Red Hat OpenShift with NetApp

NetApp Solutions

NetApp
August 18, 2021

Table of Contents

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 be utilized to 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, powered by NetApp's trusted 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
NetApp Element	Storage	12.3
NetApp Astra Control Center	Application Aware Data Management	21.08.57
NetApp Astra Trident	Storage Orchestration	21.04
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.

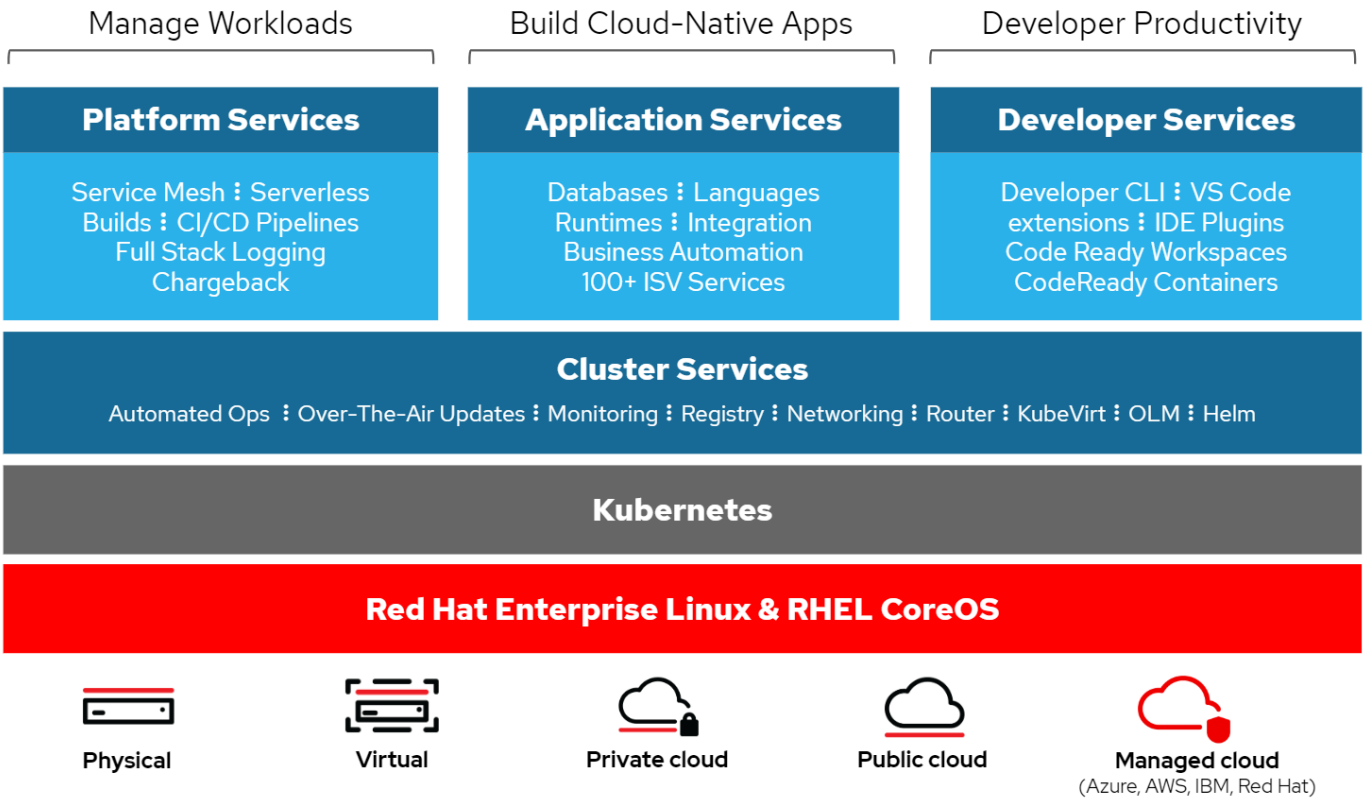
OpenShift Overview: Red Hat OpenShift with NetApp

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, encryption through TLS certificates for the various subsystems, and providing 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

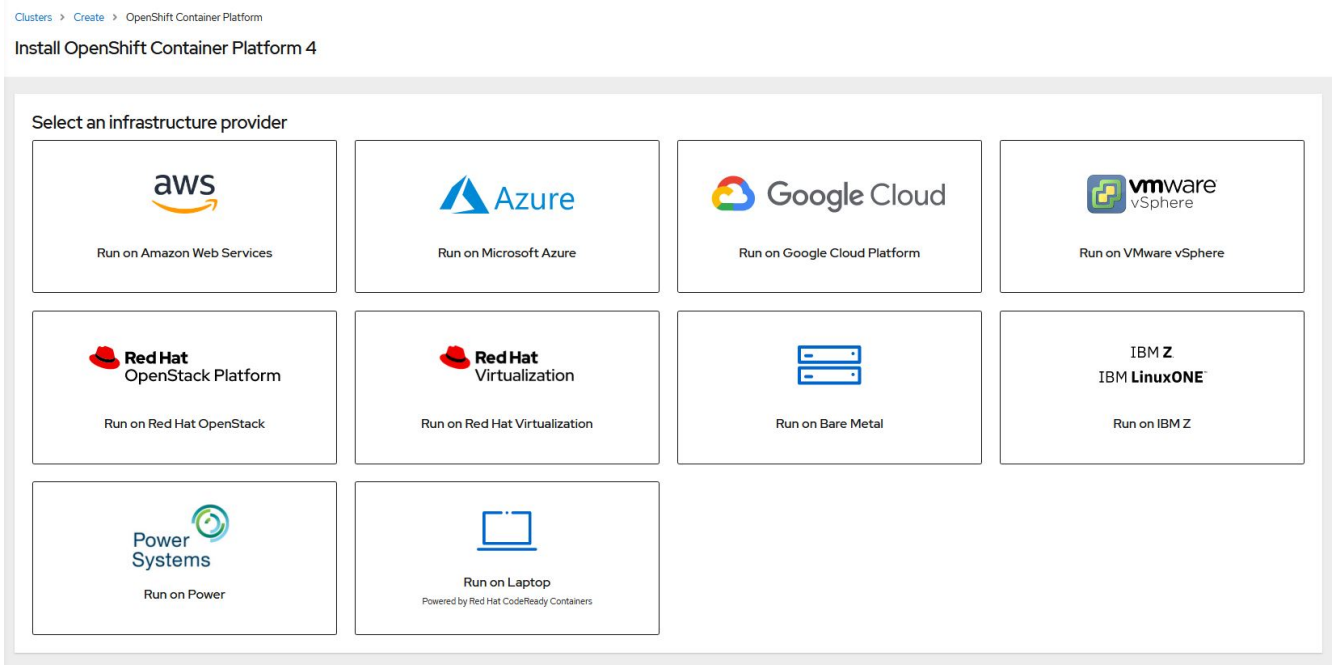
Starting with Red Hat OpenShift 4, the deployment methods for OpenShift include manual deployments using User Provisioned Infrastructure (UPI) for highly customized deployments, or fully automated deployments using Installer Provisioned Infrastructure (IPI).

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.

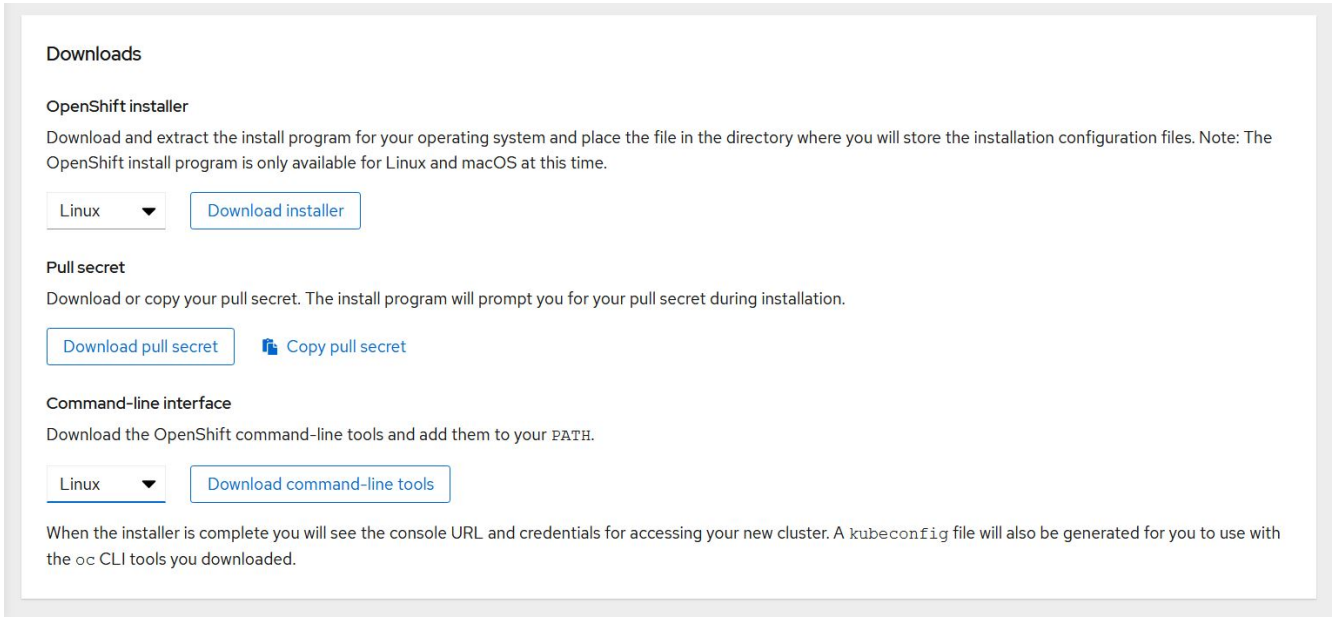
IPI installation of Red Hat OpenShift

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'd like to deploy Red Hat OpenShift into.



3. On the next screen download the installer, the unique pull secret, 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 it's 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: Red Hat OpenShift with NetApp

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

Similar to virtual deployments of OpenShift, which are quite popular because they allow for ease of deployment, rapid provisioning, and scaling of OpenShift clusters, while also supplying the need to support virtualized workloads for applications that are not ready to be containerized. By deploying on bare metal, a customer does require the extra overhead in managing the host hypervisor environment, as well as the OpenShift environment. By deploying directly on bare metal servers, the customer 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 an 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 leveragine 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 additional management switches that provide connectivity at 1Gbps for in-band management for the storage nodes and out-of-band management for IPMI functionality.

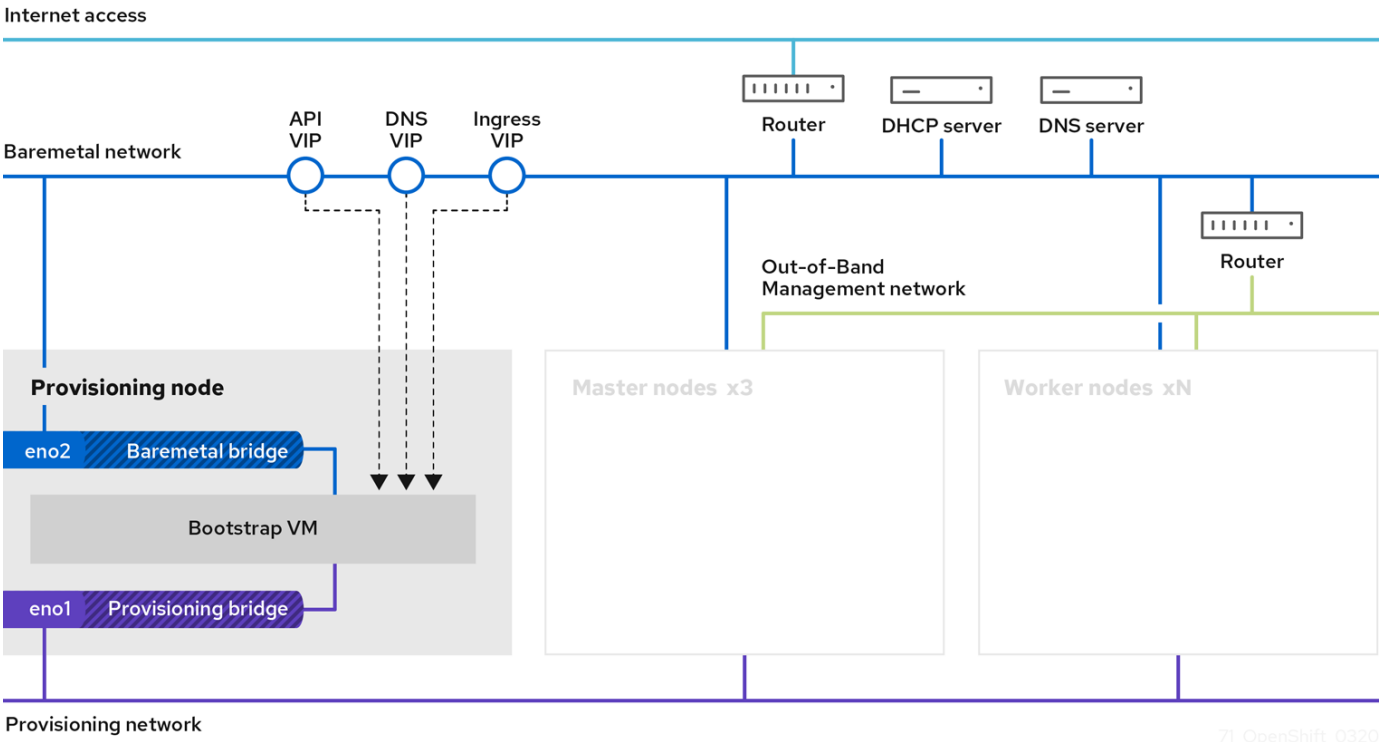
The OpenShift bare metal IPI deployment requires the customer to create a Provisioner node, a Red Hat Enterprise Linux 8 machine which will need to have network interfaces attached to separate networks.

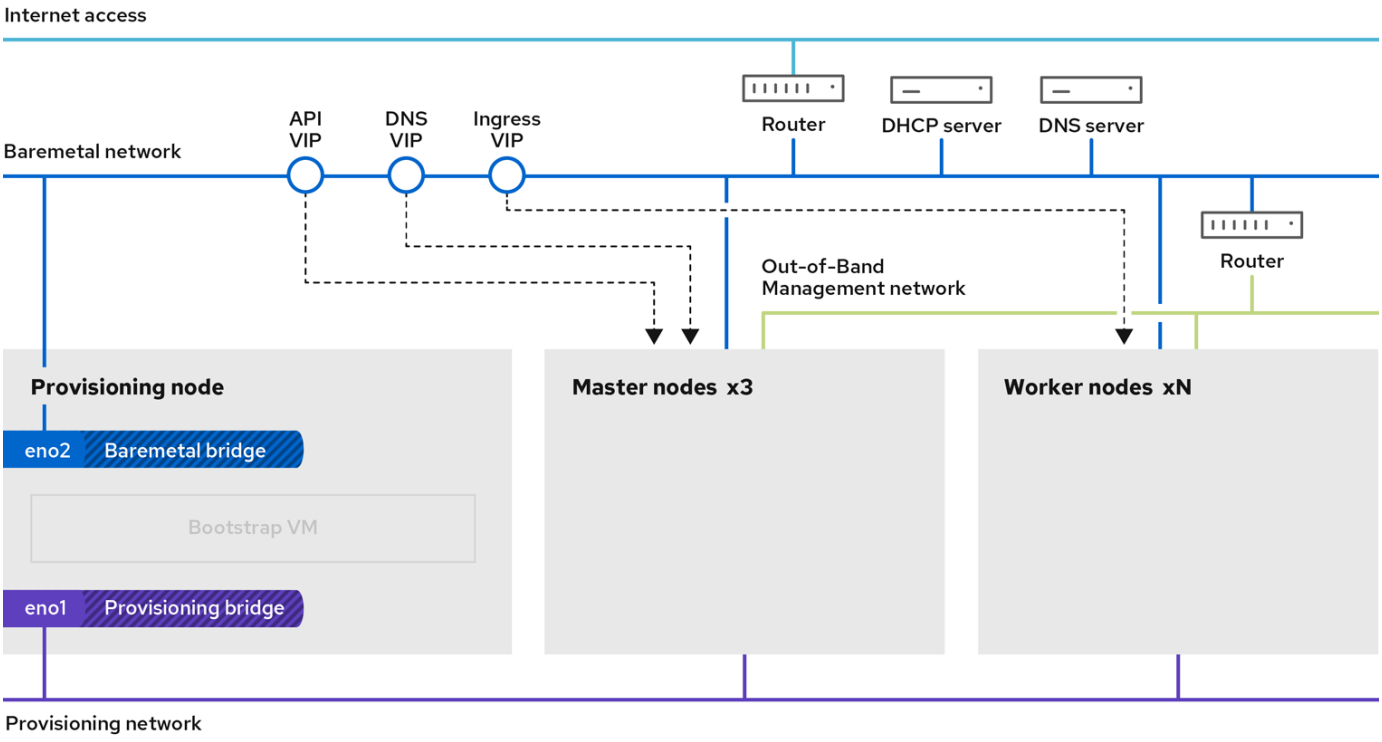
- **Provisioning Network:** This network is used to boot the bare metal nodes and install the necessary images and packages to deploy the OpenShift cluster.
- **Baremetal Network:** This network is used for public facing communication of the cluster once it is deployed.

The setup of the provisioner node will have the customer create bridge interfaces that allow the traffic to route properly on both the node itself, and for the Bootstrap VM which will be provisioned for deployment purposes.

Once the cluster is deployed the API and Ingress VIP addresses are migrated from the bootstrap node to the newly deployed cluster.

The images below display the environment both during IPI deployment, and once 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

While each of these networks were virtually separated by VLANs, each physical port needed to be set up in "access mode" with the primary VLAN assigned, as 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 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.

Next: NetApp Storage Overview.

OpenShift on Red Hat OpenStack Platform: Red Hat OpenShift with NetApp

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

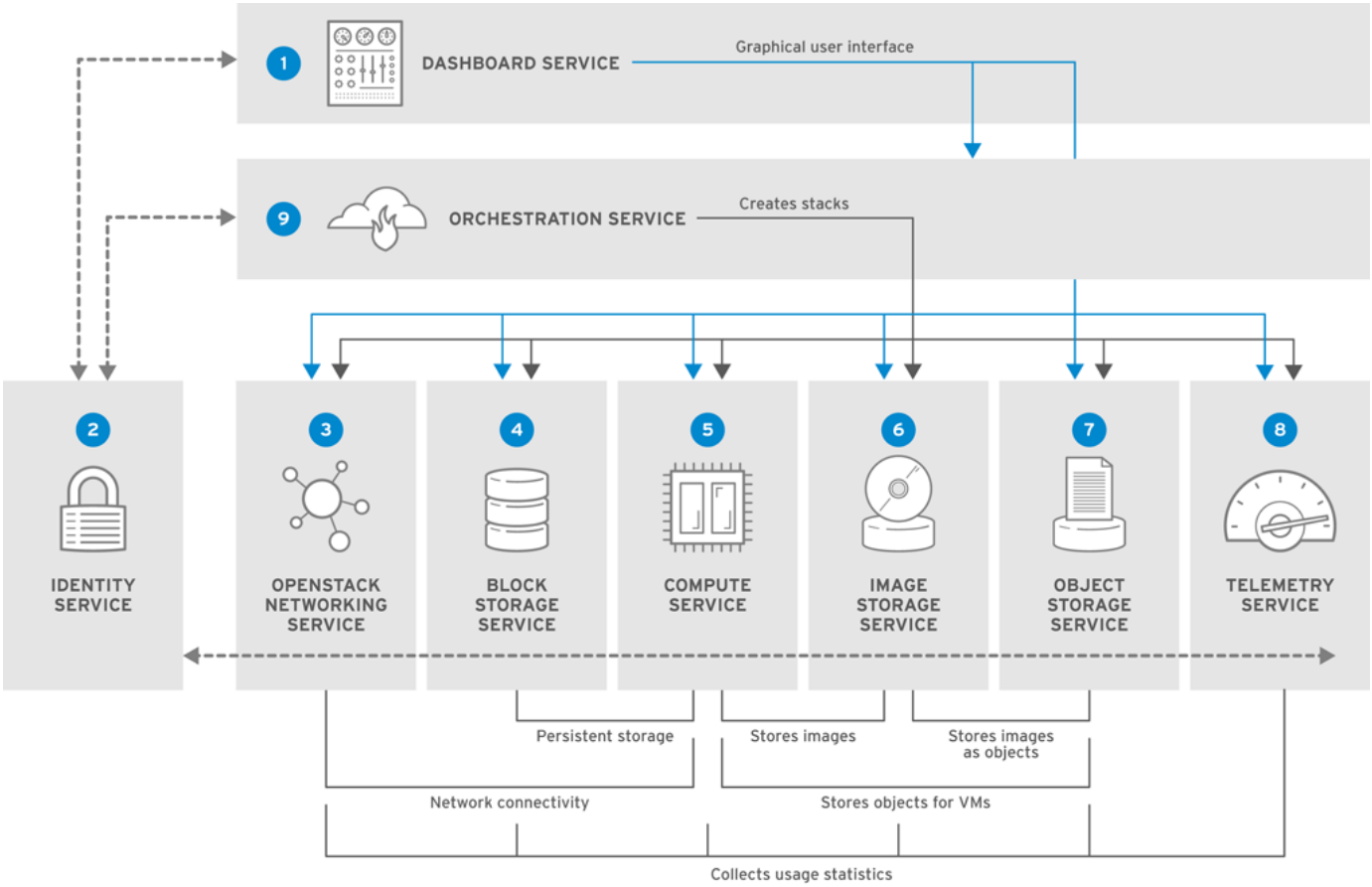
OSP is an infrastructure-as-a-service (IaaS) cloud implemented by a collection of control services that manage compute, storage, and networking resources. The environment is managed using a web-based interface that allows administrators and users to control, provision, and automate OpenStack resources. 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, and 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 backport 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 baremetal 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, and ensuring a fault tolerant configuration where both compute nodes can launch the virtual instances 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 OSP compute 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. In 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, where if there not enough resources to deploy on separate nodes, or not enough resources to allow sharing of nodes, the VM will fail 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 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 OpenStack with Customizations.

Next: NetApp Storage Overview.

OpenShift on Red Hat Virtualization: Red Hat OpenShift with NetApp

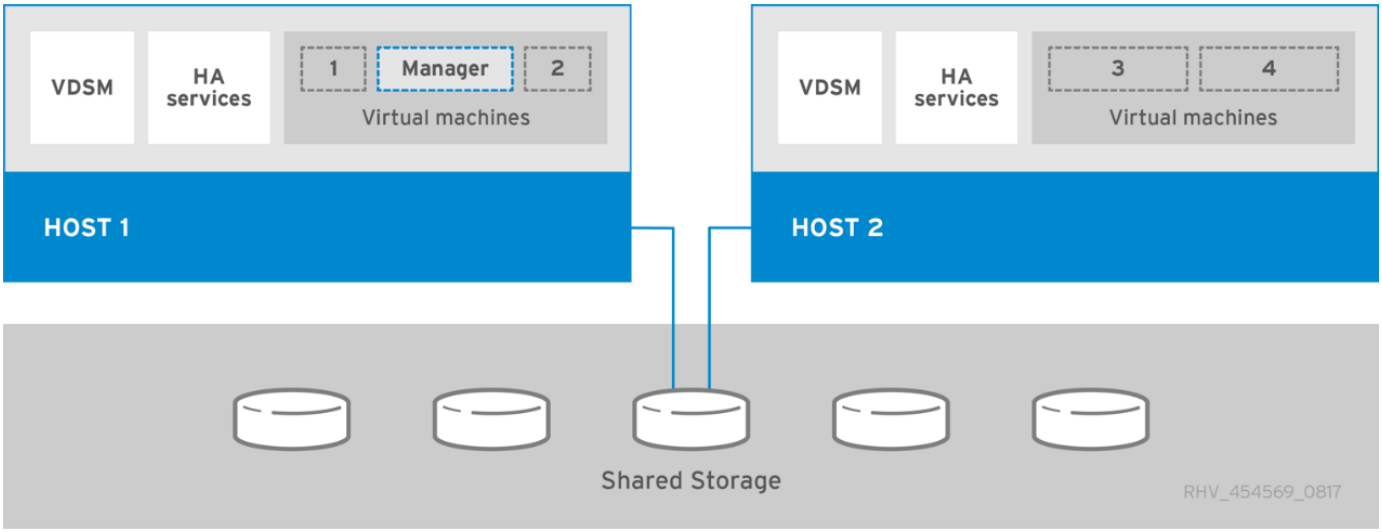
RHV is an enterprise virtual data center platform that runs on Red Hat Enterprise Linux (RHEL) and uses the KVM hypervisor.

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.



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 Virtual Machine logical network on RHV for it's 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

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, ovirtmgmt network	3343
Storage Network	Storage network for NetApp Element iSCSI	3344
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 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.

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/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: NetApp Storage Overview.

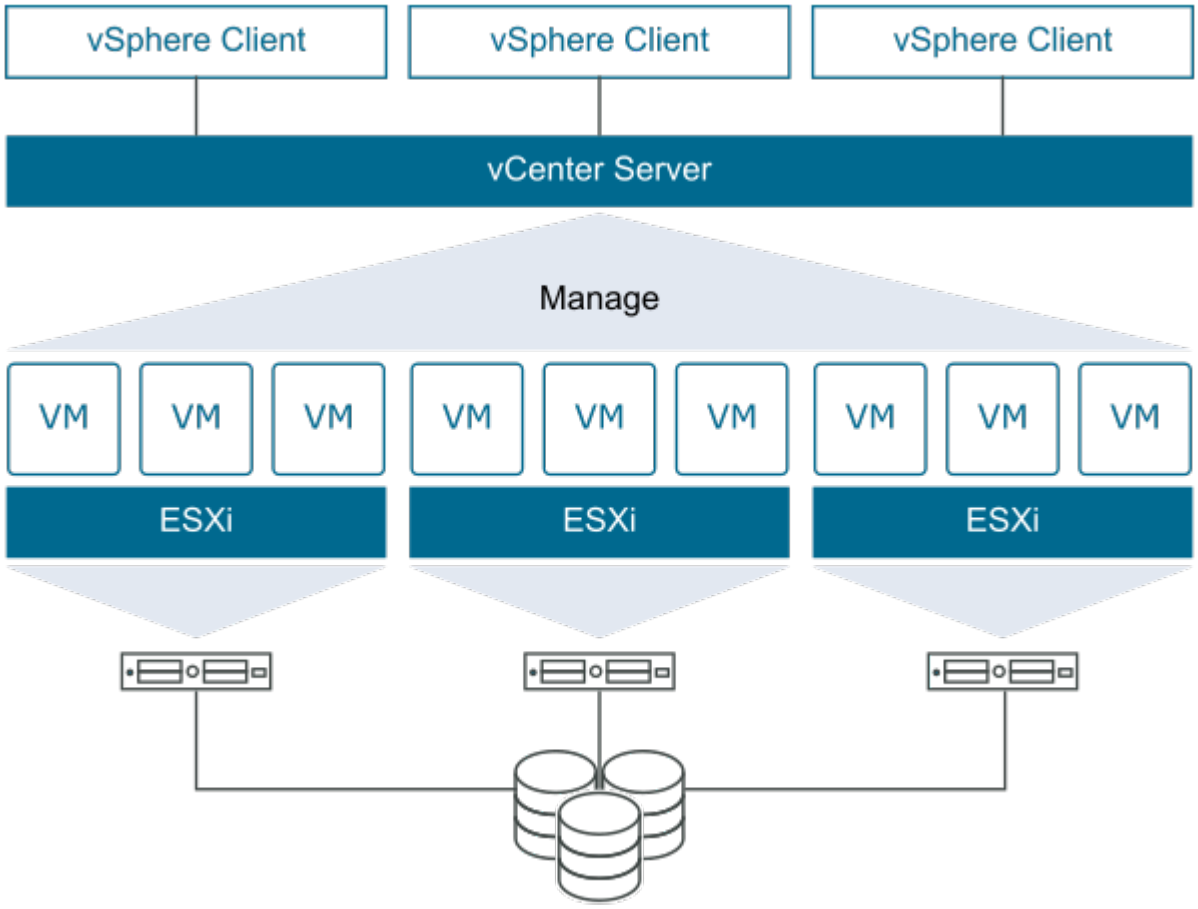
OpenShift on VMware vSphere: Red Hat OpenShift with NetApp

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 provides for the ability to hot migrate virtual machines between nodes in the cluster, upon user request in non-disruptive manner.
- **vSphere High Availability.** In event of host failures, to avoid disruption, 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 VM's it is hosting. Virtual machines with resource contentions can be hot migrated to other nodes in the cluster to ensure that there are enough resources 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 Virtual Machine logical network on VMware vSphere for it's 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

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
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 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.

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, allowing 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, 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 ESXi 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 and host affinity

Ensuring the distribution of the OpenShift masters across multiple hypervisor nodes can be achieved by enabling VM/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

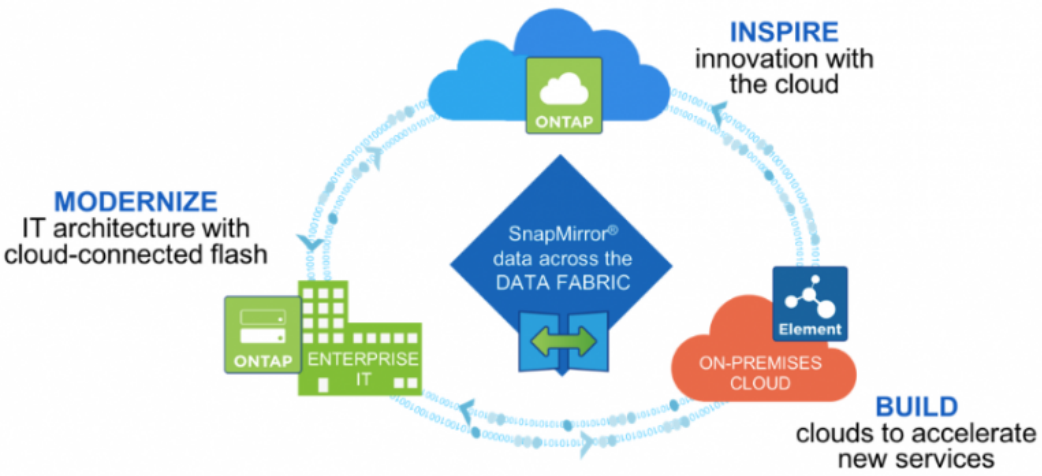
IPI makes the deployment of OpenShift clusters extremely easy through the interactive wizard discussed earlier in this document. However, it is possible that there are some default values that might need to be changed as a part of a cluster deployment.

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 vSphere with Customizations.

Next: NetApp Storage Overview.

NetApp Storage Overview: Red Hat OpenShift with NetApp

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: Red Hat OpenShift with NetApp

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.

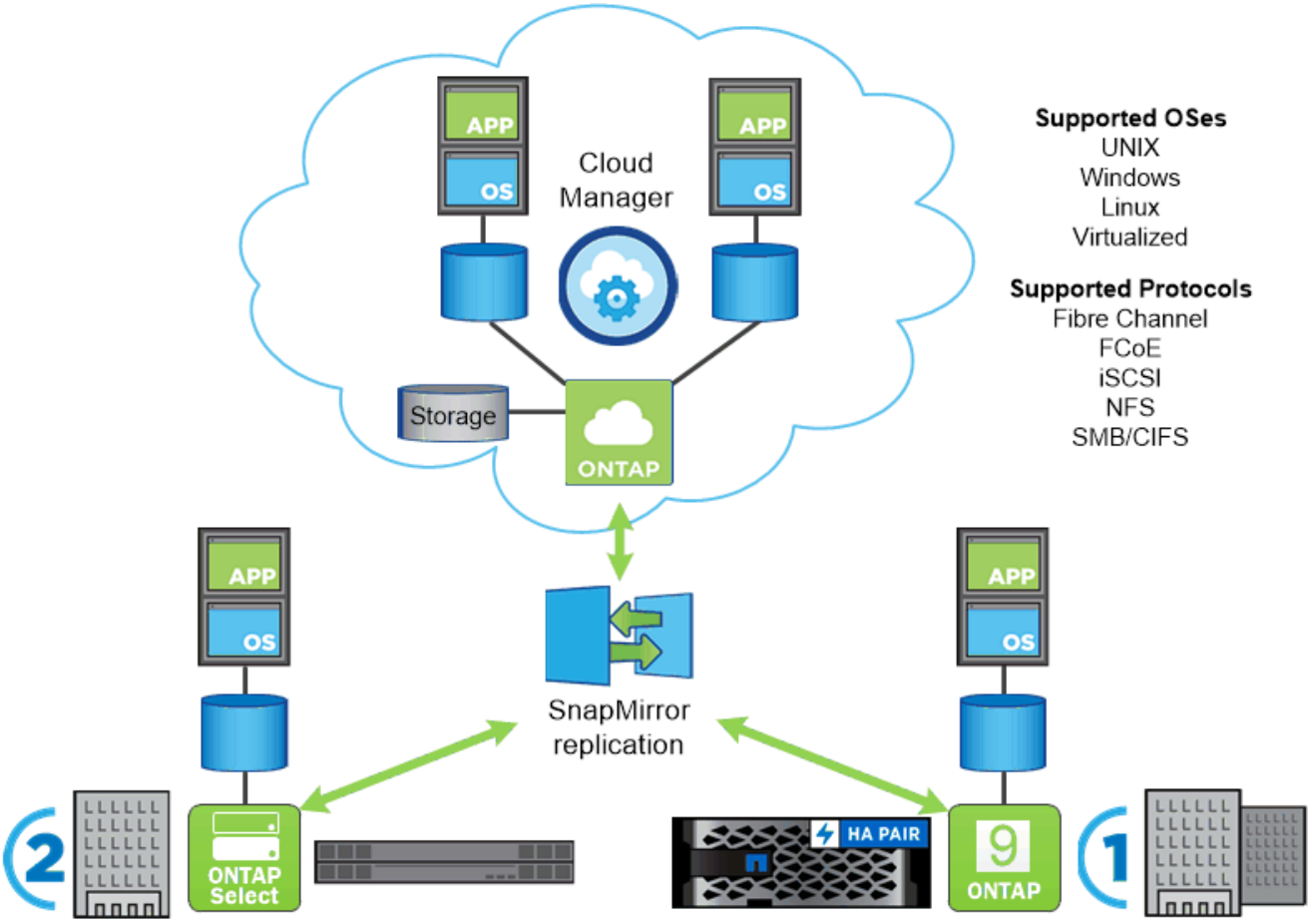
For more information about the NetApp ONTAP storage system, visit the NetApp ONTAP website.

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:

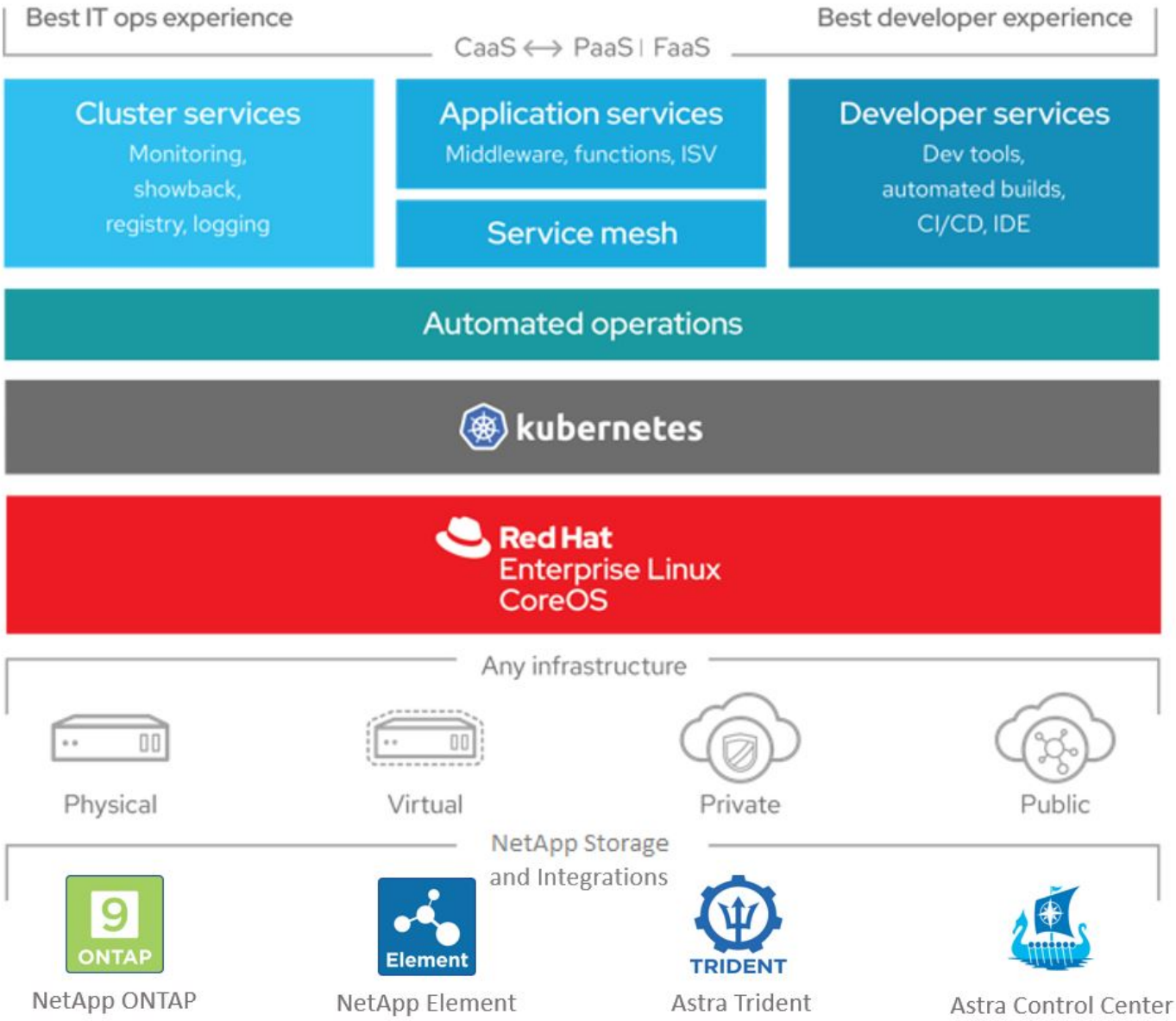
- **Deduplication.** The system only stores unique 4K blocks. Any duplicate 4K blocks are automatically associated to an already stored version of the data. Data is on block drives and is mirrored by using the NetApp Element software Helix data protection. This system significantly reduces capacity consumption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

NetApp provides a number of products which assist our customers with orchestrating and managing persistent data in container based environments, like 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's trusted data protection technology.

Astra Control Service is available to support stateful workloads in cloud-native Kubernetes deployments.

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.

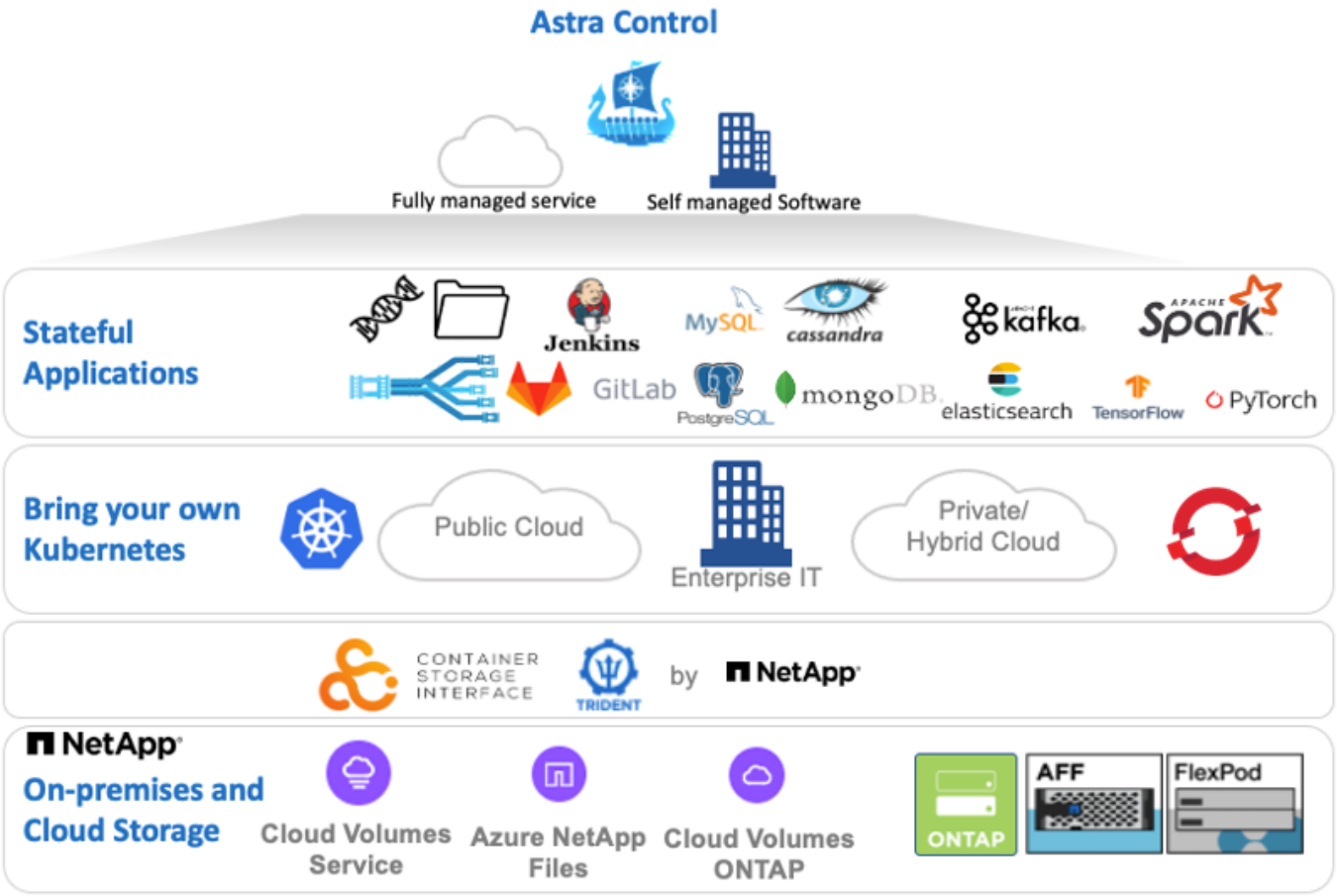
The following pages have additional information about the NetApp products that have been validated for application and persistent storage management in the Red Hat OpenShift with NetApp solution:

- NetApp Astra Control Center
- NetApp Astra Trident

Next: NetApp Astra Control Center Overview

NetApp Astra Control Center Overview: Red Hat OpenShift with NetApp

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, powered by NetApp's trusted data protection technology.



NetApp Astra Control Center can be installed to a Red Hat OpenShift cluster which has Astra Trident storage orchestrator deployed, 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 the section in 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 (7-days worth of metrics) monitoring and telemetry will be available and also exported to Kubernetes native monitoring tools (Prometheus/Grafana) through open metrics endpoints.

Astra Control Center is fully integrated into the NetApp AutoSupport/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 will be made available to customers. The evaluation version will be 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 https://cloud.netapp.com/astra.

Astra Control Center Installation Prerequisites

1. One or more Red Hat OpenShift clusters.

   Currently versions 4.6 EUS and 4.7 are supported.

2. Astra Trident must be already 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 would 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.

   Refer to 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.

   Refer 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 or Podman installations must be at least version 20.10.

Install Astra Control Center

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

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

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

```
[netapp-user@rhel7 ~]$ tar -vxzf astra-control-center-21.08.57.tar.gz
[netapp-user@rhel7 ~]$ cd astra-control-center-21.08.57
```

3. Before we start the installation, we need to 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
```

c. Create a shell script file and paste the below 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 --tls
-verify=false astra-registry.apps.ocp-vmw.cie.netapp.com
```

c. Create a shell script file and paste the below 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
```

If you are using untrusted certificates for your registry, edit the shell script and use --tls-verify=false for the docker push command - `docker 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
```

4. The next step is to upload the image registry TLS certificates to the OpenShift nodes. For that, create a configmap in 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 OpenShift internal registry with default TLS certificates from the ingress operator with a route, you will still need to follow the above 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 `acc-operator-system` for installing the Astra Control Center Operator.

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

6. Create a secret with credentials to log into the image registry in `acc-operator-system` 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 acc-operator-system
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 `[your.registry.goes.here]` and provide the name of the secret we just created. 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: acc-operator-system
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-registry.apps.ocp-vmw.cie.netapp.com/netapp-
astra/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:21.05.68
        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: 100m
            memory: 150Mi
          requests:
            cpu: 100m
            memory: 50Mi
        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 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. Next step is to edit the Astra Control Center CRD file `astra_control_center_min.yaml` and fill 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:
  astraVersion: "21.08.57"
  astraAddress: "astra-control-center.cie.netapp.com"
  autoSupport:
    enrolled: true
  email: "solutions_tme@netapp.com"
  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 above 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 like defining storageclass other than default for creating PVCs or providing SMTP details for mail notifications, you can edit the file `astra_control_center.yaml`, fill those details and use it to create the CRD.

Installation Verificaton

1. It might take several minutes for the installation to complete. Verify that all the pods and services in 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
acc-operator-system -c manager -f
```

The following message should be displayed to indicate the successful installation of Astra Control Center

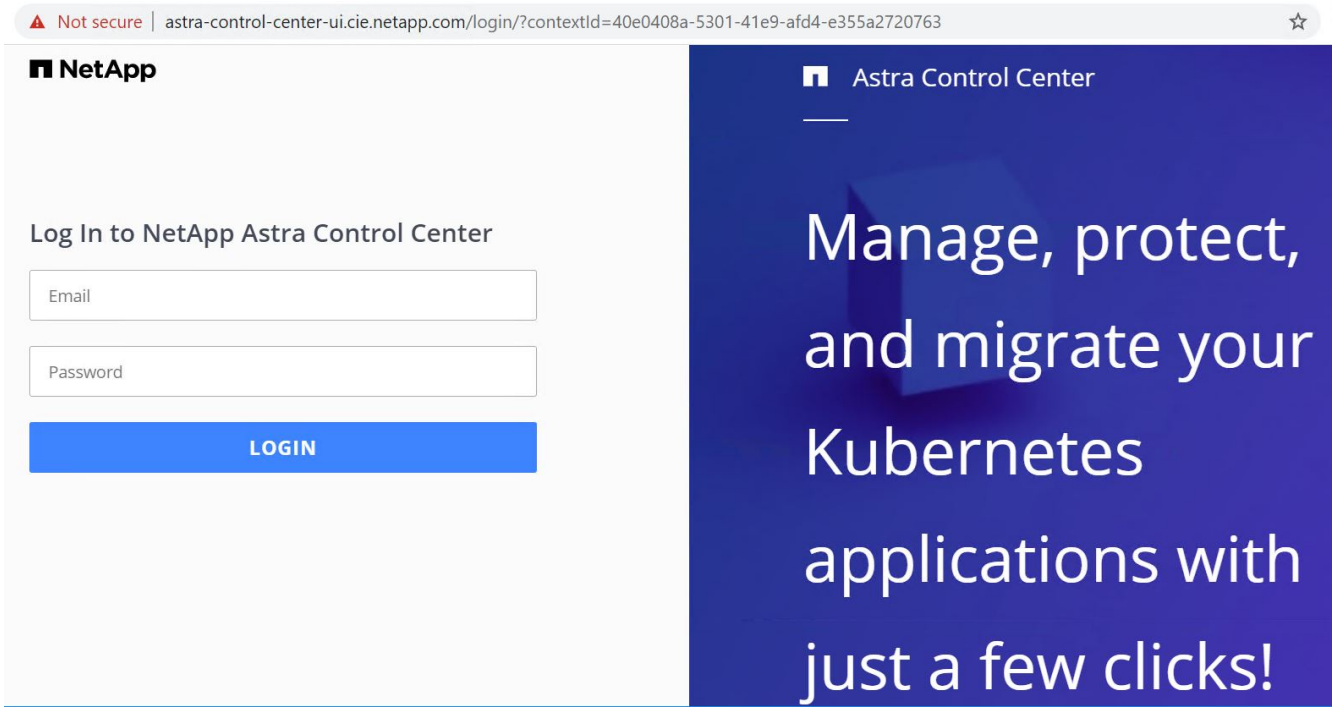
```
{"level":"info","ts":1624054318.029971,"logger":"controllers.AstraControlCenter","msg":"Successfully Reconciled AstraControlCenter in [seconds]s","AstraControlCenter":"netapp-astra-cc/astra","ae.Version":"[21.08.57]"}
```

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 –

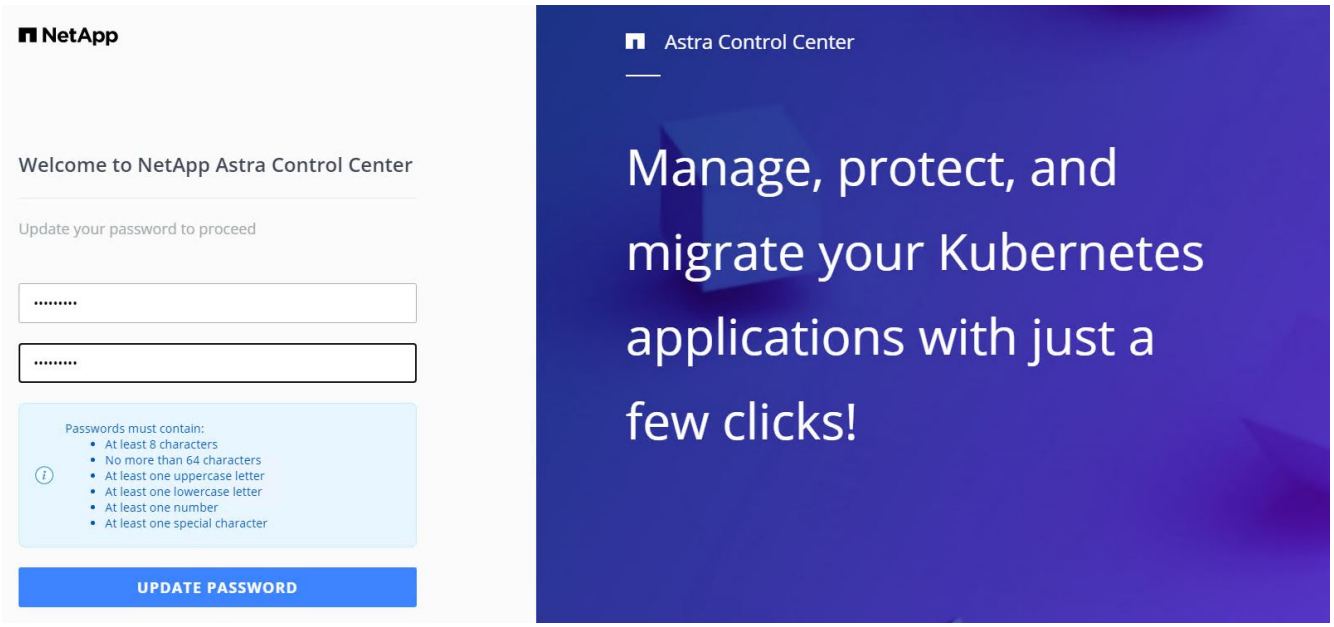
```
[netapp-user@rhel7 ~]$ oc get astracontrolcenters -n netapp-astra-cc
NAME    UUID
astra   345c55a5-bf2e-21f0-84b8-b6f2bce5e95f
```

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

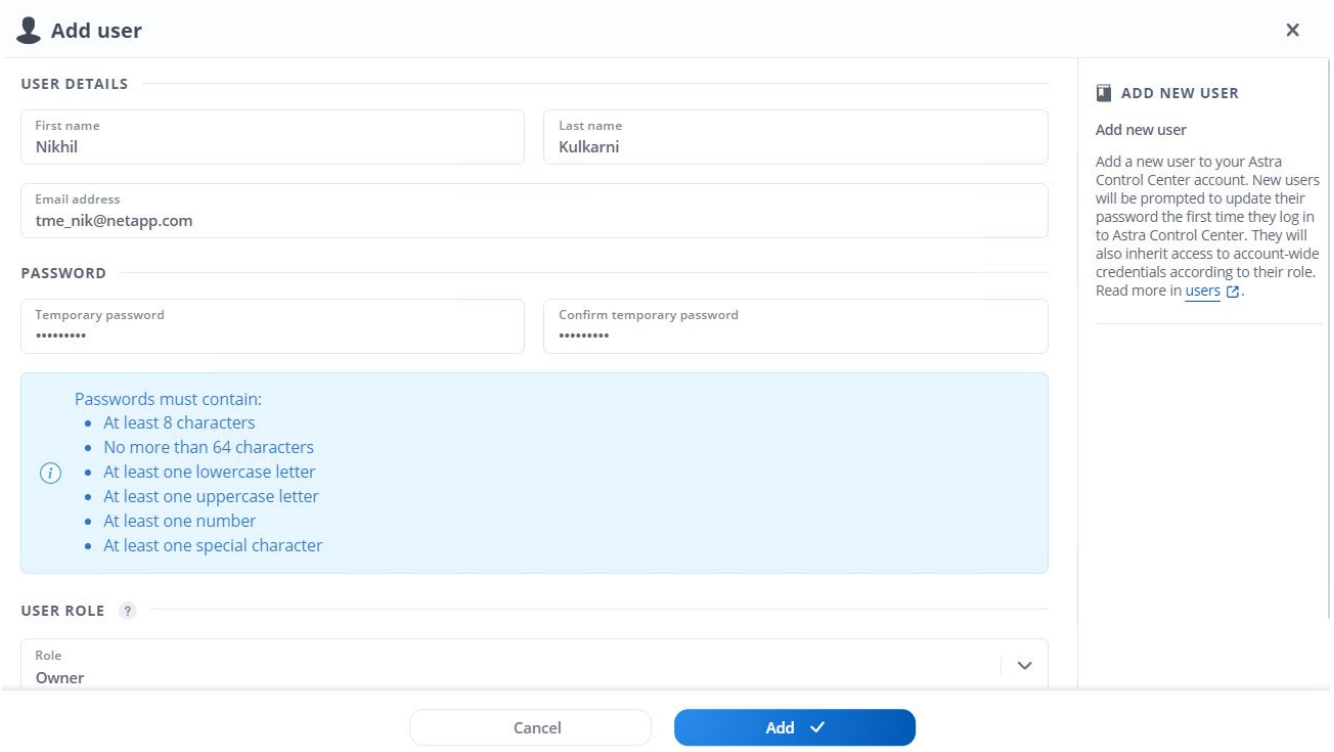
4. Now log into the Astra Control Center GUI by browsing to the FQDN you provided in the CRD file.



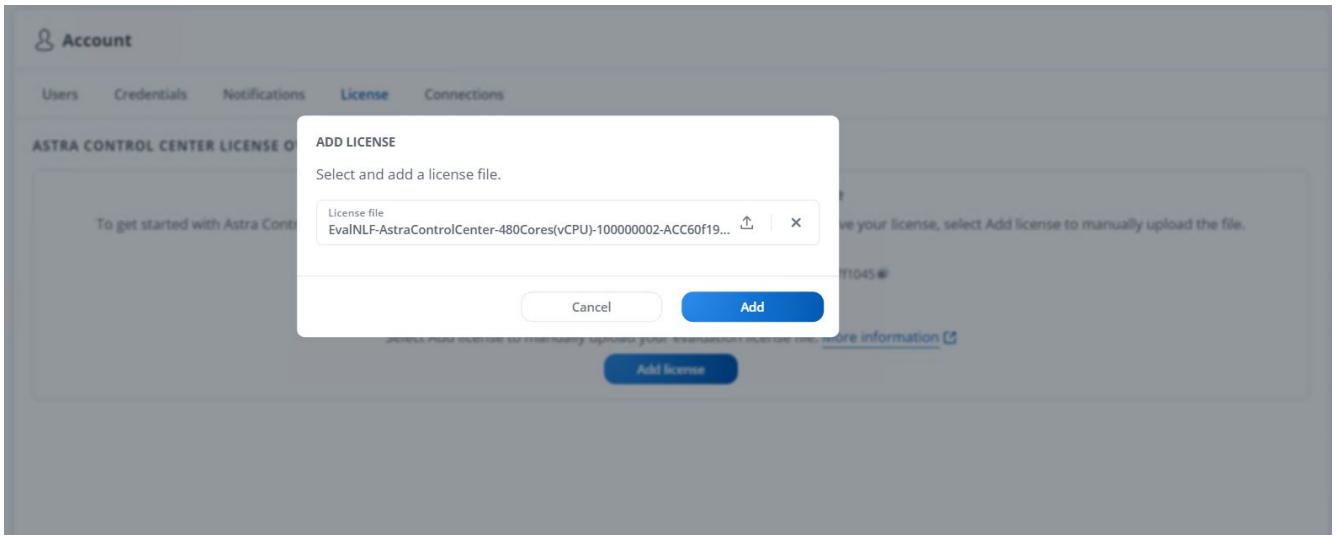
5. When you log into Astra Control Center GUI for the first time using the admin email address provided in CRD, you will need to change the password.



6. If you wish to add a user to Astra Control Center, go to `Account` → `Users` and click on `Add` and enter the details of the user and click `Add`.



7. Astra Control Center requires a license for all of it's functionalities to work. To add a license, go to `Account` → `License`, click on `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.

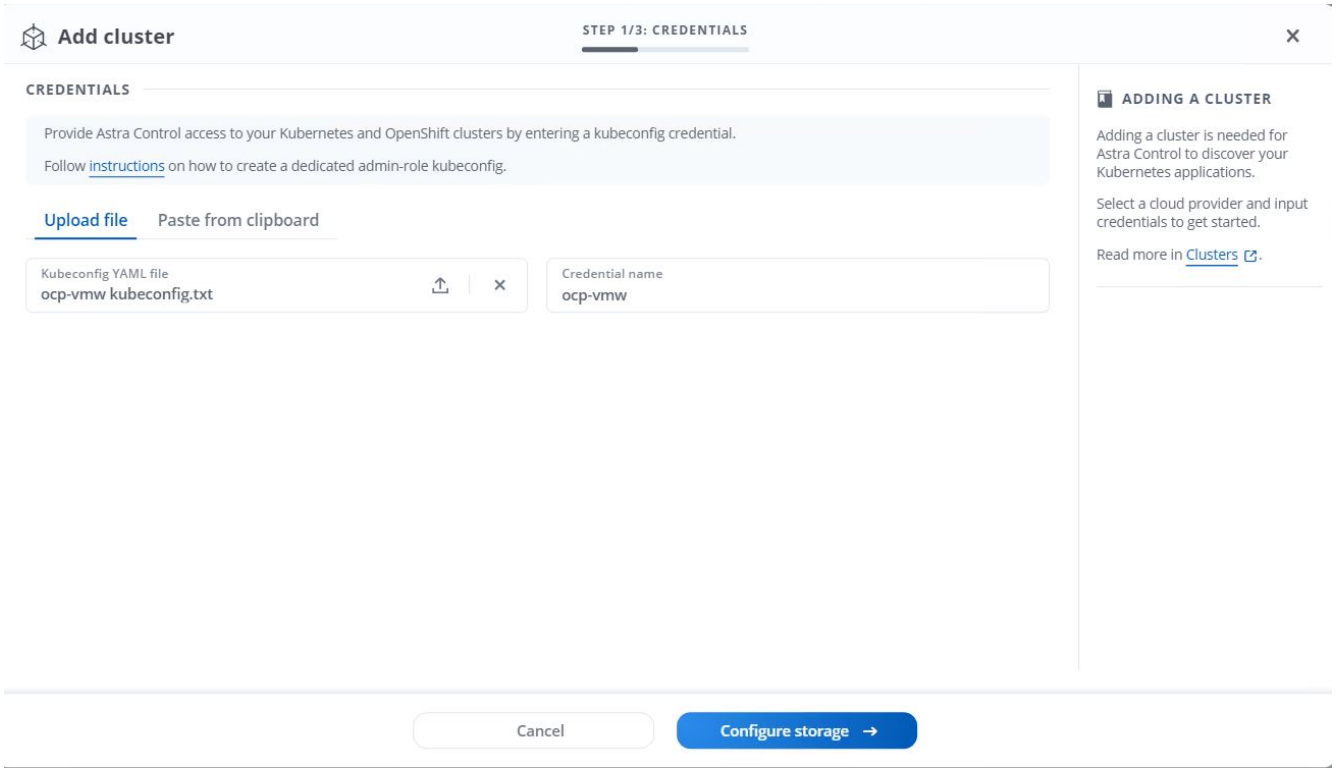
Next: Register your Red Hat OpenShift Clusters: Red Hat OpenShift with NetApp.

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.

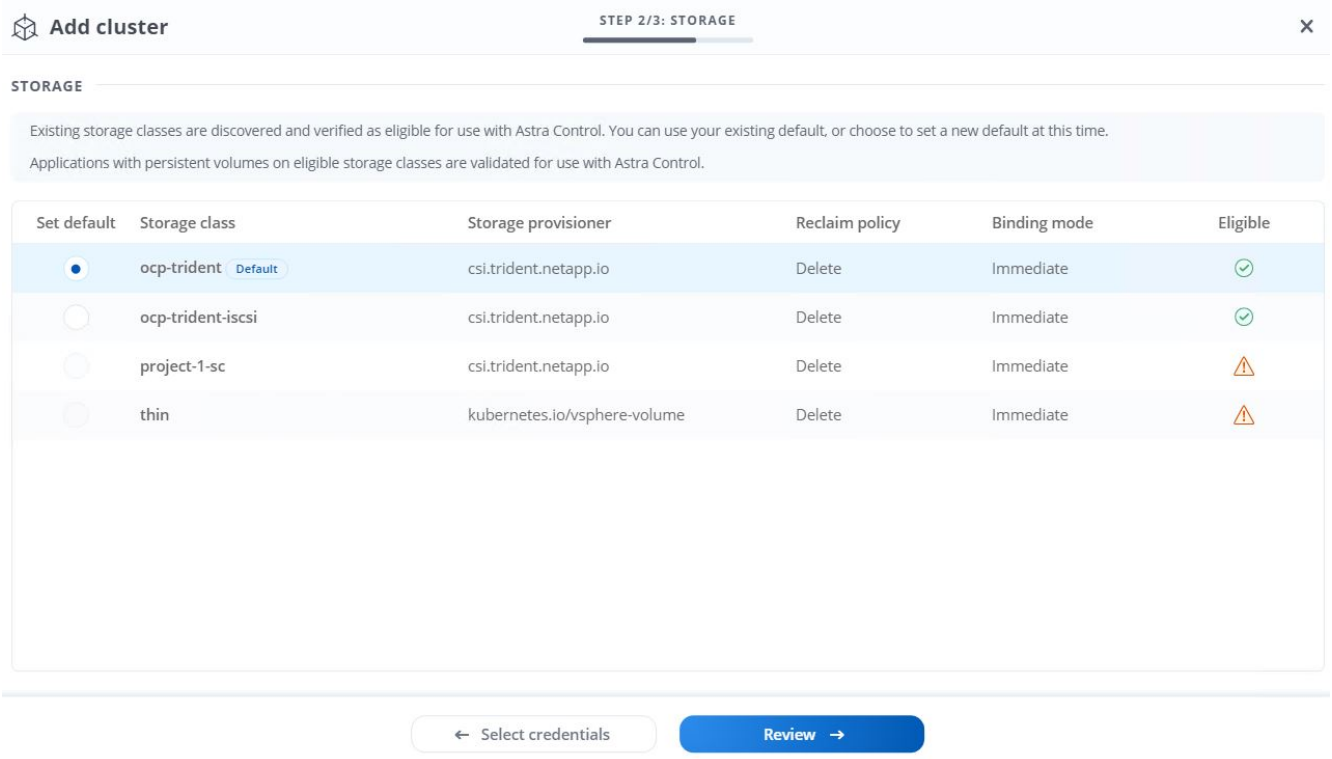
Register Red Hat OpenShift clusters

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.

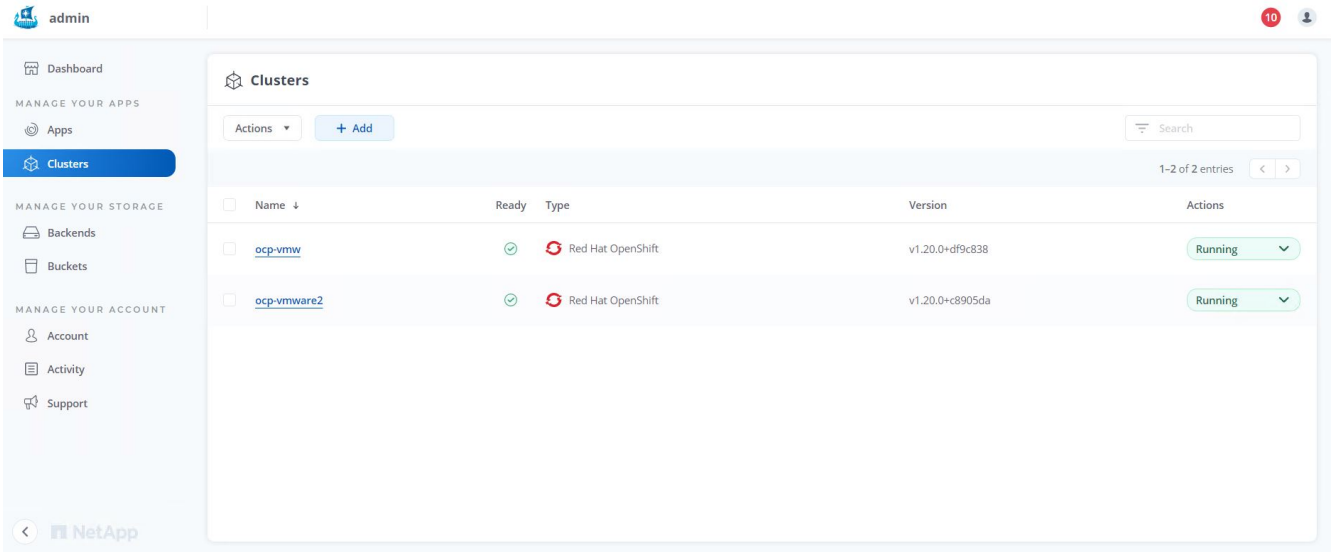


The kubeconfig file can be generated to authenticate with a username and password or a token. Tokens expire after a limited amount of time and might leave the registered cluster unreachable. NetApp recommends using a kubeconfig file with a username and password to register your OpenShift clusters to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

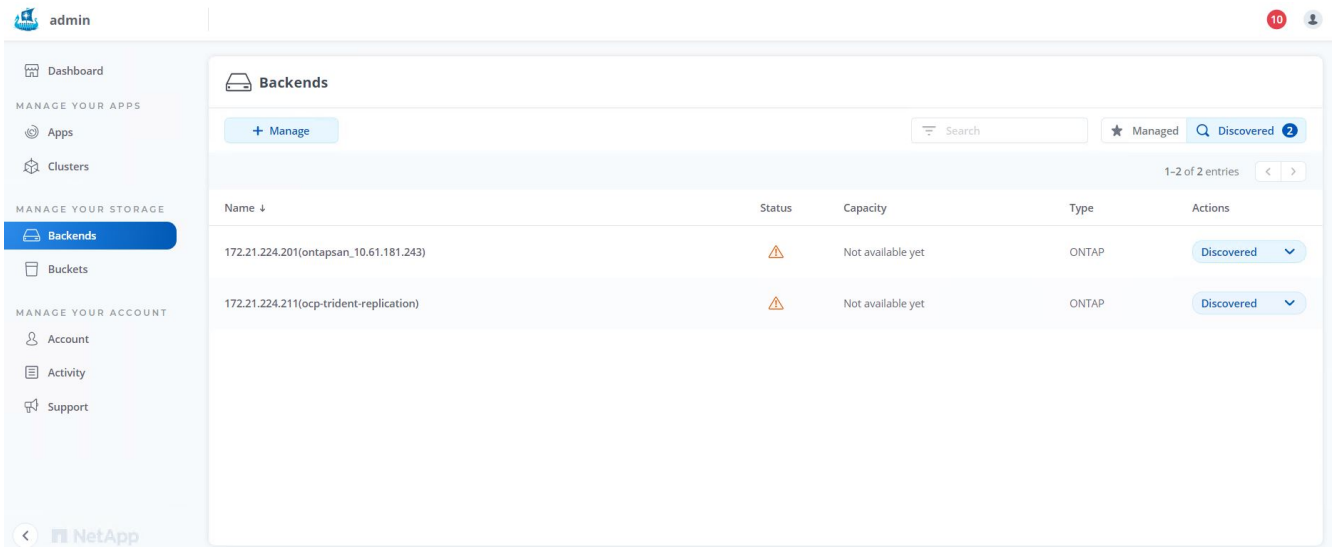


3. Register both OpenShift clusters as described in step 1. When added, the clusters move to the Discovering status while Astra Control Center inspects them and installs the necessary agents. Cluster status changes to Running after they are successfully registered.

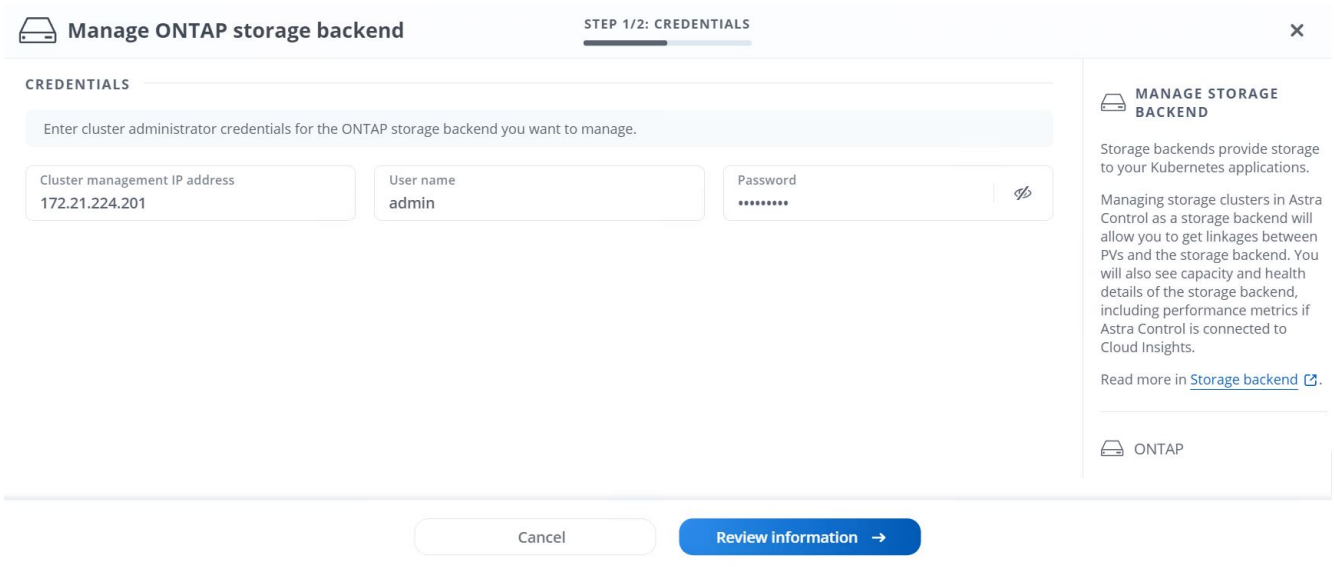


All Red Hat OpenShift clusters to be managed by Astra Control Center should have access to the image registry that was used for its installation as the agents installed on the managed clusters will pull the images from that registry.

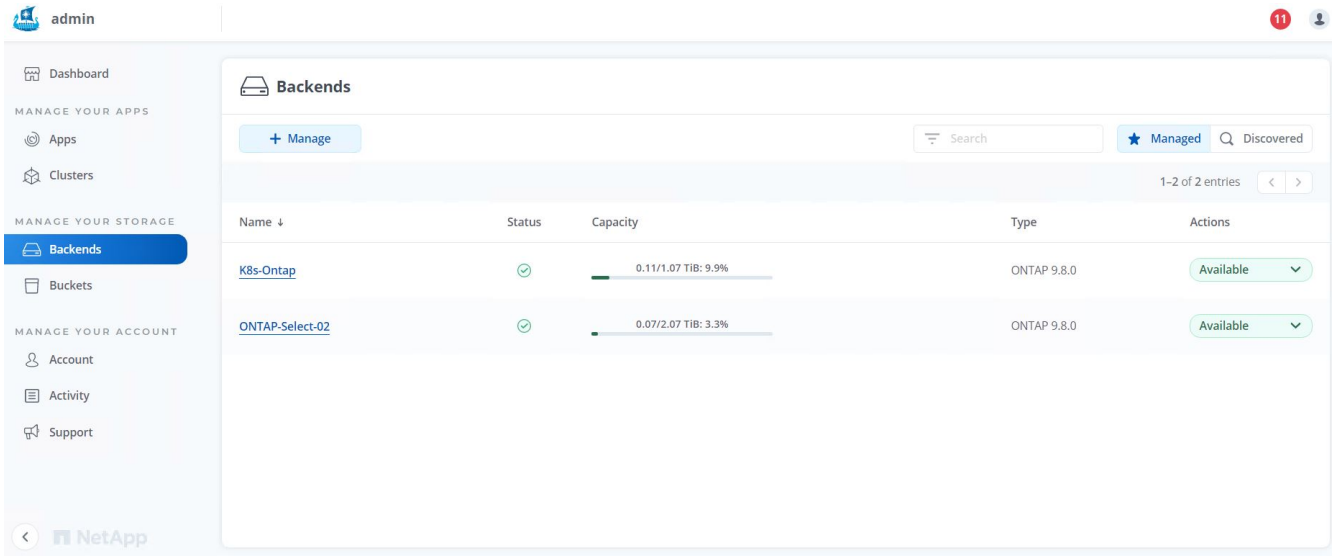
4. Import ONTAP clusters as storage resources to be managed as backends by Astra Control Center. When OpenShift clusters are added to Astra and a storageclass is configured, it automatically discovers and inspects the ONTAP cluster backing the storageclass but does not import it into the Astra Control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



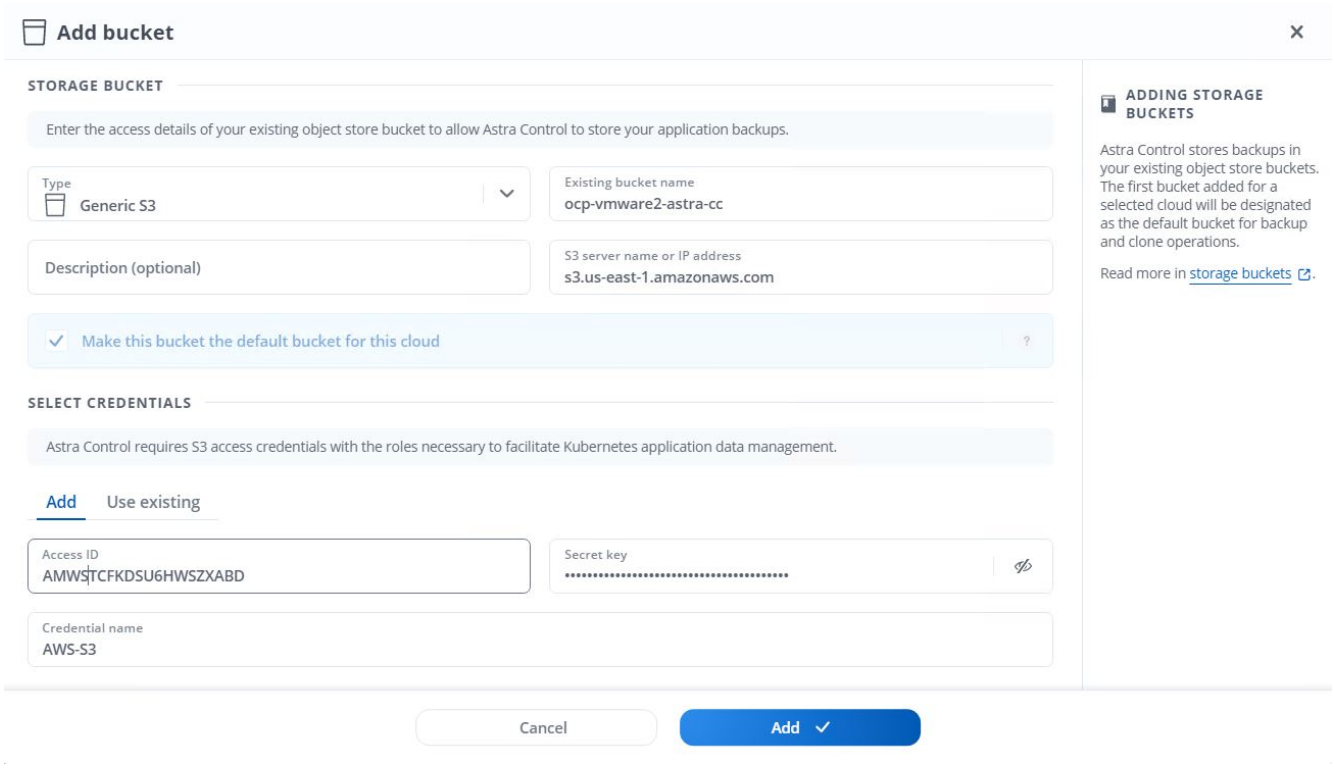
5. To import the ONTAP clusters, go to Backends, and click the dropdown, and select Manage next to the ONTAP cluster to be managed. Enter the ONTAP cluster credentials, click Review Information, and then click Import Storage Backend.



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.



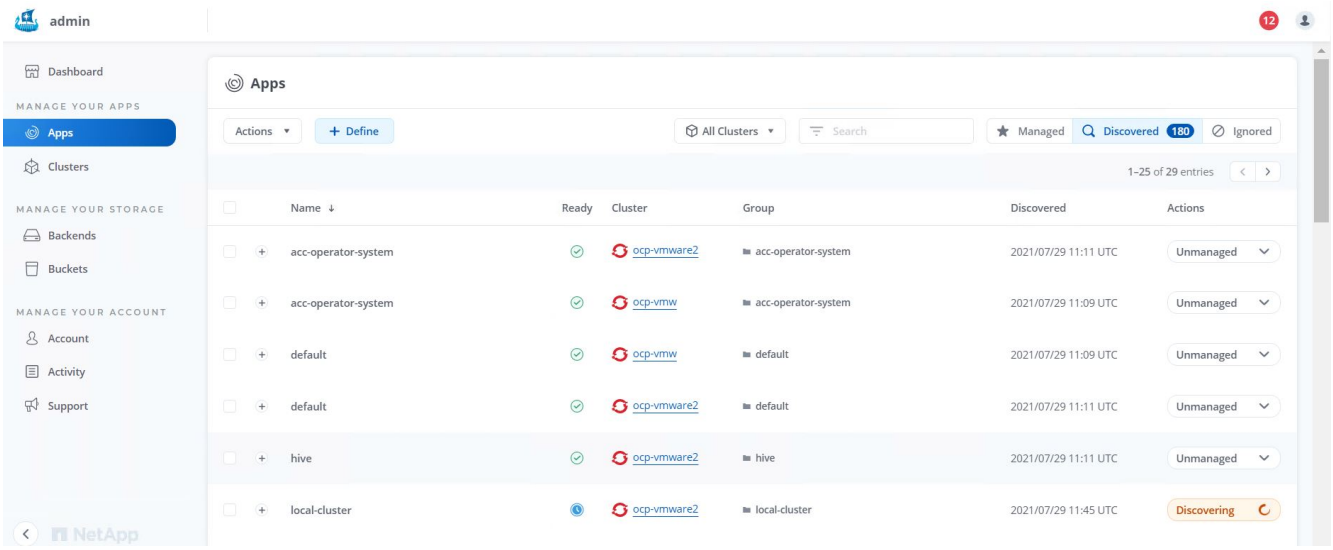
Next: Choose the Applications To Protect: Red Hat OpenShift with NetApp.

Choose the applications to protect

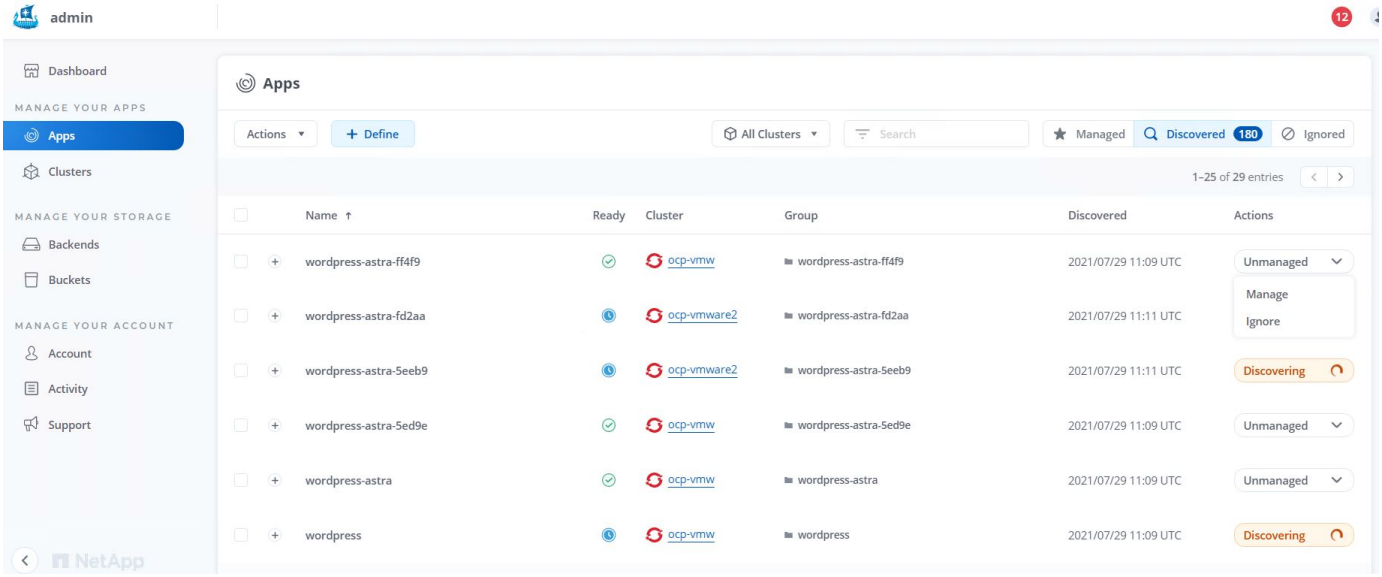
After you have registered your Red Hat OpenShift clusters, you can discover the applications that are deployed and manage them via the Astra Control Center.

Manage applications

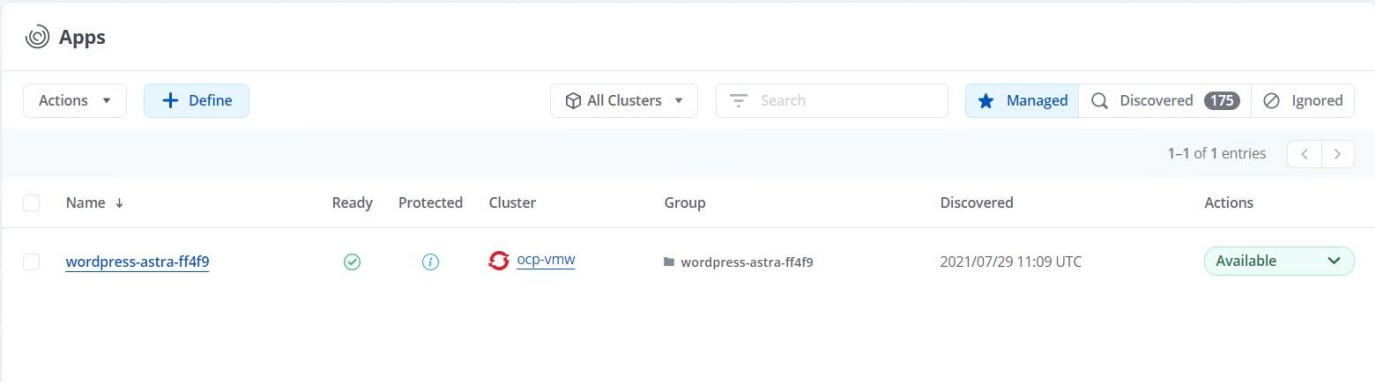
1. After the OpenShift clusters and ONTAP backends are registered with the Astra Control Center, the control center automatically starts discovering the applications in all the namespaces that are using the storageclass configured with the specified ONTAP backend.



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: Red Hat OpenShift with NetApp.

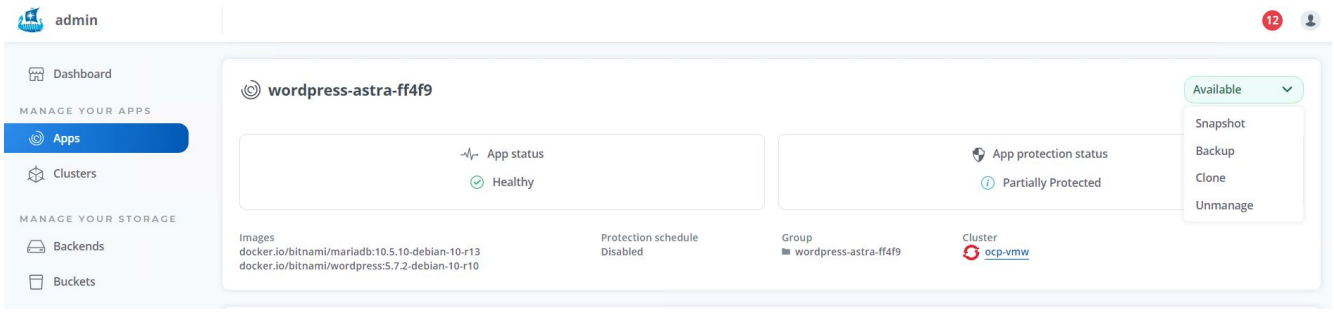
Protect your applications

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

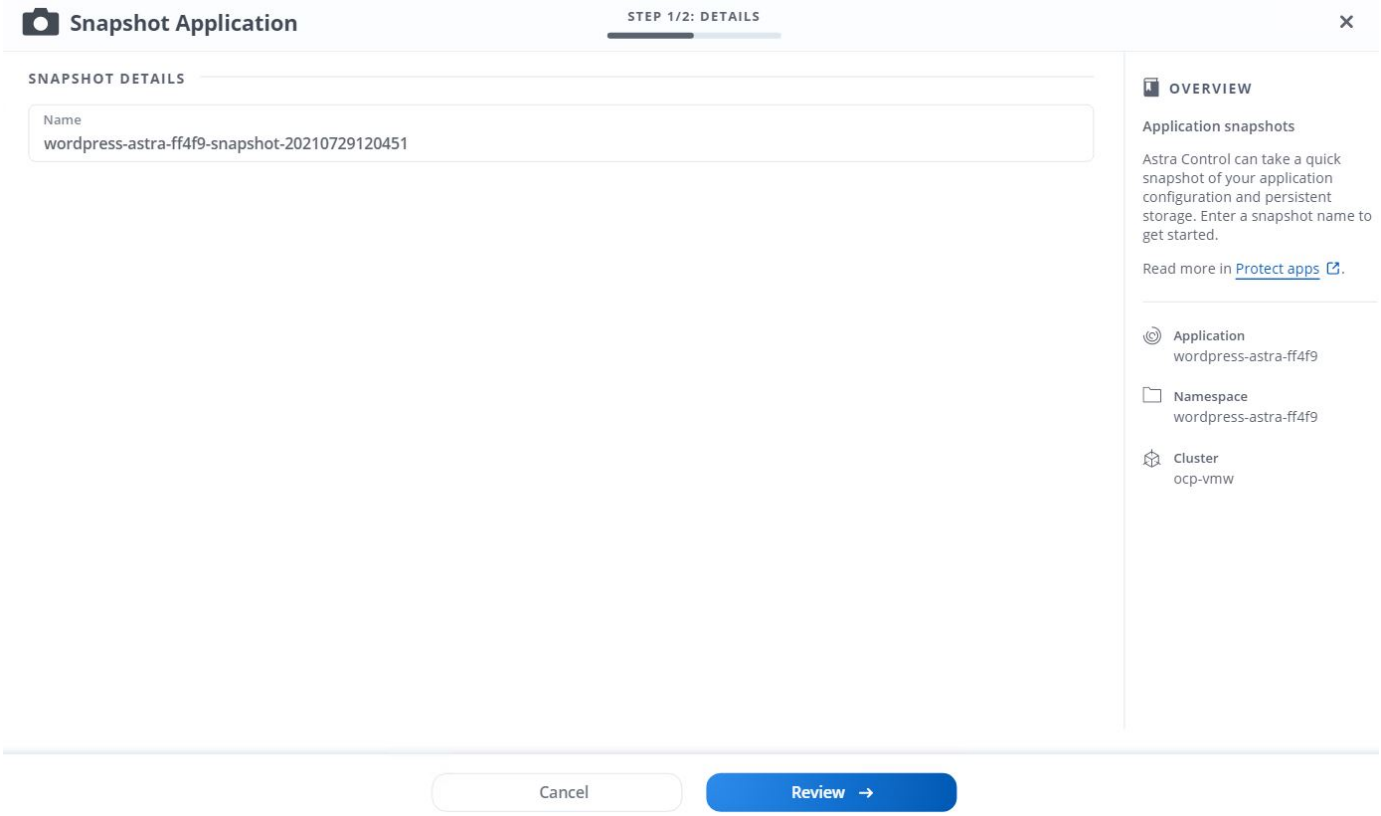
Creating an application snapshot

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.



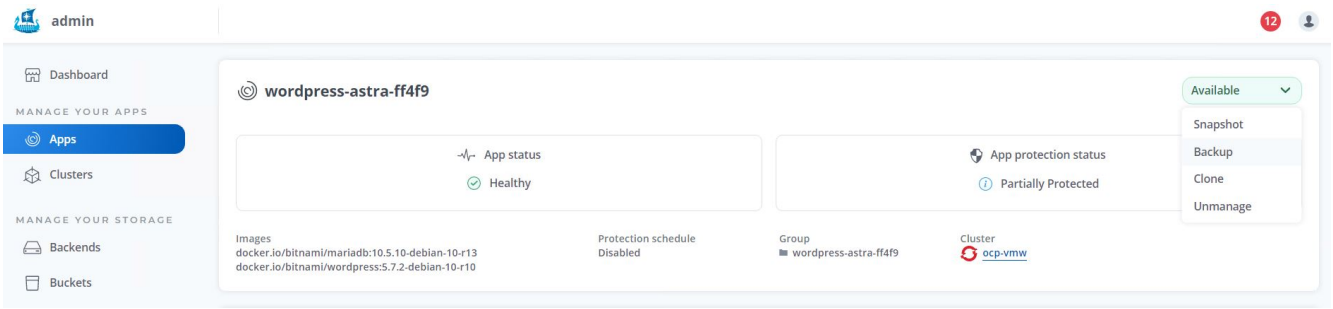
Creating an application backup

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.

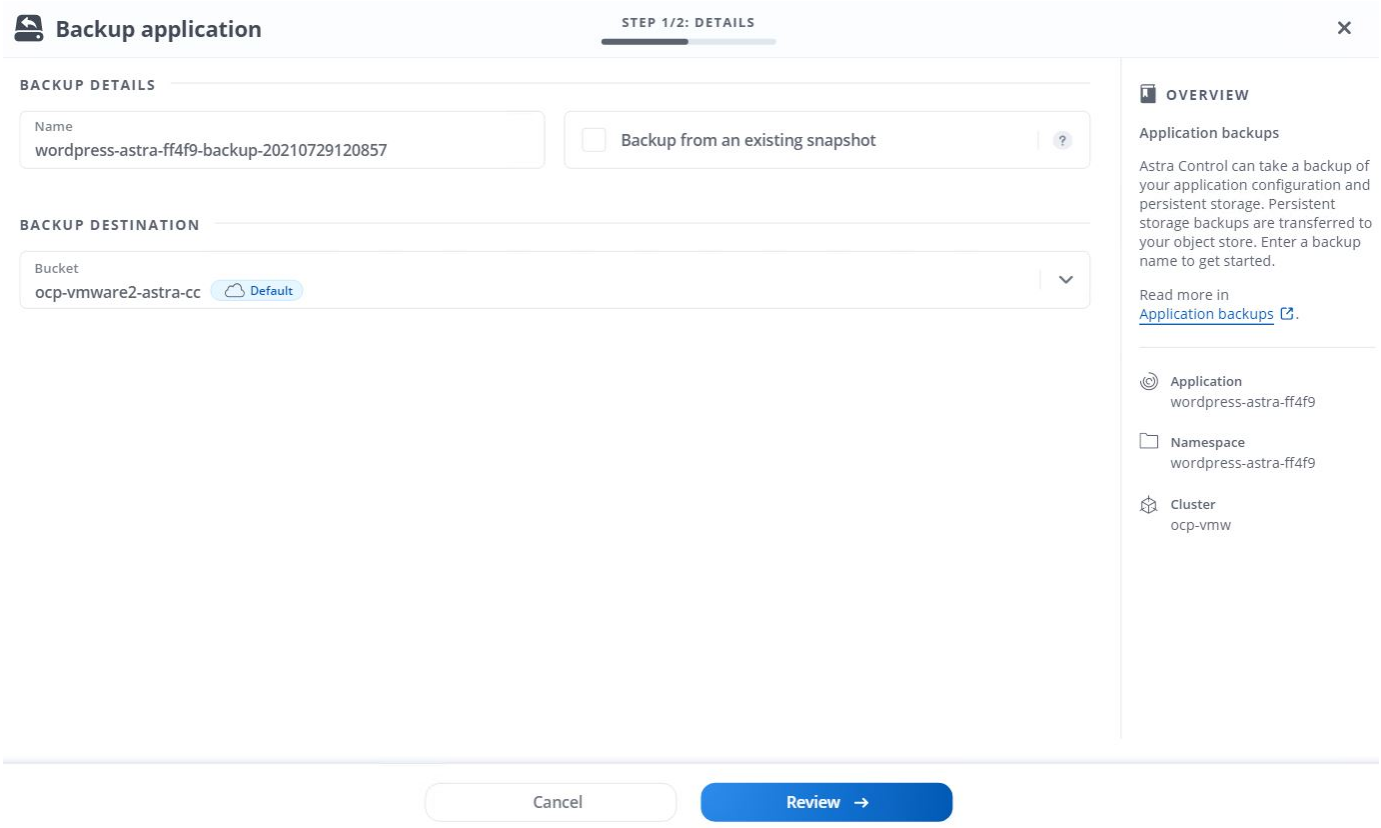
For the backup and restore of managed applications in the Astra Control Center, you must configure superuser settings for the backing ONTAP systems as a prerequisite. To do so, enter the following commands.

```
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.

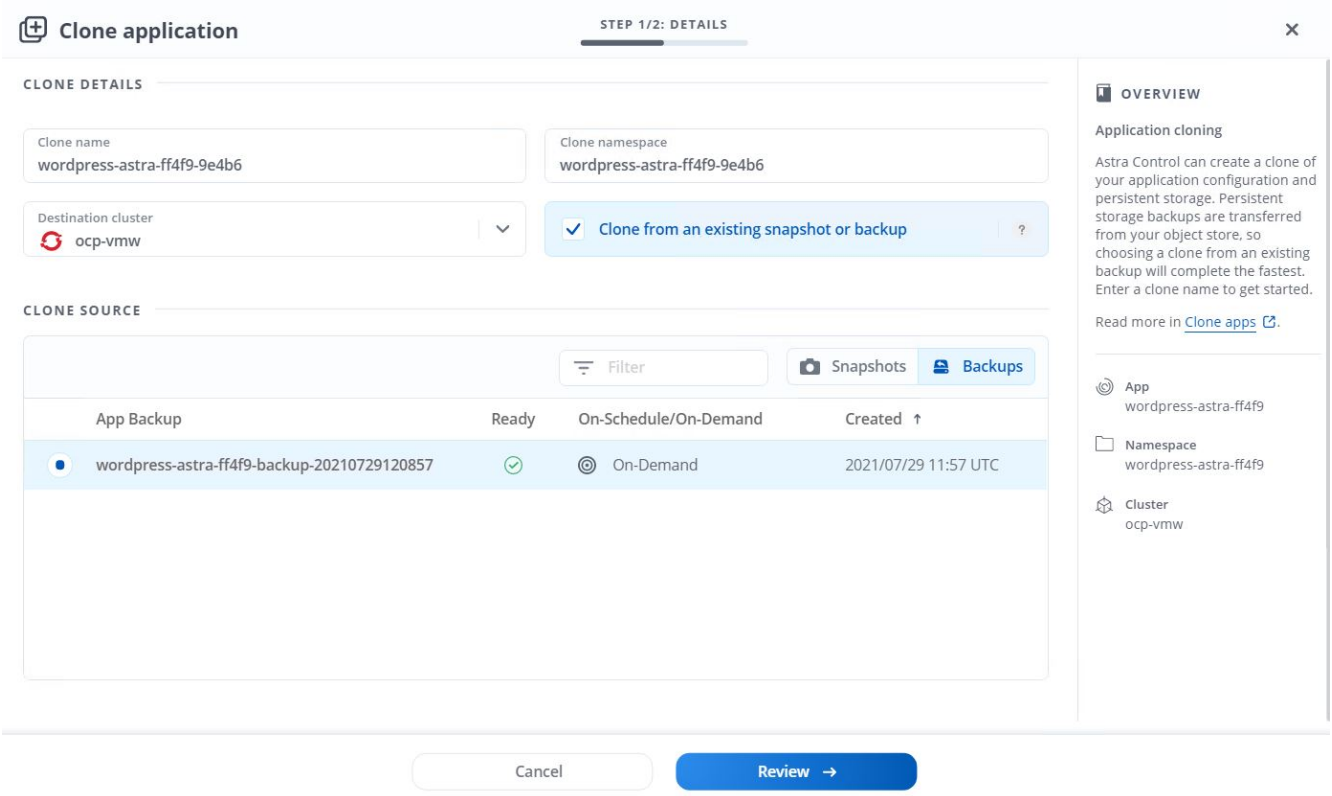


Restoring or cloning an application

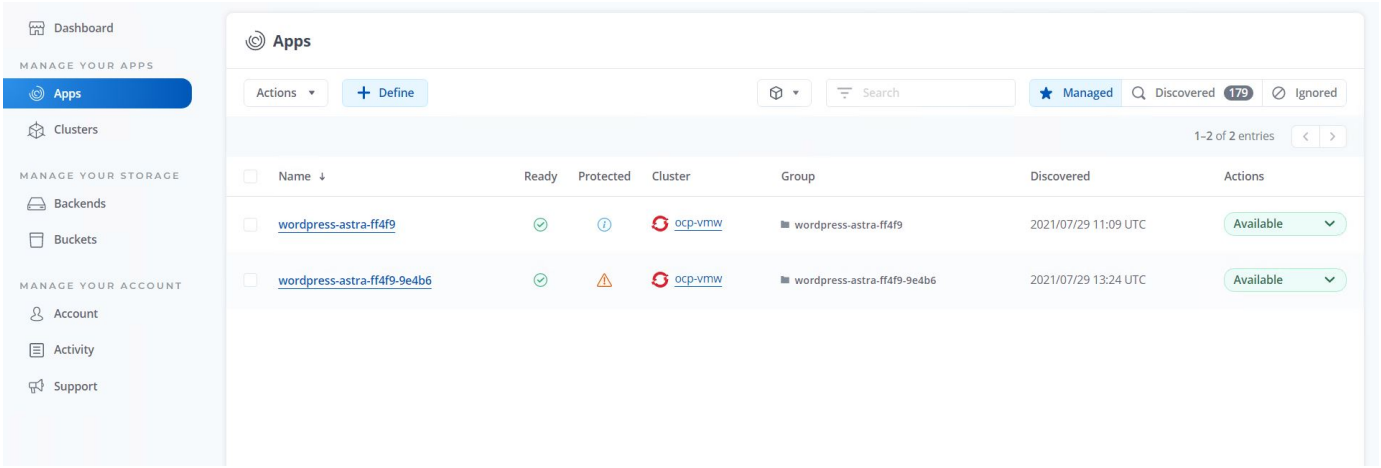
At the push of a button, you can restore an application to the originating cluster or clone it to a remote cluster for dev/test or application protection and disaster recovery purposes.

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: Red Hat OpenShift with NetApp.

Astra Trident Overview: Red Hat OpenShift with NetApp

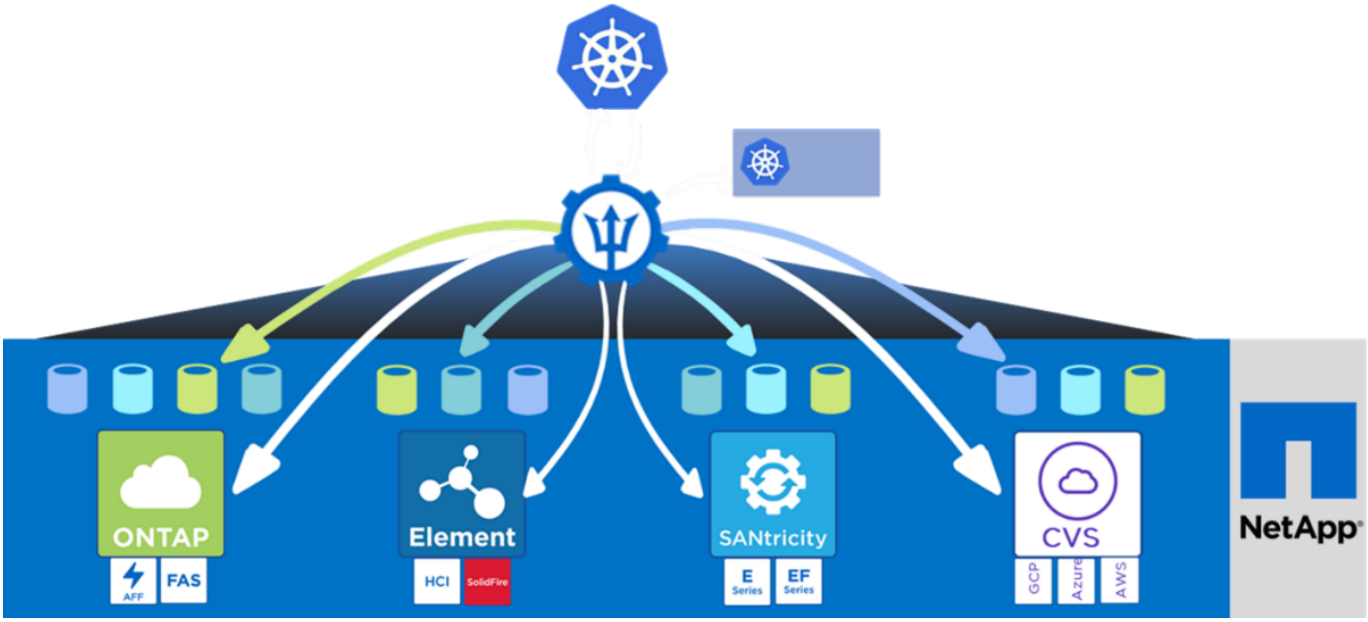
Astra Trident is an open-source and fully-supported storage orchestrator for containers and Kubernetes distributions, including Red Hat OpenShift.

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 21.04 released in April 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 21.04, which can be downloaded here.

```
[netapp-user@rhel7 ~]$ wget
https://github.com/NetApp/trident/releases/download/v21.04/trident-
installer-21.04.tar.gz
--2021-05-06 15:17:30--
https://github.com/NetApp/trident/releases/download/v21.04/trident-
installer-21.04.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-
21.04.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-
21.04.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-21.04.tar.gz'

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

2021-05-06 15:17:30 (88.5 MB/s) - 'trident-installer-21.04.tar.gz' saved
[38349341/38349341]
```

2. Extract the Trident install from the downloaded bundle.

```
[netapp-user@rhel7 ~]$ tar -xzf trident-installer-21.04.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-21.04.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 |
+----------------+----------------+
| 21.04          | 21.04          |
+----------------+----------------+
```

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:21.04
    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:21.04
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v21.04
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 |
+----------------+----------------+
| 21.04          | 21.04          |
+----------------+----------------+
```

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,ZGVmYXVsdHMgewogICAgICAgIHVzZXJfZnJpZW5kbHlfbmFtZXMgeWVzCiAgICAgI
CAgZmluZF9tdWx0aXBhdGhzIHllcwp9CgpibGFja2xpc3RfZXhjZXB0aW9ucyB7CiAgICAgICA
gcHJvcGVydHkgIihTQ1NJX0lERU5UX3xJRF9XV04pIgp9CgpibGFja2xpc3Qgewp9Cgo=
          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"
}
```

Best practice is 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: Red Hat OpenShift with NetApp.

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: Red Hat OpenShift with NetApp.

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, etc) or 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: Red Hat OpenShift with NetApp.

Advanced Configuration Options For OpenShift

Exploring load balancer options: Red Hat OpenShift with NetApp

In most of the 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

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 don't 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 pre-requisite 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: metallb-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/metallb-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.

Creating Private Image Registries: Red Hat OpenShift with NetApp

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. Edit the imageregistry operator, enter the below storage parameters to `spec` section

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

storage:
  pvc:
    claim:
```

2. Then enter the following parameters to `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 default hostname, you can just add the following parameters to `spec` section – `defaultRoute: true`

Custom TLS Certificates

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

3. Next step is to edit the imageregistry operator again and change the management state of the operator to `Managed` state, save and exit –

```
oc edit configs.imageregistry/cluster

managementState: Managed
```

4. If all the pre-requisites are satisfied, it should start creating PVCs, pods and services 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
```

5. If you are using the default TLS certificates of Ingress operator OpenShift registry route, you can fetch the TLS certificates using the below command.

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

6. To allow OpenShift nodes to access and pull the images from the registry, you need to add the certificates

to the docker client on the OpenShift nodes. Create a configmap in `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
```

7. 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/group of users to pull images from the registry, the user/s must have 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/group of users to write or push images, the user/s must have 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
   ```

8. For OpenShift nodes to access the registry and push/pull the images, you will 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
```

9. This pull secret can then be patched to serviceaccounts or be referenced in the corresponding pod definition.
   a. To patch it to service accounts

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

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

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

10. To push/pull an image from workstations apart from OpenShift node.

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 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 via podman/docker command.

podman

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

docker

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

d. Push/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.

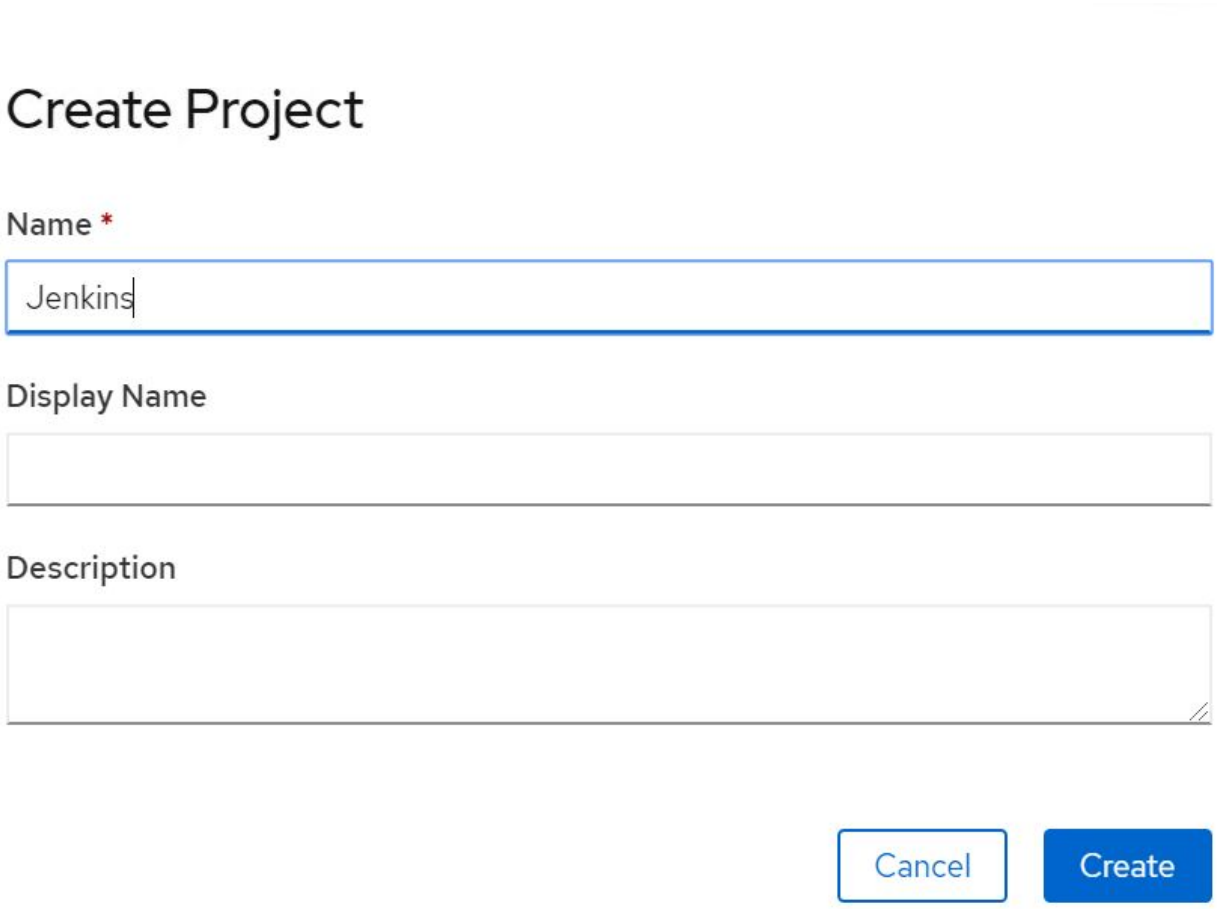
Deploy a Jenkins CI/CD Pipeline with Persistent Storage: Red Hat OpenShift with NetApp

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

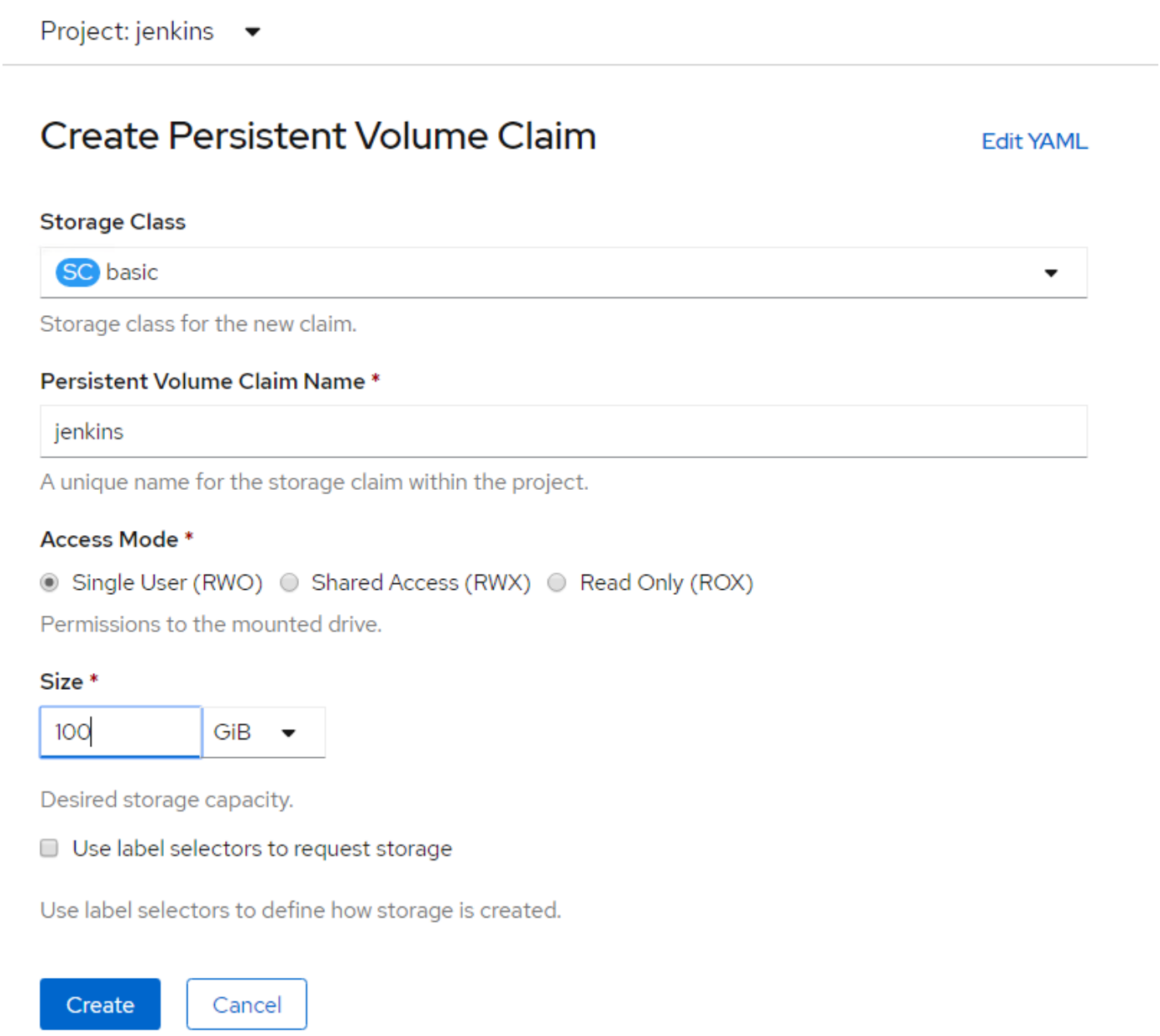
Create the resources required for Jenkins deployment

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

1. Create a new project named Jenkins.



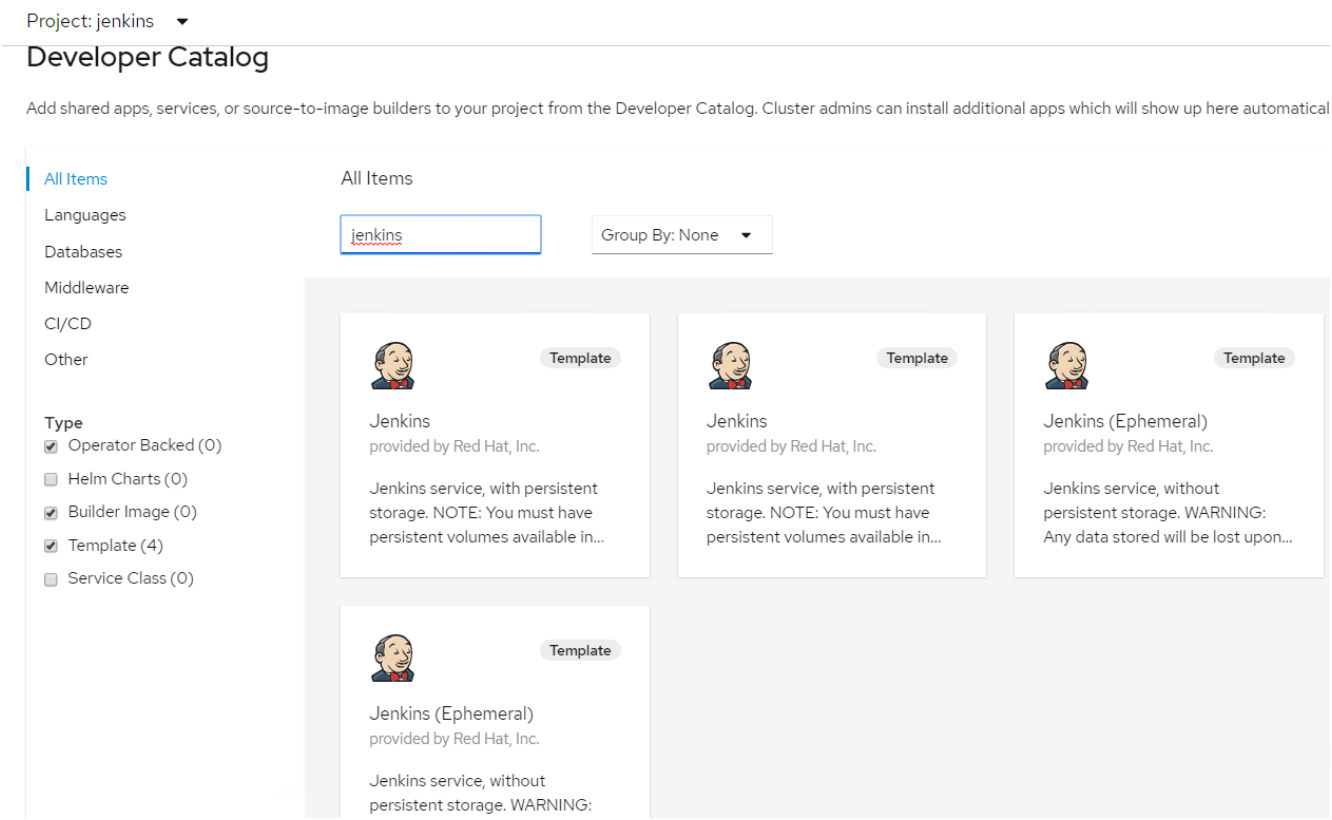
2. In this example, we deployed Jenkins with persistent storage. To support the Jenkins build, create the PVC. Navigate to `Storage > Persistent Volume Claims` and click `Create Persistent Volume Claim`. Select the storage class that was created, make sure that the Persistent Volume Claim Name is jenkins,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



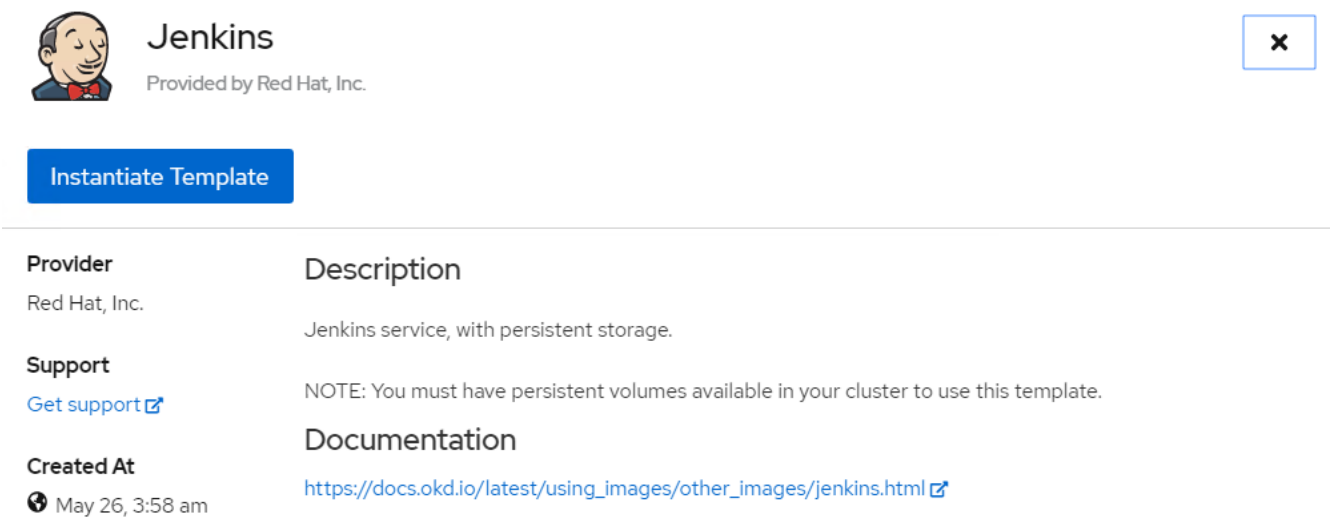
Deploy Jenkins with Persistent Storage

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.

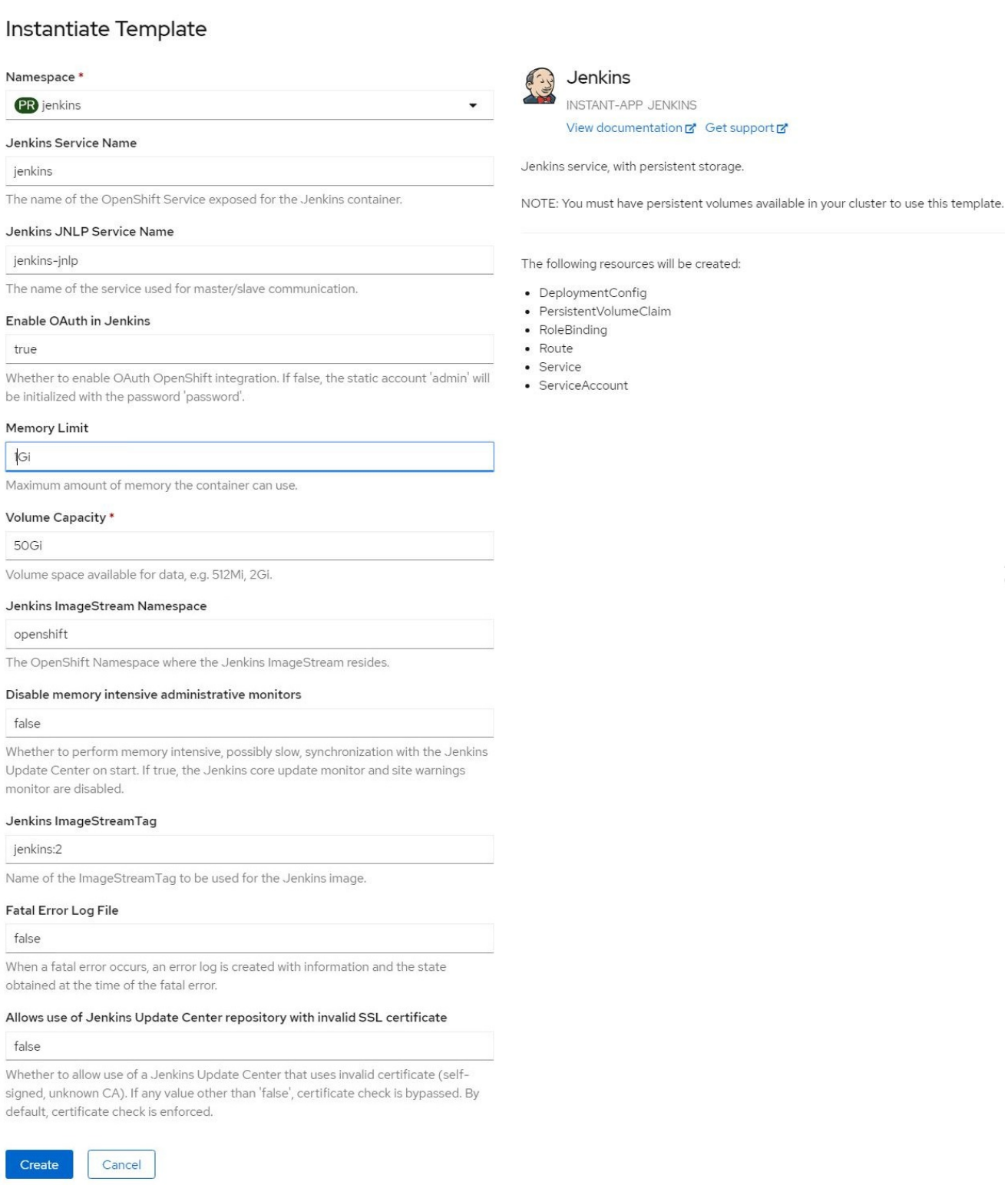


2. Click `Instantiate Template`.

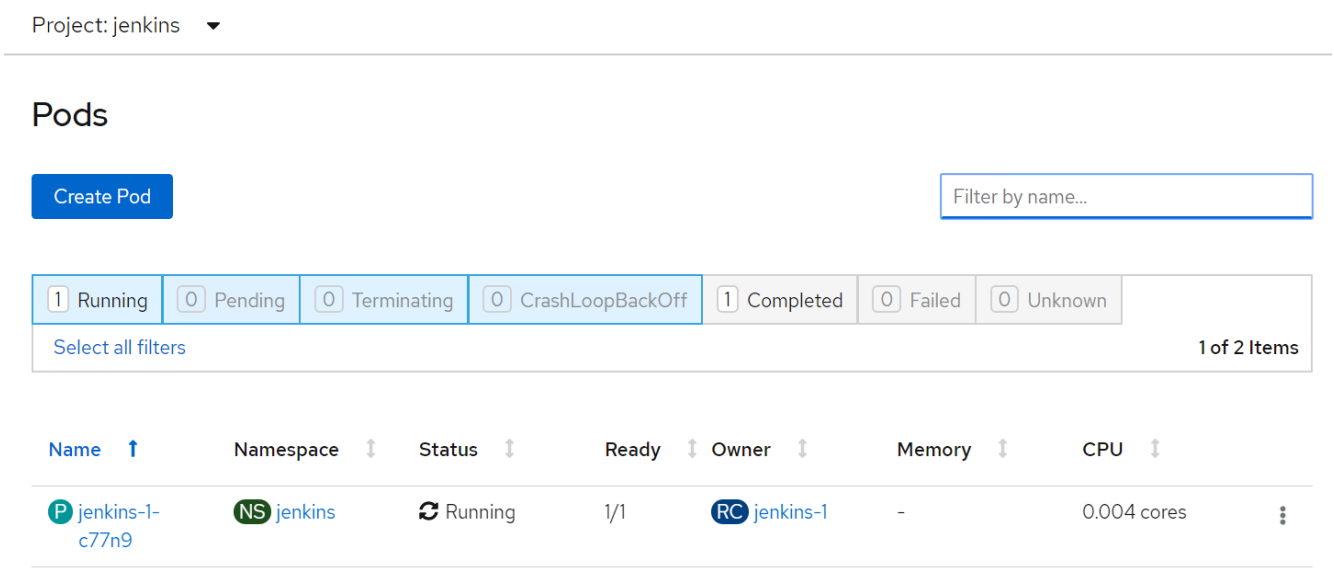


3. By default, the details for the Jenkins application are populated. Based on your requirements, modify the parameters and click Create. This process creates all the required resources for supporting Jenkins on

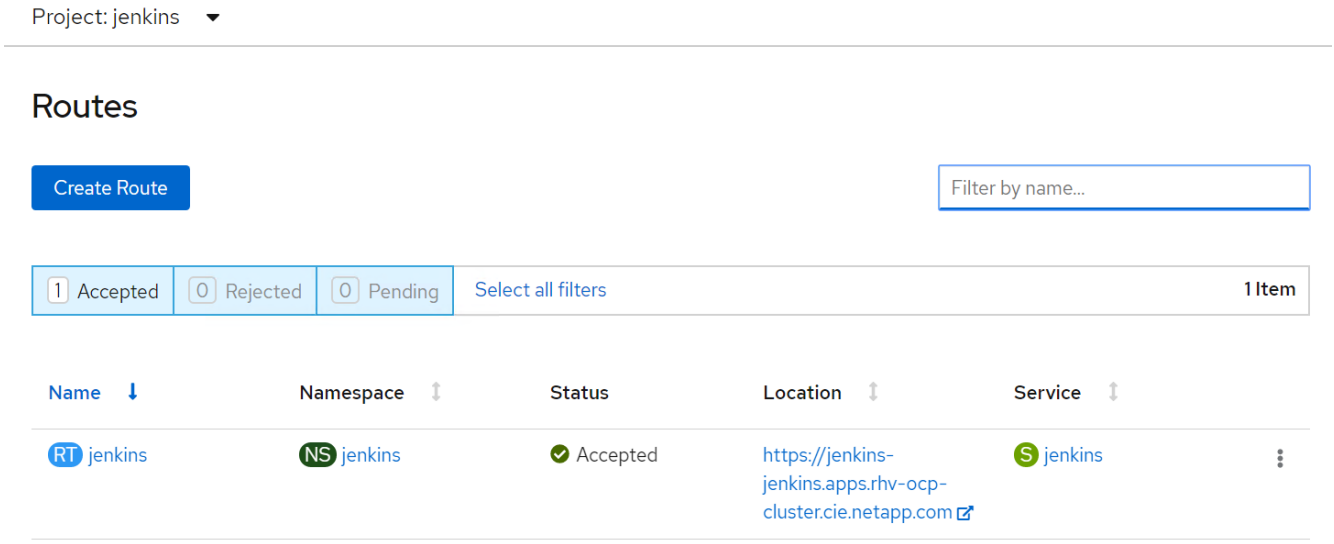
OpenShift.



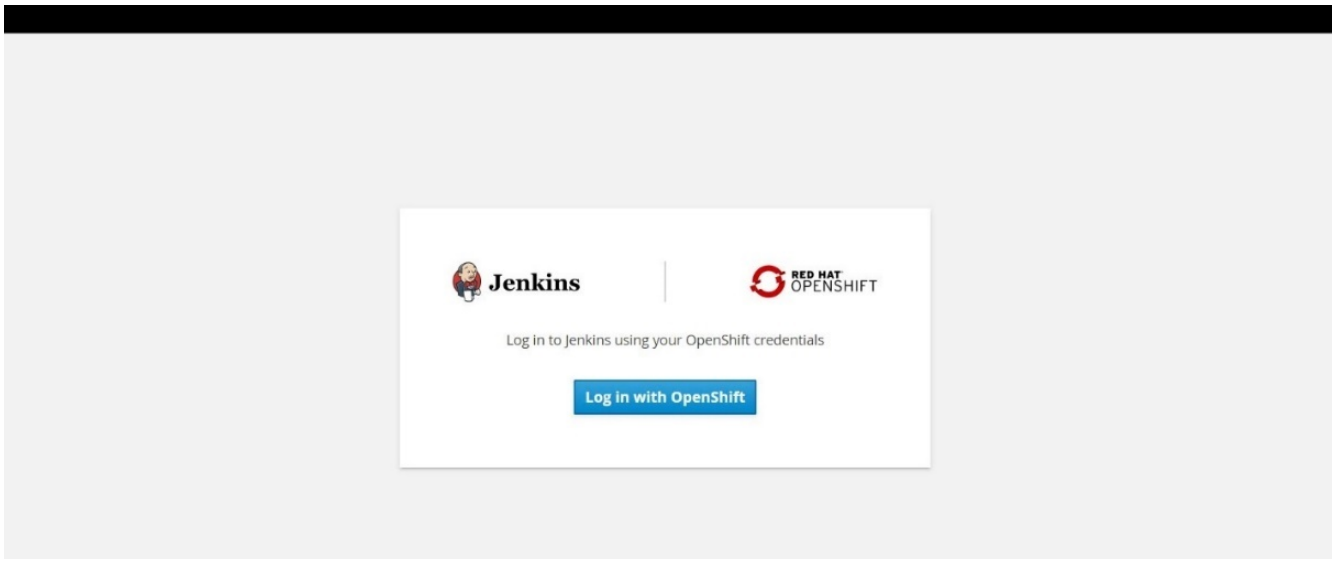
4. The Jenkins pods take approximately 10–12 minutes to enter the Ready state.



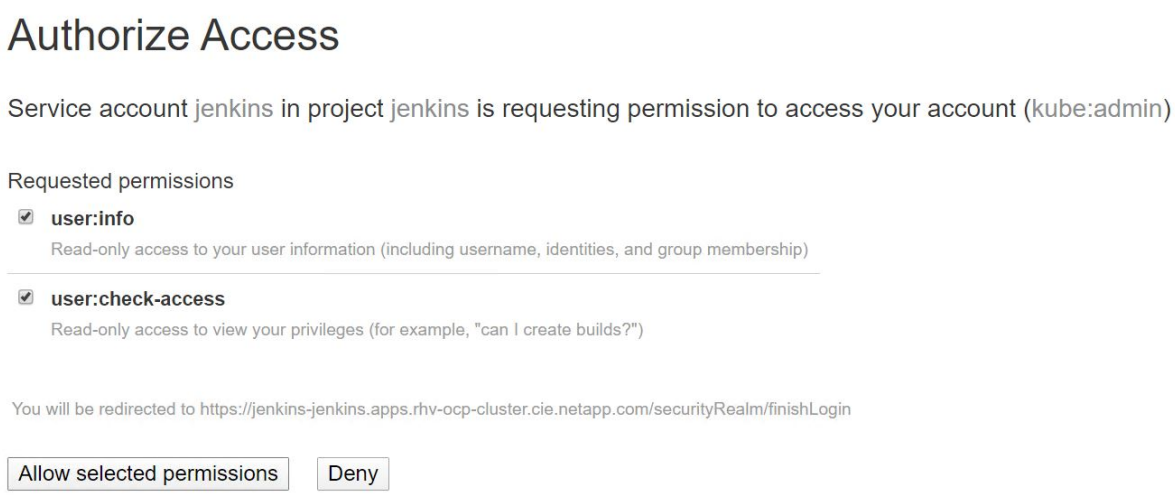
5. After the pods are instantiated, navigate to `Networking > Routes`. To open the Jenkins webpage, click the URL provided for the jenkins route.



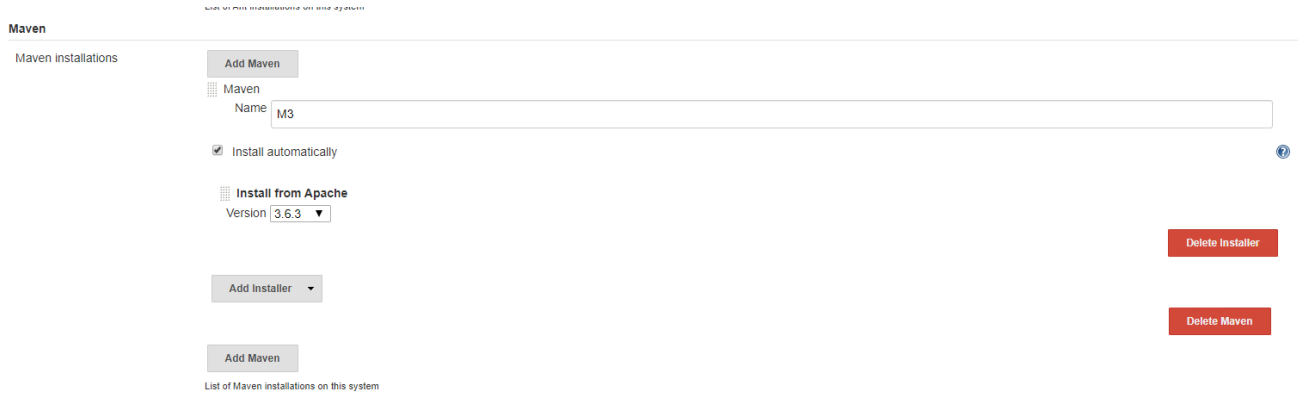
6. Because OpenShift OAuth was used while creating the Jenkins app, click `Log in with OpenShift`.



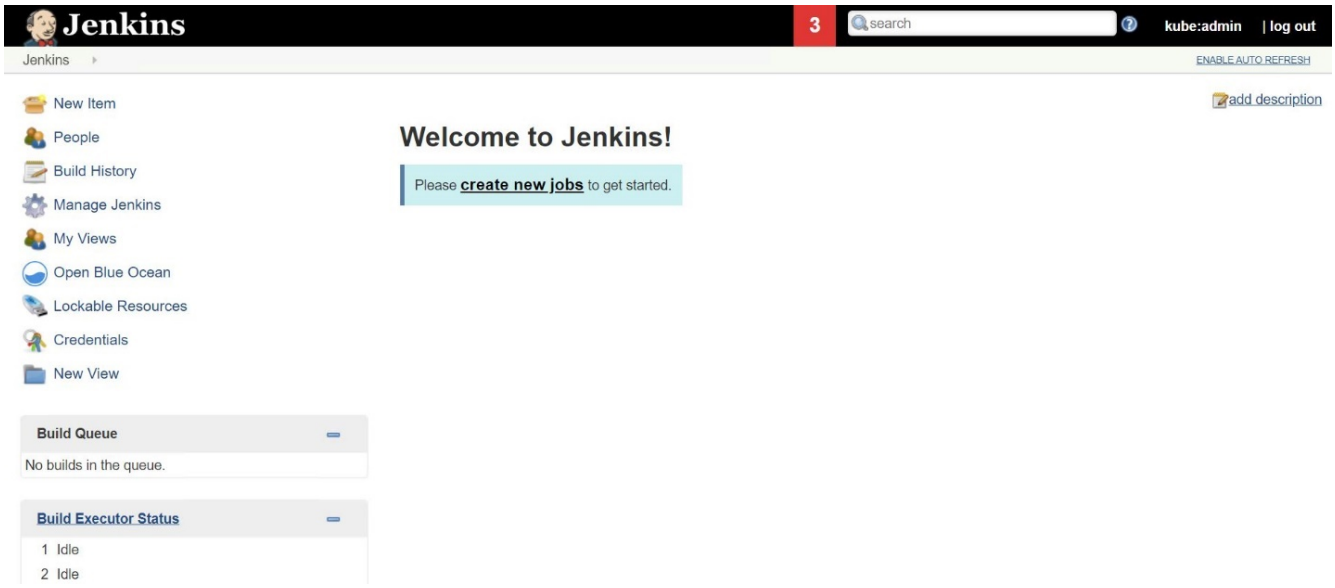
7. Authorize the Jenkins service-account to access the OpenShift users.



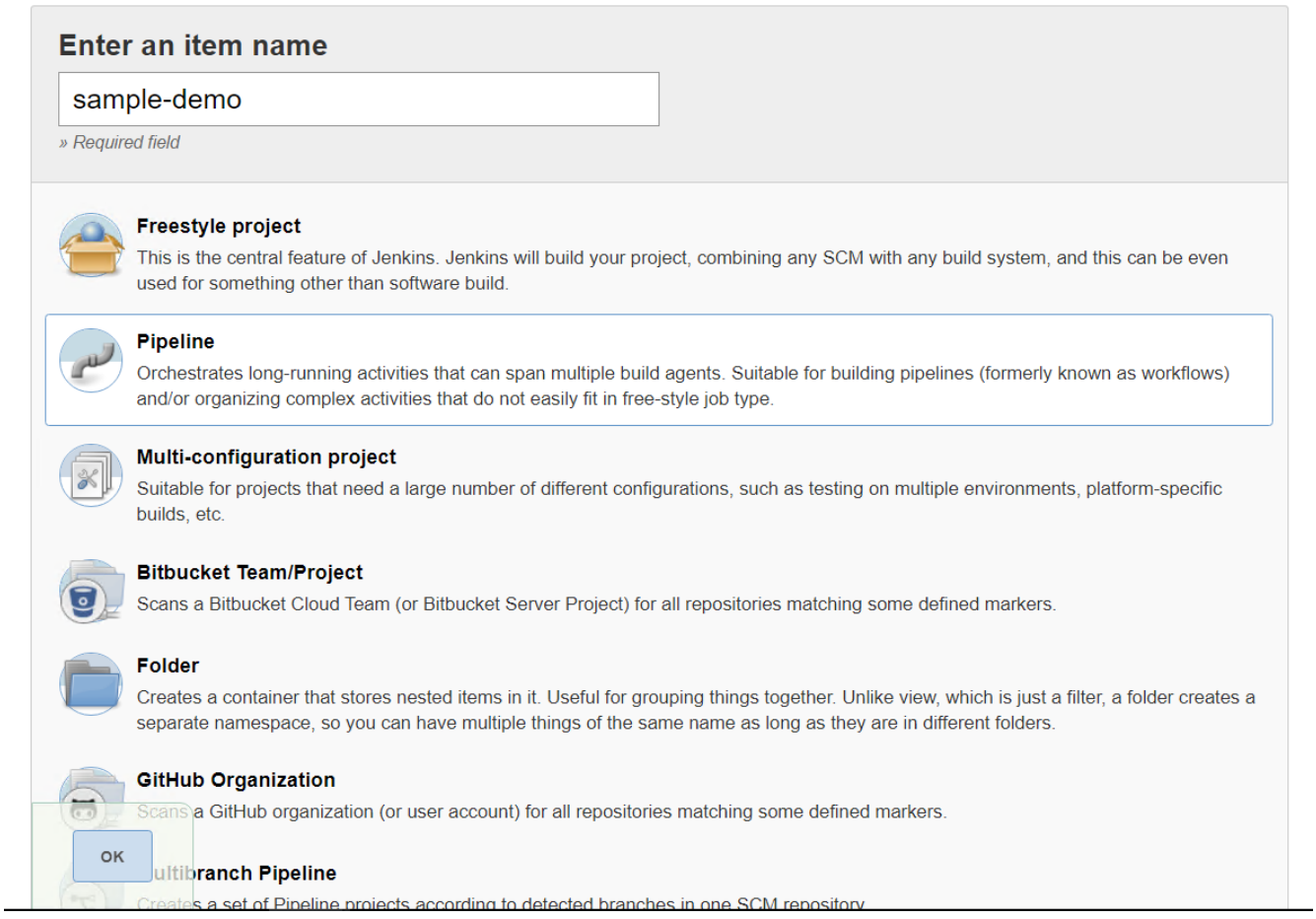
8. The Jenkins welcome page is displayed. Because we are using a Maven build, complete the Maven installation first. Navigate to `Manage Jenkins > Global Tool Configuration`, and then, in the Maven subhead, click `Add Maven`. Enter the name of your choice and make sure that the Install Automatically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



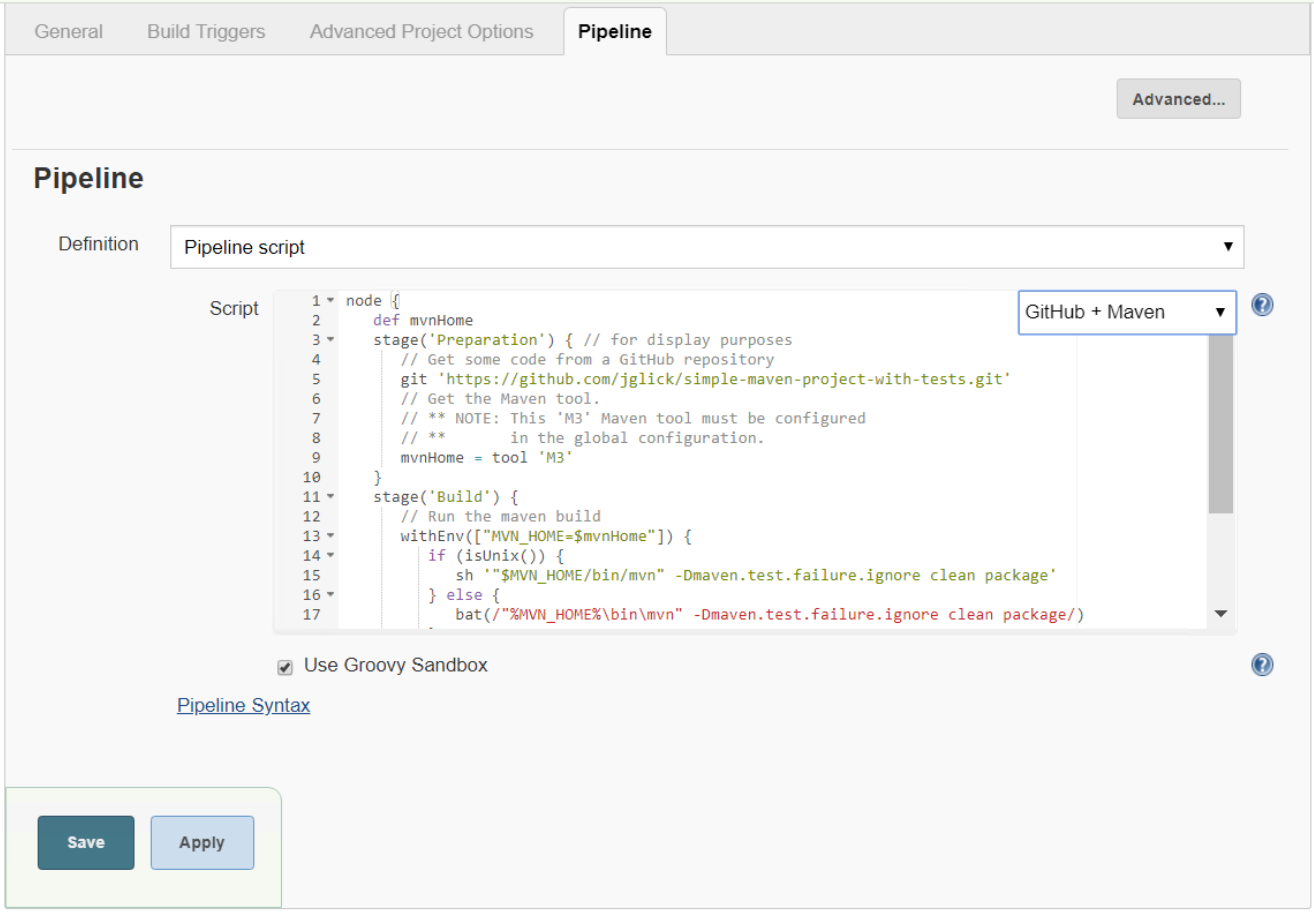
9. You can now create a pipeline to demonstrate the CI/CD workflow. On the home page, click `Create New Jobs` or `New Item` from the left-hand menu.



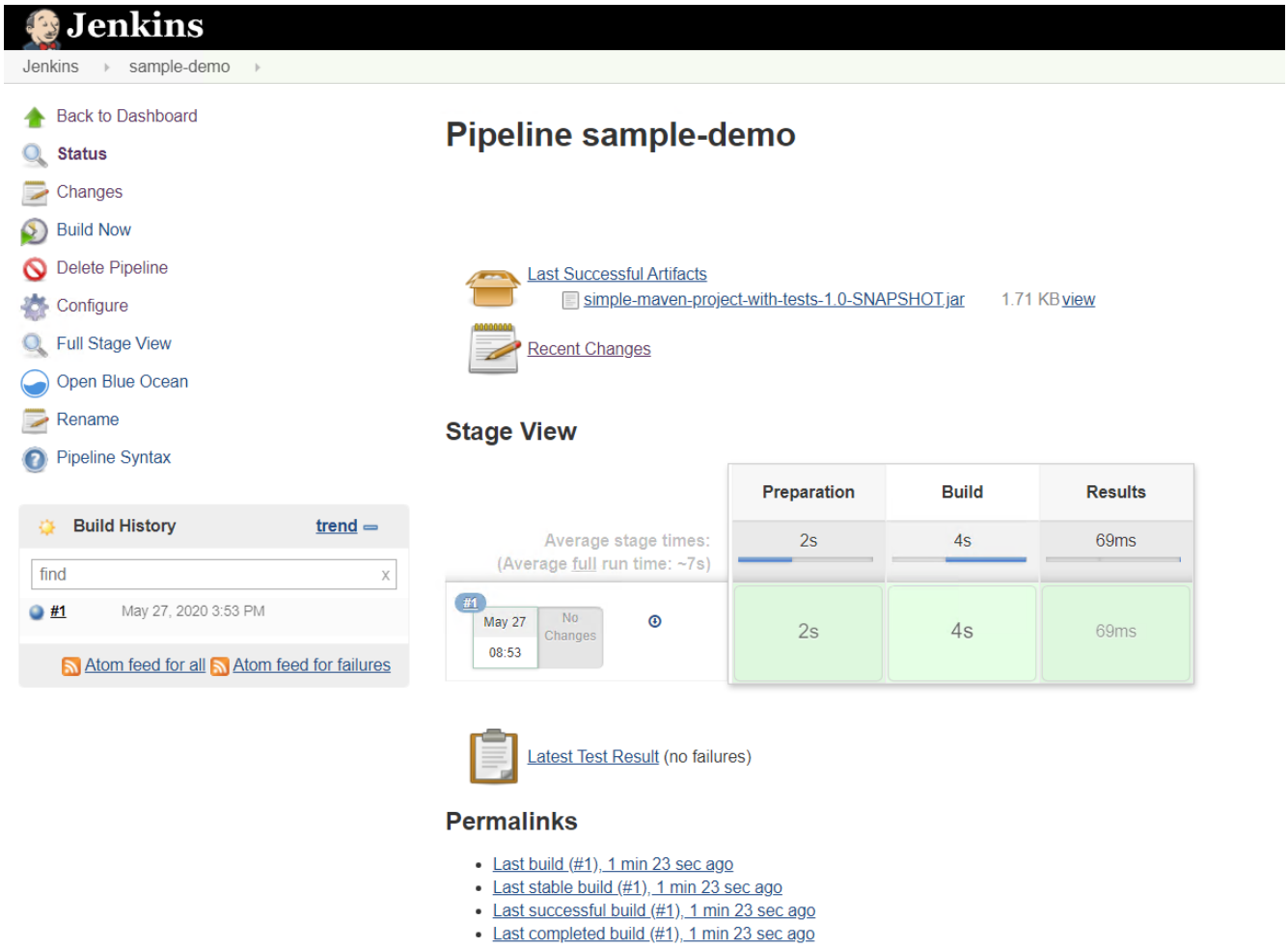
10. On the Create Item page, enter the name of your choice, select Pipeline, and click Ok.



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: 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/workload. This allows the organizations to implement strict isolation for the application/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/workloads in a single Red Hat OpenShift cluster. But in such an architecture, resource isolation and application security vulnerabilities pose themselves as 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.

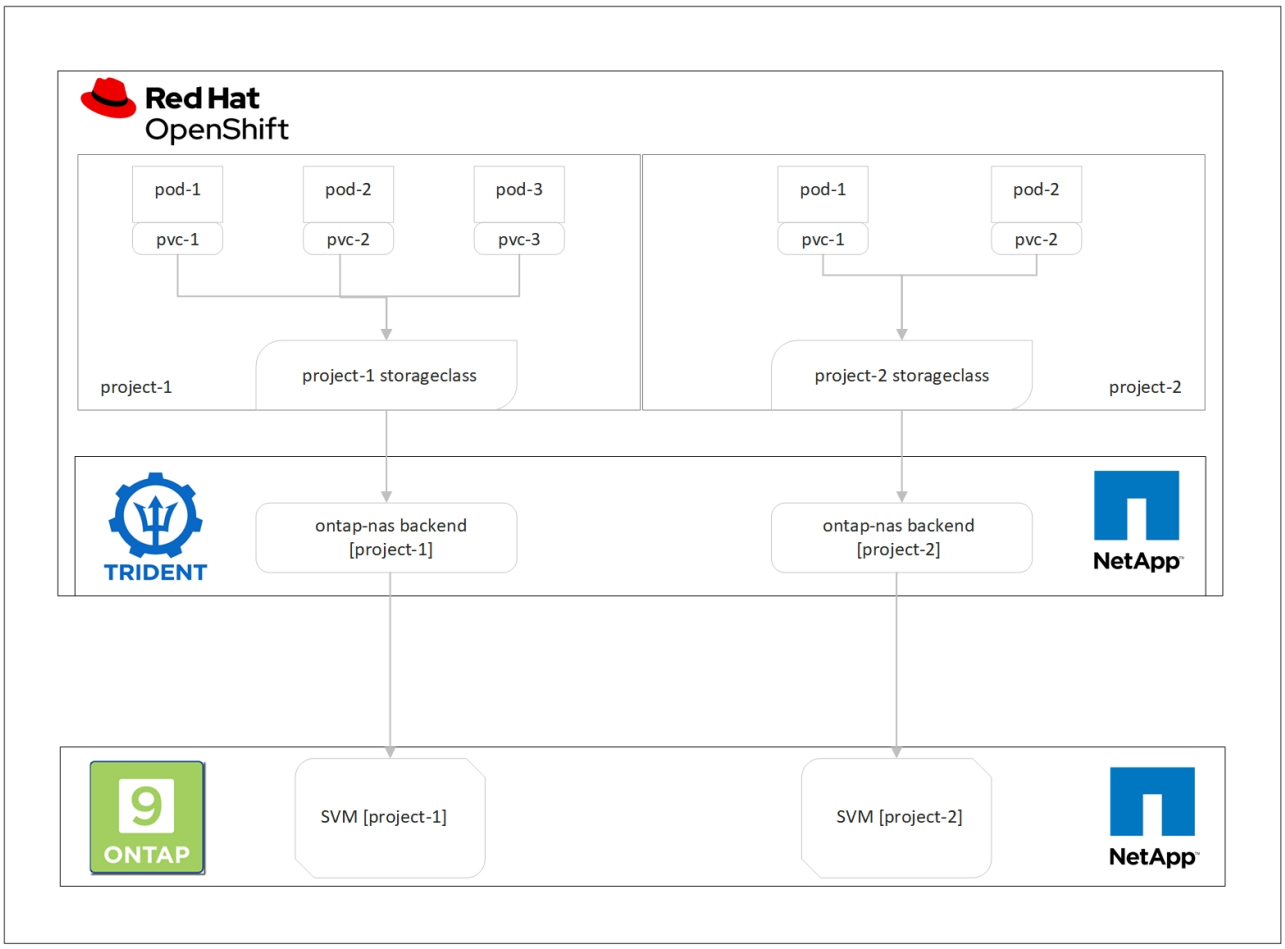
Next: Architecture.

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 multi-tenancy. To better understand designing a multi-tenant 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 multi-tenant 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 multi-tenant 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 multi-tenancy.

Next up is to configure RBAC in the cluster. The best practice is to have all the developers working on a single project/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 user group/s working on that project, hence restricting unauthorized access to any cluster resources. To learn more about IdP integration with Red Hat OpenShift, refer the documentation here.

NetApp ONTAP

It is important to isolate the shared storage serving as persistent storage provider for Red Hat OpenShift cluster to ensure 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/workloads and dedicate each SVM to a workload.

Astra Trident

After we have different SVMs for different projects created on NetApp ONTAP, we need to 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, protocol driver for the backend at the minimum. Optionally, it allows us to define how the volumes are provisioned on the storage and to set limits for the size of volumes or usage of aggregates etc. Details of defining the Trident backend for NetApp ONTAP can be found here.

Red Hat OpenShift – storage resources

After configuring the Trident backends, 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 storagePools parameter while defining the storage class. The details to define a storage class can be found here. Thus, there will be 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 will be served by the SVM dedicated to that project only.

But since storage classes are not namespaced resources, how do we ensure that storage claims to storage class of one project by pods in another namespace/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/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/patch PVCs/pods in assigned project
	Validate access to create/patch PVCs/pods in another project
	Validate access to view/edit Projects, ResourceQuotas, and StorageClasses

Next: Prerequisites.

Configuration

Pre-requisites

- 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 role must be defined based on the 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 sync'd with cluster groups. The following example shows that the identity provider has been integrated with the cluster and sync'd 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 roles.

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
```

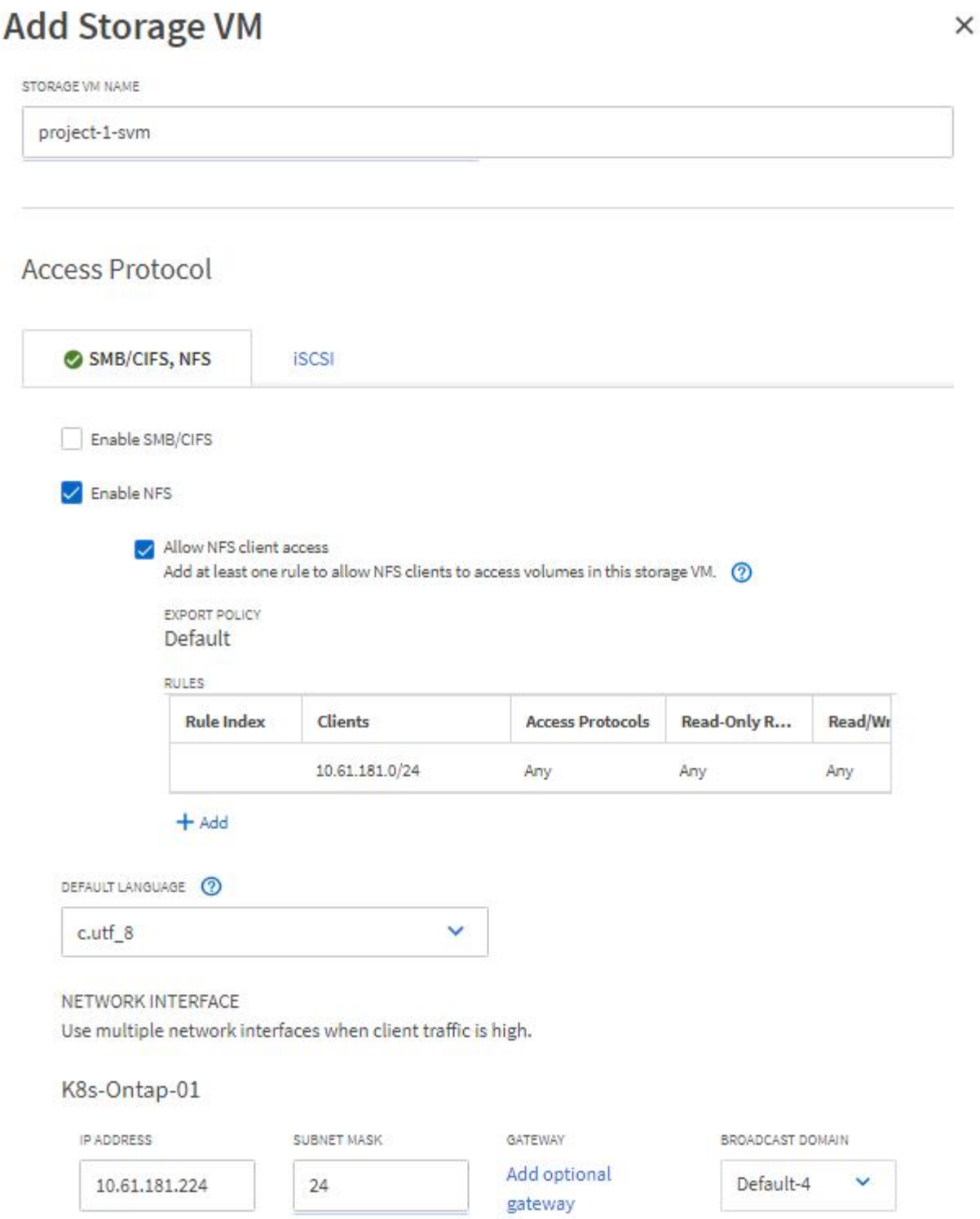
10. Similarly, create RoleBindings for developers binding the developer roles to the corresponding user group in project-2.

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, providing the required details. Also create an vsadmin account to manage the SVM and its resources.



3. Login to 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 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/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/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/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.

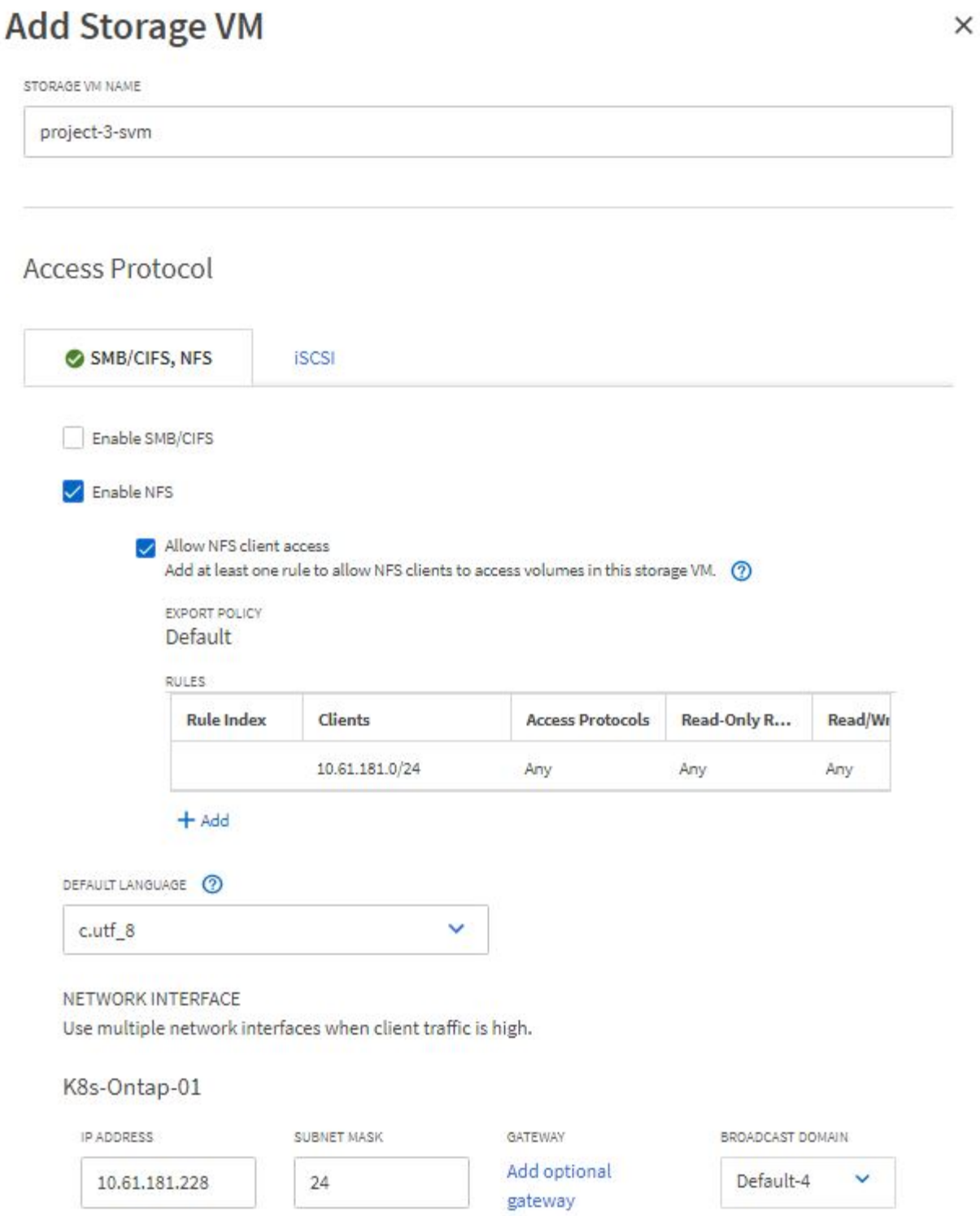
```
oc edit sc project-1-sc
```

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 sync'd 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.

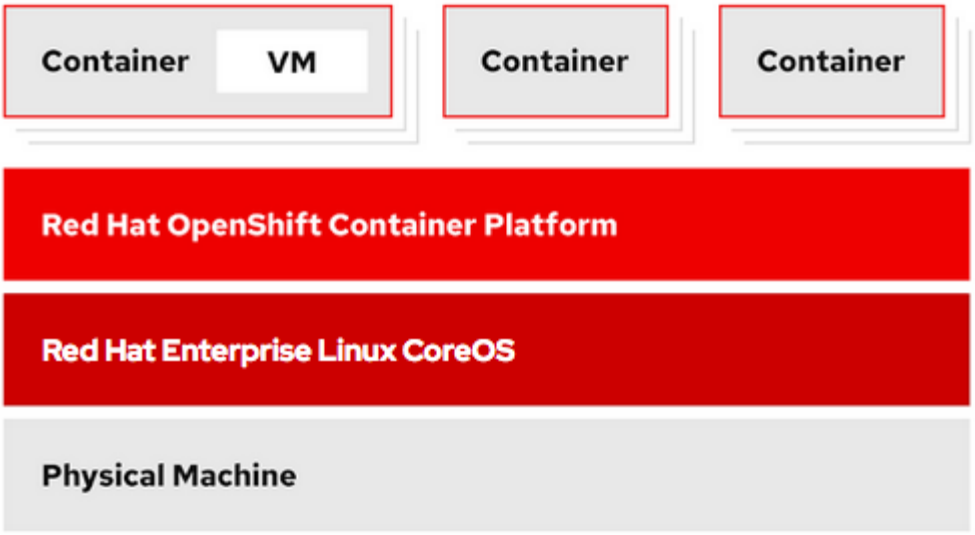
```
oc patch resourcequotas project-1-sc-rq -n project-1 --patch
'{"spec":{"hard":{ "project-3-
sc.storageclass.storage.k8s.io/persistentvolumeclaims": 0}}}'
oc patch resourcequotas project-2-sc-rq -n project-2 --patch
'{"spec":{"hard":{ "project-3-
sc.storageclass.storage.k8s.io/persistentvolumeclaims": 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) are considered to offer an optimal platform 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.

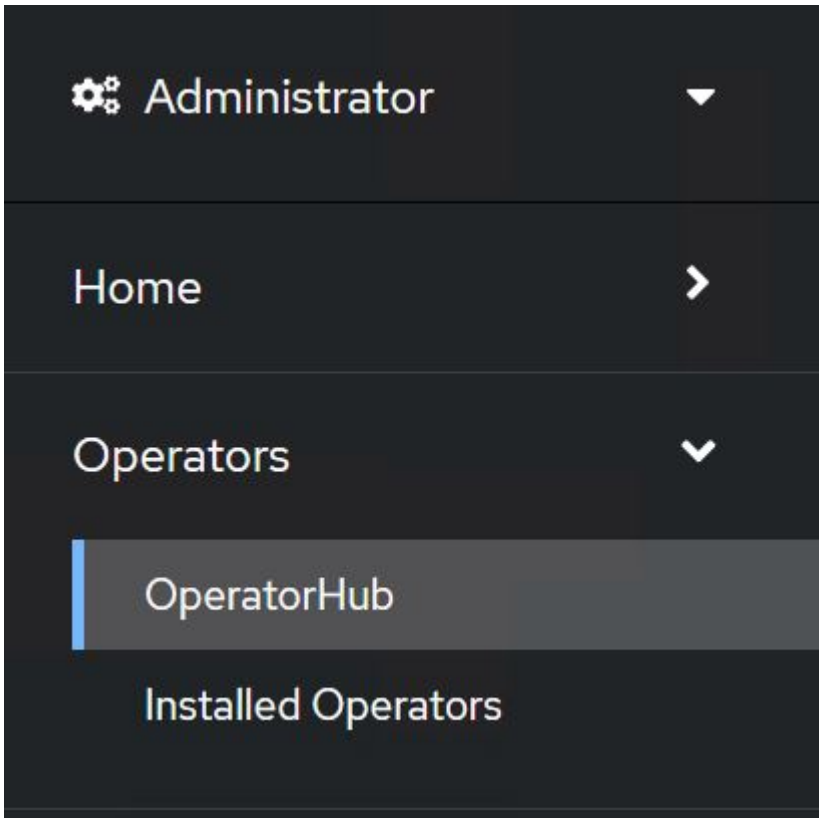
For the storage backing OpenShift Virtualization, NetApp recommends having a dedicated StorageClass that requests storage from a particular Trident backend, which in turn is backed by a dedicated SVM. This maintains a level of multitenancy with regard to the data being served for VM-based workloads on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

Next: Deploy via operator.

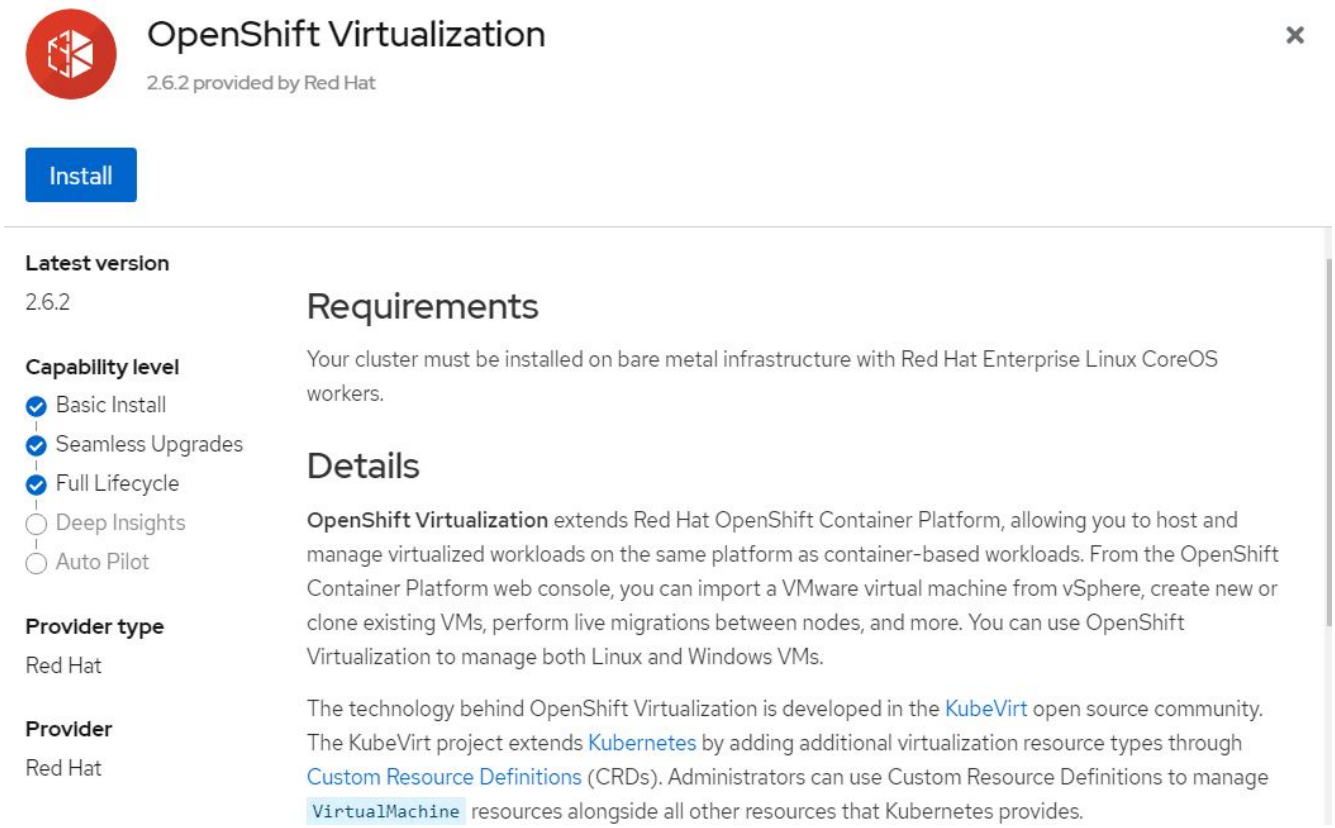
Deploy Red Hat OpenShift Virtualization with NetApp ONTAP

To install OpenShift Virtualization, complete the following steps:

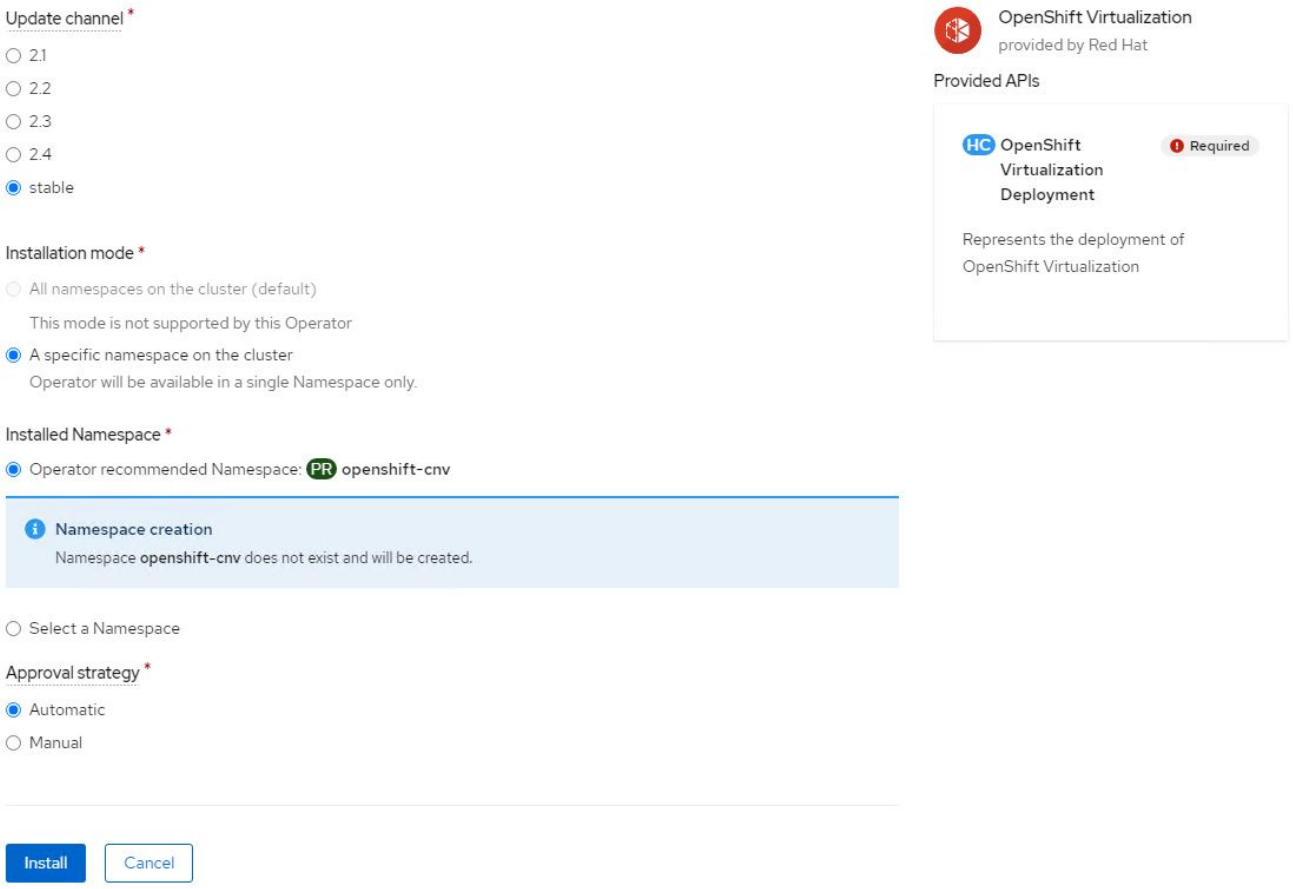
1. Log into the Red Hat OpenShift bare-metal cluster with cluster-admin access.
2. Select Administrator from the Perspective drop down.
3. Navigate to `Operators → OperatorHub` and search for OpenShift Virtualization.



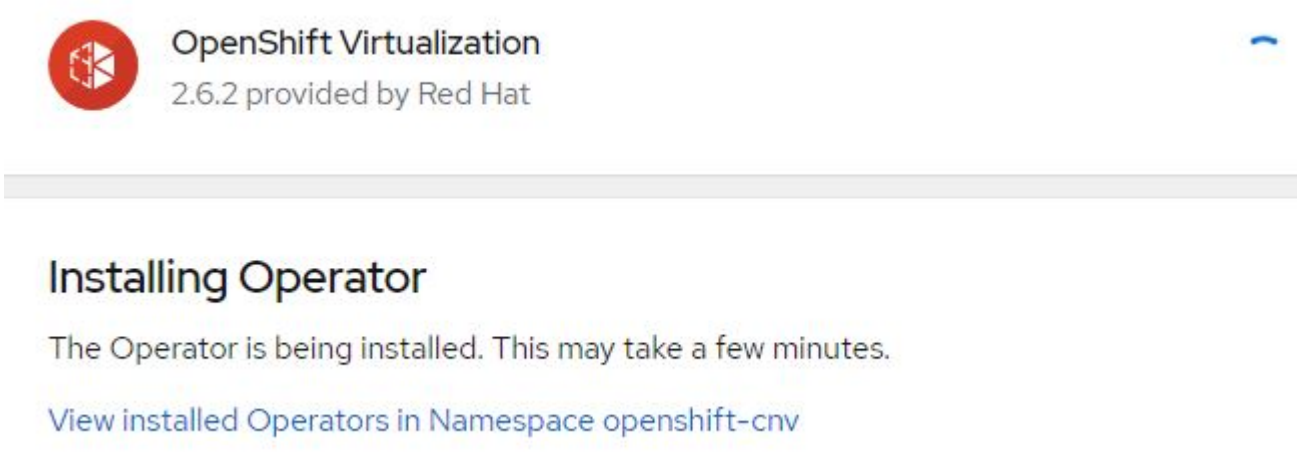
4. Select the OpenShift Virtualization tile and click Install.



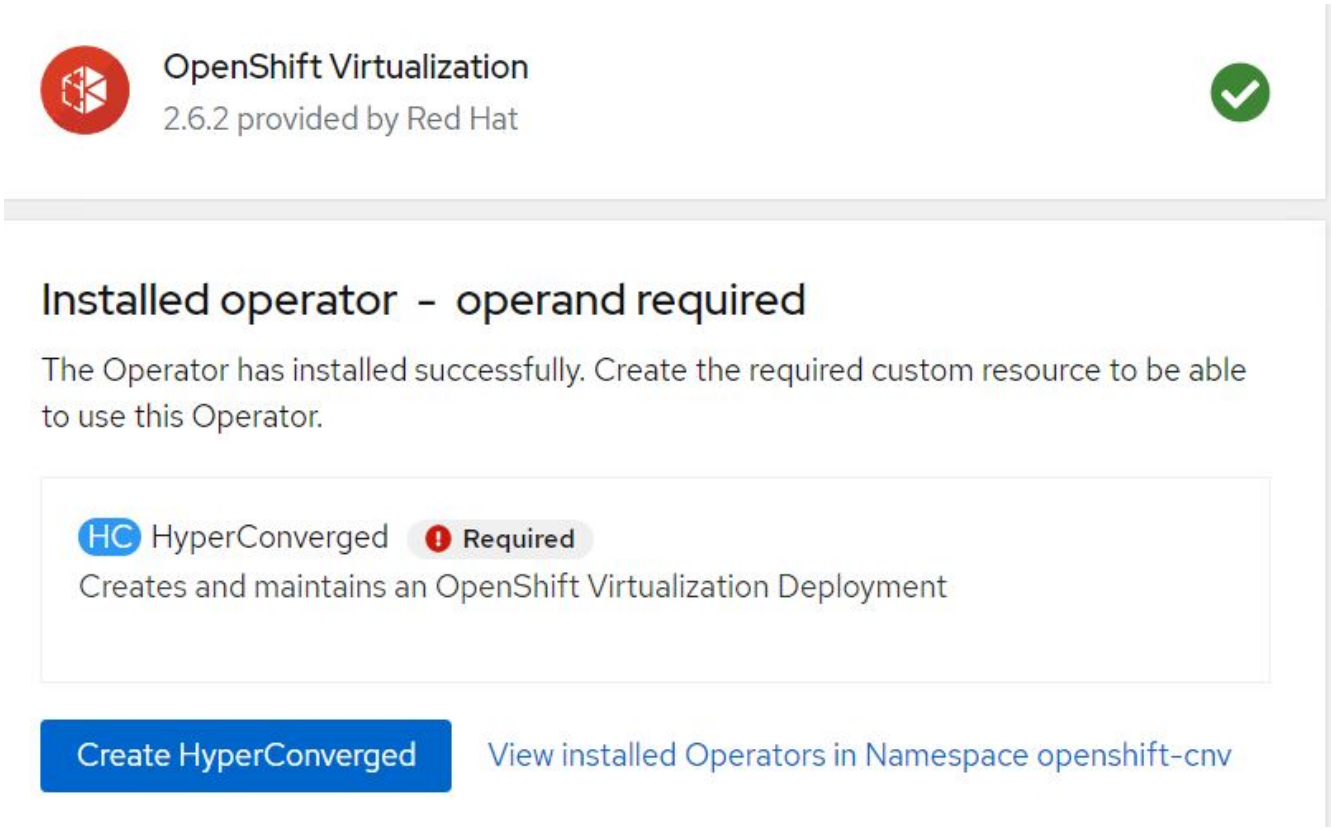
5. On the Install Operator screen, leave all default parameters and click Install.



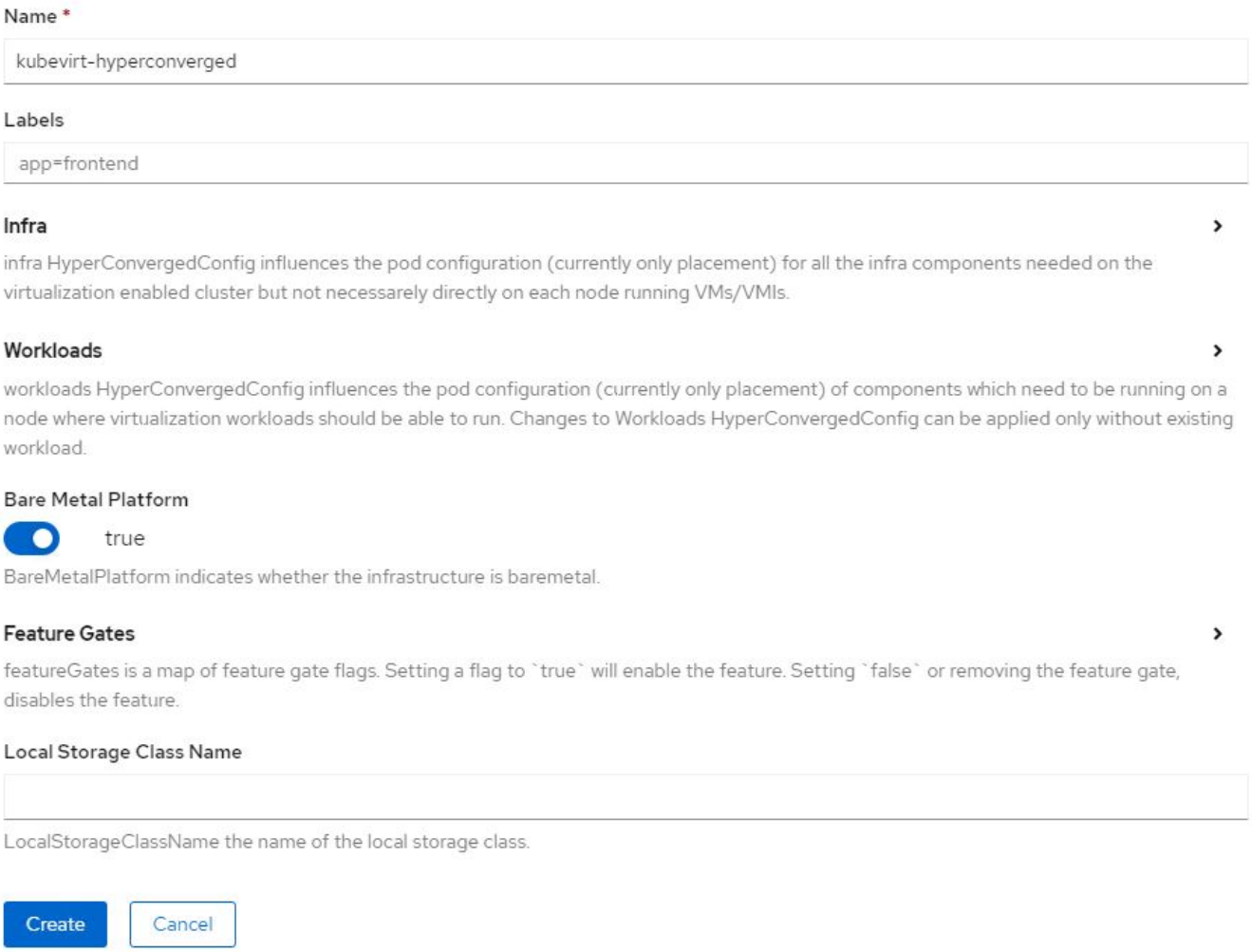
6. Wait for the operator installation to complete.



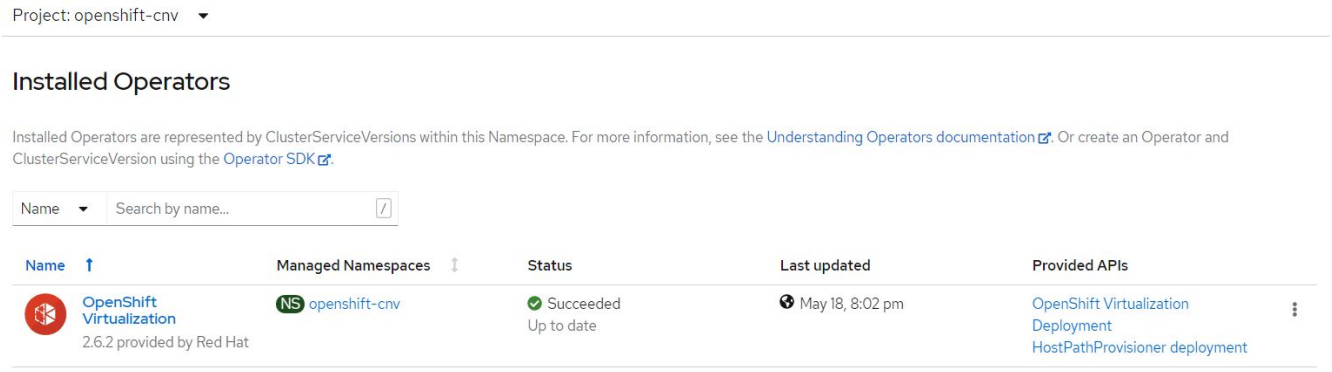
7. After the operator has installed, click Create HyperConverged.



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.



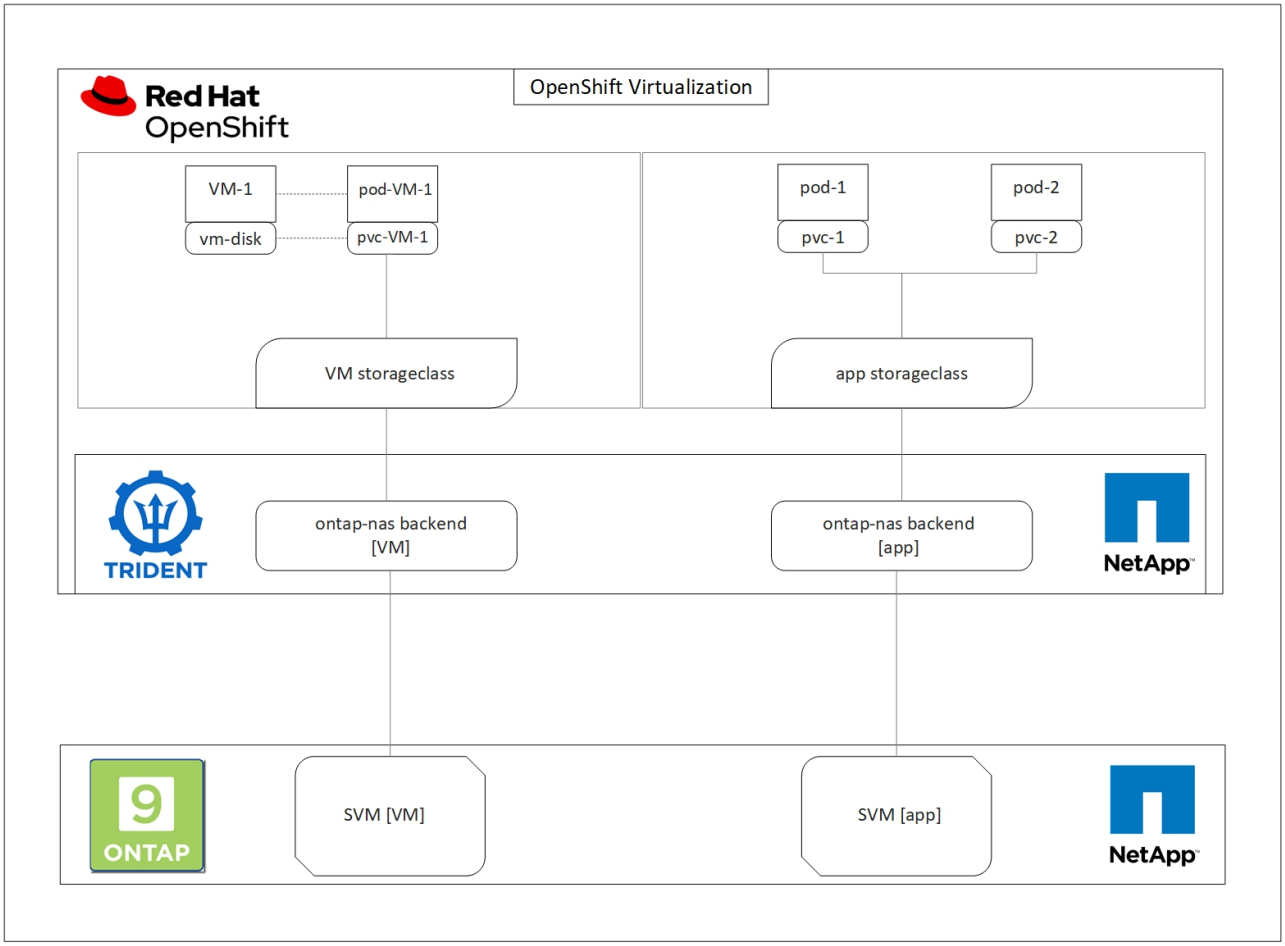
Next: Workflows: Create VM.

Workflows

Workflows: Red Hat OpenShift Virtualization with NetApp ONTAP

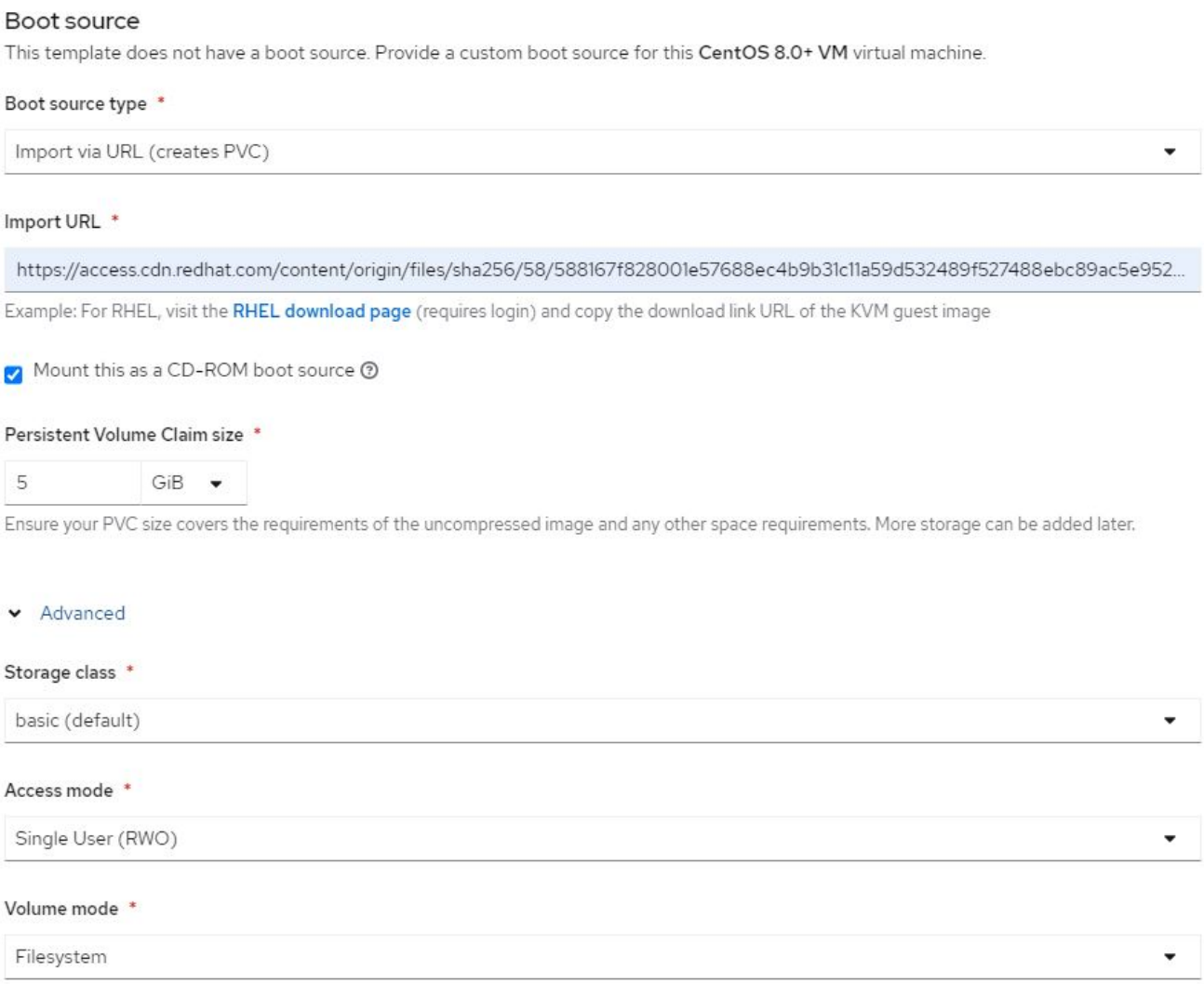
Create VM

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.

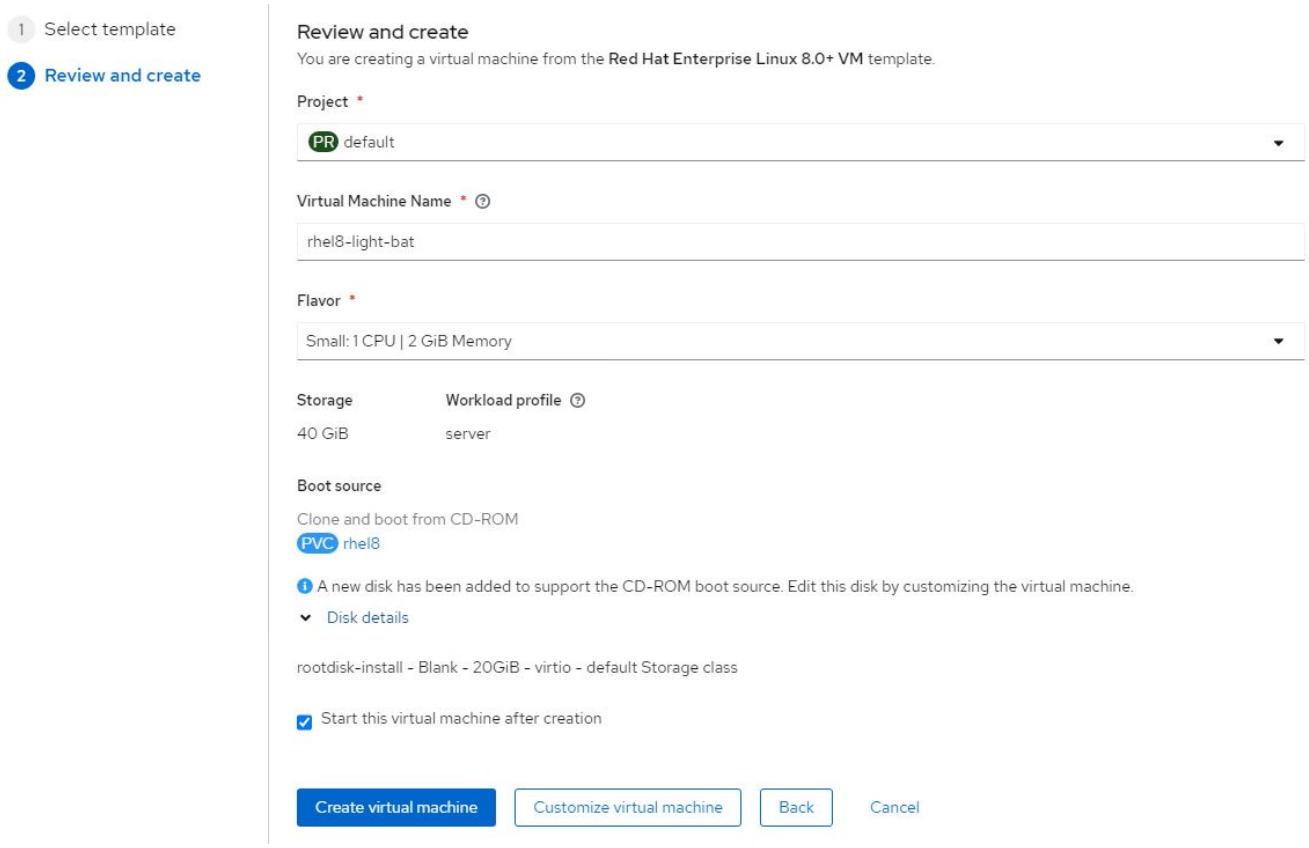


To create a virtual machine on the OpenShift cluster, complete the following steps:

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.

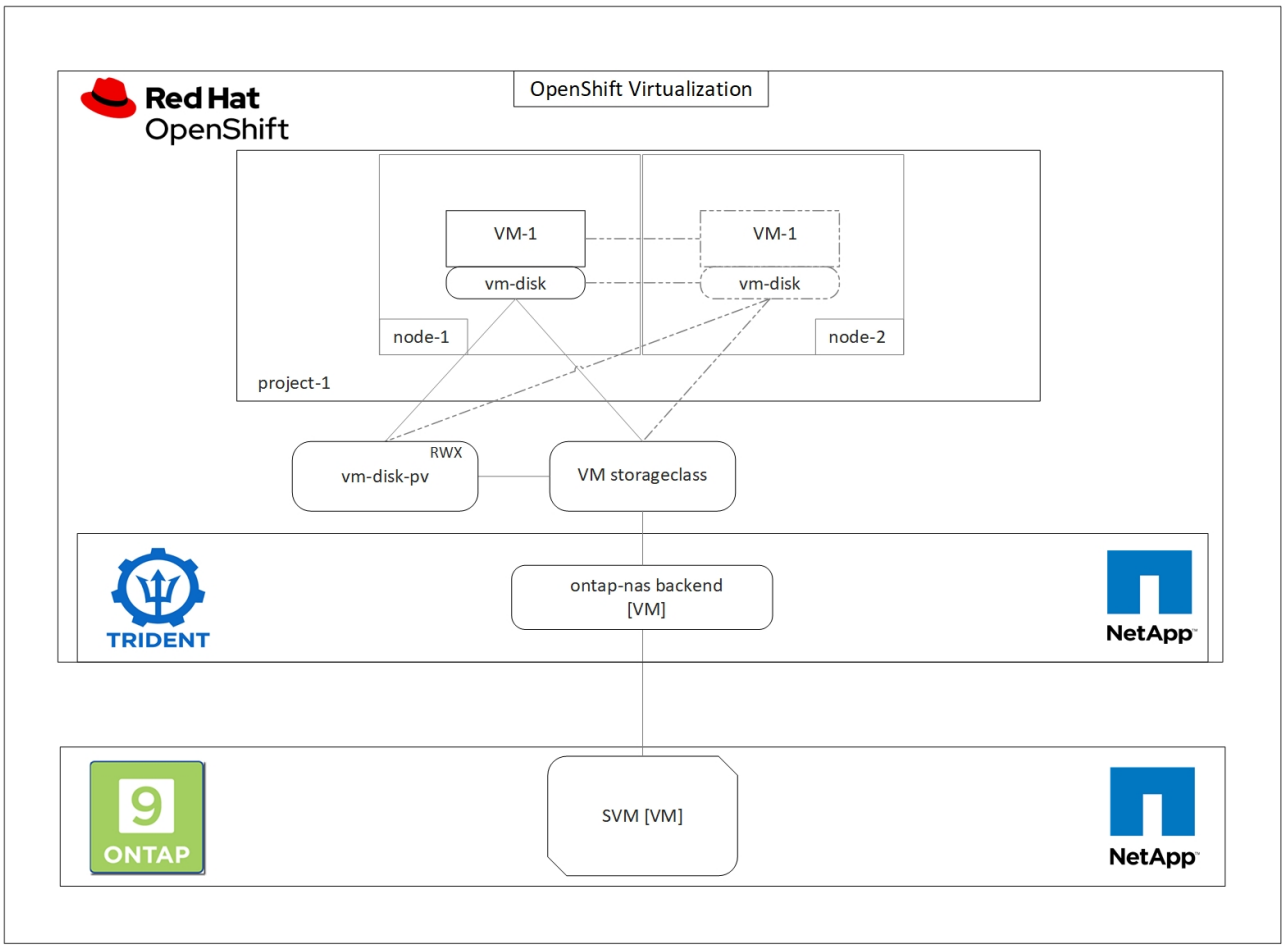
When a boot source is configured for a template or an operating system from an URL or from a registry, it creates a PVC in the `openshift-virtualization-os-images` project and downloads the KVM guest image to the PVC. You must make sure that template PVCs have enough provisioned space to accommodate 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

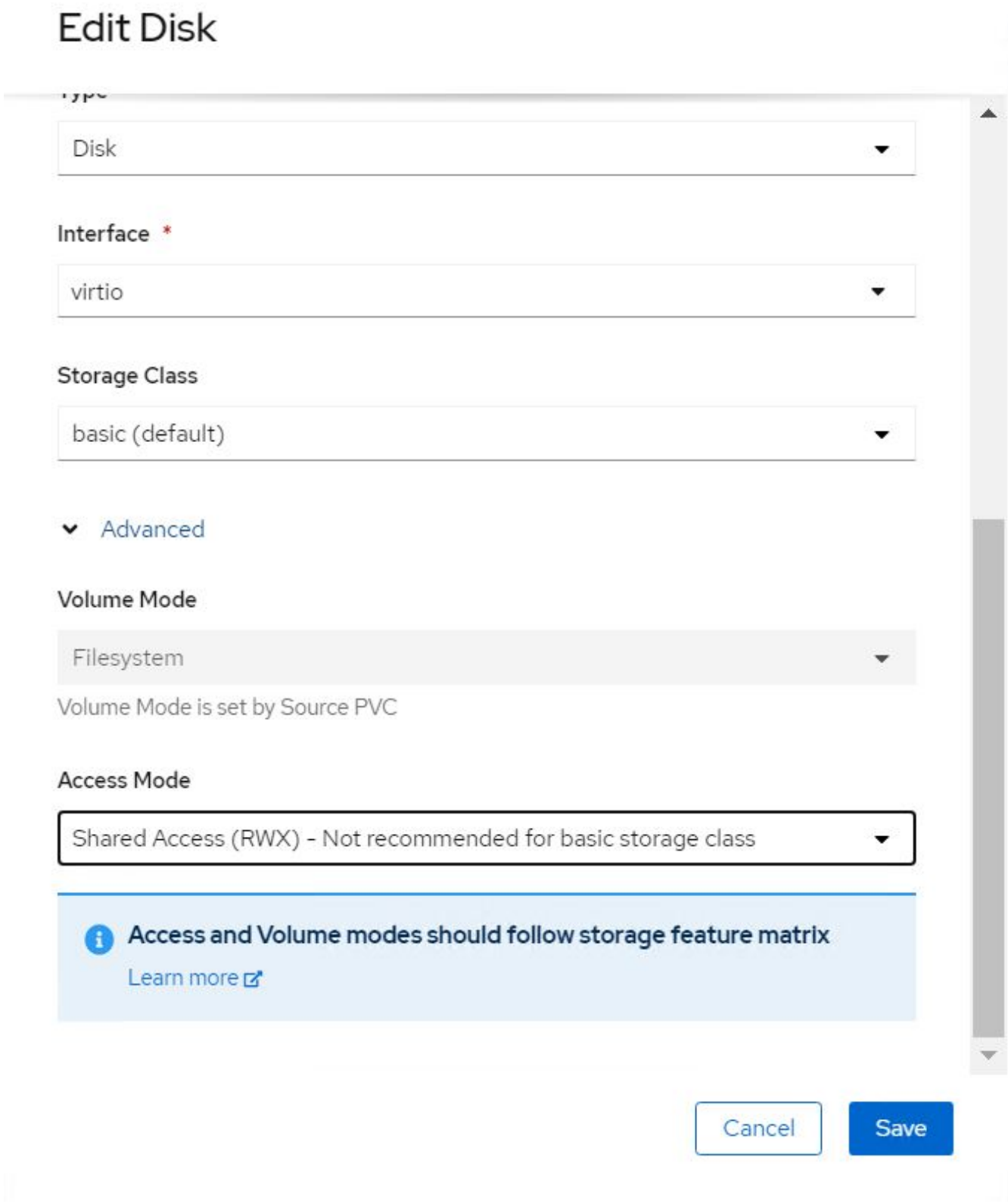
VM Live Migration

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 on the ellipsis next to rootdisk, 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.

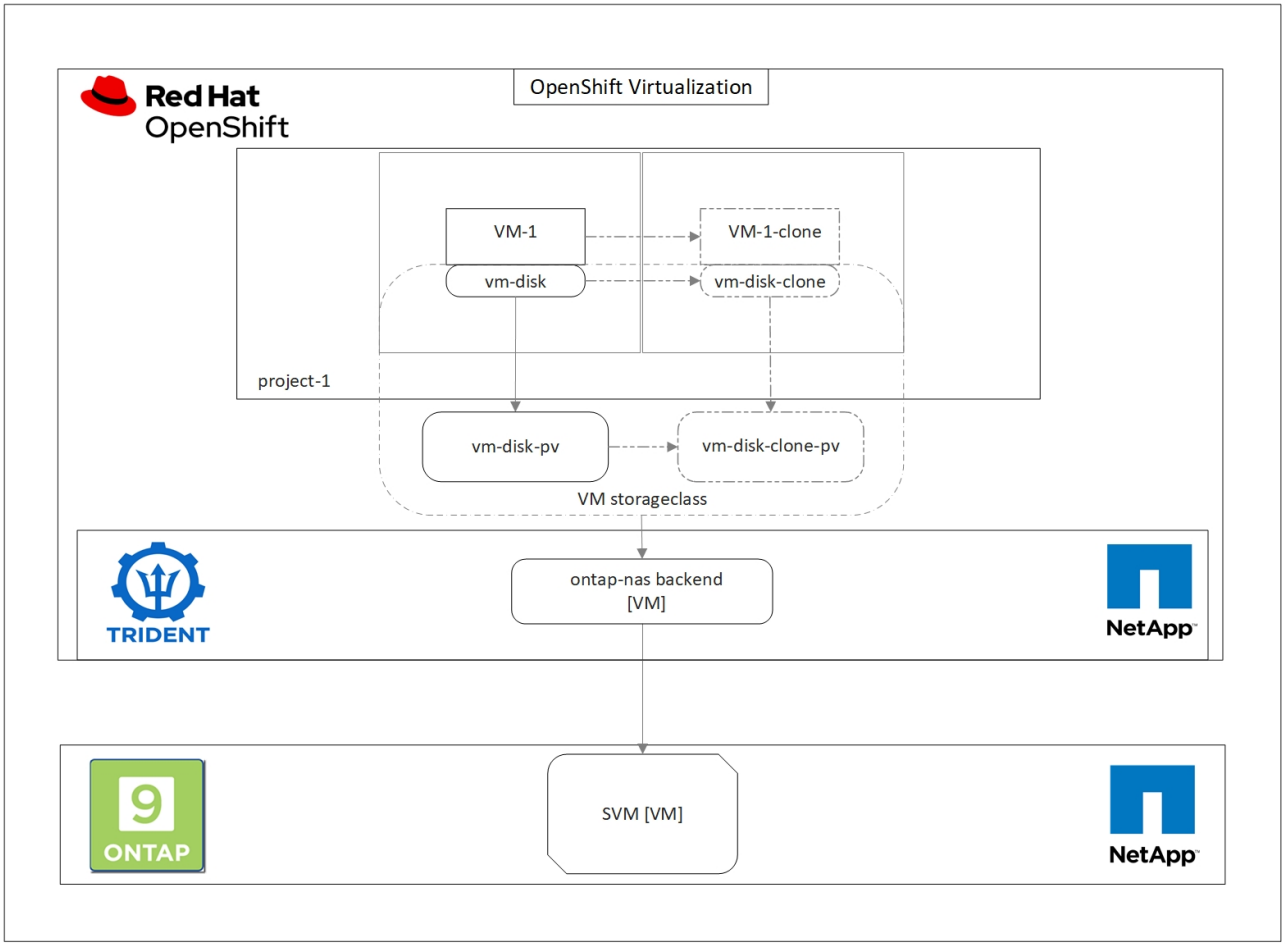
A VM instance in an OpenShift cluster automatically migrates to another node when the original node is placed into maintenance mode if the evictionStrategy is set to LiveMigrate.

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.

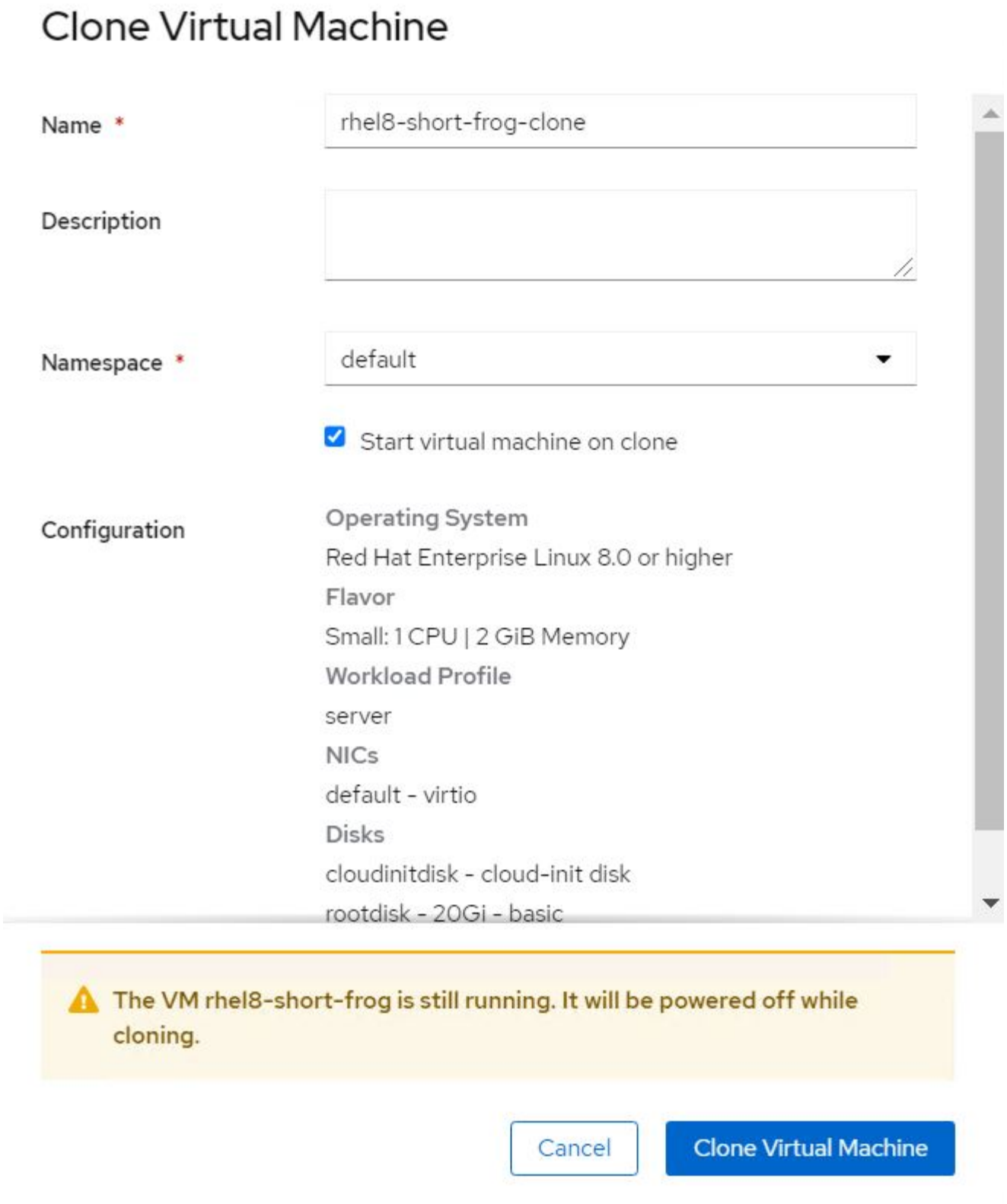
VMs in an OpenShift cluster can be cloned in two ways:

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.

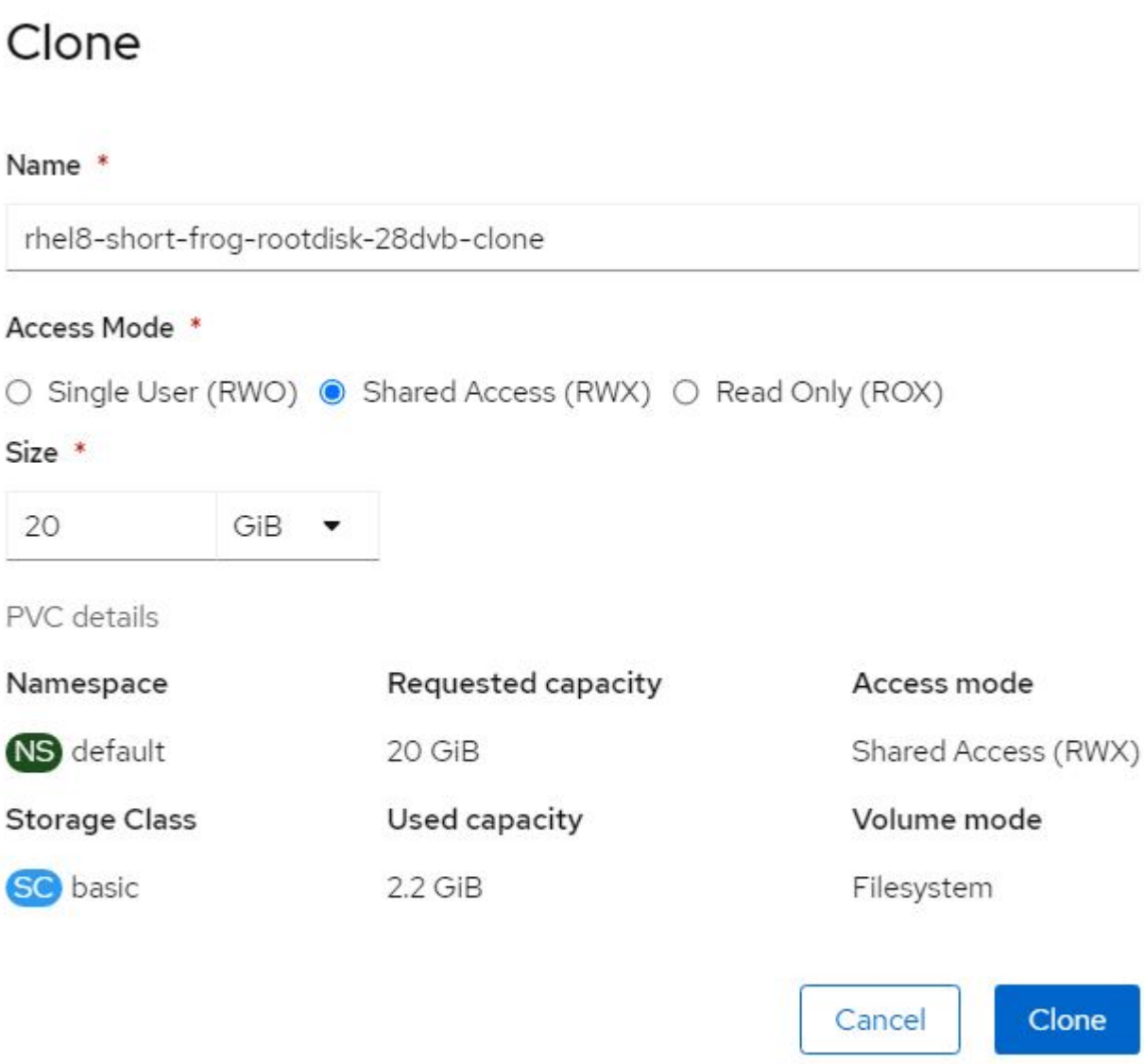


3. Click Clone Virtual Machine; this shuts down the source VM and initiates the creation of the clone VM.
4. After this step is completed, you can access and verify the content of the cloned VM.

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.

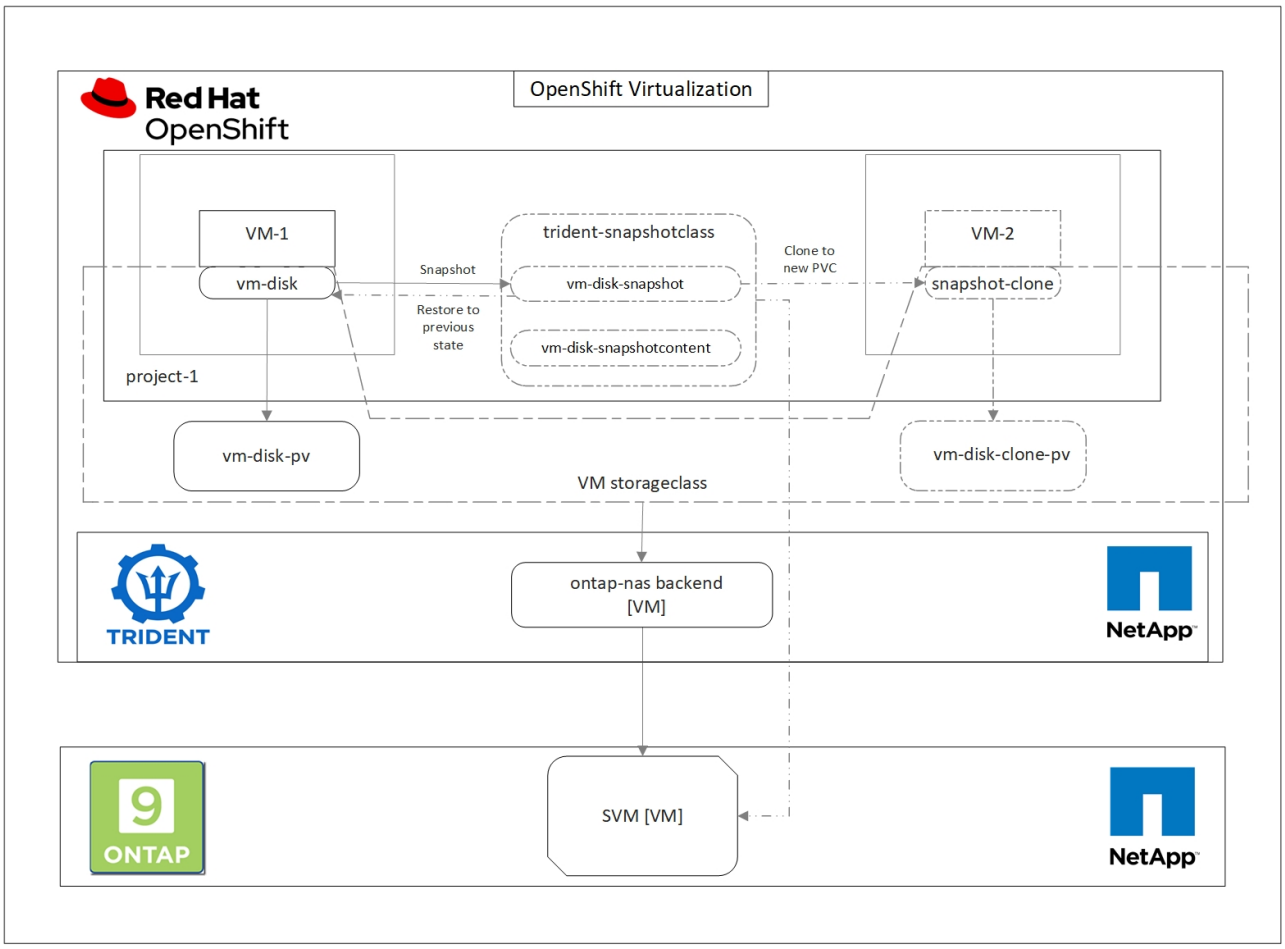
Next: Workflows: Create VM from a Snapshot.

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:

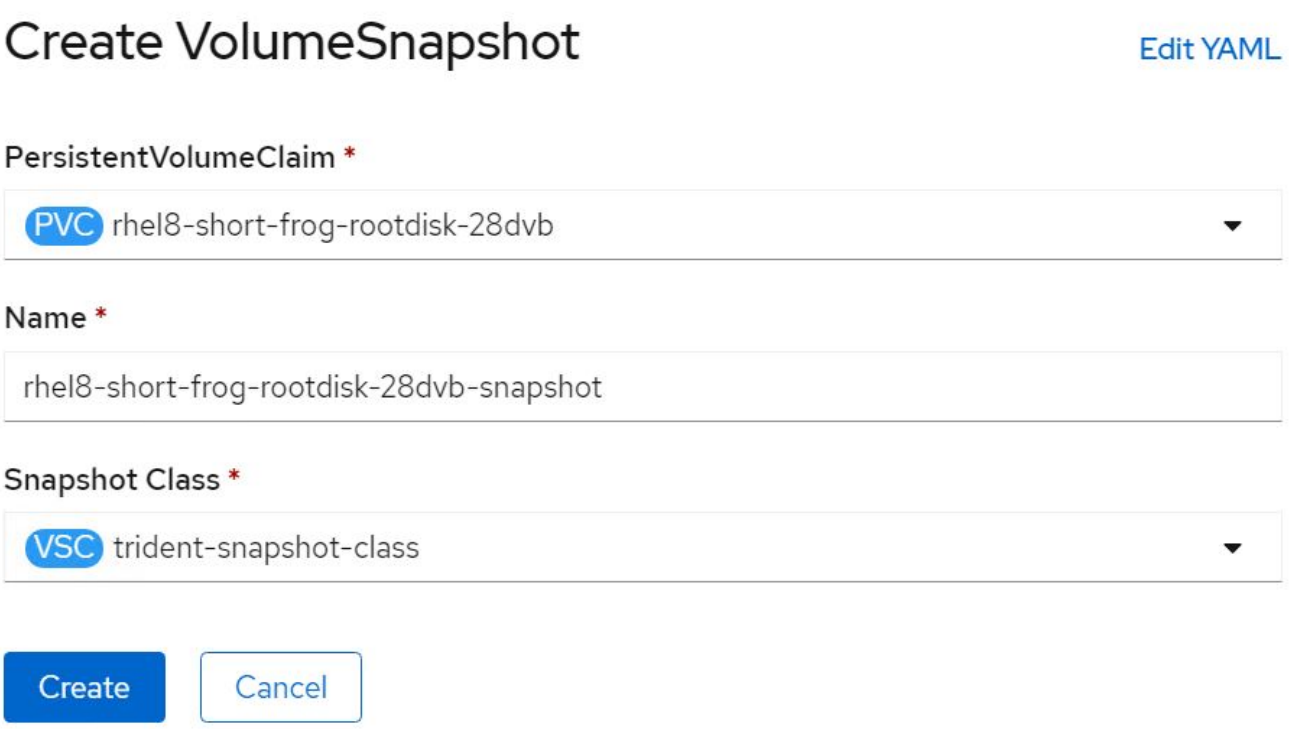
1. Create a VolumeSnapshotClass that can then be used to create a VolumeSnapshot. Navigate to `Storage > VolumeSnapshotClasses` and click Create VolumeSnapshotClass.
2. Enter the name of the Snapshot Class, enter csi.trident.netapp.io for the driver, and click Create.

View shortcuts

```
1  apiVersion: snapshot.storage.k8s.io/v1
2  kind: VolumeSnapshotClass
3  metadata:
4    name: trident-snapshot-class
5  driver: csi.trident.netapp.io
6  deletionPolicy: Delete
7
```

Create Cancel Download

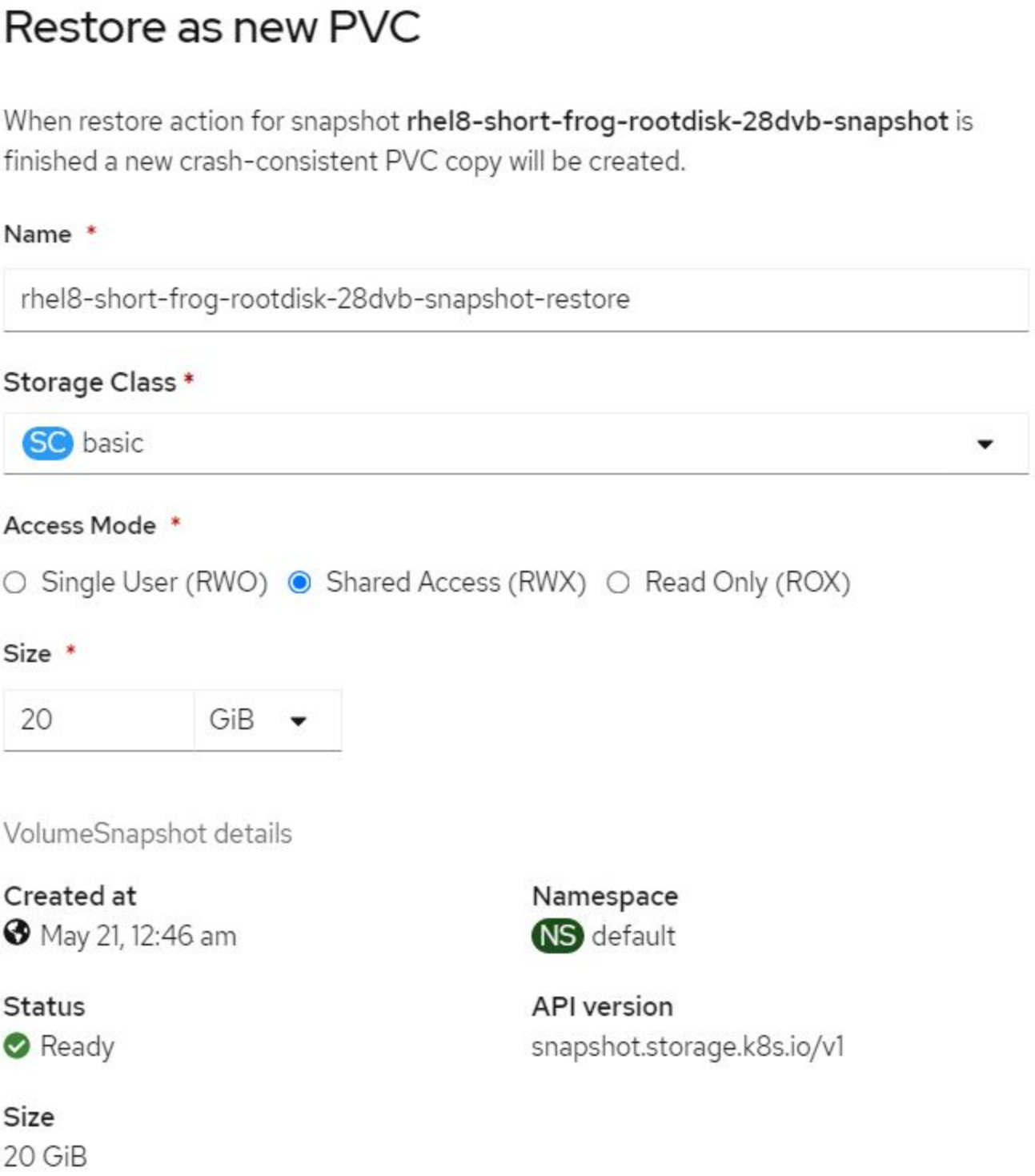
3. Identify the PVC that is attached to the source VM and then create a Snapshot of that PVC. Navigate to `Storage > VolumeSnapshots` and click Create VolumeSnapshots.
4. Select the PVC that you want to create the Snapshot for, enter the name of the Snapshot or accept the default, and select the appropriate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

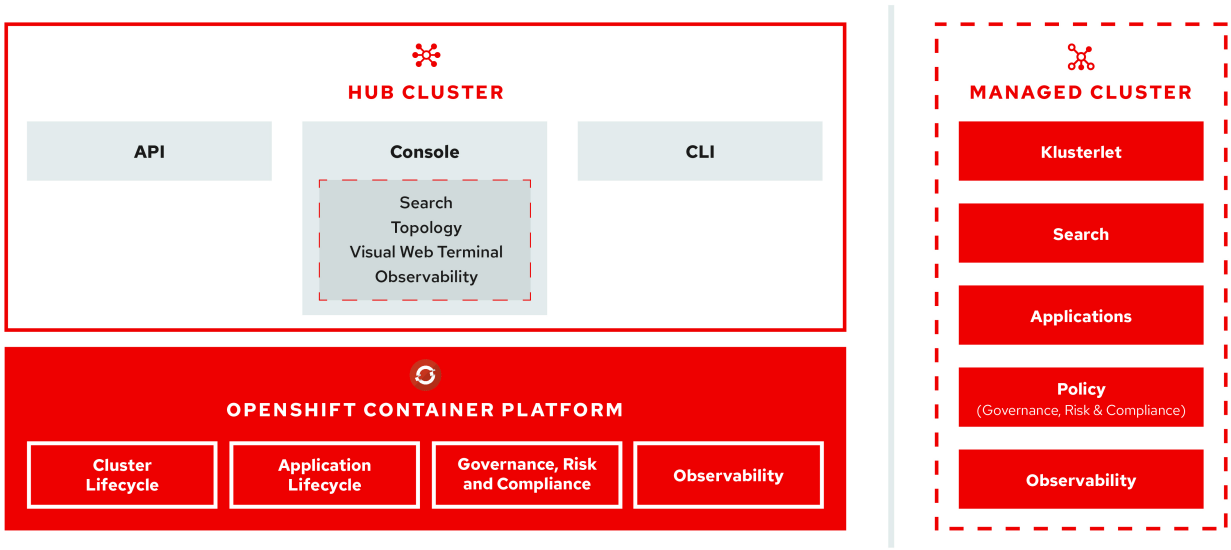
Advanced Cluster Management for Kubernetes: Red Hat OpenShift with NetApp

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 allows the users to:

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 analyse 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: Red Hat OpenShift with NetApp

Prerequisites

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

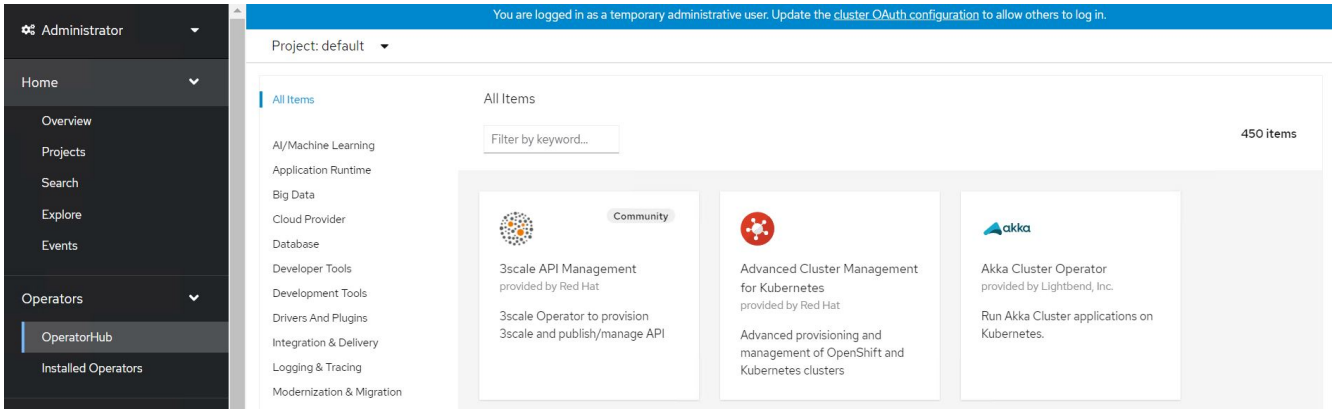
Optionally, if the hub cluster has dedicated nodes for hosting infrastructure components and you would like to install Advanced Cluster Management resources only on those nodes, you need to add tolerations and selectors to those nodes accordingly. For more details, see the documentation here.

Next: Installation.

Deploy Advanced Cluster Management for Kubernetes: Red Hat OpenShift with NetApp

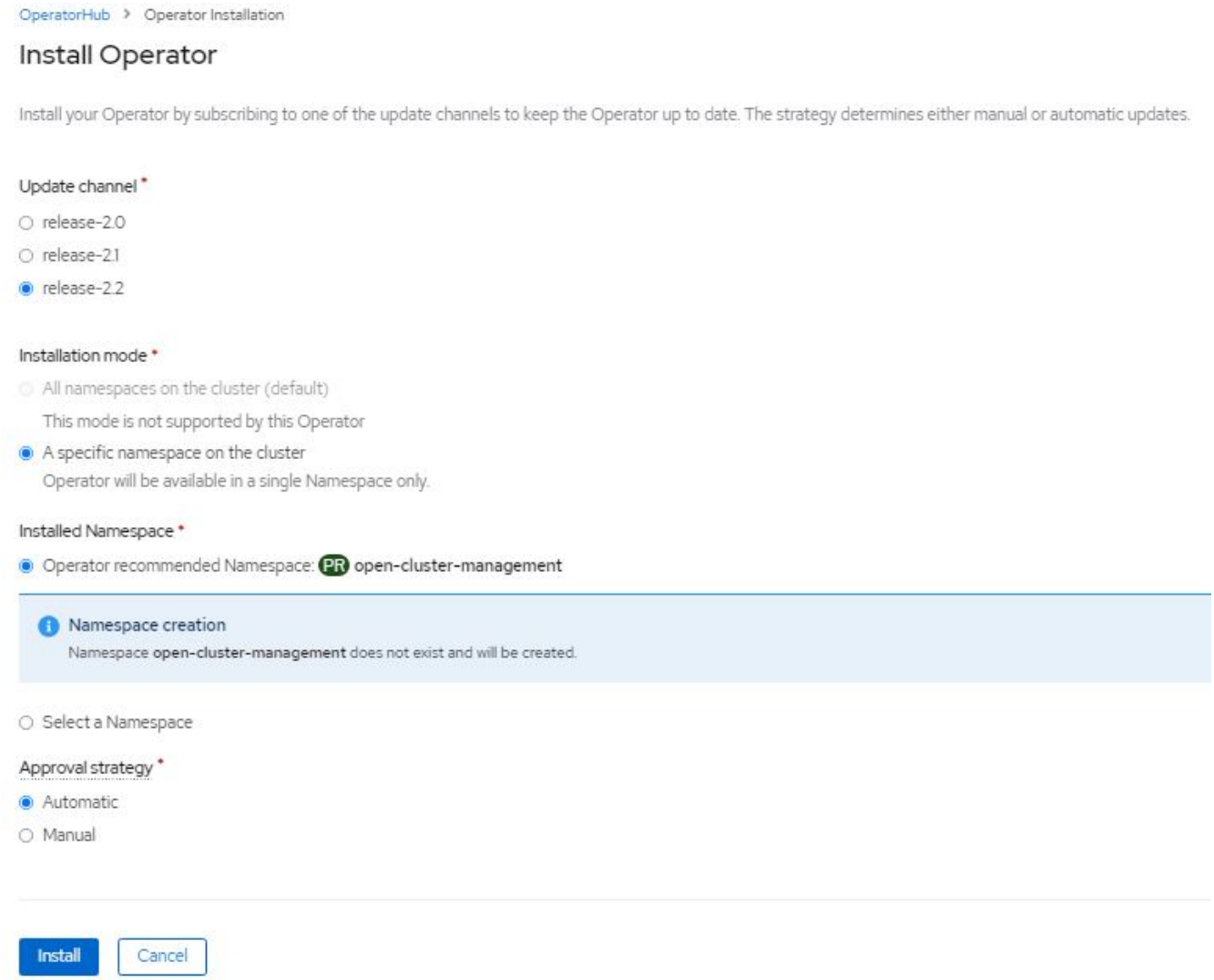
To install Advanced Cluster Management for Kubernetes on an OpenShift cluster, complete the following steps:

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`.

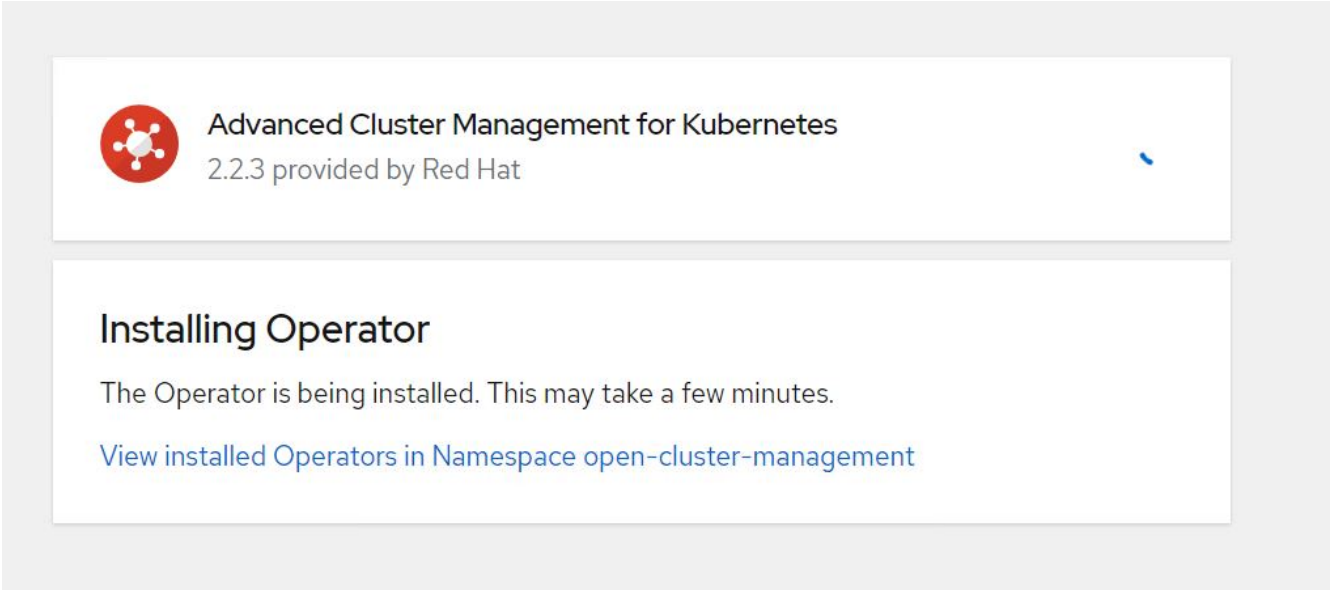


3. Select the `Advanced Cluster Management for Kubernetes` and click `Install`.

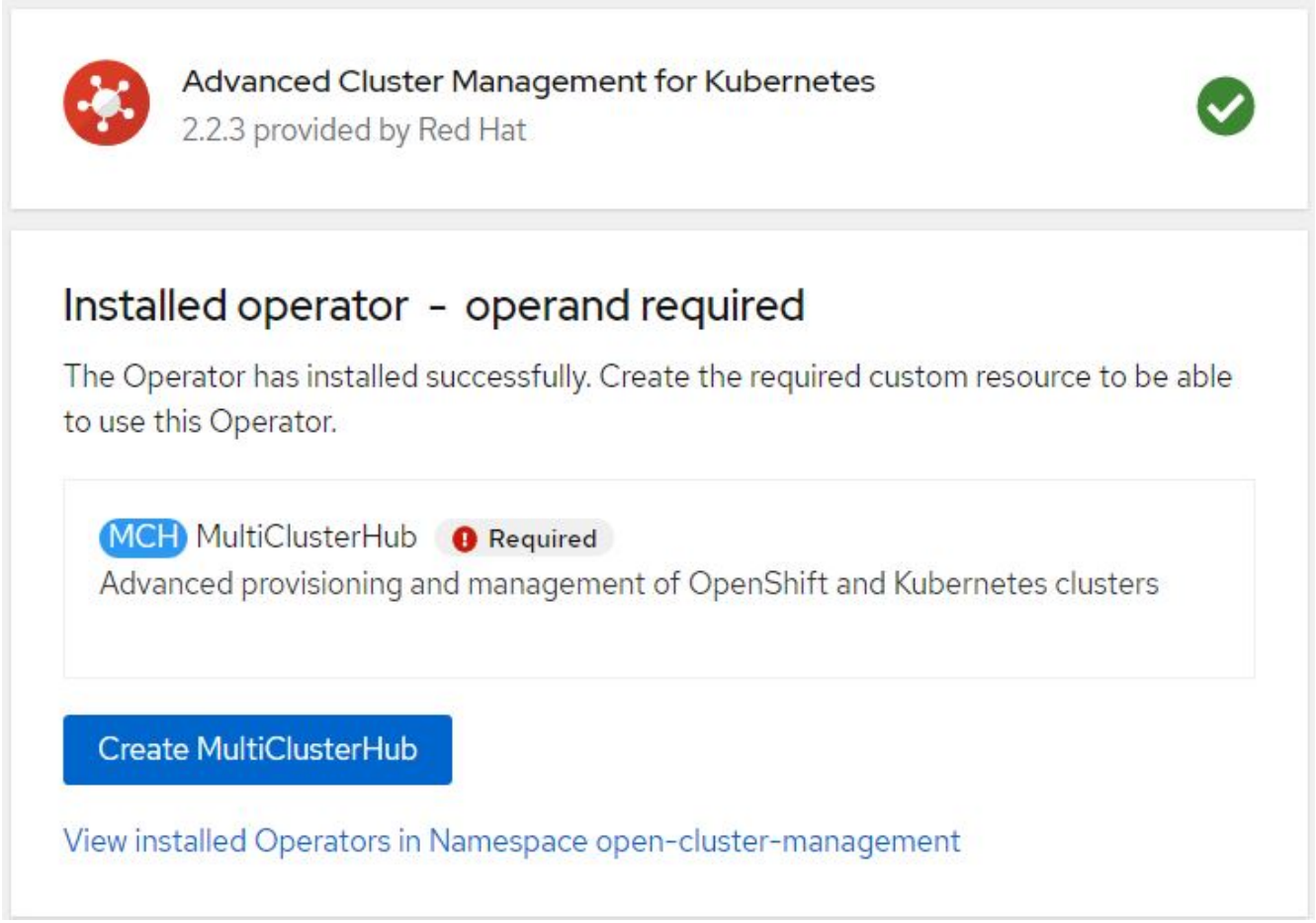
4. On the Install Operator screen, provide the necessary details (NetApp recommends retaining the default parameters) and click `Install`.



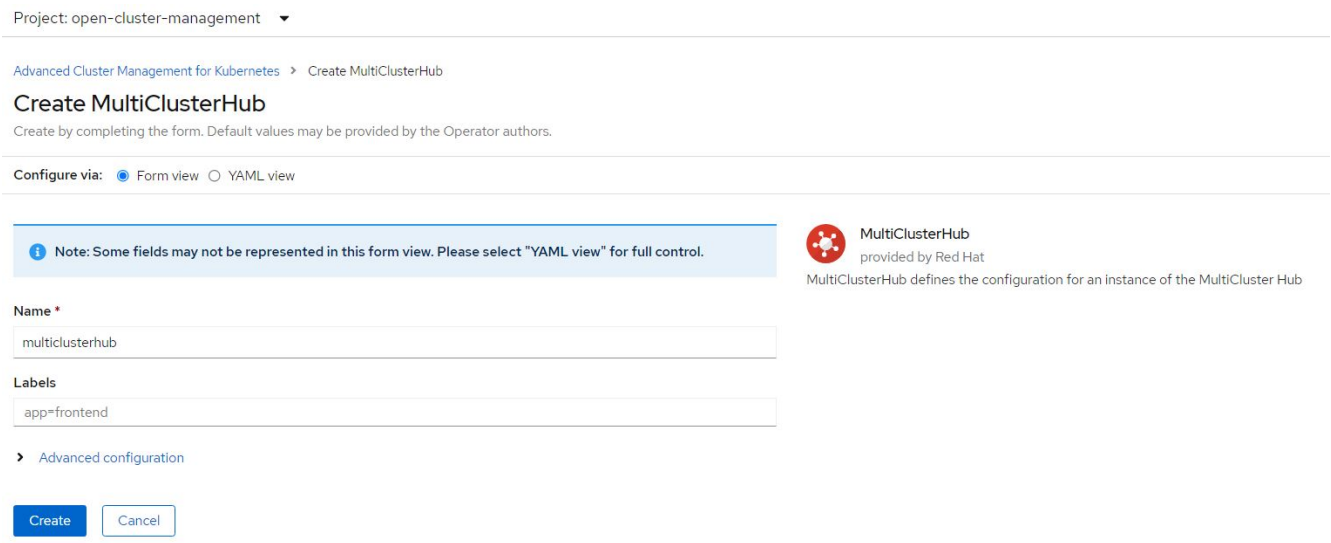
5. Wait for the operator installation to complete.



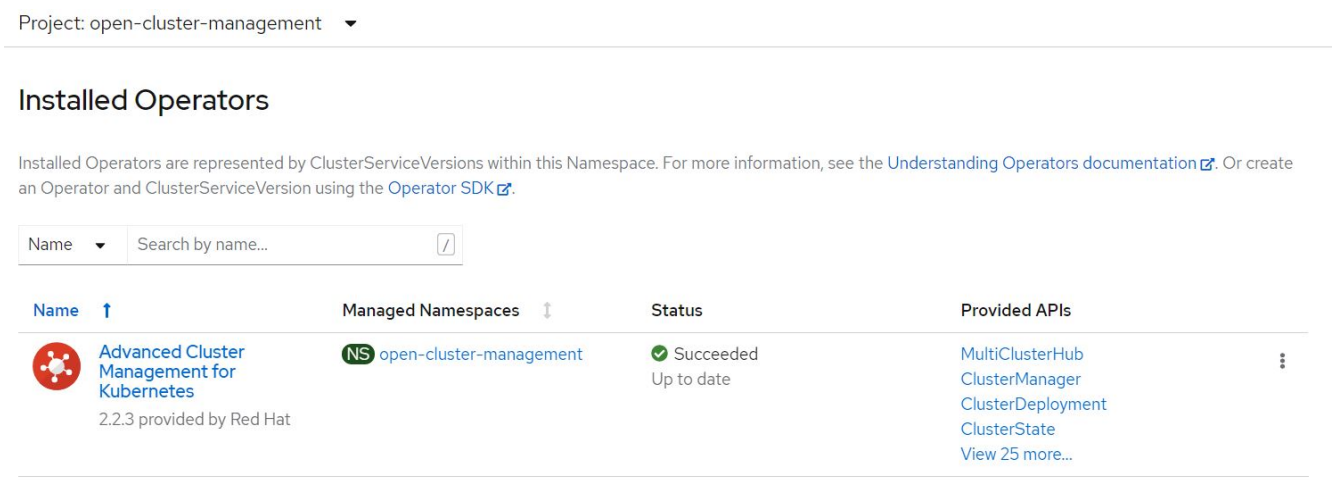
6. After the operator is installed, click `Create MultiClusterHub`.



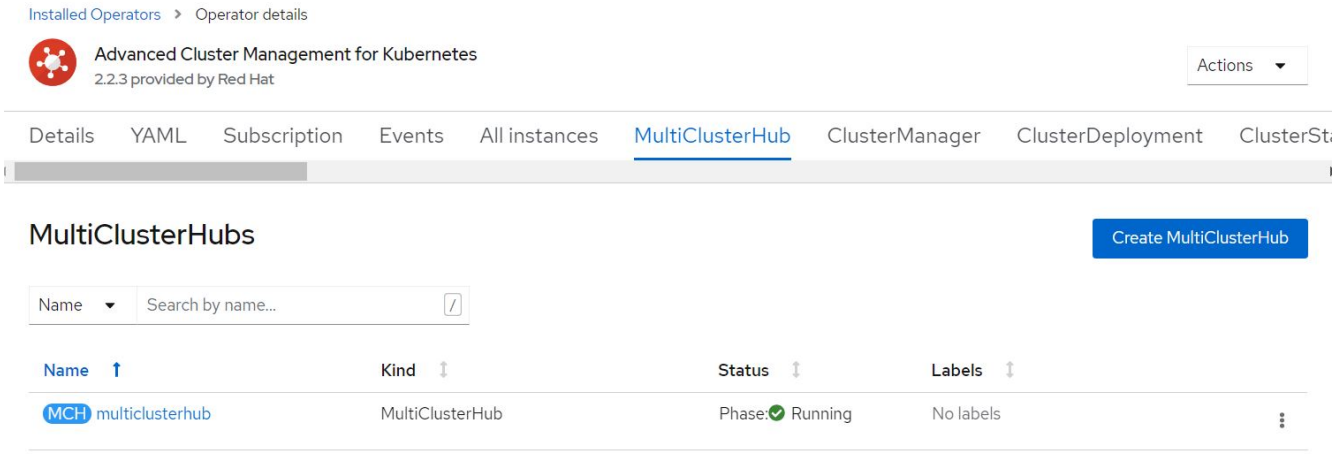
7. On the `Create MultiClusterHub` screen, click `Create` after furnishing the details. This initiates the installation of a multi-cluster hub.



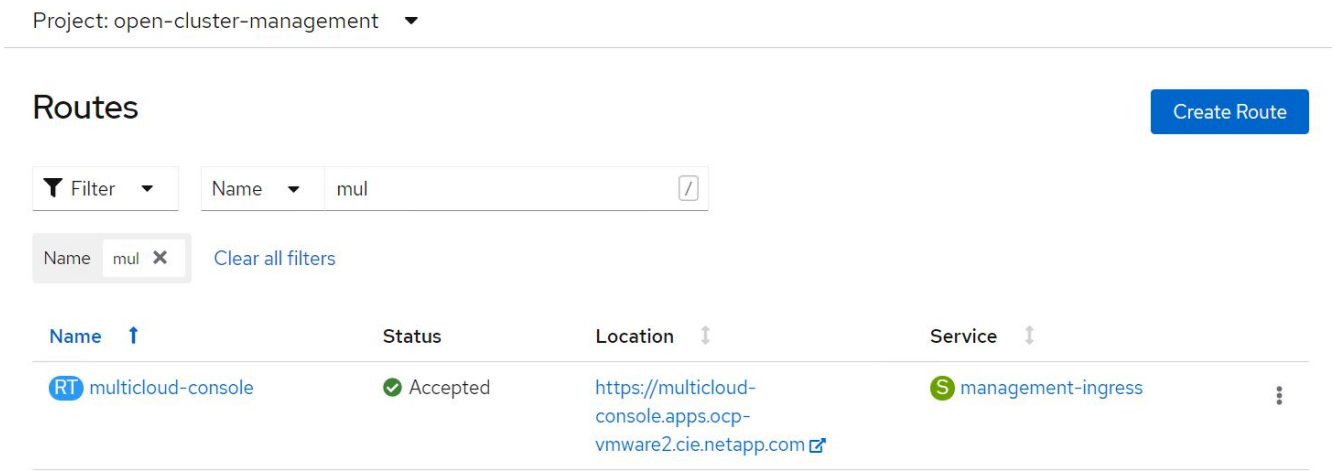
8. After all the pods move to the `Running` state in the open-cluster-management namespace and the operator moves to the `Succeeded` state, Advanced Cluster Management for Kubernetes is installed.



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.

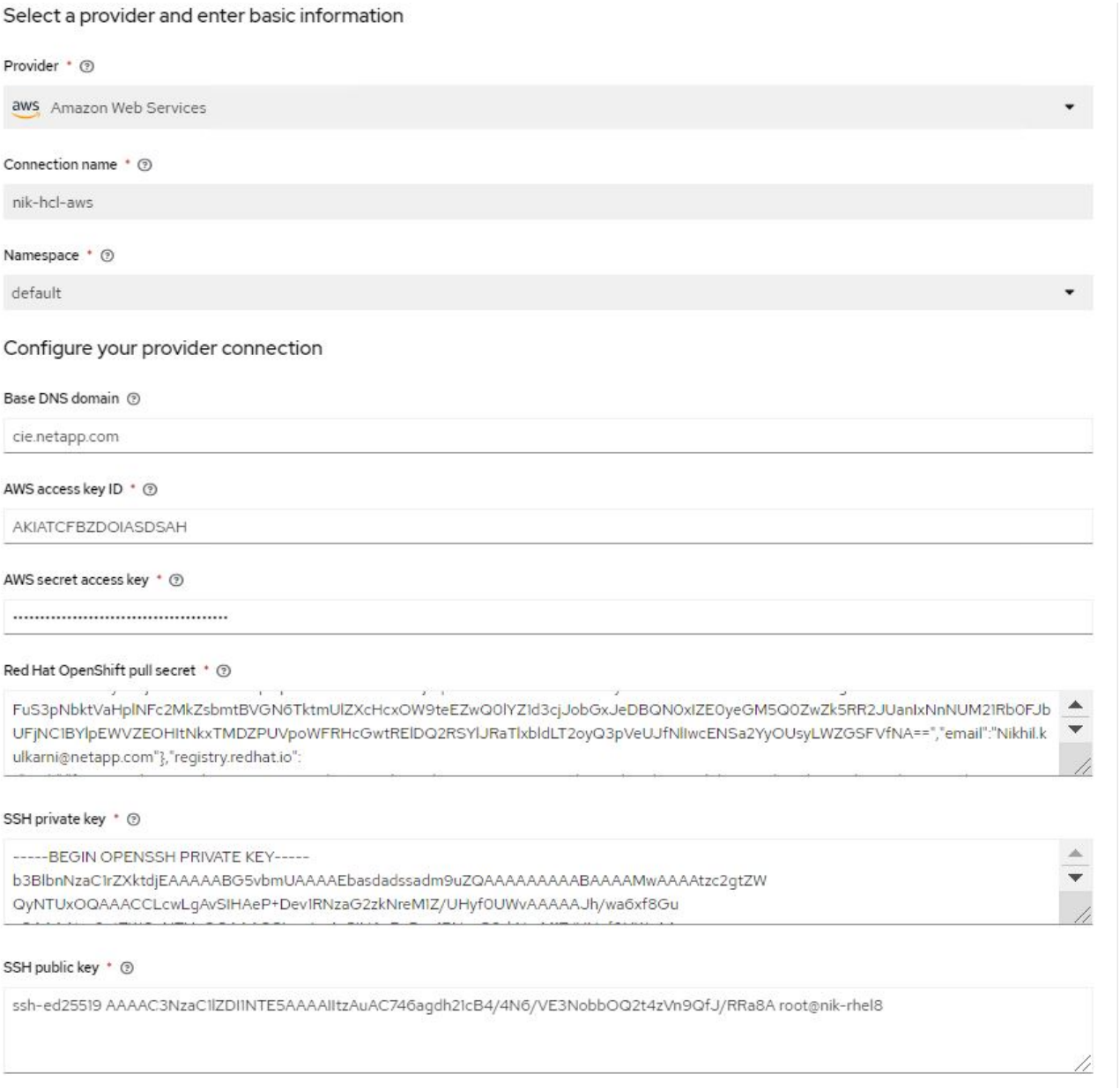
Features

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

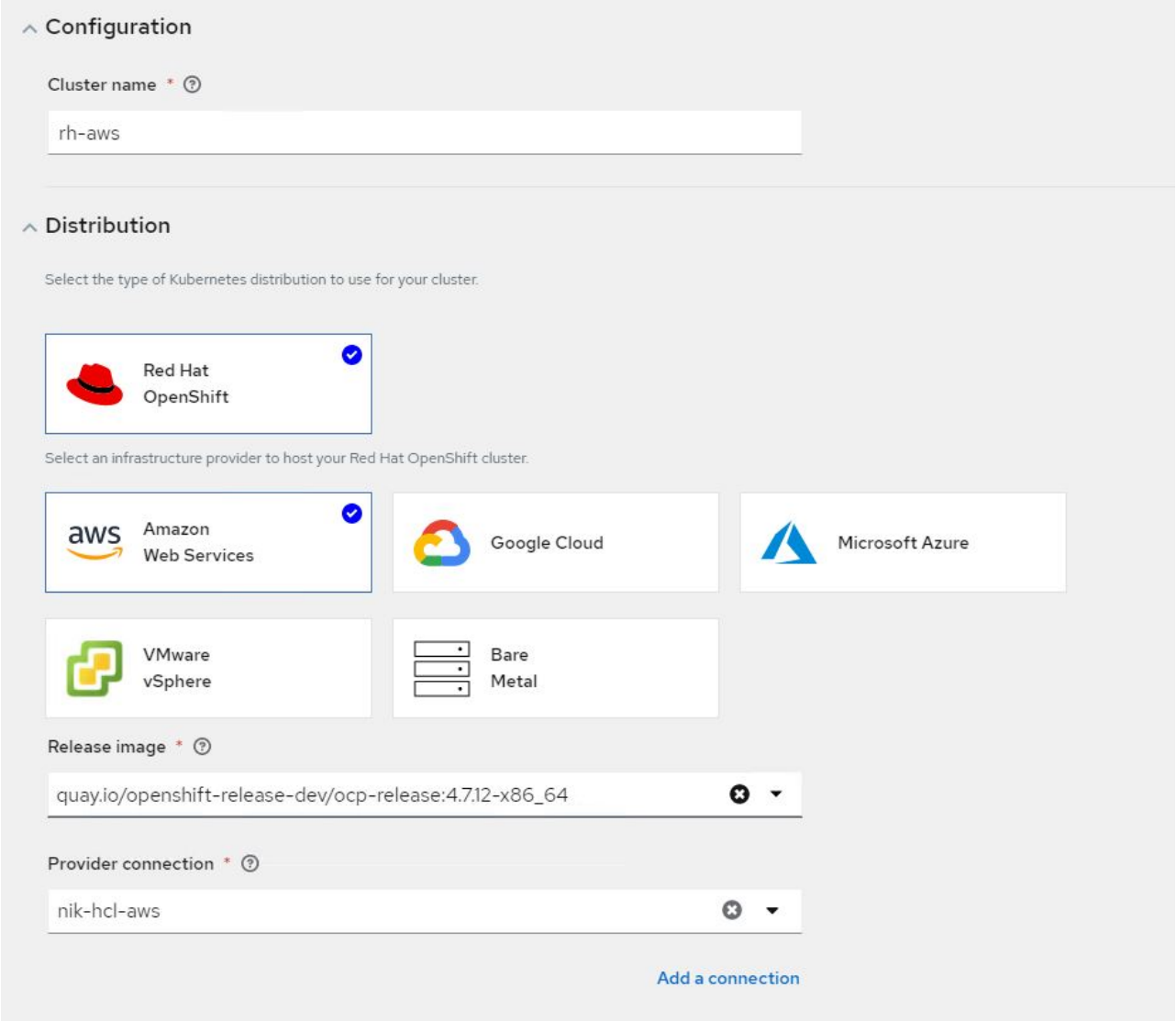
Cluster Lifecycle Management

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 on `Add a connection`, provide all the details corresponding to the selected provider type and click on `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`.



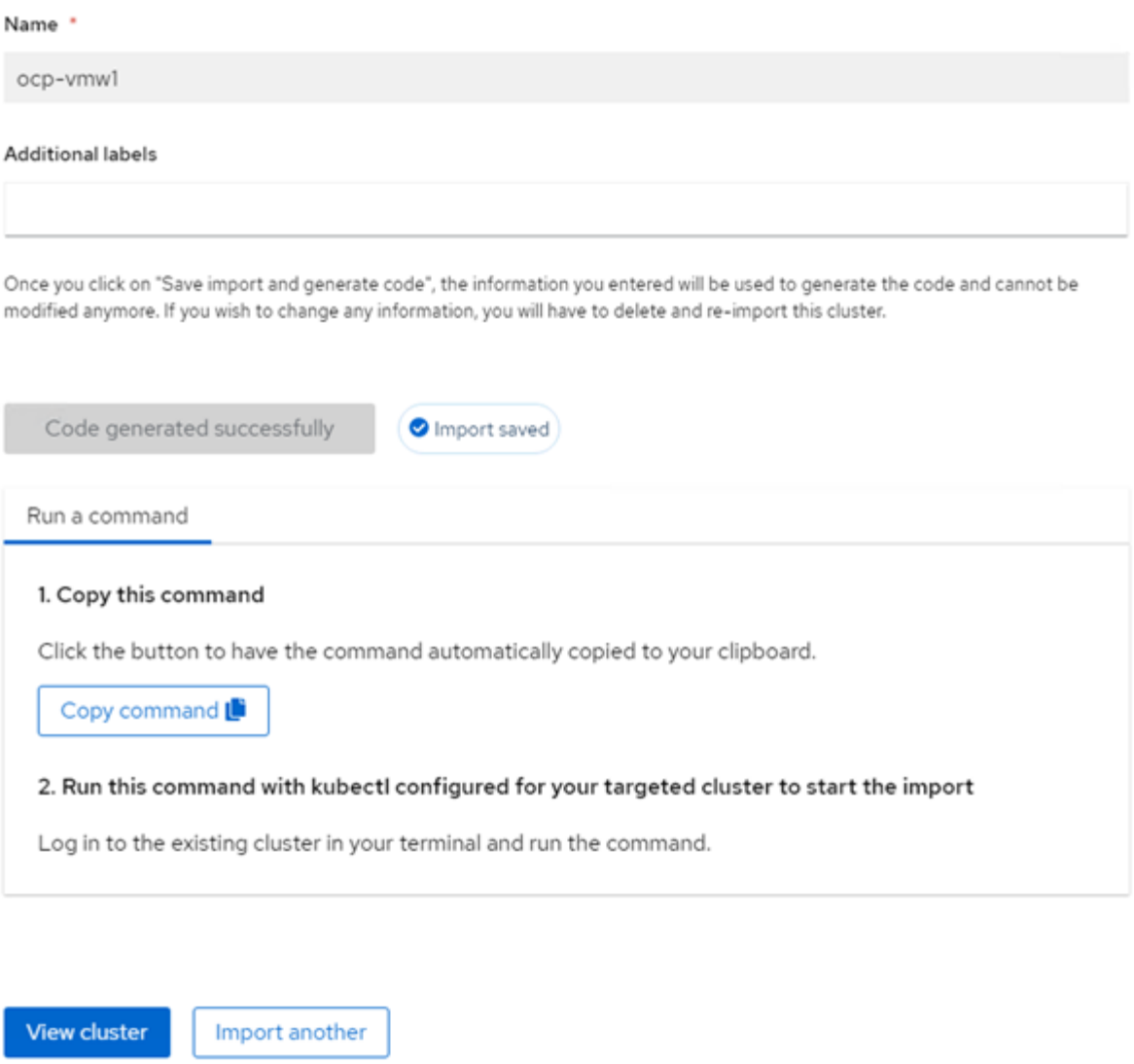
c. After the cluster is created, it appears in the cluster list with the status `Ready`.

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.

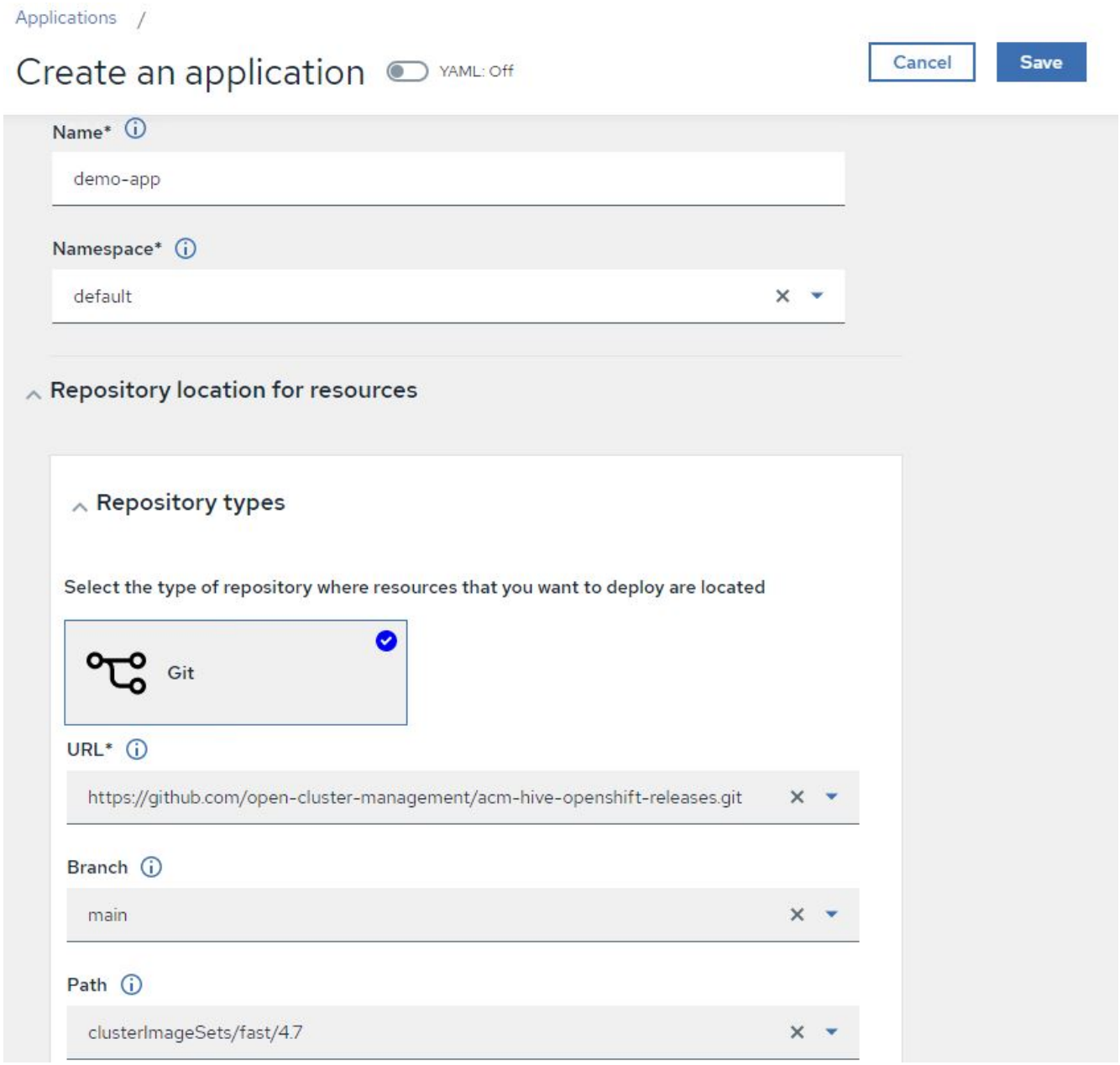
Next: Features - Application Lifecycle Management.

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

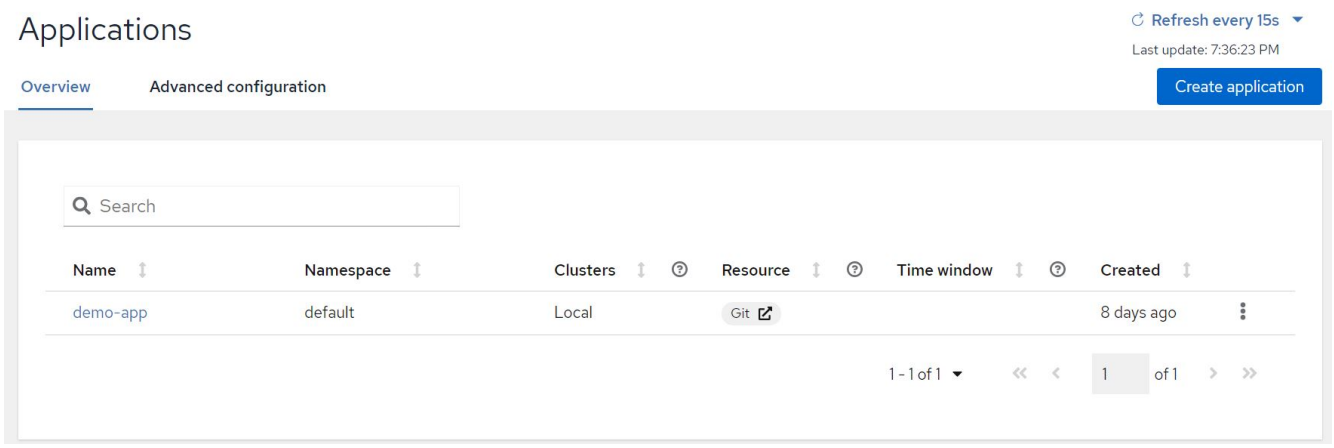
Application lifecycle management

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.

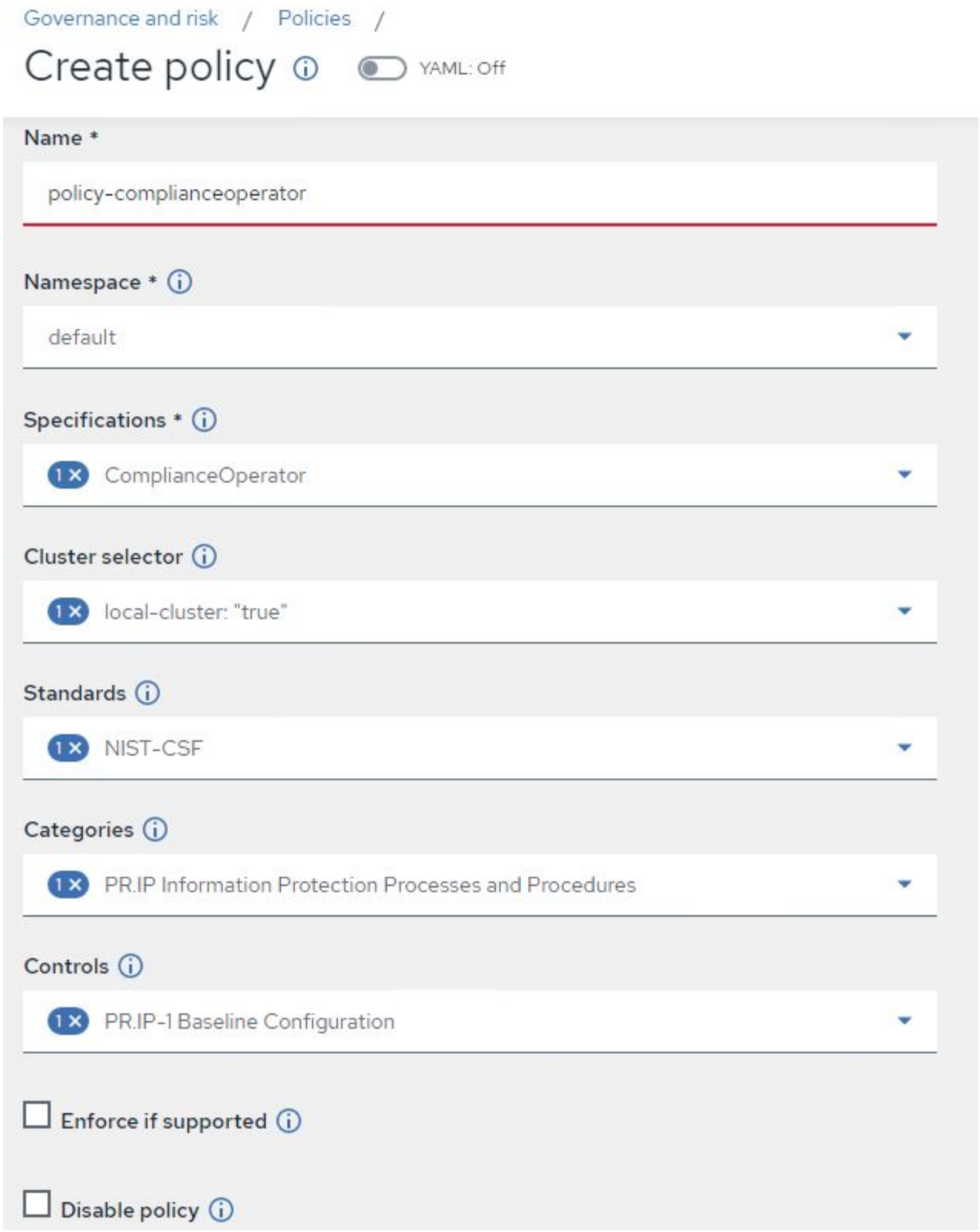
Next: Features - governance and risk.

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

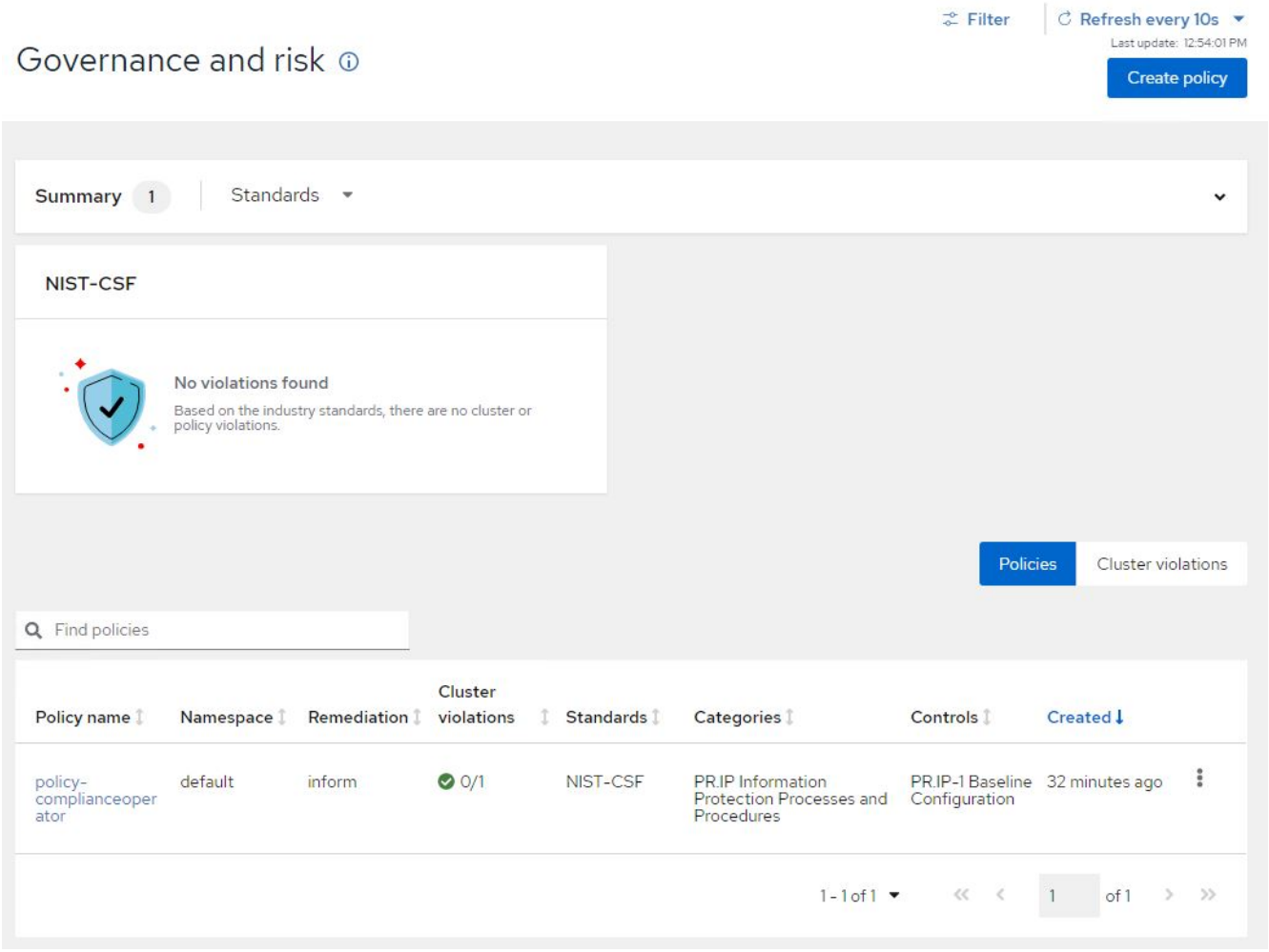
Governance and Risk

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.



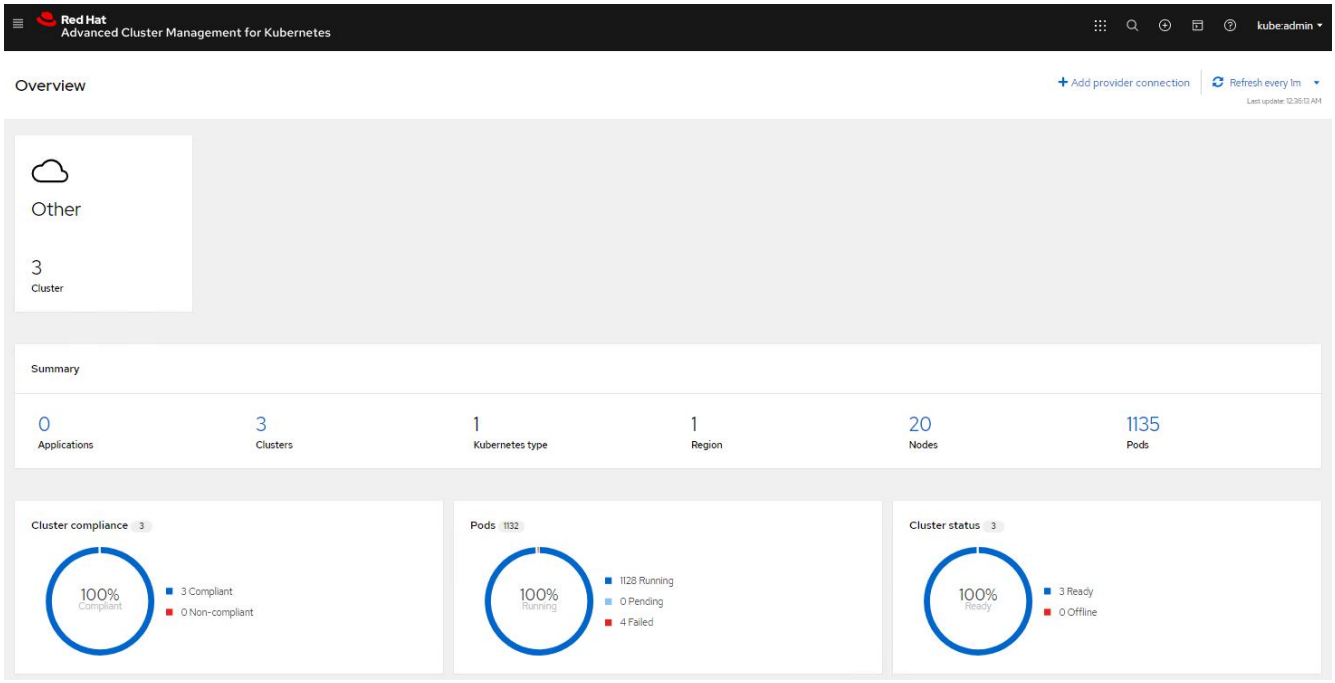
Next: Features - Observability.

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

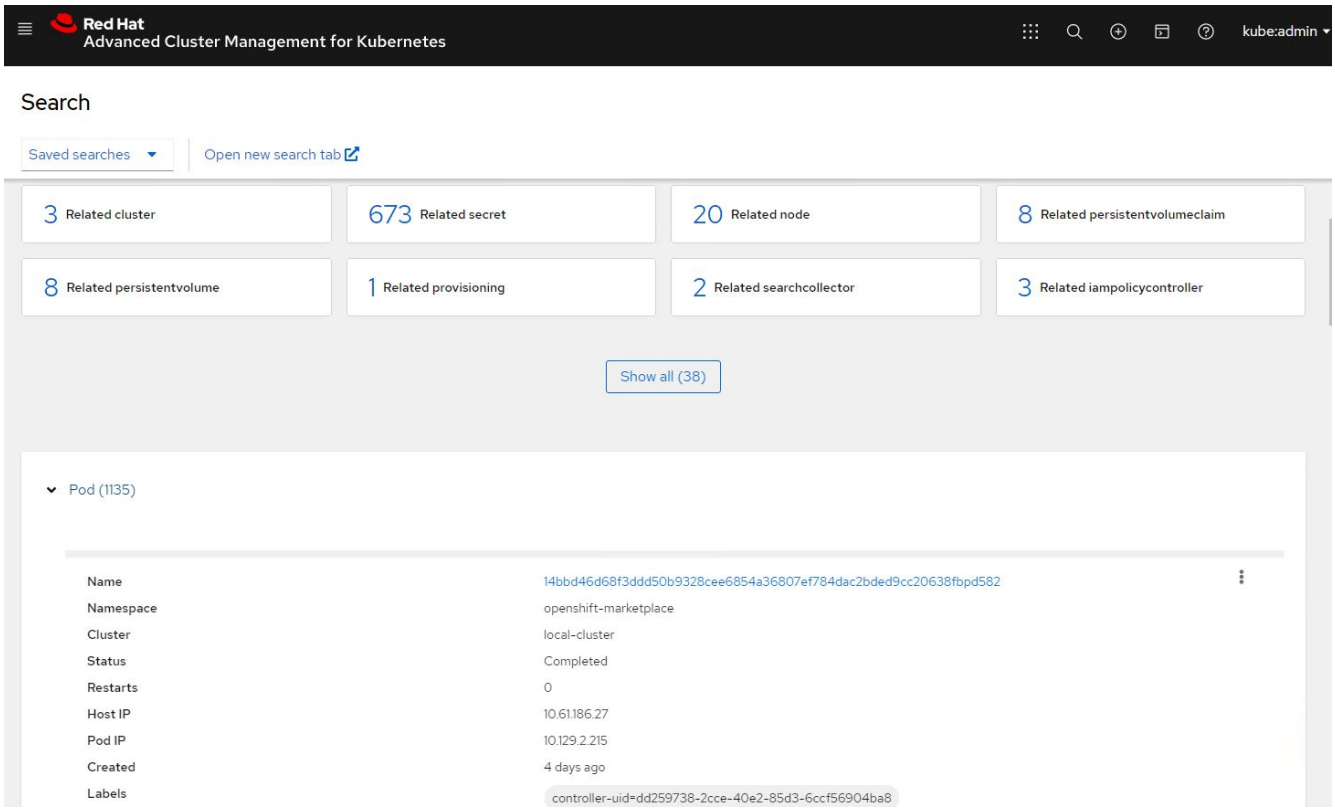
Observability

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

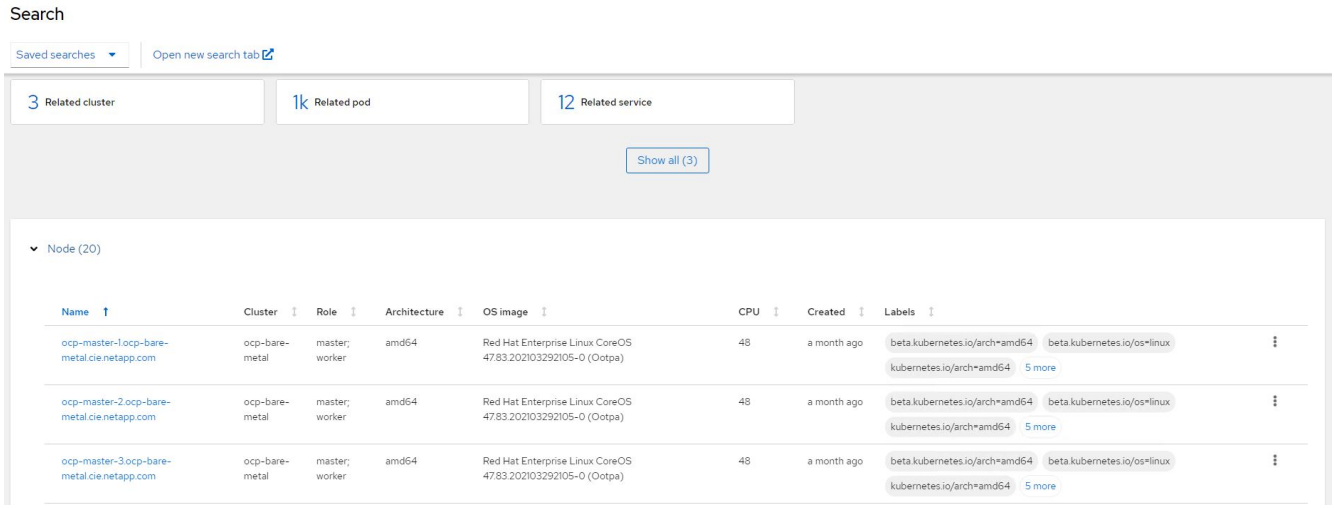
1. Navigate to `Observe Environments` > `Overview`.



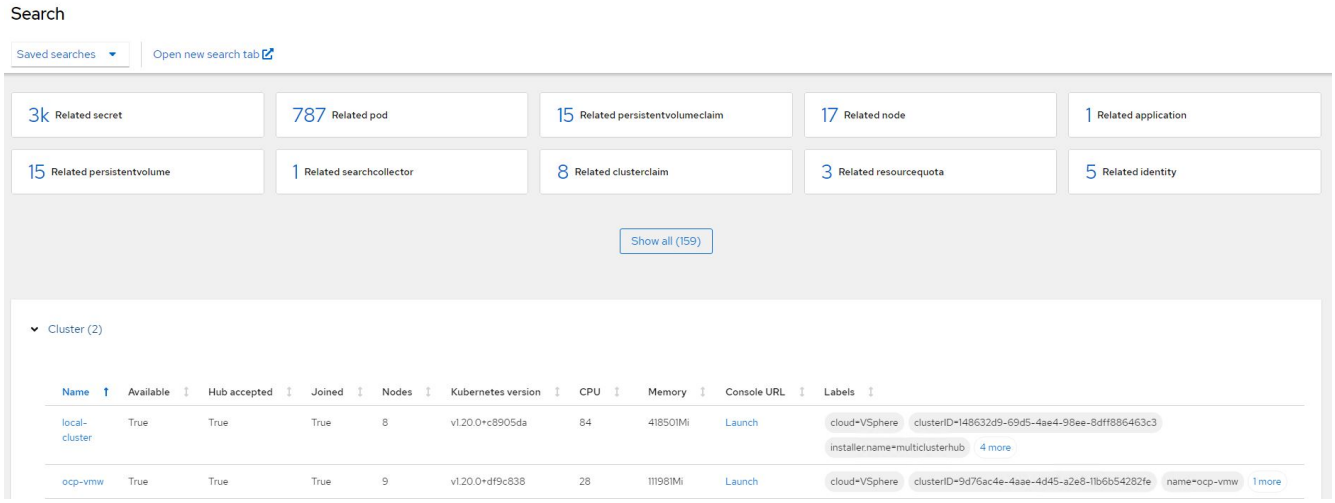
2. All pods and workloads across all clusters are monitored and sorted based on a variety of filters. Click `Pods` to view the corresponding data.



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.

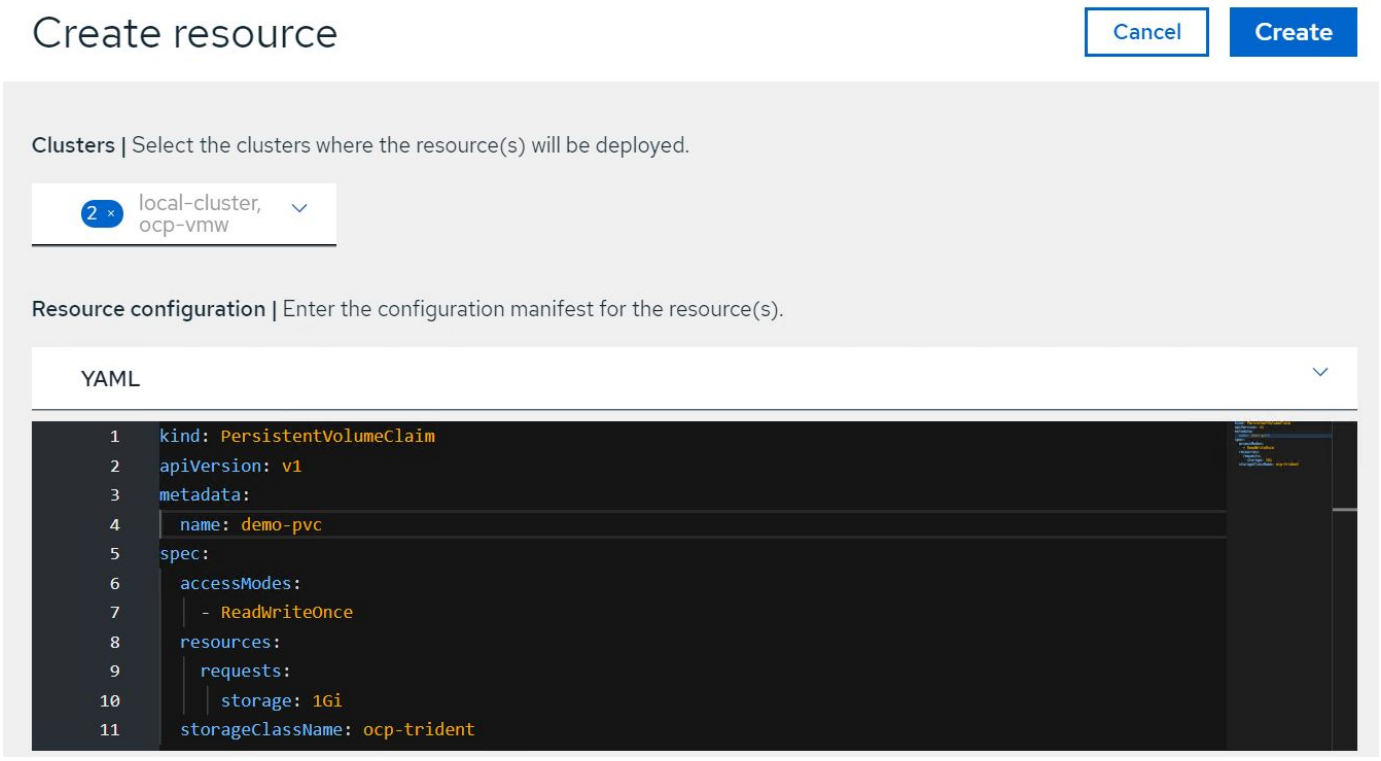


Next: Features - Create Resources.

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 - 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-v21.04/

- 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/