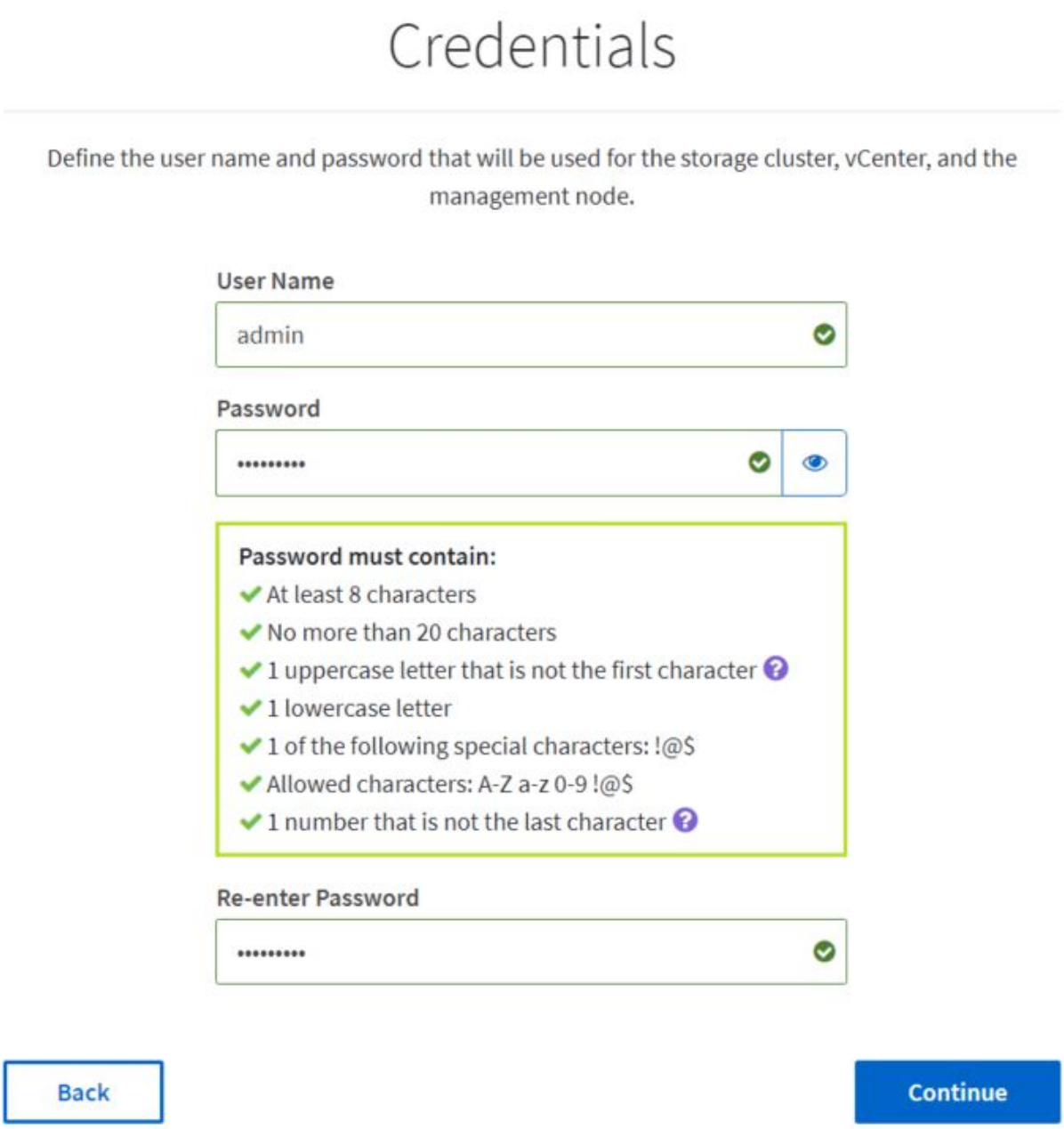
5. The Bond 10G interface must have its MTU setting changed to enable jumbo frames and its bond mode changed to LACP.



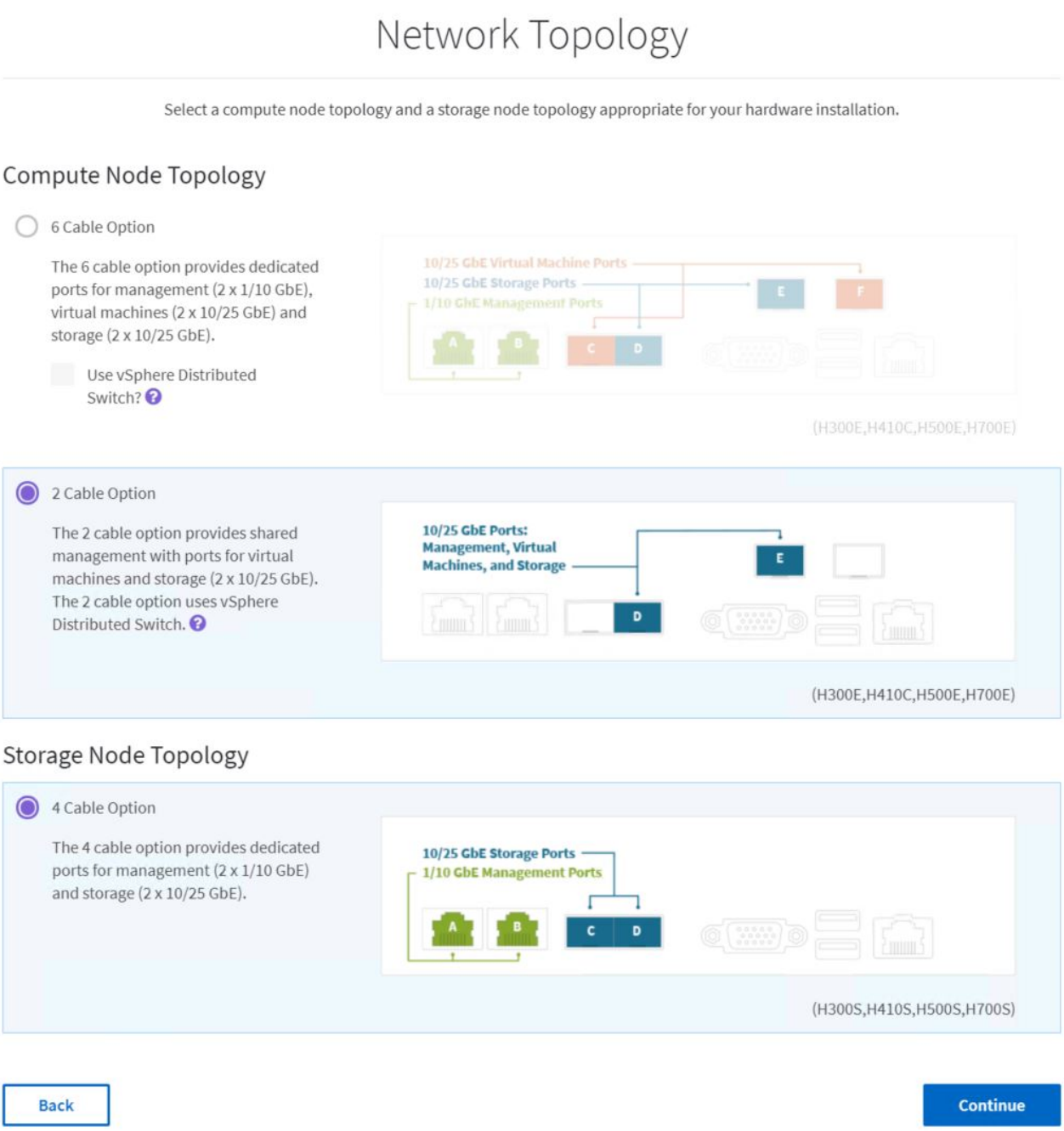
6. After completion, open a web browser and visit the IP address you configured for the management port to start NetApp HCI configuration with NDE.
7. On the Welcome to NetApp HCI page, click the Get Started button.
8. Check each associated box on the Prerequisites page and click Continue.
9. The next 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.



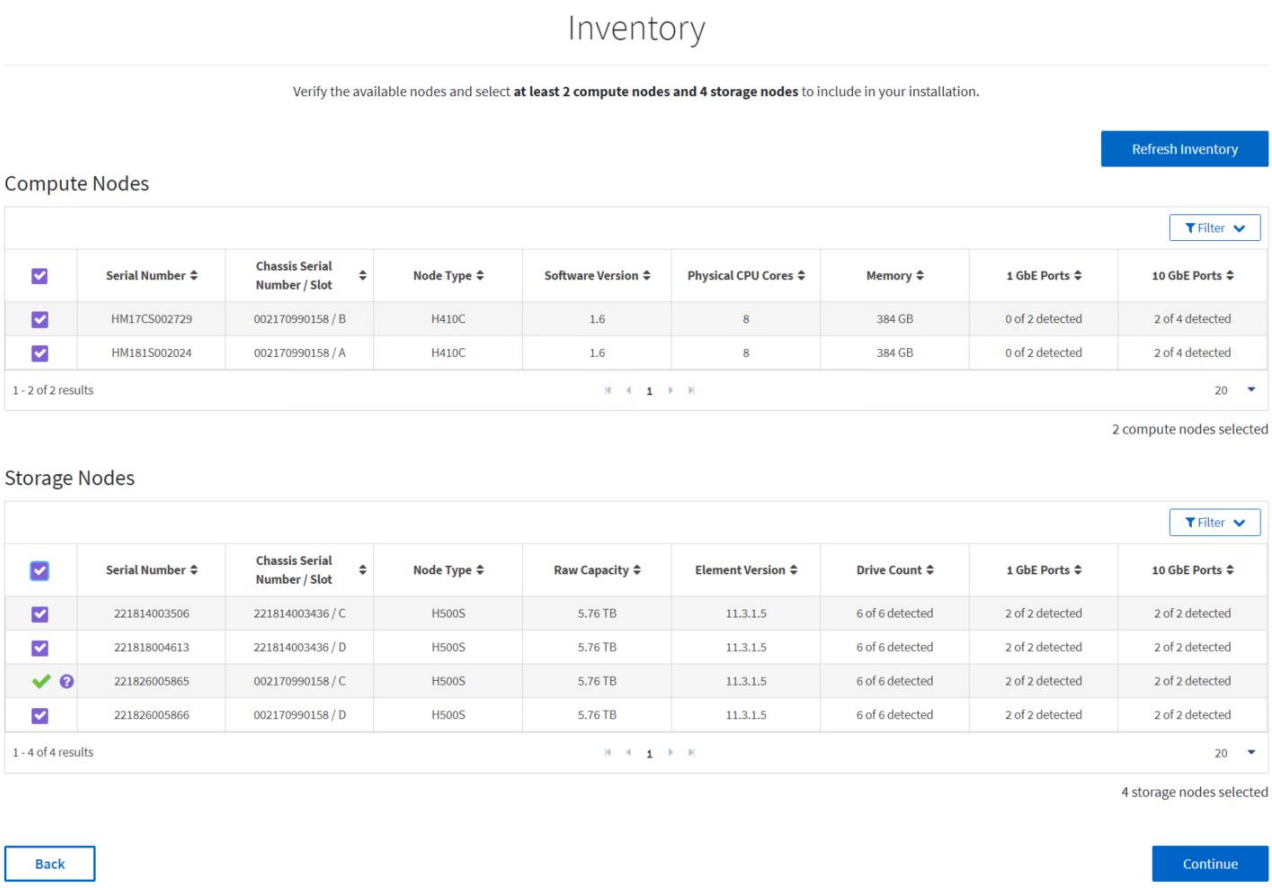
11. NDE asks for the credentials to be used in the environment. This is used for VMware vSphere, the NetApp Element storage cluster, and the NetApp Mnode, which provides management functionality for the cluster. When you are finished, click Continue.



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.

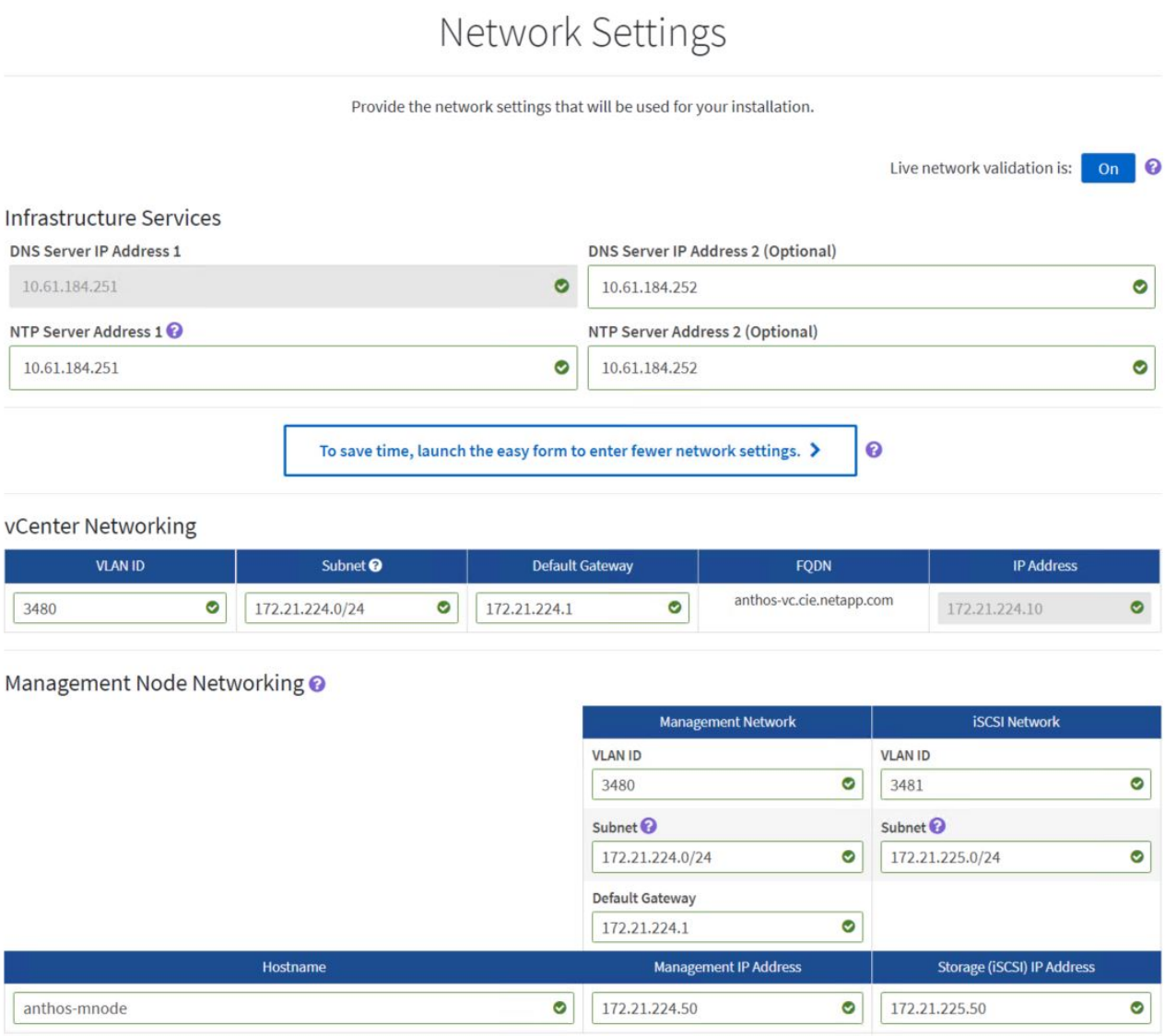


13. The next page presented by NDE is the inventory of the environment as discovered by the APIPA addressed on the storage network. The storage node that is currently running NDE is already selected with a green check mark. Select the corresponding boxes to add additional nodes to the NetApp HCI environment.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

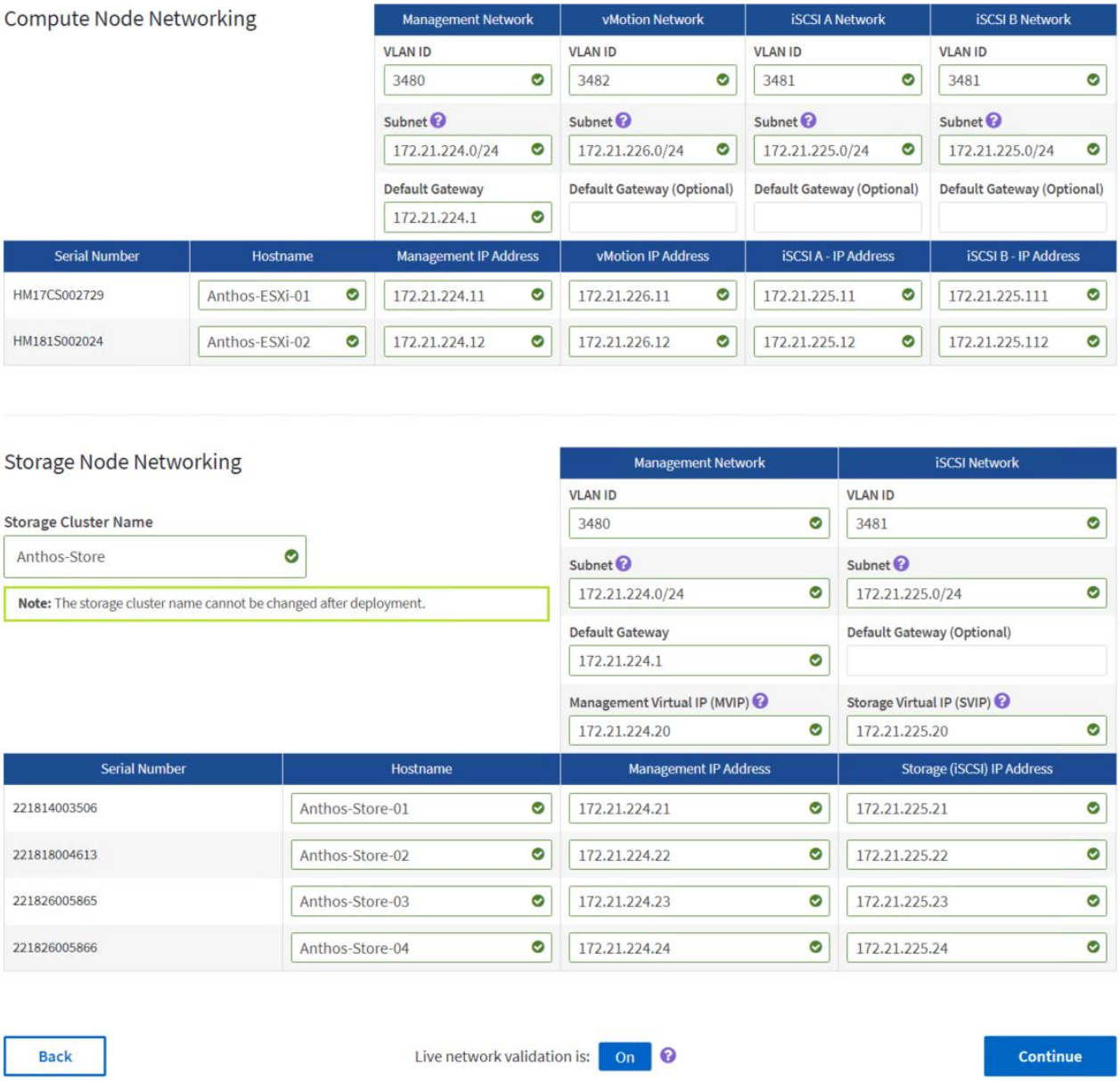


If there are any nodes missing from the inventory screen, wait a few minutes and click Refresh Inventory. If the node still fails to appear, additional investigation of environment networking might be required.

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.



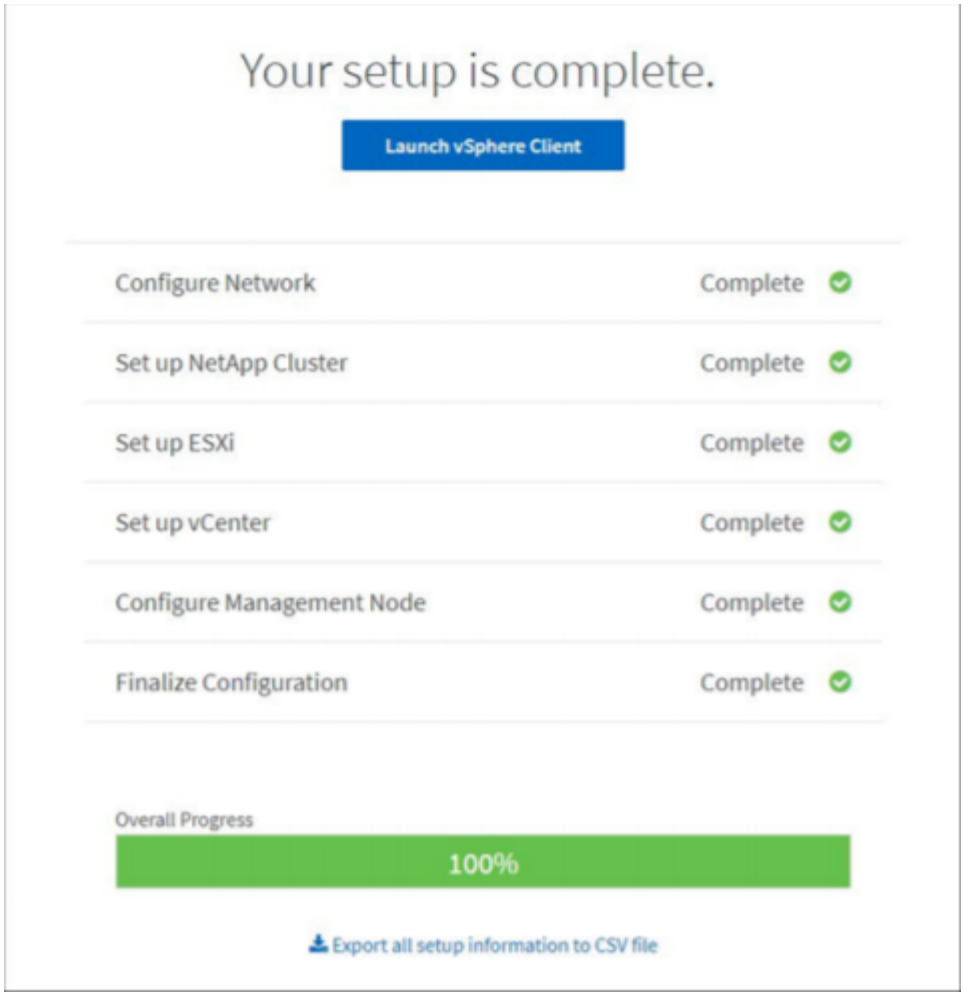
15. The next page allows you to configure each node in the environment. For the compute nodes, it allows you to configure the host name, management network, vMotion network, and storage network. For the storage nodes, name the storage cluster and configure the management and storage networks being used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

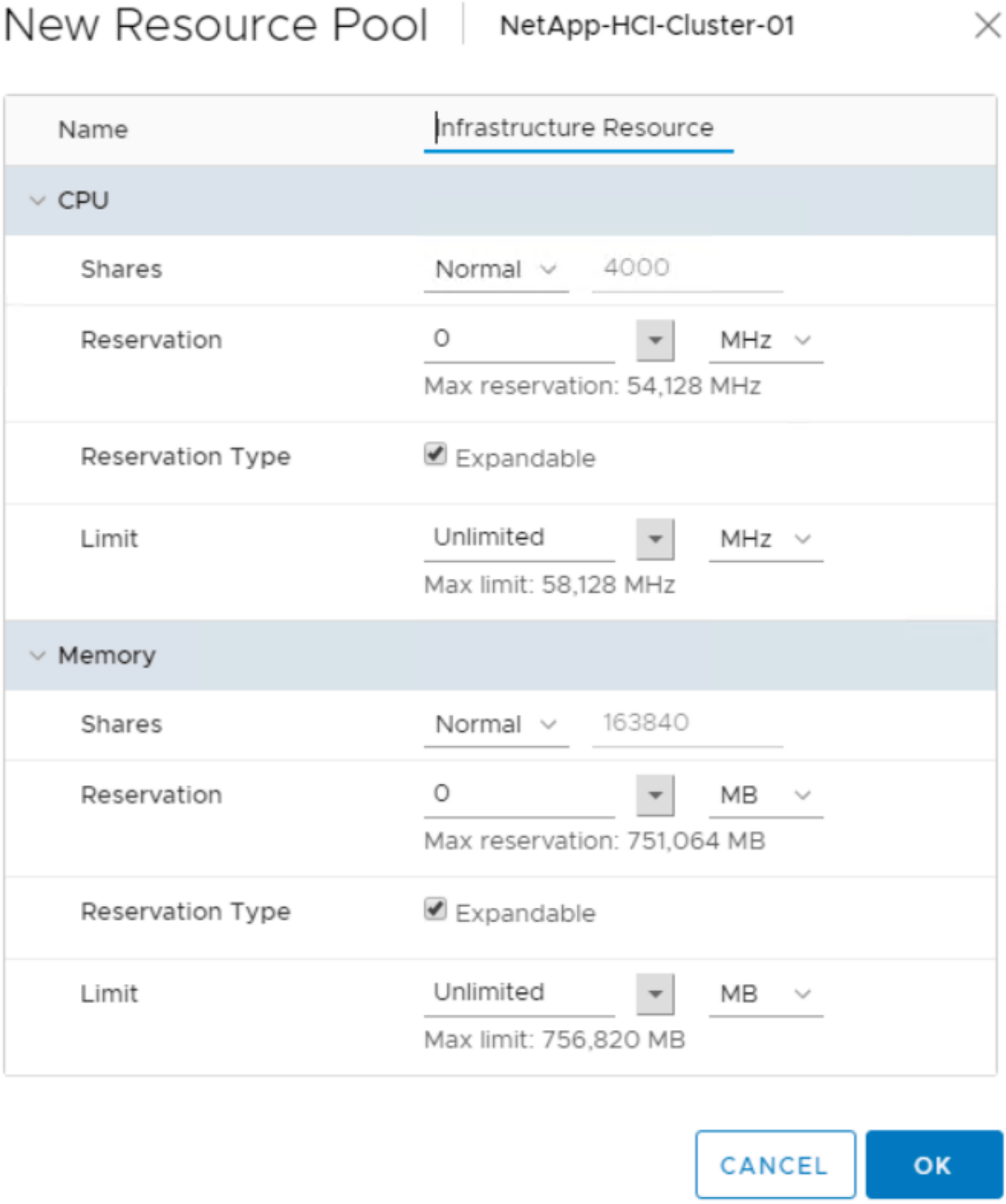
Note that NDE deploys vSphere 6.7U1. You can upgrade the Virtual Appliance and individual ESXi hosts by following the instructions from VMware here.

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:

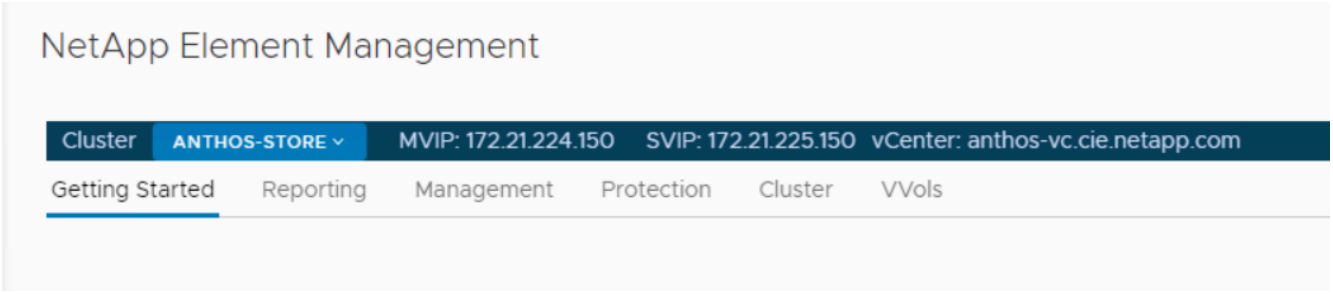
1. Log into the VMware vCenter server using the Administrator@vsphere.local account and the password chosen for the admin user during NDE configuration.

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.

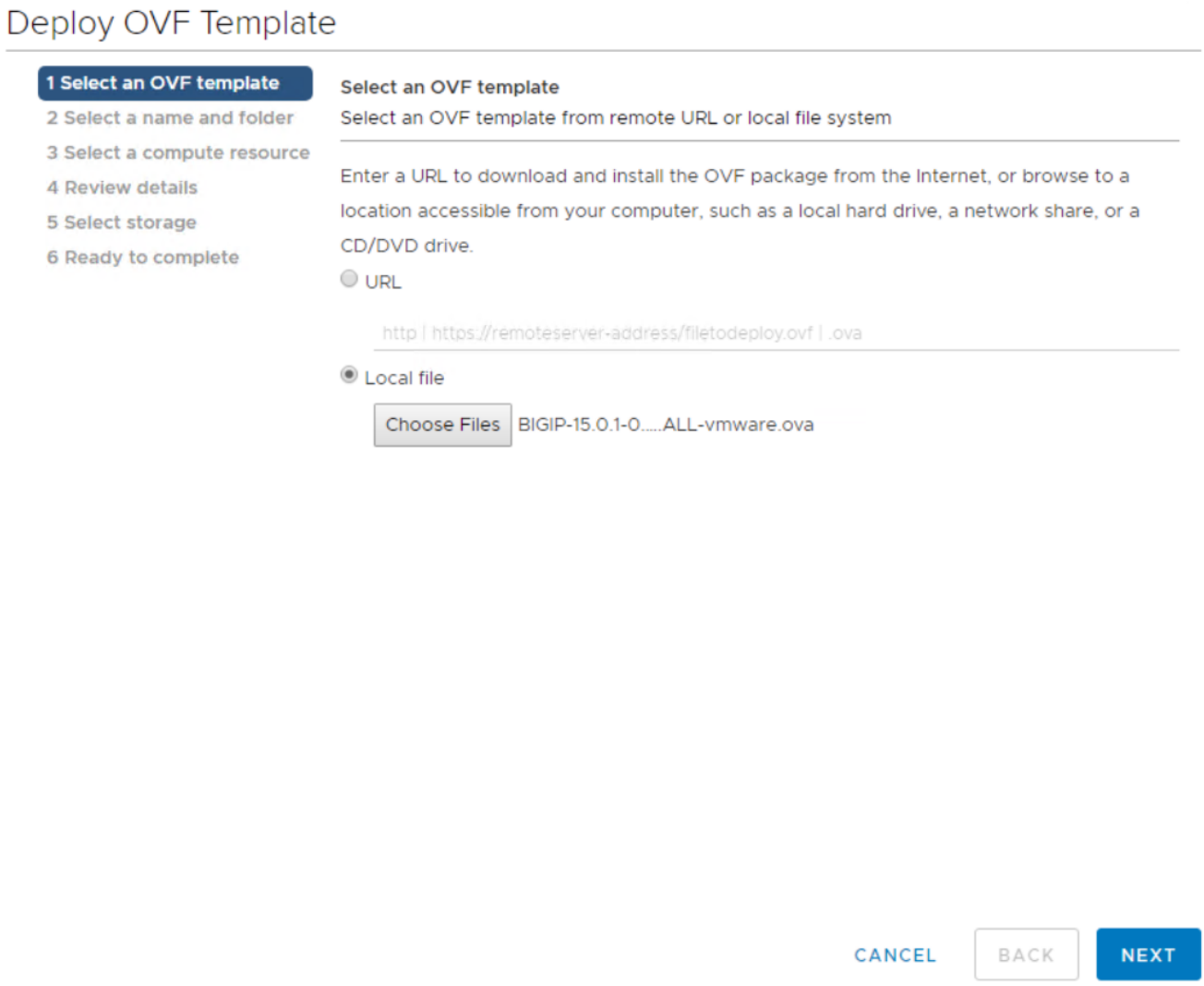
This solution makes use of the virtual appliance deployed in VMware vSphere as deployed by NDE. Networking for the F5 Big-IP virtual appliance can be configured in a two-armed or three-armed configuration based on your network environment. The deployment in this document is based on the two-armed configuration.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

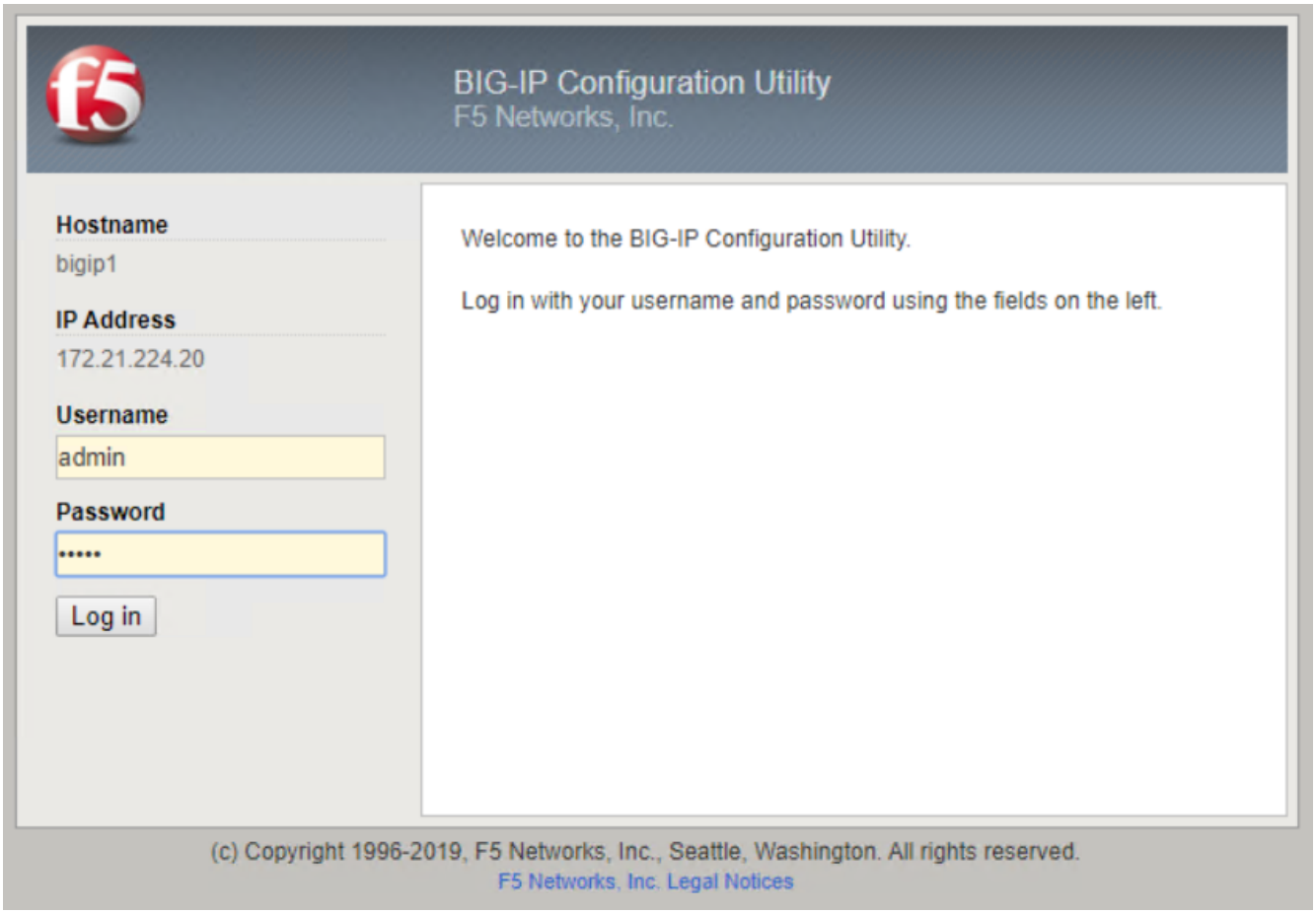


3. Click Next to continue through each step and accept the default values for each screen presented until you reach the storage selection screen. Select the VM_Datastore that was created earlier, and then click Next.
4. The next screen presented by the wizard allows you to customize the virtual networks 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.

    [Big-IP Configuration]

    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.

    [Big-IP Configuration]

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 user clusters

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 20 user clusters, and each user cluster can support up to 250 nodes and 7500 pods.

To configure user clusters for your deployment, complete the following steps:

1. When the anthos-admin workstation is deployed, a file called `user-cluster.yaml` is created that can be used to deploy a number of additional user clusters for running workloads. Start by copying this default file with a new name for each cluster you intend to deploy.

```
ubuntu@gke-admin-ws-200915-151421:~ $ cp config.yaml anthos-cluster01-
config.yaml
```

2. Edit the `anthos-cluster01-config.yaml` file so that it is specific for the environment that is being deployed.
   a. In a manner similar to the `admin-config.yaml` used earlier, most of the variables are already filled in or they reference the admin-cluster for the information needed to deploy. This first section confirms the information for the version of Anthos being deployed and the vCenter instance it is being deployed on.

```
apiVersion: v1
kind: UserCluster
# (Required) A unique name for this cluster
name: "anthos-cluster01"
# (Required) GKE on-prem version (example: 1.3.0-gke.16)
gkeOnPremVersion: 1.6.0-gke.7
# # (Optional) vCenter configuration (default: inherit from the admin
cluster)
# vCenter:
#   resourcePool: ""
#   datastore: ""
#   # Provide the path to vCenter CA certificate pub key for SSL
verification
#   caCertPath: ""
#   # The credentials to connect to vCenter
#   credentials:
#     username: ""
#     password: ""
```

   b. You must 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 to supply addresses similar to the admin-cluster configuration.

The items under the hostConfig section are global for any time static IPs are used in a deployment. This includes both static IPs for the cluster and those used for the SeeSaw load balancers, which are configured later.

```
# (Required) Network configuration; vCenter section is optional and
inherits from
# the admin cluster if not specified
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: 10.96.0.0/12
  # (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. Next fill out the load balancer section. This can vary depending on the type of load balancer being deployed.

SeeSaw Example:

```
# (Required) Load balancer configuration
loadBalancer:
```

```
  # (Required) The VIPs to use for load balancing
  vips:
    # Used to connect to the Kubernetes API
    controlPlaneVIP: "10.63.172.156"
    # Shared by all services for ingress traffic
    ingressVIP: "10.63.172.157"
  # (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: 30243
  #   # NodePort for ingress service's https (only needed for user
cluster)
  #   ingressHTTPSNodePort: 30879
  #   # NodePort for control plane service
  #   controlPlaneNodePort: 30562
  #   # NodePort for addon service (only needed for admin cluster)
  #   addonsNodePort: 0
  # # (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: "anthos-cluster01-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: 101
    # (Required) The IP announced by the master of Seesaw group
    masterIP: "10.63.172.153"
```

```
    # (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 `anthos-cluster01-seesaw-block.yaml` that was referenced in this configuration section.

```
blocks:
  - netmask: "255.255.255.0"
    gateway: "10.63.172.1"
    ips:
    - ip: "10.63.172.154"
      hostname: "anthos-cluster01-seesaw-vm"
```

F5 BigIP Example:

```
loadBalancer:
  # (Required) The VIPs to use for load balancing
  vips:
    # Used to connect to the Kubernetes API
    controlPlaneVIP: "10.63.172.158"
    # Shared by all services for ingress traffic
    ingressVIP: "10.63.172.159"
  # (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: 30243
  #   # NodePort for ingress service's https (only needed for user
```

```
cluster)
  #   ingressHTTPSNodePort: 30879
  #   # NodePort for control plane service
  #   controlPlaneNodePort: 30562
  #   # NodePort for addon service (only needed for admin cluster)
  #   addonsNodePort: 0
  # # (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: "Anthos-Cluster-01"
  #   # # (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 final section describes the resources for the nodes that the cluster is deploying, including creating a node pool that we can use for dynamic scaling later. This section also supplies the service account keys to register the cluster with GKE once deployed.

```
# (Optional) User cluster master nodes must have either 1 or 3
replicas (default:
# 4 CPUs; 16384 MB memory; 1 replica)
masterNode:
  cpus: 4
  memoryMB: 8192
  # How many machines of this type to deploy
  replicas: 1
# (Required) List of node pools. The total un-tainted replicas across
all node pools
# must be greater than or equal to 3
nodePools:
- name: anthos-cluster01
  # # Labels to apply to Kubernetes Node objects
  # labels: {}
  # # Taints to apply to Kubernetes Node objects
  # taints:
  # - key: ""
  #   value: ""
  #   effect: ""
  cpus: 4
  memoryMB: 8192
  # How many machines of this type to deploy
  replicas: 3
# Spread nodes across at least three physical hosts (requires at
least three hosts)
antiAffinityGroups:
  # Set to false to disable DRS rule creation
  enabled: false
# # (Optional): Configure additional authentication
# authentication:
#   # (Optional) Configure OIDC authentication
#   oidc:
#     issuerURL: ""
#     kubectlRedirectURL: ""
#     clientID: ""
#     clientSecret: ""
#     username: ""
#     usernamePrefix: ""
#     group: ""
#     groupPrefix: ""
#     scopes: ""
#     extraParams: ""
#     # Set value to string "true" or "false"
#     deployCloudConsoleProxy: ""
```

```
#     # # The absolute or relative path to the CA file (optional)
#     # caPath: ""
#   # (Optional) Provide an additional serving certificate for the
API server
#   sni:
#     certPath: ""
#     keyPath: ""
#   # (Optional) Configure LDAP authentication (preview feature)
#   ldap:
#     name: ""
#     host: ""
#     # Only support "insecure" for now (optional)
#     connectionType: insecure
#     # # The absolute or relative path to the CA file (optional)
#     # caPath: ""
#     user:
#       baseDN: ""
#       userAttribute: ""
#       memberAttribute: ""
# (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) Specify which GCP project to connect your GKE clusters
to
gkeConnect:
  projectID: "anthos-dev"
  # The absolute or relative path to the key file for a GCP service
account used to
  # register the cluster
  registerServiceAccountKeyPath: "/home/ubuntu/connect-register-
key.json"
  # The absolute or relative path to the key file for a GCP service
account used by
  # the GKE connect agent
  agentServiceAccountKeyPath: "/home/ubuntu/component-access-
key.json"
# (Optional) Specify Cloud Run configuration
```

```
cloudRun:
  enabled: false
# # (Optional/Alpha) Configure the GKE usage metering feature
# usageMetering:
#   bigQueryProjectID: ""
#   # The ID of the BigQuery Dataset in which the usage metering data
will be stored
#   bigQueryDatasetID: ""
#   # The absolute or relative path to the key file for a GCP service
account used by
#   # gke-usage-metering to report to BigQuery
#   bigQueryServiceAccountKeyPath: ""
#   # Whether or not to enable consumption-based metering
#   enableConsumptionMetering: false
# # (Optional/Alpha) 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: ""
```

3. After the edits to the configuration file are complete, NetApp recommends that the file be checked for proper syntax and spacing. You can check the config file you just created. This command references the `kubeconfig` file created by the admin-cluster.

```
ubuntu@gke-admin-200915-151421:~$ gkectl check-config --kubeconfig
kubeconfig --config anthos-cluster01-config.yaml
```

4. If you are using a SeeSaw load balancer, you need to create it prior to deploying the user cluster.

```
ubuntu@gke-admin-200915-151421:~$ gkectl create loadbalancer
--kubeconfig kubeconfig --config anthos-cluster-01-config.yaml
```

5. Create the user cluster. Just as we did with the admin cluster, the process can be accelerated by skipping the additional validations because we have already run the checks in the prior step.

```
ubuntu@gke-admin-200915-151421:~$ gkectl create cluster --config anthos-
cluster-01-config.yaml --skip-validation-all
```

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

```
ubuntu@gke-admin-ws-200915-151421:~$ kubectl get nodes --kubeconfig
anthos-cluster01-kubeconfig
NAME                    STATUS   ROLES    AGE   VERSION
anthos-cluster01-7b5995cc45-ftrdw   Ready    <none>   5m   v1.18.6-
gke.6600
anthos-cluster01-7b5995cc45-z7q9b   Ready    <none>   5m   v1.18.6-
gke.6600
anthos-cluster01-7b5995cc45-zw6sv   Ready    <none>   6m   v1.18.6-
gke.6600
ubuntu@gke-admin-ws-200915-151421:~/ $ gkectl diagnose cluster
--kubeconfig kubeconfig --cluster-name anthos-cluster01
Diagnosing user cluster "anthos-cluster01"...

- Validation Category: User Cluster VCenter
Checking Credentials...SUCCESS
Checking VSphere CSI Driver...SUCCESS
Checking Version...SUCCESS
Checking Datacenter...SUCCESS
Checking Datastore...SUCCESS
Checking Resource pool...SUCCESS
Checking Folder...SUCCESS
Checking Network...SUCCESS
Checking Datastore...SUCCESS

- Validation Category: User Cluster
Checking onpremusercluster and onpremnodepool...SUCCESS
Checking cluster object...SUCCESS
Checking machine deployment...SUCCESS
Checking machineset...SUCCESS
Checking machine objects...SUCCESS
Checking control place pods...SUCCESS
Checking gke-connect pods...SUCCESS
Checking config-management-system pods...Warning: No pod is running in
namespace "config-management-system"...SUCCESS
Checking kube-system pods...SUCCESS
Checking gke-system pods...SUCCESS
Checking storage...SUCCESS
Checking resource...System pods on UserNode cpu resource request report:
total 3059m nodeCount 3 min 637m max 1224m avg 1019m tracked amount in
bundle 4000m
System pods on UserNode memory resource request report: total 6464Mi
nodeCount 3 min 1670Mi max 2945Mi avg 2259331754 tracked amount in
bundle 8192Mi
SUCCESS
Cluster is healthy.
```

Next: Enable access to the cluster with the GKE console.

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.

> 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
```

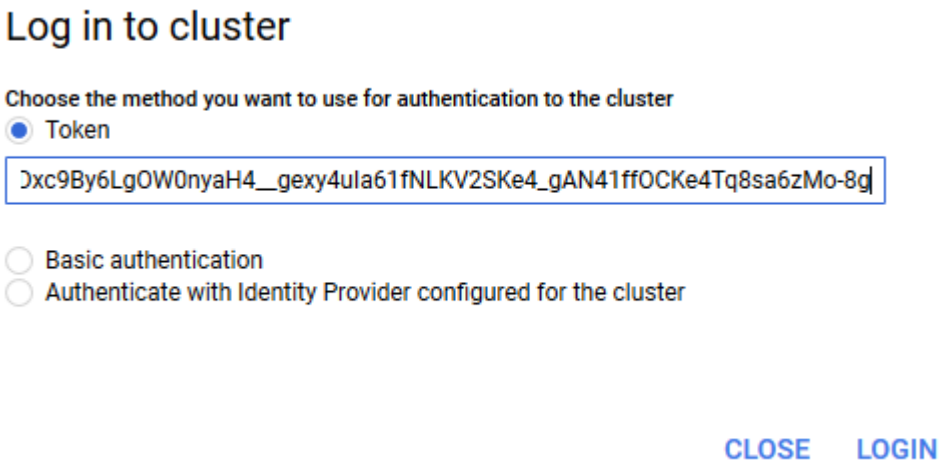
5. If you need to extend permissions further, you can grant the KSA user a role with cluster admin permissions in a similar manner.

```
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/netapp-solutions/media/Anthos-Deploy-App-Demo.mp4 *(video)*

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

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

Where to Find Additional Information: NetApp HCI with Anthos

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

- Anthos Documentation
- NetApp HCI Documentation
- NetApp NDE 1.8 Deployment Guide
- NetApp Trident Documentation
- VMware vSphere 6.7U3 Documentation
- F5 Big-IP Documentation