



Red Hat OpenShift with NetApp

NetApp container solutions

NetApp
January 22, 2026

Table of Contents

Red Hat OpenShift with NetApp	1
NVA-1160: Red Hat OpenShift with NetApp	1
Use cases	1
Business value	1
Technology overview	2
Advanced configuration options	2
Current support matrix for validated releases	2
Red Hat Openshift	2
OpenShift Overview	2
OpenShift on Bare Metal	6
OpenShift on Red Hat OpenStack Platform	8
OpenShift on Red Hat Virtualization	12
OpenShift on VMware vSphere	14
Red Hat OpenShift Service on AWS	17
NetApp storage systems	17
NetApp ONTAP	17
NetApp Element: Red Hat OpenShift with NetApp	19
NetApp storage integrations	21
Learn about NetApp Trident integration with Red Hat OpenShift	21
NetApp Trident	22
Advanced configuration options	40
Explore load balancer options	40
Creating Private Image Registries	60
Solution validation and use cases	66
Solution Validation and Use Cases: Red Hat OpenShift with NetApp	66
Deploy a Jenkins CI/CD Pipeline with Persistent Storage: Red Hat OpenShift with NetApp	66
Configure multi-tenancy	76
Advanced Cluster Management for Kubernetes	96
Advanced Cluster Management for Kubernetes: Red Hat OpenShift with NetApp - Overview	96
Deploy ACM for Kubernetes	97
Data protection for Container Apps and VMs using Trident protect	111
Data protection for Container Apps and VMs using 3rd party tools	111
Additional resources to learn about Red Hat OpenShift Virtualization integration with NetApp storage	112

Red Hat OpenShift with NetApp

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 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 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 Red Hat OpenShift IPI in the customer's choice of data center environment.

Technology overview

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

Red Hat OpenShift Container Platform

Red Hat OpenShift Container Platform is a fully supported enterprise Kubernetes platform. Red Hat makes several enhancements to open-source Kubernetes to deliver an application platform with all the components fully integrated to build, deploy, and manage containerized applications.

For more information visit the OpenShift website [here](#).

NetApp storage systems

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

For more information visit the NetApp website [here](#).

NetApp storage integrations

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

For more information, visit the Trident website [here](#).

Advanced configuration options

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

Current support matrix for validated releases

Technology	Purpose	Software version
NetApp ONTAP	Storage	9.8, 9.9.1, 9.12.1
NetApp Element	Storage	12.3
NetApp Trident	Storage Orchestration	22.01.0, 23.04, 23.07, 23.10, 24.02
Red Hat OpenShift	Container orchestration	4.6 EUS, 4.7, 4.8, 4.10, 4.11, 4.12, 4.13, 4.14
VMware vSphere	Data center virtualization	7.0, 8.0.2

Red Hat OpenShift

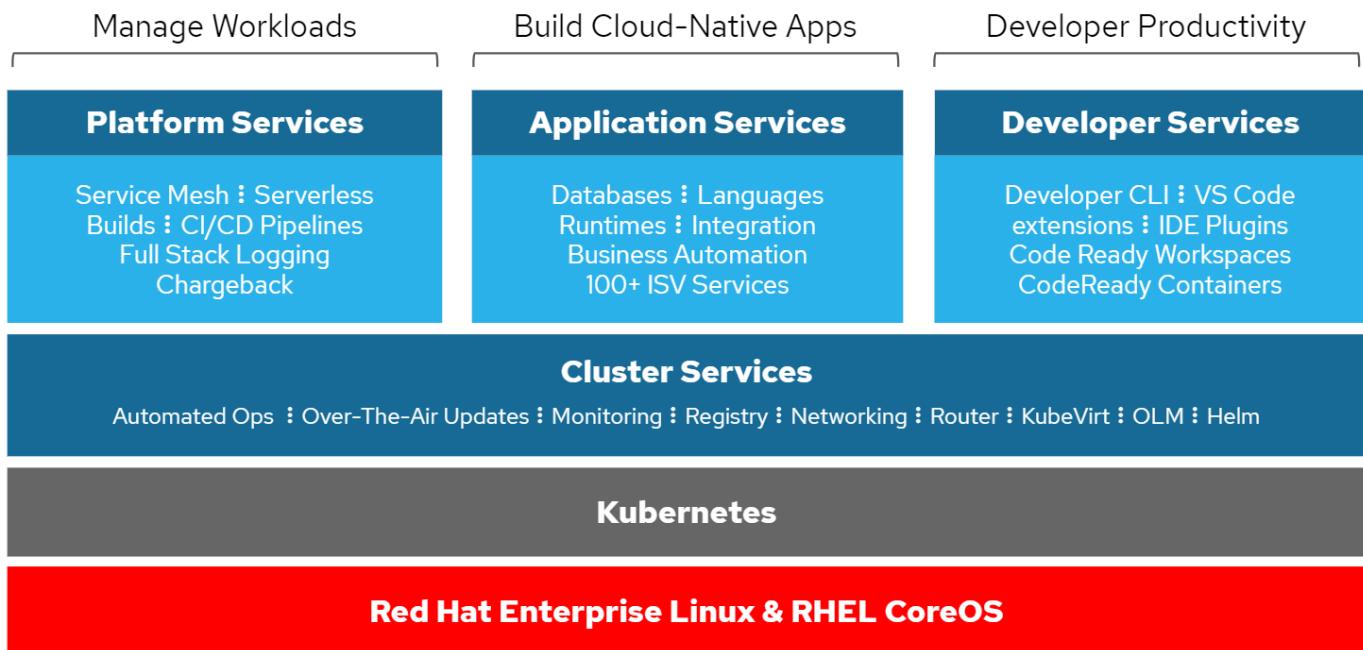
OpenShift Overview

The Red Hat OpenShift Container Platform unites development and IT operations on a single platform to build, deploy, and manage applications consistently across on-premises

and hybrid cloud infrastructures. Red Hat OpenShift is built on open-source innovation and industry standards, including Kubernetes and Red Hat Enterprise Linux CoreOS, the world's leading enterprise Linux distribution designed for container-based workloads. OpenShift is part of the Cloud Native Computing Foundation (CNCF) Certified Kubernetes program, providing portability and interoperability of container workloads.

Red Hat OpenShift provides the following capabilities:

- **Self-service provisioning** Developers can quickly and easily create applications on demand from the tools that they use most, while operations retain full control over the entire environment.
- **Persistent storage** By providing support for persistent storage, OpenShift Container Platform allows you to run both stateful applications and cloud-native stateless applications.
- **Continuous integration and continuous development (CI/CD)** This source-code platform manages build and deployment images at scale.
- **Open-source standards** These standards incorporate the Open Container Initiative (OCI) and Kubernetes for container orchestration, in addition to other open-source technologies. You are not restricted to the technology or to the business roadmap of a specific vendor.
- **CI/CD pipelines** OpenShift provides out-of-the-box support for CI/CD pipelines so that development teams can automate every step of the application delivery process and make sure it's executed on every change that is made to the code or configuration of the application.
- **Role-Based Access Control (RBAC)** This feature provides team and user tracking to help organize a large developer group.
- **Automated build and deploy** OpenShift gives developers the option to build their containerized applications or have the platform build the containers from the application source code or even the binaries. The platform then automates deployment of these applications across the infrastructure based on the characteristic that was defined for the applications. For example, how quantity of resources that should be allocated and where on the infrastructure they should be deployed in order for them to be compliant with third-party licenses.
- **Consistent environments** OpenShift makes sure that the environment provisioned for developers and across the lifecycle of the application is consistent from the operating system, to libraries, runtime version (for example, Java runtime), and even the application runtime in use (for example, tomcat) in order to remove the risks originated from inconsistent environments.
- **Configuration management** Configuration and sensitive data management is built in to the platform to make sure that a consistent and environment agnostic application configuration is provided to the application no matter which technologies are used to build the application or which environment it is deployed.
- **Application logs and metric.** Rapid feedback is an important aspect of application development. OpenShift integrated monitoring and log management provides immediate metrics back to developers in order for them to study how the application is behaving across changes and be able to fix issues as early as possible in the application lifecycle.
- **Security and container catalog** OpenShift offers multitenancy and protects the user from harmful code execution by using established security with Security-Enhanced Linux (SELinux), CGroups, and Secure Computing Mode (seccomp) to isolate and protect containers. It also provides encryption through TLS certificates for the various subsystems and access to Red Hat certified containers (access.redhat.com/containers) that are scanned and graded with a specific emphasis on security to provide certified, trusted, and secure application containers to end users.



Physical



Virtual



Private cloud



Public cloud



Managed cloud
(Azure, AWS, IBM, Red Hat)

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 OpenShift clusters for dev, test, and production environments.

IPI installation of Red Hat OpenShift

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

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

Install OpenShift Container Platform 4

Select an infrastructure provider

 Run on Amazon Web Services	 Run on Microsoft Azure	 Run on Google Cloud Platform	 Run on VMware vSphere
 Run on Red Hat OpenStack	 Run on Red Hat Virtualization	 Run on Bare Metal	 Run on IBM Z
 Run on Power	 Run on Laptop Powered by Red Hat CodeReady Containers		

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

Downloads

OpenShift installer
Download and extract the install program for your operating system and place the file in the directory where you will store the installation configuration files. Note: The OpenShift install program is only available for Linux and macOS at this time.

Linux

Pull secret
Download or copy your pull secret. The install program will prompt you for your pull secret during installation.

Command-line interface
Download the OpenShift command-line tools and add them to your PATH.

Linux

When the installer is complete you will see the console URL and credentials for accessing your new cluster. A kubeconfig file will also be generated for you to use with the `oc` CLI tools you downloaded.

4. Follow the [installation instructions](#) provided by Red Hat to deploy to your environment of choice.

NetApp validated OpenShift deployments

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

- [OpenShift on Bare Metal](#)
- [OpenShift on Red Hat OpenStack Platform](#)
- [OpenShift on Red Hat Virtualization](#)
- [OpenShift on VMware vSphere](#)

OpenShift on Bare Metal

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

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

OpenShift on Bare Metal provides the following features:

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

Network design

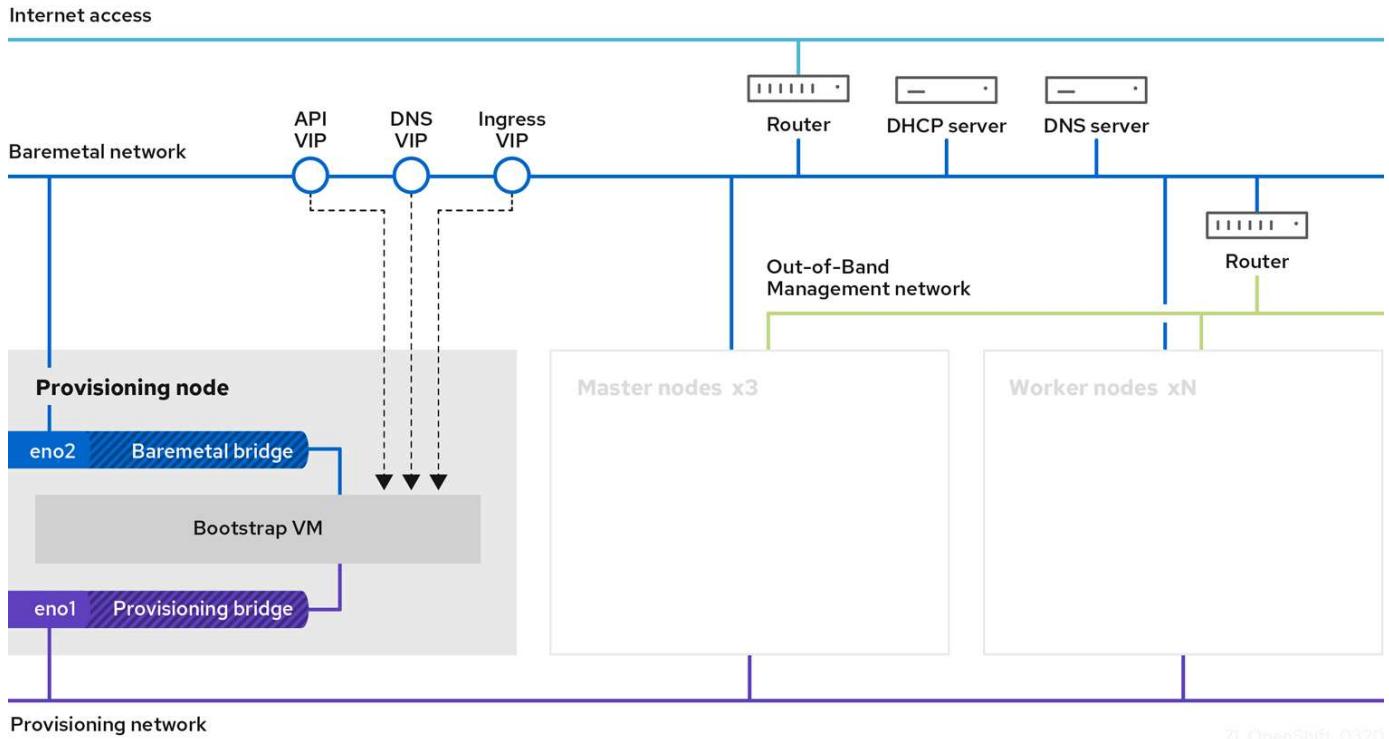
The Red Hat OpenShift on NetApp solution uses two data switches to provide primary data connectivity at 25Gbps. It also uses two management switches that provide connectivity at 1Gbps for in-band management for the storage nodes and out-of-band management for IPMI functionality.

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

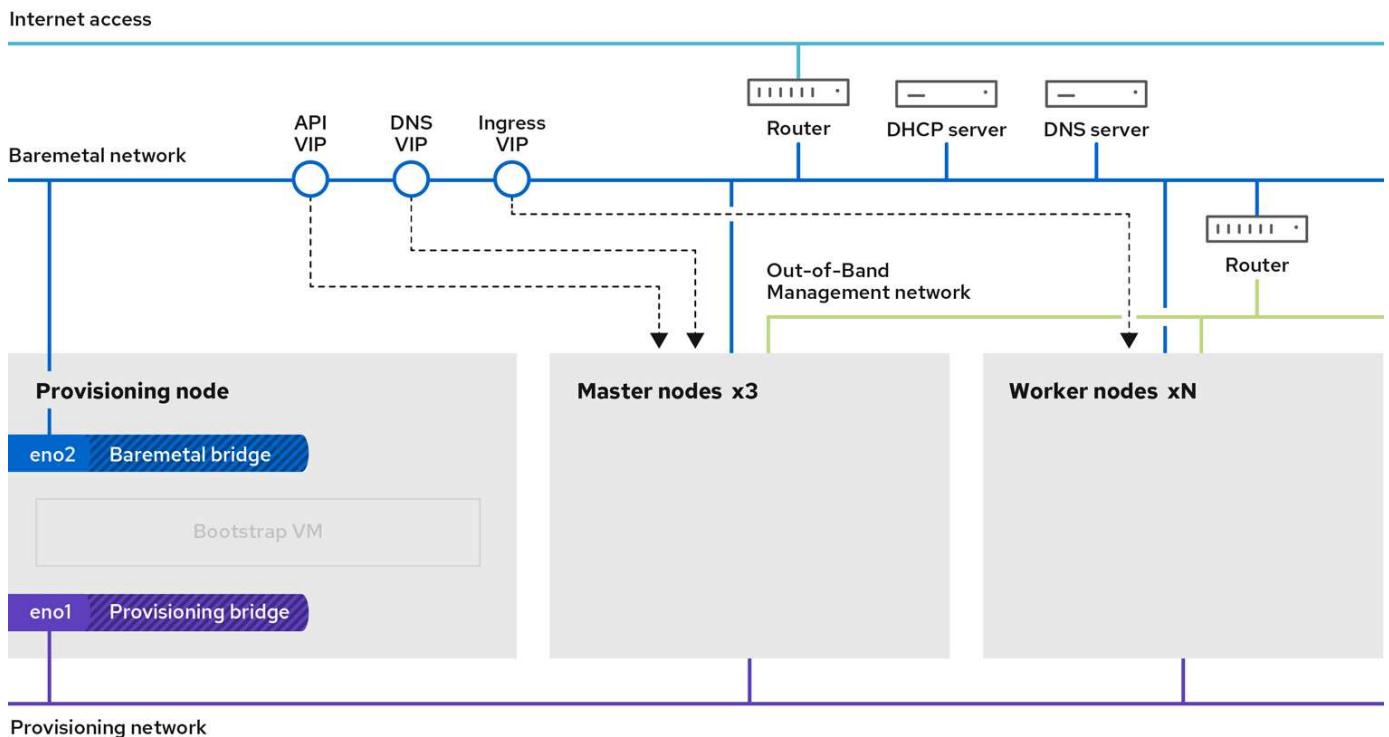
- **Provisioning network** This network is used to boot the bare-metal nodes and install the necessary images and packages to deploy the OpenShift cluster.
- **Bare-metal network** This network is used for public-facing communication of the cluster after it is deployed.

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

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



7L_OpenShift_0320



VLAN requirements

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

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



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

Network infrastructure support resources

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

- At least one DNS server that provides a full host-name resolution accessible from the in-band management network and the VM network.
- At least one NTP server that is accessible from the in-band management network and the VM network.
- (Optional) Outbound internet connectivity for both the in-band management network and the VM network.

OpenShift on Red Hat OpenStack Platform

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

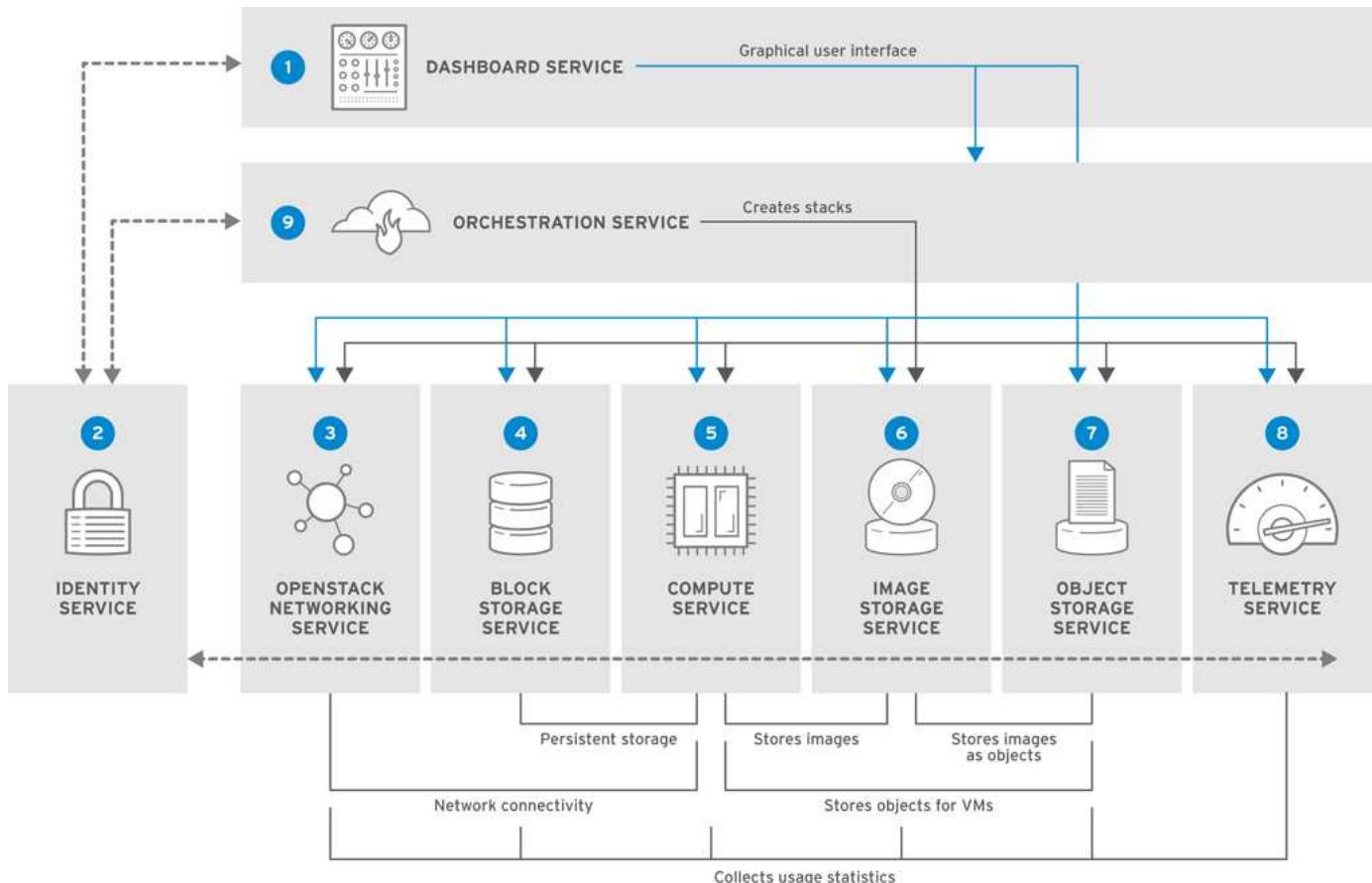
OSP is an infrastructure-as-a-service (IaaS) cloud implemented by a collection of control services that manage compute, storage, and networking resources. The environment is managed using a web-based interface that allows administrators and users to control, provision, and automate OpenStack resources.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

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

For more information about OSP see the [Red Hat OpenStack Platform website](#).

OpenStack services

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



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

Network design

The Red Hat OpenShift with NetApp solution uses two data switches to provide primary data connectivity at 25Gbps. It also uses two additional management switches that provide connectivity at 1Gbps for in-band

management for the storage nodes and out-of-band management for IPMI functionality.

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

VLAN requirements

Red Hat OpenShift with NetApp is designed to logically separate network traffic for different purposes by using virtual local area networks (VLANs). This configuration can be scaled to meet customer demands or to provide further isolation for specific network services. The following table lists the VLANs that are required to implement the solution while validating the solution at NetApp.

VLANs	Purpose	VLAN ID
Out-of-band management network	Network used for management of physical nodes and IPMI service for Ironic.	16
Storage infrastructure	Network used for controller nodes to map volumes directly to support infrastructure services like Swift.	201
Storage Cinder	Network used to map and attach block volumes directly to virtual instances deployed in the environment.	202
Internal API	Network used for communication between the OpenStack services using API communication, RPC messages, and database communication.	301
Tenant	Neutron provides each tenant with their own networks via tunneling through VXLAN. Network traffic is isolated within each tenant network. Each tenant network has an IP subnet associated with it, and network namespaces mean that multiple tenant networks can use the same address range without causing conflicts.	302
Storage management	OpenStack Object Storage (Swift) uses this network to synchronize data objects between participating replica nodes. The proxy service acts as the intermediary interface between user requests and the underlying storage layer. The proxy receives incoming requests and locates the necessary replica to retrieve the requested data.	303
PXE	The OpenStack Director provides PXE boot as a part of the Ironic bare metal provisioning service to orchestrate the installation of the OSP Overcloud.	3484
External	Publicly available network which hosts the OpenStack Dashboard (Horizon) for graphical management and allows for public API calls to manage OpenStack services.	3485
In-band management network	Provides access for system administration functions such as SSH access, DNS traffic, and Network Time Protocol (NTP) traffic. This network also acts as a gateway for non-controller nodes.	3486

Network infrastructure support resources

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

- At least one DNS server which provides a full host-name resolution.
- At least three NTP servers which can keep time synchronized for the servers in the solution.

- (Optional) Outbound internet connectivity for the OpenShift environment.

Best practices for production deployments

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

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

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

Because Red Hat OpenShift initially deploys with three master nodes, a two-node configuration might cause at least two masters to occupy the same node, which can lead to a possible outage for OpenShift if that specific node becomes unavailable. Therefore, it is a Red Hat best practice to deploy at least three OSP compute nodes so that the OpenShift masters can be distributed evenly and the solution receives an added degree of fault tolerance.

Configure virtual machine/host affinity

Distributing the OpenShift masters across multiple hypervisor nodes can be achieved by enabling VM/host affinity.

Affinity is a way to define rules for a set of VMs and/or hosts that determine whether the VMs run together on the same host or hosts in the group or on different hosts. It is applied to VMs by creating affinity groups that consist of VMs and/or hosts with a set of identical parameters and conditions. Depending on whether the VMs in an affinity group run on the same host or hosts in the group or separately on different hosts, the parameters of the affinity group can define either positive affinity or negative affinity. In the Red Hat OpenStack Platform, host affinity and anti-affinity rules can be created and enforced by creating server groups and configuring filters so that instances deployed by Nova in a server group deploy on different compute nodes.

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

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

To configure affinity groups, see [How do I configure Affinity and Anti-Affinity for OpenStack instances?](#).

Use a custom install file for OpenShift deployment

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

In these instances, you can run and task the wizard without immediately deploying a cluster; instead it creates a configuration file from which the cluster can be deployed later. This is very useful if you need to change any IPI defaults, or if you want to deploy multiple identical clusters in your environment for other uses such as multitenancy. For more information about creating a customized install configuration for OpenShift, see [Red Hat OpenShift Installing a Cluster on OpenStack with Customizations](#).

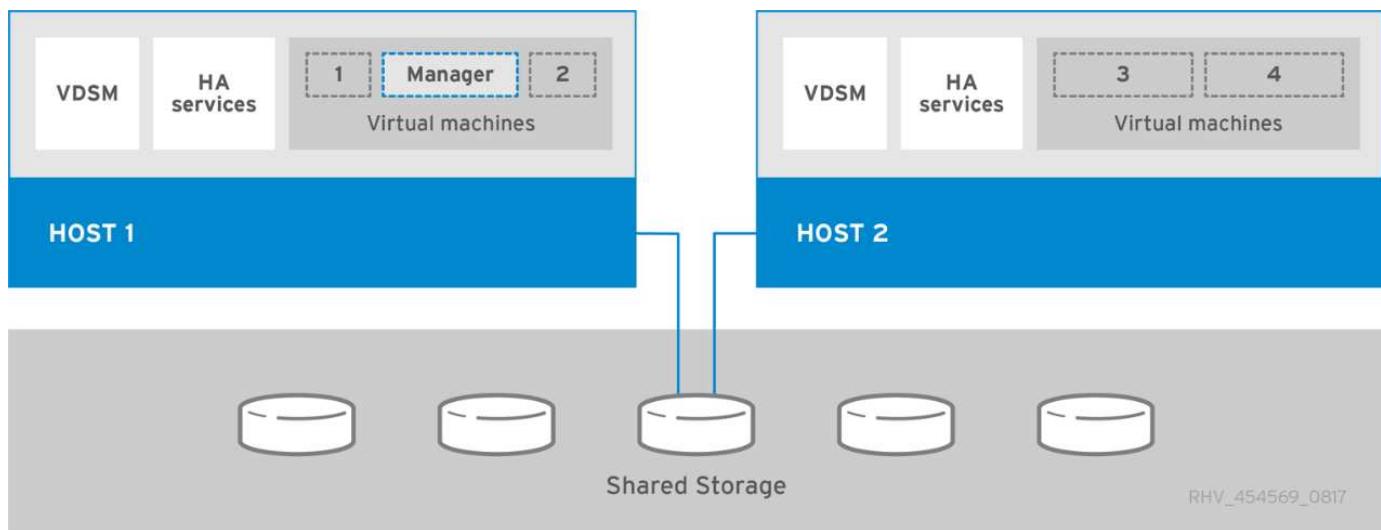
OpenShift on Red Hat Virtualization

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

For more information about RHV, see the [Red Hat Virtualization website](#).

RHV provides the following features:

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



Network design

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

VLAN requirements

Red Hat OpenShift on RHV is designed to logically separate network traffic for different purposes by using virtual local area networks (VLANs). This configuration can be scaled to meet customer demands or to provide

further isolation for specific network services. The following table lists the VLANs that are required to implement the solution while validating the solution at NetApp.

VLANs	Purpose	VLAN ID
Out-of-band management network	Management for physical nodes and IPMI	16
VM Network	Virtual guest network access	1172
In-band management network	Management for RHV-H nodes, RHV-Manager, and ovirtmgmt network	3343
Storage network	Storage network for NetApp Element iSCSI	3344
Migration network	Network for virtual guest migration	3345

Network infrastructure support resources

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

- At least one DNS server providing full host-name resolution that is accessible from the in-band management network and the VM network.
- At least one NTP server that is accessible from the in-band management network and the VM network.
- (Optional) Outbound internet connectivity for both the in-band management network and the VM network.

Best practices for production deployments

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

Deploy OpenShift to an RHV cluster of at least three nodes

The verified architecture described in this document presents the minimum hardware deployment suitable for HA operations by deploying two RHV-H hypervisor nodes and ensuring a fault tolerant configuration where both hosts can manage the hosted-engine and deployed VMs can migrate between the two hypervisors.

Because Red Hat OpenShift initially deploys with three master nodes, it is ensured in a two-node configuration that at least two masters will occupy the same node, which can lead to a possible outage for OpenShift if that specific node becomes unavailable. Therefore, it is a Red Hat best practice that at least three RHV-H hypervisor nodes be deployed as part of the solution so that the OpenShift masters can be distributed evenly and the solution receives an added degree of fault tolerance.

Configure virtual machine/host affinity

You can distribute the OpenShift masters across multiple hypervisor nodes by enabling VM/host affinity.

Affinity is a way to define rules for a set of VMs and/or hosts that determine whether the VMs run together on the same host or hosts in the group or on different hosts. It is applied to VMs by creating affinity groups that consist of VMs and/or hosts with a set of identical parameters and conditions. Depending on whether the VMs in an affinity group run on the same host or hosts in the group or separately on different hosts, the parameters of the affinity group can define either positive affinity or negative affinity.

The conditions defined for the parameters can be either hard enforcement or soft enforcement. Hard enforcement ensures that the VMs in an affinity group always follows the positive or negative affinity strictly without any regards to external conditions. Soft enforcement ensures that a higher preference is set for the VMs in an affinity group to follow the positive or negative affinity whenever feasible. In the two or three

hypervisor configuration described in this document, soft affinity is the recommended setting. In larger clusters, hard affinity can correctly distribute OpenShift nodes.

To configure affinity groups, see the [Red Hat 6.11. Affinity Groups documentation](#).

Use a custom install file for OpenShift deployment

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

In these instances, you can run and task the wizard without immediately deploying a cluster. Rather, a configuration file is created from which the cluster can be deployed later. This is very useful if you want to change any IPI defaults or if you want to deploy multiple identical clusters in your environment for other uses such as multitenancy. For more information about creating a customized install configuration for OpenShift, see [Red Hat OpenShift Installing a Cluster on RHV with Customizations](#).

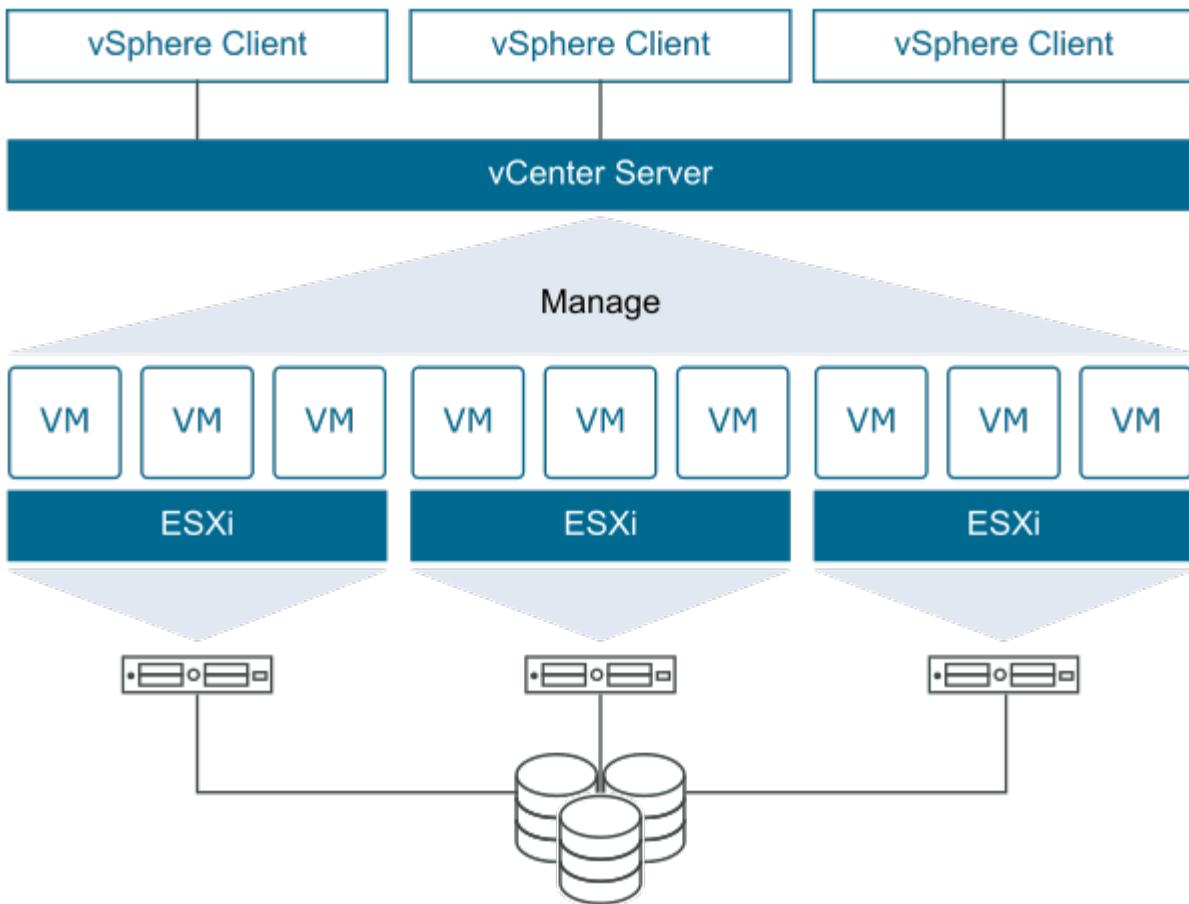
OpenShift on VMware vSphere

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

For more information about VMware vSphere, see the [VMware vSphere website](#).

VMware vSphere provides the following features:

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



Network design

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

VLAN requirements

Red Hat OpenShift on VMware vSphere is designed to logically separate network traffic for different purposes by using virtual local area networks (VLANs). This configuration can be scaled to meet customer demands or to provide further isolation for specific network services. The following table lists the VLANs that are required to implement the solution while validating the solution at NetApp.

VLANs	Purpose	VLAN ID
Out-of-band management network	Management for physical nodes and IPMI	16
VM Network	Virtual guest network access	181
Storage network	Storage network for ONTAP NFS	184
Storage network	Storage network for ONTAP iSCSI	185
In-band management network	Management for ESXi Nodes, VCenter Server, ONTAP Select	3480

VLANs	Purpose	VLAN ID
Storage network	Storage network for NetApp Element iSCSI	3481
Migration network	Network for virtual guest migration	3482

Network infrastructure support resources

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

- At least one DNS server providing full host-name resolution that is accessible from the in-band management network and the VM network.
- At least one NTP server that is accessible from the in-band management network and the VM network.
- (Optional) Outbound internet connectivity for both the in-band management network and the VM network.

Best practices for production deployments

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

Deploy OpenShift to an ESXi cluster of at least three nodes

The verified architecture described in this document presents the minimum hardware deployment suitable for HA operations by deploying two ESXi hypervisor nodes and ensuring a fault tolerant configuration by enabling VMware vSphere HA and VMware vMotion. This configuration allows deployed VMs to migrate between the two hypervisors and reboot should one host become unavailable.

Because Red Hat OpenShift initially deploys with three master nodes, at least two masters in a two-node configuration can occupy the same node under some circumstances, which can lead to a possible outage for OpenShift if that specific node becomes unavailable. Therefore, it is a Red Hat best practice that at least three ESXi hypervisor nodes must be deployed so that the OpenShift masters can be distributed evenly, which provides an added degree of fault tolerance.

Configure virtual machine and host affinity

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

Affinity or anti-affinity is a way to define rules for a set of VMs and/or hosts that determine whether the VMs run together on the same host or hosts in the group or on different hosts. It is applied to VMs by creating affinity groups that consist of VMs and/or hosts with a set of identical parameters and conditions. Depending on whether the VMs in an affinity group run on the same host or hosts in the group or separately on different hosts, the parameters of the affinity group can define either positive affinity or negative affinity.

To configure affinity groups, see the [vSphere 9.0 Documentation: Using DRS Affinity Rules](#).

Use a custom install file for OpenShift deployment

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

In these instances, you can run and task the wizard without immediately deploying a cluster, but instead the wizard creates a configuration file from which the cluster can be deployed later. This is very useful if you need

to changes any IPI defaults, or if you want to deploy multiple identical clusters in your environment for other uses such as multitenancy. For more information about creating a customized install configuration for OpenShift, see [Red Hat OpenShift Installing a Cluster on vSphere with Customizations](#).

Red Hat OpenShift Service on AWS

Red Hat OpenShift Service on AWS (ROSA) is a managed service that you can use to build, scale, and deploy containerized applications with the Red Hat OpenShift enterprise Kubernetes platform on AWS. ROSA streamlines moving on-premises Red Hat OpenShift workloads to AWS, and offers tight integration with other AWS services.

For more information about ROSA, see the documentation here:

[Red Hat OpenShift Service on AWS \(AWS documentation\)](#).

[Red Hat OpenShift Service on AWS \(Red Hat documentation\)](#).

NetApp storage systems

NetApp ONTAP

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

For more information about the NetApp ONTAP storage system, visit the [NetApp ONTAP website](#).

ONTAP provides the following features:

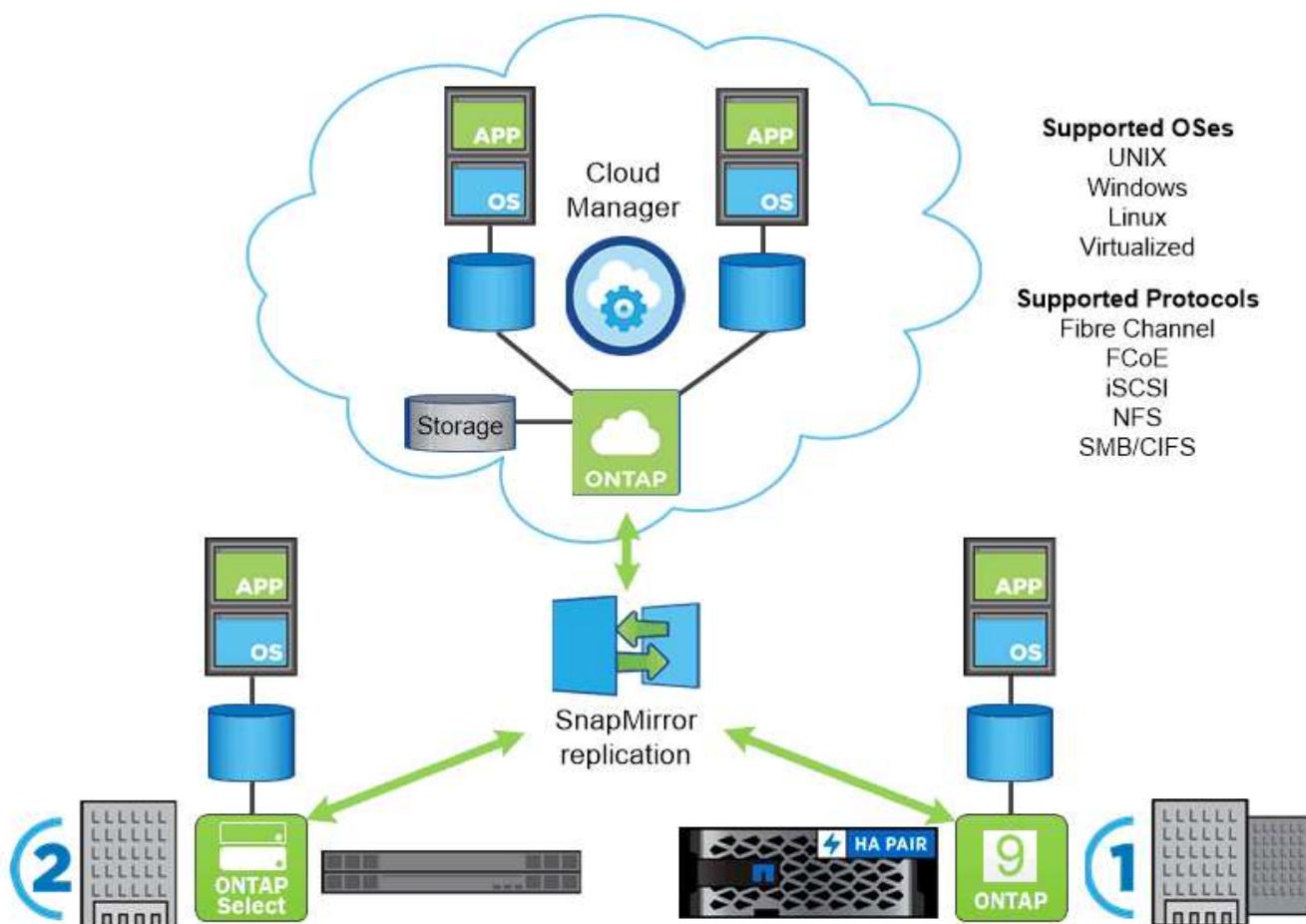
- A unified storage system with simultaneous data access and management of NFS, CIFS, iSCSI, FC, FCoE, and FC-NVMe protocols.
- Different deployment models include on-premises on all-flash, hybrid, and all-HDD hardware configurations; VM-based storage platforms on a supported hypervisor such as ONTAP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

- **NetApp SnapRestore.** Provides fast restoration of backed-up data on demand from Snapshot copies.
- **NetApp FlexClone.** Provides instantaneous provisioning of a fully readable and writeable copy of a NetApp volume based on a Snapshot copy.

For more information about ONTAP, see the [ONTAP 9 Documentation Center](#).



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



NetApp platforms

NetApp AFF/FAS

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

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

For more information about NETAPP AFF/FAS platforms, click [here](#).

ONTAP Select

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

For more information about ONTAP Select, click [here](#).

Cloud Volumes ONTAP

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

For more information about Cloud Volumes ONTAP, click [here](#).

Amazon FSx ONTAP

Amazon FSx ONTAP provides fully managed shared storage in the AWS Cloud with the popular data access and management capabilities of ONTAP. For more information about Amazon FSx ONTAP, click [here](#).

Azure NetApp Files

Azure NetApp Files is an Azure native, first-party, enterprise-class, high-performance file storage service. It provides Volumes as a service for which you can create NetApp accounts, capacity pools, and volumes. You can also select service and performance levels and manage data protection. You can create and manage high-performance, highly available, and scalable file shares by using the same protocols and tools that you're familiar with and rely on on-premises. For more information about Azure NetApp Files, click [here](#).

Google Cloud NetApp Volumes

Google Cloud NetApp Volumes is a fully managed, cloud-based data storage service that provides advanced data management capabilities and highly scalable performance. It lets you move file-based applications to Google Cloud. It has support for Network File System (NFSv3 and NFSv4.1) and Server Message Block (SMB) protocols built-in, so you don't need to re-architect your applications and can continue to get persistent storage for your applications. For more information about Google Cloud NetApp VolumesP, click [here](#).

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.



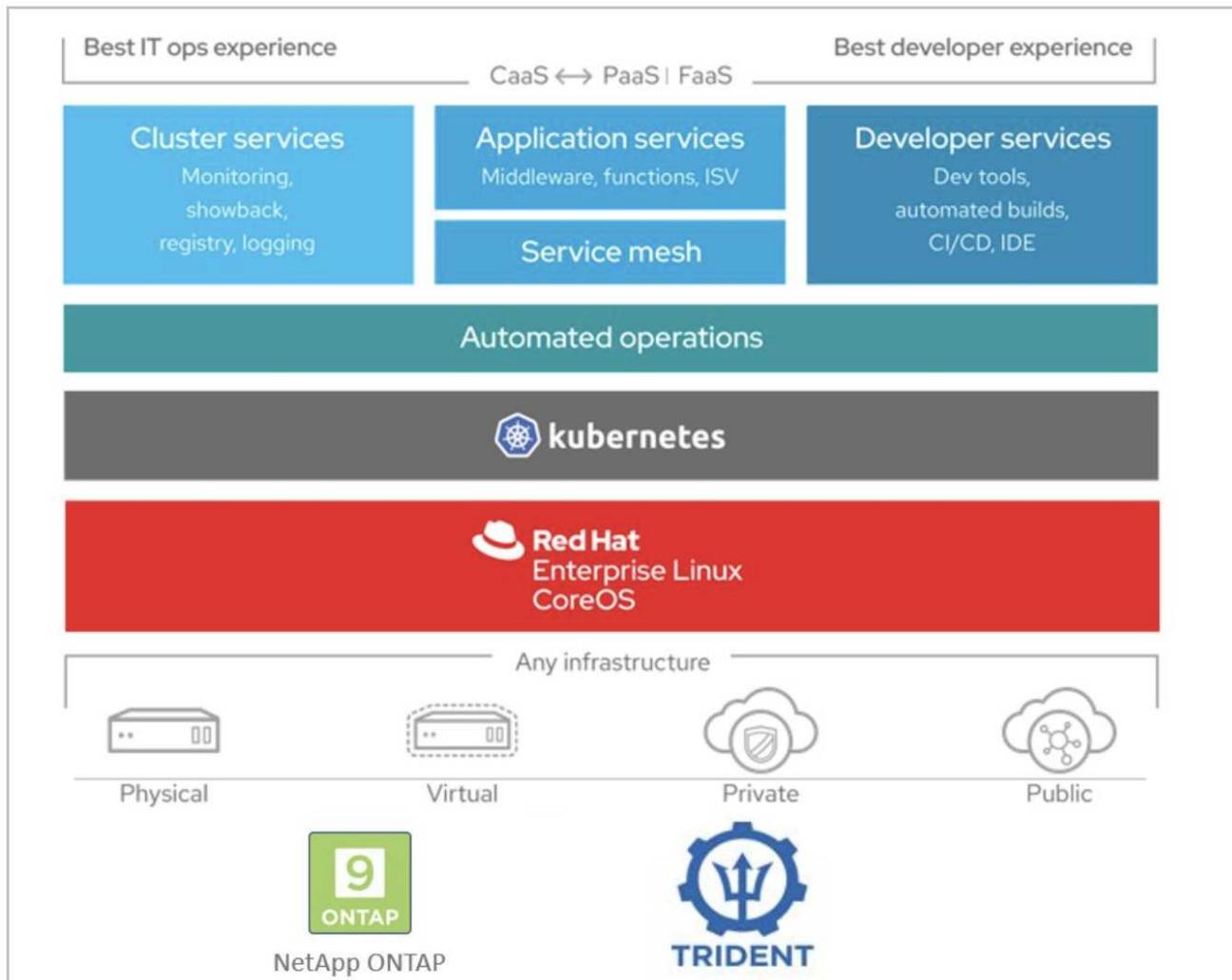
Element was designed for automation. All the storage features are available through APIs. These APIs are the only method that the UI uses to control the system.

NetApp storage integrations

Learn about NetApp Trident integration with Red Hat OpenShift

Learn about NetApp Trident protect that has been validated for application and persistent storage management for the OpenShift Virtualization solution.

Trident, an open source storage provisioner and orchestrator maintained by NetApp and NetApp Trident protect help you with orchestrating and managing persistent data in container based environments, such as Red Hat OpenShift.



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:

- [Trident Documentation](#)
- [Trident protect documentation](#)

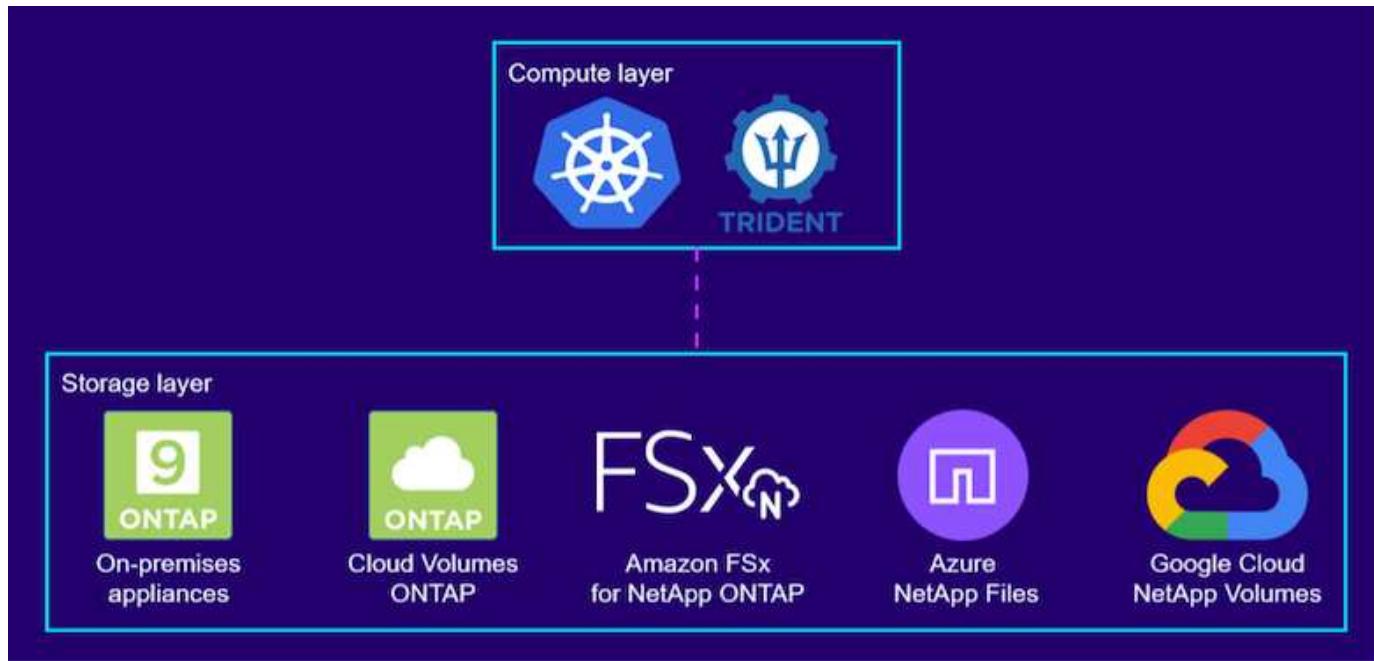
NetApp Trident

Trident Overview

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.



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

A support matrix for what version of Trident has been tested with which Kubernetes distribution can be found [here](#).

Please refer to the [Trident product documentation](#) for installation and configuration details.

Download 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 can be downloaded [here](#).
2. Extract the Trident install from the downloaded bundle.

```
[netapp-user@rhel7 ~]$ tar -xzf trident-installer-22.01.0.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-22.01.0.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-5z451          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 |
+-----+-----+
| 22.01.0          | 22.01.0          |
+-----+-----+
```

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

Manually install the Trident Operator

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

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

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

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

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

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

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

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

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

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

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

```
[netapp-user@rhel7 trident-installer]$ oc create -f
deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created
[netapp-user@rhel7 trident-installer]$ oc describe torc trident
Name:           trident
Namespace:
Labels:          <none>
Annotations:    <none>
API Version:   trident.netapp.io/v1
Kind:           TridentOrchestrator
Metadata:
  Creation Timestamp: 2021-05-07T17:00:28Z
  Generation:        1
  Managed Fields:
    API Version:   trident.netapp.io/v1
    Fields Type:   FieldsV1
    fieldsV1:
      f:spec:
        ..
      f:debug:
      f:namespace:
    Manager:        kubectl-create
    Operation:      Update
    Time:          2021-05-07T17:00:28Z
    API Version:   trident.netapp.io/v1
```

```

Fields Type: FieldsV1
fieldsV1:
  f:status:
    .:
  f:currentInstallationParams:
    .:
    f:IPv6:
    f:autosupportHostname:
    f:autosupportImage:
    f:autosupportProxy:
    f:autosupportSerialNumber:
    f:debug:
    f:enableNodePrep:
    f:imagePullSecrets:
    f:imageRegistry:
    f:k8sTimeout:
    f:kubeletDir:
    f:logFormat:
    f:silenceAutosupport:
    f:tridentImage:
    f:message:
    f:namespace:
    f:status:
    f:version:
  Manager:          trident-operator
  Operation:        Update
  Time:            2021-05-07T17:00:28Z
  Resource Version: 931421
  Self Link:
  /apis/trident.netapp.io/v1/tridentorchestrators/trident
  UID:            8a26a7a6-dde8-4d55-9b66-a7126754d81f
Spec:
  Debug:          true
  Namespace:      trident
Status:
  Current Installation Params:
    IPv6:            false
    Autosupport Hostname:
    Autosupport image:      netapp/trident-autosupport:21.01
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:          true
    Enable Node Prep:    false
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:       30

```

```

Kubelet Dir:          /var/lib/kubelet
Log Format:          text
Silence Autosupport: false
Trident image:       netapp/trident:22.01.0
Message:              Trident installed
Namespace:            trident
Status:               Installed
Version:              v22.01.0

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 |
+-----+-----+
| 22.01.0        | 22.01.0        |
+-----+-----+

```

Prepare worker nodes for storage

NFS

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

However, for NFSv3, there is no mechanism to negotiate concurrency between the client and the server. Hence the maximum number of client-side sunrpc slot table entries must be manually synced with supported value on the server to ensure the best performance for the NFS connection without the server having to decrease the window size of the connection.

For ONTAP, the supported maximum number of sunrpc slot table entries is 128 i.e. ONTAP can serve 128

concurrent NFS requests at a time. However, by default, Red Hat CoreOS/Red Hat Enterprise Linux has a maximum of 65,536 sunrpc slot table entries per connection. We need to set this value to 128 and this can be done using Machine Config Operator (MCO) in OpenShift.

To modify the maximum sunrpc slot table entries in OpenShift worker nodes, 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.

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 98-worker-nfs-rpc-slot-tables
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
        - contents:
            source: data:text/plain;charset=utf-8;base64,b3B0aW9ucyBzdW5ycGMgdGNwX21heF9zbG90X3RhYmx1X2VudHJpZXM9MTI4Cg==
        =
          filesystem: root
          mode: 420
          path: /etc/modprobe.d/sunrpc.conf
```

2. After the MCO is created, the configuration needs to be applied on all worker nodes and rebooted one by one. The whole process takes approximately 20 to 30 minutes. Verify whether the machine config is applied by using `oc get mcp` and make sure that the machine config pool for workers is updated.

```
[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
```

iSCSI

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 iSCSI 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,ZGVmYXVsdHMgewogICAgICAgIHZXJfZnJpZW5kbH1fbmFtZXgbm8KICAgICA
            gICBmaW5kX211bHRpcGF0aHMgbm8KfQoKYmxhY2tsaXN0X2V4Y2VwdG1vbnMgewogICAgICA
            gIHByb3BlcnR5ICIoU0NTSV9JREVOVF98SURfV1dOKSIKfQoKYmxhY2tsaXN0IHsKfQoK
      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 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 Trident.

- [NetApp ONTAP NFS](#)
- [NetApp ONTAP iSCSI](#)
- [NetApp Element iSCSI](#)

NetApp ONTAP NFS configuration

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

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

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

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

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



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

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

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

- 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
```

- 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-nas"
```



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

- 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-input` 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
```

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
```

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 |
+-----+-----+
+-----+-----+
```

- 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
```

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

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



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

- 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-input` 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
```

Advanced configuration options

Explore load balancer options

Exploring load balancer options: Red Hat OpenShift with NetApp

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

OpenShift router to provide this named connectivity to external clients.

However in some cases, applications require the deployment and configuration of customized load balancers to expose the appropriate services. One example of this is NetApp Trident Protect. To meet this need, we have evaluated a number of custom load balancer options. Their installation and configuration are described in this section.

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

- [MetalLB](#)
- [F5 BIG-IP](#)

Installing MetalLB load balancers: Red Hat OpenShift with NetApp

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

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

MetalLB configuration options

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

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



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

Installing The MetalLB Load Balancer

1. Download the MetalLB resources.

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

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

Lines to be deleted:

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

3. Create the `metallb-system` namespace.

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

4. Create the MetalLB CR.

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

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

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

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

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

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

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

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



If you wish to configure MetalLB in BGP mode, skip step 6 above and follow the procedure in the MetalLB documentation [here](#).

Installing F5 BIG-IP Load Balancers

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

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

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

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

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



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



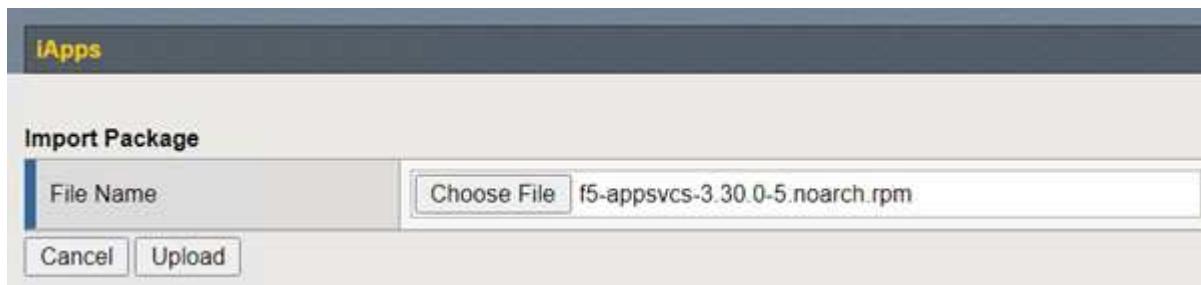
An F5 BIG-IP system can be deployed on dedicated hardware, in the cloud, or as a virtual appliance on-premises with versions greater than 12.x for it to be integrated with F5 CIS. For the purpose of this document, the F5 BIG-IP system was validated as a virtual appliance, for example using the BIG-IP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

Validated releases

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

Installation

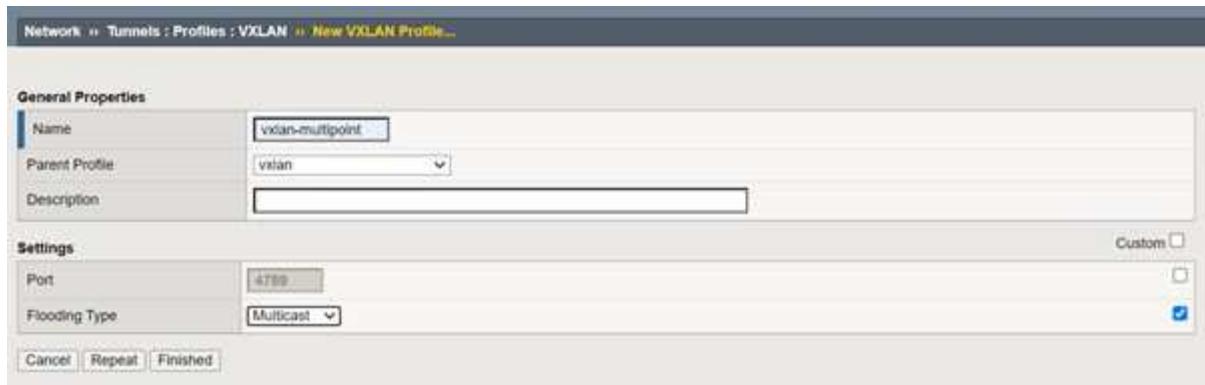
1. Install the F5 Application Services 3 extension to allow BIG-IP systems to accept configurations in JSON instead of imperative commands. Go to [F5 AS3 GitHub repository](#), and download the latest RPM file.
2. Log into F5 BIG-IP system, navigate to iApps > Package Management LX and click Import.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



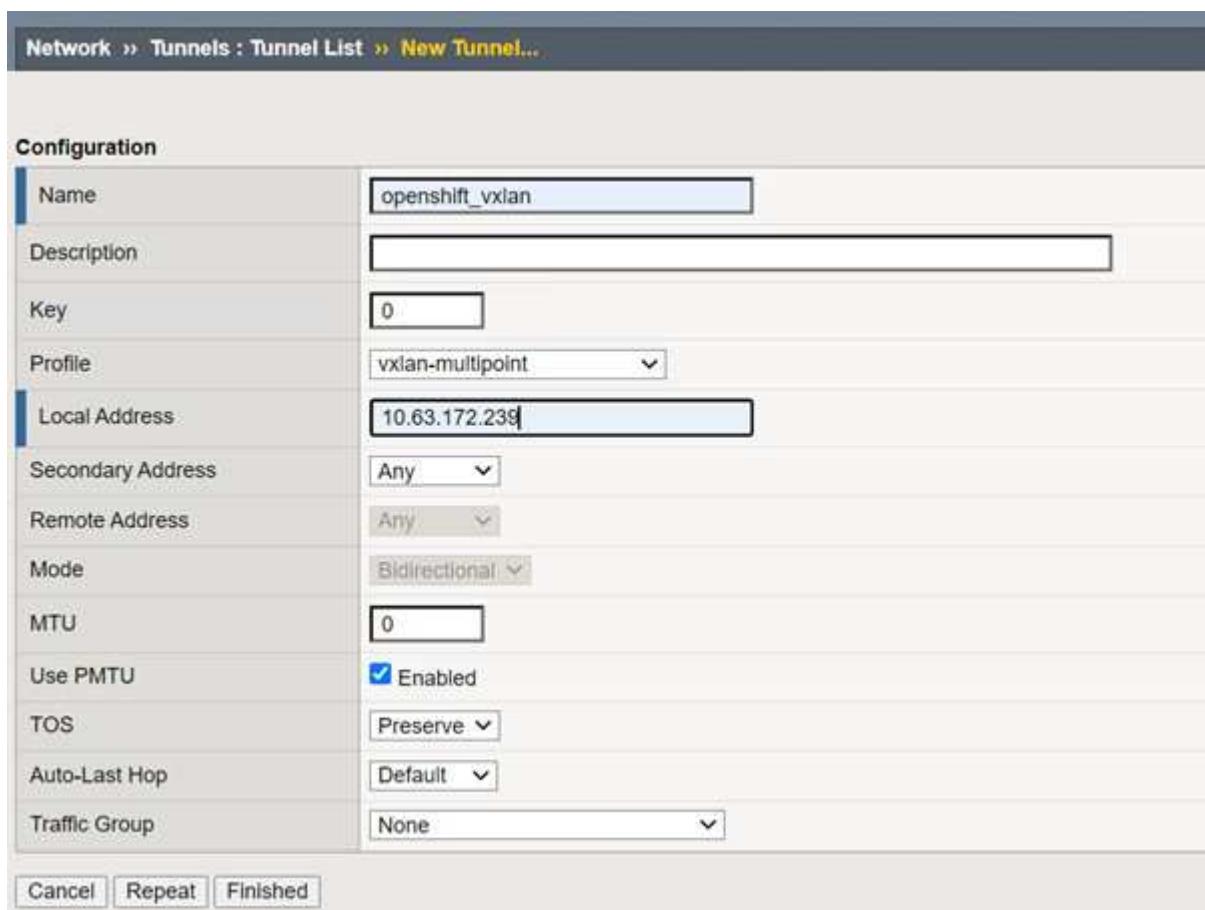
4. Confirm that the AS3 extension is installed successfully.



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



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



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

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

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

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



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

10. Create the HostSubnet resource.

```
[admin@rhel-7 ~]$ oc create -f f5-kctlr-openshift-hostsubnet.yaml
hostsubnet.network.openshift.io/f5-server created
```

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

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

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

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

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

System » Users : Partition List » New Partition...

Properties

Partition Name	ocp-vmw
Partition Default Route Domain	0
Description	<input type="checkbox"/> Extend Text Area <input type="checkbox"/> Wrap Text

Redundant Device Configuration

Device Group	<input checked="" type="checkbox"/> Inherit device group from root folder None
Traffic Group	<input checked="" type="checkbox"/> Inherit traffic group from root folder traffic-group-1 (floating)

Buttons

Cancel Repeat Finished



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

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

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

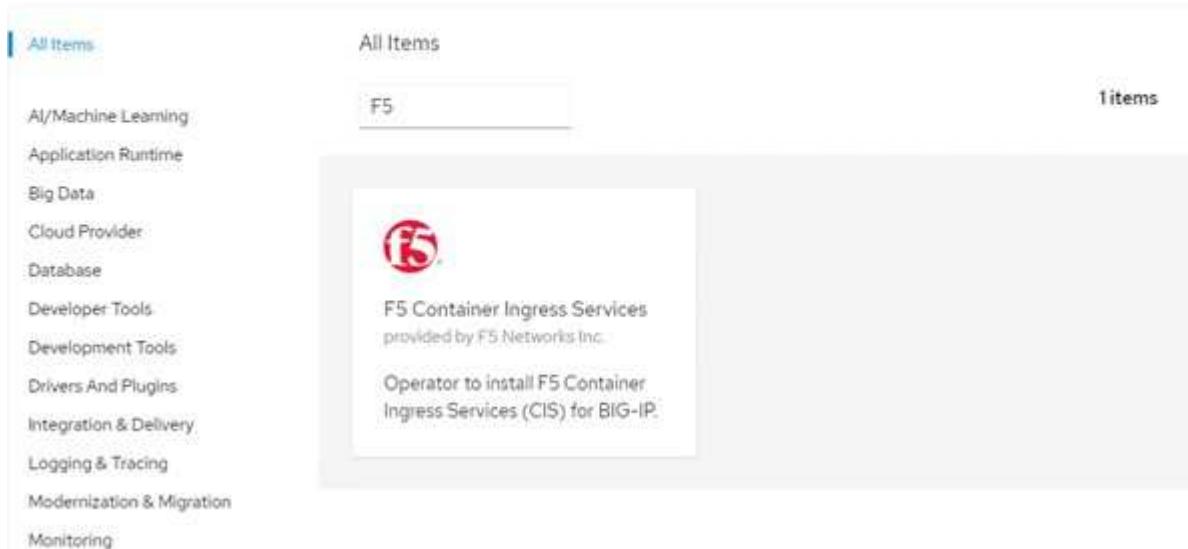
15. Install the F5 CIS CRDs.

```
[admin@rhel-7 ~]$ oc apply -f  
https://raw.githubusercontent.com/F5Networks/k8s-bigip-  
ctlr/master/docs/config_examples/crd/Install/customresourcedefinitions.y  
ml  
  
customresourcedefinition.apiextensions.k8s.io/virtualservers.cis.f5.com  
created  
customresourcedefinition.apiextensions.k8s.io/tlsprofiles.cis.f5.com  
created  
customresourcedefinition.apiextensions.k8s.io/transportservers.cis.f5.co  
m created  
customresourcedefinition.apiextensions.k8s.io/externaldnss.cis.f5.com  
created  
customresourcedefinition.apiextensions.k8s.io/ingresslinks.cis.f5.com  
created
```

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

OperatorHub

Discover Operators from the Kubernetes community and Red Hat partners, curated by Red Hat. You can purchase commercial software through [Red Hat Marketplace](#). You can install Operators on your clusters to provide optional add-ons and shared services to your developers. After installation, the Operator capabilities will appear in the [Developer Catalog](#) providing a self-service experience.



The screenshot shows the OperatorHub interface. On the left, there is a sidebar with a list of categories: All Items, AI/Machine Learning, Application Runtime, Big Data, Cloud Provider, Database, Developer Tools, Development Tools, Drivers And Plugins, Integration & Delivery, Logging & Tracing, Modernization & Migration, and Monitoring. The main area has a search bar with the text 'F5' and a results table. The table has three columns: Category, Name, and Description. There is one item listed: 'F5 Container Ingress Services' provided by 'F5 Networks Inc.' with the description 'Operator to install F5 Container Ingress Services (CIS) for BIG-IP.'

Category	Name	Description
All Items	F5 Container Ingress Services	Operator to install F5 Container Ingress Services (CIS) for BIG-IP.

17. Read the operator information and click Install.

F5 Container Ingress Services
1.8.0 provided by F5 Networks Inc.

Install

Latest version
1.8.0

Capability level
 Basic Install
 Seamless Upgrades
 Full Lifecycle
 Deep Insights
 Auto Pilot

Provider type
Certified

Provider
F5 Networks Inc.

Repository
<https://github.com/F5Networks/k8s-bigip-ctlr>

Container Image
registry.connect.redhat.com/f5networks/k8s-bigip-ctlr

Introduction
This Operator installs F5 Container Ingress Services (CIS) for BIG-IP in your Cluster. This enables to configure and deploy CIS using Helm Charts.

F5 Container Ingress Services for BIG-IP
F5 Container Ingress Services (CIS) integrates with container orchestration environments to dynamically create L4/L7 services on F5 BIG-IP systems, and load balance network traffic across the services. Monitoring the orchestration API server, CIS is able to modify the BIG-IP system configuration based on changes made to containerized applications.

Documentation
Refer to F5 documentation

- CIS on OpenShift (<https://clouddocs.f5.com/containers/latest/userguide/openshift/>) - OpenShift Routes (<https://clouddocs.f5.com/containers/latest/userguide/routes.html>)

Prerequisites
Create BIG-IP login credentials for use with Operator Helm charts. A basic way be,

```
oc create secret generic <SECRET-NAME> -n kube-system --from-literal=username=<USERNAME> --from-literal=password=<PASSWORD>
```

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

Install Operator

Install your Operator by subscribing to one of the update channels to keep the Operator up to date. The strategy determines either manual or automatic updates.

Update channel *

beta

Installation mode *

All namespaces on the cluster (default)
Operator will be available in all Namespaces.

A specific namespace on the cluster
Operator will be available in a single Namespace only.

Installed Namespace *

openshift-operators

Approval strategy *

Automatic

Manual

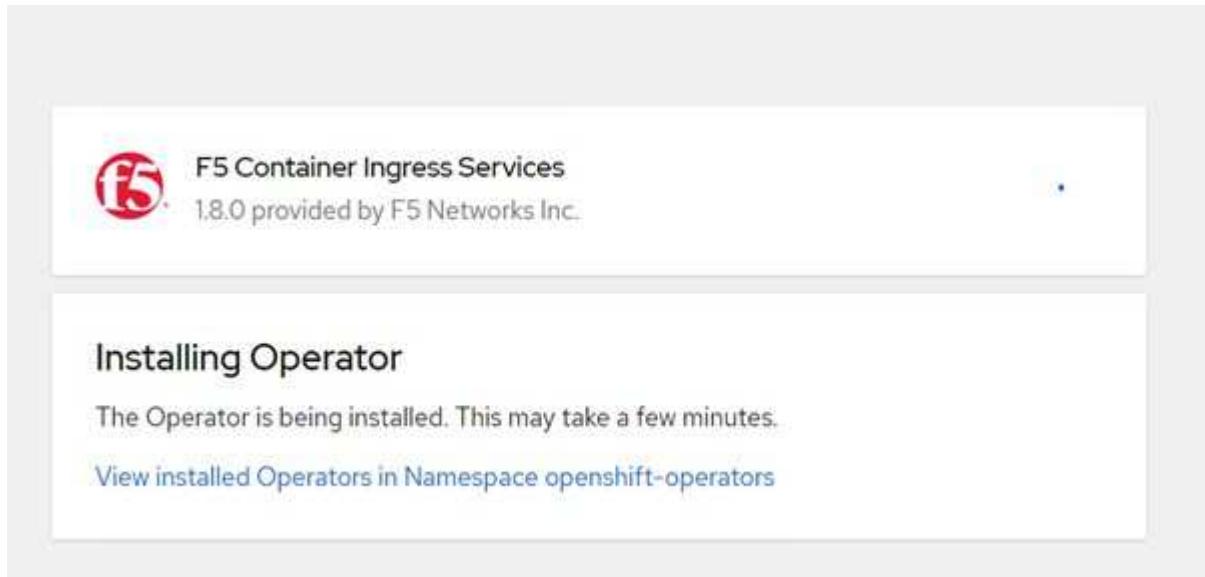
 F5 Container Ingress Services
provided by F5 Networks Inc.

Provided APIs

 F5BigIpCtr
This CRD provides kind `F5BigIpCtr` to configure and deploy F5 BIG-IP Controller.

Install **Cancel**

19. It takes a while to install the operator.



20. After the operator is installed, the Installation Successful message is displayed.

21. Navigate to Operators > Installed Operators, click F5 Container Ingress Service, and then click Create Instance under the F5BigIpCtr tile.



F5 Container Ingress Services

1.8.0 provided by F5 Networks Inc.

[Details](#)[YAML](#)[Subscription](#)[Events](#)[F5BigIpCtlr](#)

Provided APIs

F5BigIpCtlr

This CRD provides kind `F5BigIpCtlr` to configure and deploy F5 BIG-IP Controller.

 [Create instance](#)

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



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

```

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

```

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

Pods

Create Pod

Name	Status	Ready	Restarts	Owner	Memory	CPU
f5-server-f5-bigip-ctlr-5d7578667d-qxdgj	Running	1/1	0	RS f5-server-f5-bigip-ctlr-5d7578667d	61.1 MiB	0.003 cores



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

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

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

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

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

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

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

26. Create the resources.

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

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

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



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



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

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

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

28. Create the F5 IPAM controller deployment.

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

29. Verify the F5 IPAM controller pods are running.

```
[admin@rhel-7 ~]$ oc get pods -n kube-system  
  
NAME                                READY   STATUS    RESTARTS  
AGE  
f5-ipam-controller-5986cff5bd-2bvn6   1/1     Running   0  
30s  
f5-server-f5-bigip-ctlr-5d7578667d-qxdgj   1/1     Running   0  
14m
```

30. Create the F5 IPAM schema.

```
[admin@rhel-7 ~]$ oc create -f  
https://raw.githubusercontent.com/F5Networks/f5-ipam-  
controller/main/docs/_static/schemas/ipam_schema.yaml  
  
customresourcedefinition.apiextensions.k8s.io/ipams.fic.f5.com
```

Verification

1. Create a service of type LoadBalancer

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

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

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

service/f5-demo-test created
```

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

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

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

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

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

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

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

deployment/f5-demo-test created
```

4. Check if the pods are running.

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

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

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



Creating Private Image Registries

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

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

Trident Protect 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 Trident Protect.

Creating A private image registry

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

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

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

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

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

storage:
  pvc:
    claim:
```

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

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



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

Custom TLS certificates

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

- 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
```

- 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
```

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

```
oc edit configs.imageregistry/cluster

managementState: Managed
```

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

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

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

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP
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
kubernetes.io/os=linux	90d			

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

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-78bfb7f59	0	0			
0	15h				
replicaset.apps/image-registry-7fcc8d6cc8	0	0			
0	80m				
replicaset.apps/image-registry-864f88f5b	0	0			
0	15h				
replicaset.apps/image-registry-cb47fffb	0	0			
0	10h				
NAME			COMPLETIONS	DURATION	AGE
job.batch/image-pruner-1627257600			1/1	10s	2d9h
job.batch/image-pruner-1627344000			1/1	6s	33h
job.batch/image-pruner-1627430400			1/1	5s	9h
NAME		SCHEDULE	SUSPEND	ACTIVE	LAST
SCHEDULE	AGE				
cronjob.batch/image-pruner	0 0 * * *		False	0	9h
90d					
NAME			HOST/PORT		
PATH	SERVICES	PORT	TERMINATION	WILDCARD	
route.route.openshift.io/public-routes	astra-registry.apps.ocp-				
vmw.cie.netapp.com	image-registry	<all>	reencrypt	None	

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

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

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

```
[netapp-user@rhel7 ~]$ oc create configmap astra-ca -n openshift-config  
--from-file=astra-registry.apps.ocp-vmw.cie.netapp.com=tls.crt  
  
[netapp-user@rhel7 ~]$ oc patch image.config.openshift.io/cluster  
--patch '{"spec":{"additionalTrustedCA":{"name":"astra-ca"}}}'  
--type=merge
```

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

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

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

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

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

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

10. This pull secret can then be patched to serviceaccounts or be referenced in the corresponding pod definition.

- a. To patch it to service accounts, run the following command.

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

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

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

11. To push or pull an image from workstations apart from OpenShift node, complete the following steps.

a. Add the TLS certificates to the docker client.

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

b. Log into OpenShift using the `oc login` command.

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

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

podman

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

+

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

docker

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

+

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

d. Push or pull the images.

podman

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

docker

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

Solution validation and use cases

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](#)

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.

Create Project

Name *

Display Name

Description

Cancel

Create

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.

Create Persistent Volume Claim

[Edit YAML](#)**Storage Class** SC basic

Storage class for the new claim.

Persistent Volume Claim Name * jenkins

A unique name for the storage claim within the project.

Access Mode * Single User (RWO) Shared Access (RWX) Read Only (ROX)

Permissions to the mounted drive.

Size * 100

GiB ▾

Desired storage capacity.

 Use label selectors to request storage

Use label selectors to define how storage is created.

Create**Cancel**

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**.

Developer Catalog

Add shared apps, services, or source-to-image builders to your project from the Developer Catalog. Cluster admins can install additional apps which will show up here automatically.

All Items

Languages

Databases

Middleware

CI/CD

Other

Type

- Operator Backed (0)
- Helm Charts (0)
- Builder Image (0)
- Template (4)
- Service Class (0)

All Items

Group By: None ▾

Template

Jenkins

provided by Red Hat, Inc.

Jenkins service, with persistent storage. NOTE: You must have persistent volumes available in...

Template

Jenkins

provided by Red Hat, Inc.

Jenkins service, with persistent storage. NOTE: You must have persistent volumes available in...

Template

Jenkins (Ephemeral)

provided by Red Hat, Inc.

Jenkins service, without persistent storage. WARNING: Any data stored will be lost upon...

Template

Jenkins (Ephemeral)

provided by Red Hat, Inc.

Jenkins service, without persistent storage. WARNING:

2. Click Instantiate Template.

Jenkins

Provided by Red Hat, Inc.

Instantiate Template

Provider	Description
Red Hat, Inc.	Jenkins service, with persistent storage.
Support	NOTE: You must have persistent volumes available in your cluster to use this template.
Get support ↗	
Created At	Documentation
⌚ May 26, 3:58 am	https://docs.okd.io/latest/using_images/other_images/jenkins.html ↗

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.

Instantiate Template

Namespace *

PR jenkins

Jenkins Service Name

jenkins

The name of the OpenShift Service exposed for the Jenkins container.

Jenkins JNLP Service Name

jenkins-jnlp

The name of the service used for master/slave communication.

Enable OAuth in Jenkins

true

Whether to enable OAuth OpenShift integration. If false, the static account 'admin' will be initialized with the password 'password'.

Memory Limit

1Gi

Maximum amount of memory the container can use.

Volume Capacity *

50Gi

Volume space available for data, e.g. 512Mi, 2Gi.

Jenkins ImageStream Namespace

openshift

The OpenShift Namespace where the Jenkins ImageStream resides.

Disable memory intensive administrative monitors

false

Whether to perform memory intensive, possibly slow, synchronization with the Jenkins Update Center on start. If true, the Jenkins core update monitor and site warnings monitor are disabled.

Jenkins ImageStreamTag

jenkins:2

Name of the ImageStreamTag to be used for the Jenkins image.

Fatal Error Log File

false

When a fatal error occurs, an error log is created with information and the state obtained at the time of the fatal error.

Allows use of Jenkins Update Center repository with invalid SSL certificate

false

Whether to allow use of a Jenkins Update Center that uses invalid certificate (self-signed, unknown CA). If any value other than 'false', certificate check is bypassed. By default, certificate check is enforced.

Create **Cancel**

Jenkins
INSTANT-APP JENKINS
[View documentation](#) [Get support](#)

Jenkins service, with persistent storage.

NOTE: You must have persistent volumes available in your cluster to use this template.

The following resources will be created:

- DeploymentConfig
- PersistentVolumeClaim
- RoleBinding
- Route
- Service
- ServiceAccount

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

Pods

[Create Pod](#)

1 Running	0 Pending	0 Terminating	0 CrashLoopBackOff	1 Completed	0 Failed	0 Unknown
Select all filters						1 of 2 Items

Name	Namespace	Status	Ready	Owner	Memory	CPU
jenkins-1-c77n9	jenkins	Running	1/1	jenkins-1	-	0.004 cores

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

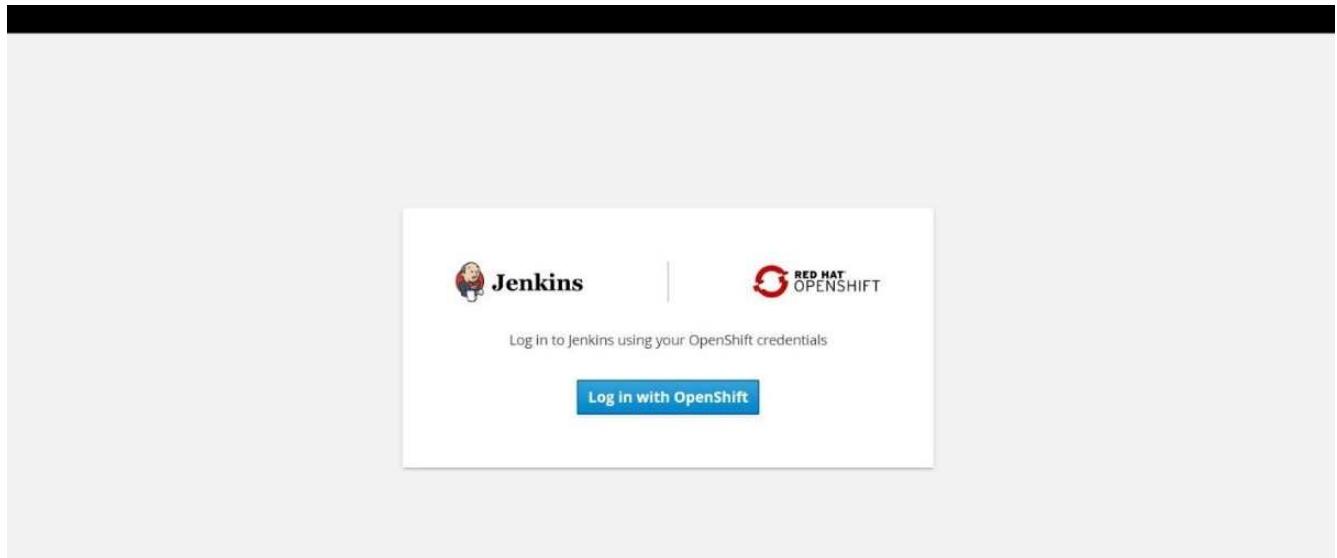
Routes

[Create Route](#)

1 Accepted	0 Rejected	0 Pending	Select all filters	1 Item
---	---	--	--------------------	--------

Name	Namespace	Status	Location	Service
jenkins	jenkins	Accepted	https://jenkins-jenkins.apps.rhv-ocp-cluster.cie.netapp.com	jenkins

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.

Authorize Access

Service account jenkins in project jenkins is requesting permission to access your account (kube:admin)

Requested permissions

user:info

Read-only access to your user information (including username, identities, and group membership)

user:check-access

Read-only access to view your privileges (for example, "can I create builds?")

You will be redirected to <https://jenkins-jenkins.apps.rhv-ocp-cluster.cie.netapp.com/securityRealm/finishLogin>

[Allow selected permissions](#) [Deny](#)

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.

Maven
Maven installations

Add Maven

Maven
Name <input type="text" value="M3"/>
<input checked="" type="checkbox"/> Install automatically
Install from Apache
Version <input type="text" value="3.6.3"/>

[Delete Installer](#)

[Add Maven](#)

Add Maven

List of Maven installations on this system

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.

The screenshot shows the Jenkins home page. At the top, there is a navigation bar with the Jenkins logo, a search bar, and a user 'kube:admin' logged in. A red box highlights the number '3' in a red box, indicating three new items. Below the navigation, a sidebar on the left lists various Jenkins management options: New Item, People, Build History, Manage Jenkins, My Views, Open Blue Ocean, Lockable Resources, Credentials, and New View. The main content area features a 'Welcome to Jenkins!' message with a sub-instruction: 'Please [create new jobs](#) to get started.' Below this, there are two collapsed sections: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle).

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

The screenshot shows the 'Enter an item name' page. The input field contains 'sample-demo' with a note '» Required field'. Below the input field, there is a list of project types with their descriptions and icons:

- Freestyle project** (Icon: box with a gear): This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Pipeline** (Icon: pipe): Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project** (Icon: wrench and gear): Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Bitbucket Team/Project** (Icon: Bitbucket logo): Scans a Bitbucket Cloud Team (or Bitbucket Server Project) for all repositories matching some defined markers.
- Folder** (Icon: folder): Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- GitHub Organization** (Icon: GitHub logo): Scans a GitHub organization (or user account) for all repositories matching some defined markers.

At the bottom of the list, there is a large blue 'OK' button and a note: 'Creates a set of Pipeline projects according to detected branches in one SCM repository'.

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

General Build Triggers Advanced Project Options **Pipeline** Advanced...

Pipeline

Definition Pipeline script

Script

```

1  node {
2      def mvnHome
3      stage('Preparation') { // for display purposes
4          // Get some code from a GitHub repository
5          git 'https://github.com/jglick/simple-maven-project-with-tests.git'
6          // Get the Maven tool.
7          // ** NOTE: This 'M3' Maven tool must be configured
8          // ** in the global configuration.
9          mvnHome = tool 'M3'
10     }
11    stage('Build') {
12        // Run the maven build
13        withEnv(["MVN_HOME=$mvnHome"]) {
14            if (isUnix()) {
15                sh '$MVN_HOME/bin/mvn' -Dmaven.test.failure.ignore clean package'
16            } else {
17                bat(/"%MVN_HOME%\bin\mvn" -Dmaven.test.failure.ignore clean package/)
18            }
19        }
20    }
21  }

```

GitHub + Maven

Use Groovy Sandbox

[Pipeline Syntax](#)

Save **Apply**

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.

 **Jenkins**

Jenkins > sample-demo >

[Back to Dashboard](#) [Status](#) [Changes](#) [Build Now](#) [Delete Pipeline](#) [Configure](#) [Full Stage View](#) [Open Blue Ocean](#) [Rename](#) [Pipeline Syntax](#)

Pipeline sample-demo

 [Last Successful Artifacts](#)
[simple-maven-project-with-tests-1.0-SNAPSHOT.jar](#) 1.71 KB [view](#)

 [Recent Changes](#)

Stage View

Average stage times:
 (Average full run time: ~7s)

Preparation	Build	Results
2s	4s	69ms
2s	4s	69ms

 [Latest Test Result \(no failures\)](#)

Permalinks

- [Last build \(#1\), 1 min 23 sec ago](#)
- [Last stable build \(#1\), 1 min 23 sec ago](#)
- [Last successful build \(#1\), 1 min 23 sec ago](#)
- [Last completed build \(#1\), 1 min 23 sec ago](#)

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.

Configure multi-tenancy

Configuring multitenancy on Red Hat OpenShift with NetApp

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

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

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

workload.

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

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

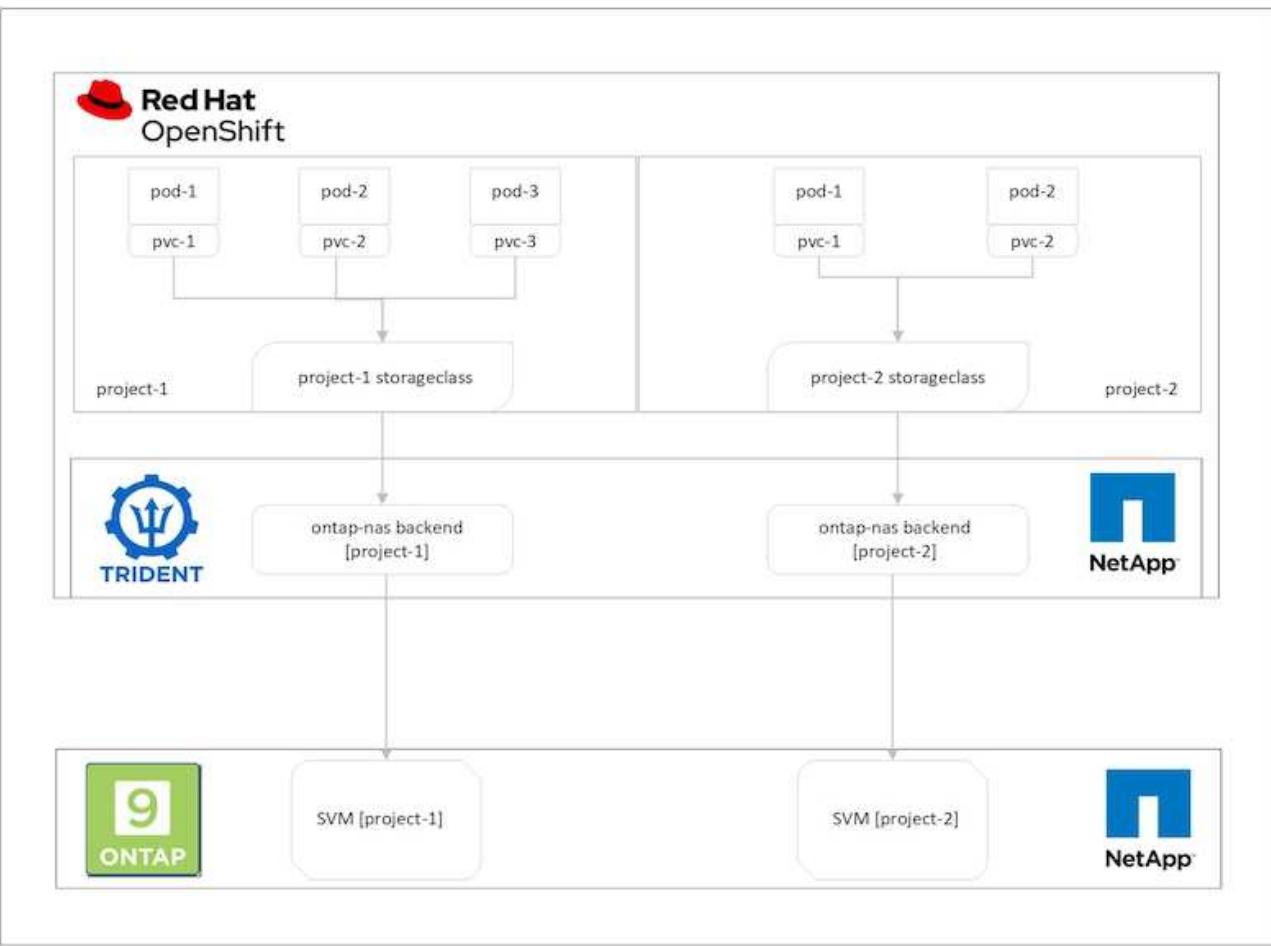
For a fully realized multitenant OpenShift cluster, quotas and restrictions must be configured for cluster resources belonging to different resource buckets: compute, storage, networking, security, and so on. Although we cover certain aspects of all the resource buckets in this solution, we focus on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

Architecture

Although Red Hat OpenShift and Trident backed by NetApp ONTAP do not provide isolation between workloads by default, they offer a wide range of features that can be used to configure multitenancy. To better understand designing a multitenant solution on a Red Hat OpenShift cluster with 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 Trident on a NetApp ONTAP NAS backend. The organization has a requirement to design a multitenant solution for these two workloads and isolate the resources used for these projects to make sure that security and performance is maintained, primarily focused on the data that serves those applications.

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



Technology requirements

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

Red Hat OpenShift – Cluster resources

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

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

NetApp ONTAP

It is important to isolate the shared storage serving as a persistent storage provider for a Red Hat OpenShift cluster to make sure that the volumes created on the storage for each project appear to the hosts as if they are created on separate storage. To do this, create as many SVMs (storage virtual machines) on NetApp ONTAP

as there are projects or workloads, and dedicate each SVM to a workload.

Trident

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

Red Hat OpenShift – storage resources

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

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

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

Configuration

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

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

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

Configuration

Following are the prerequisites for Configuring Multitenancy on Red Hat OpenShift with NetApp.

Prerequisites

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

Configuration: cluster-admin tasks

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

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

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

3. Create the developer role for project-1.

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

```

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

```



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

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

```

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

```

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

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

8. Configure ClusterRoleBindings for storage admins.

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



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

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

```

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

```

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

Configuration: Storage-admin tasks

The following resources must be configured by a storage administrator:

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

Add Storage VM

X

STORAGE VM NAME

project-1-svm

Access Protocol

SMB/CIFS, NFS

iSCSI

Enable SMB/CIFS

Enable NFS

Allow NFS client access

Add at least one rule to allow NFS clients to access volumes in this storage VM. [?](#)

EXPORT POLICY

Default

RULES

Rule Index	Clients	Access Protocols	Read-Only R...	Read/Wr...
	10.61.181.0/24	Any	Any	Any

[+ Add](#)

DEFAULT LANGUAGE

[?](#)

c.utf_8



NETWORK INTERFACE

Use multiple network interfaces when client traffic is high.

K8s-Ontap-01

IP ADDRESS

10.61.181.224

SUBNET MASK

24

GATEWAY

Add optional
gateway

BROADCAST DOMAIN

Default-4



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

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



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



We assume that Trident is installed in the trident project.

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

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

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

```

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

```

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

Validation

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

Validate access to create PVCs or pods in assigned project

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

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

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

```

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

```

4. Check the PV associated with the PVC.

```
oc get pv
```

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

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

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

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

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

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

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

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

```

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

```

3. Create a PVC in project-2.

```

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

```

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

```

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

```

5. Create a pod in project-2.

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

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

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

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

3. Validate access to view projects.

```
oc get ns
```

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

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

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

```
oc get sc
```

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

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

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.

Add Storage VM

X

STORAGE VM NAME

project-3-svm

Access Protocol

SMB/CIFS, NFS

iSCSI

Enable SMB/CIFS

Enable NFS

Allow NFS client access

Add at least one rule to allow NFS clients to access volumes in this storage VM. [?](#)

EXPORT POLICY

Default

RULES

Rule Index	Clients	Access Protocols	Read-Only R...	Read/W...
	10.61.181.0/24	Any	Any	Any

[+ Add](#)

DEFAULT LANGUAGE [?](#)

c.utf_8

NETWORK INTERFACE

Use multiple network interfaces when client traffic is high.

K8s-Ontap-01

IP ADDRESS

10.61.181.228

SUBNET MASK

24

GATEWAY

Add optional
gateway

BROADCAST DOMAIN

Default-4

3. Log into the Red Hat OpenShift cluster as cluster admin.

4. Create a new project.

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

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

```
oc get groups
```

6. Create the developer role for project-3.

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

```

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

```

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

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

```

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

```

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

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



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



We assume that Trident is installed in the trident project.

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

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

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

```

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

```

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

```

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

```

Advanced Cluster Management for Kubernetes

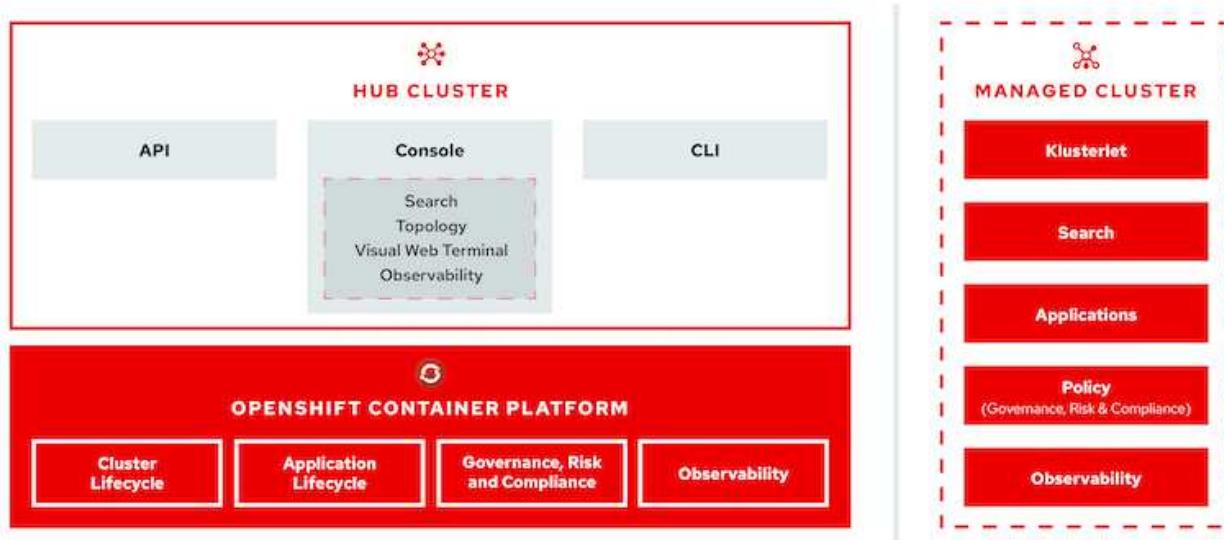
Advanced Cluster Management for Kubernetes: Red Hat OpenShift with NetApp - Overview

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

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

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

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



For more information, see the documentation [here](#).

Deploy ACM for Kubernetes

Deploy Advanced Cluster Management for Kubernetes

This section covers advanced cluster management for Kubernetes on Red Hat OpenShift with NetApp.

Prerequisites

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

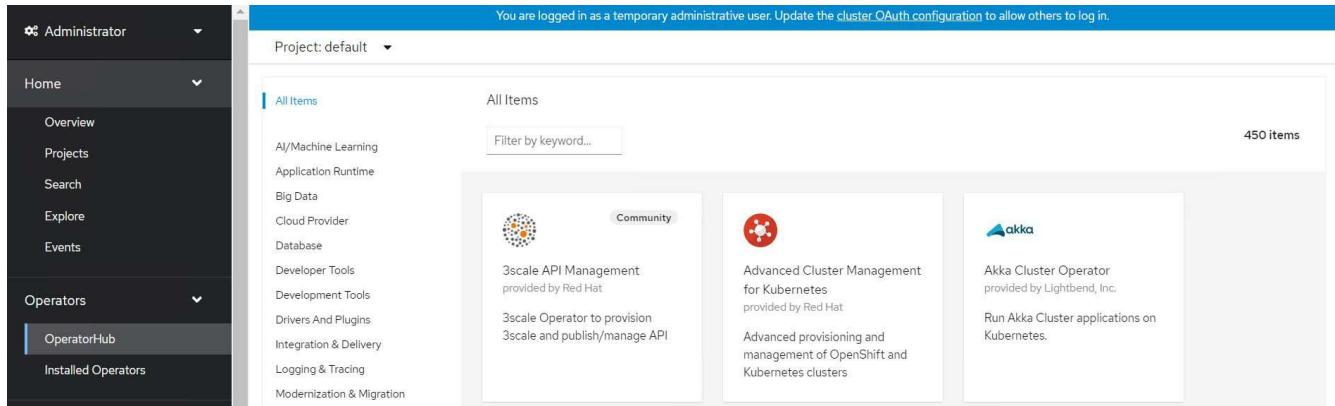
Advanced Cluster Management is an add-on on for the OpenShift cluster, so there are certain requirements and restrictions on the hardware resources based on the features used across the hub and managed clusters. You need to take these issues into account when sizing the clusters. See the documentation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

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

Deploy Advanced Cluster Management for Kubernetes

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

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



The screenshot shows the Operators Hub interface. The left sidebar is titled 'Administrator' and includes 'Home', 'Operators' (selected), and 'OperatorHub' (highlighted). The main area is titled 'Project: default' and shows a search bar with 'All Items' and 'Filter by keyword...'. There are 450 items listed. Three operators are visible in the grid: '3scale API Management' (Community), 'Advanced Cluster Management for Kubernetes' (provided by Red Hat), and 'Akka Cluster Operator' (provided by Lightbend, Inc.).

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



The screenshot shows the 'Advanced Cluster Management for Kubernetes' operator page. It includes a red circular icon with a white cluster symbol, the operator name, its version (2.2.3), and the provider (Red Hat). A large blue 'Install' button is prominent. Below the button, the 'Latest version' is listed as 2.2.3. The 'Capability level' section shows 'Basic Install' and 'Seamless Upgrades' as checked options, while 'Full Lifecycle', 'Deep Insights', and 'Auto Pilot' are unselected. The 'Provider type' is Red Hat. The 'Provider' is Red Hat. The 'Infrastructure features' are listed as 'Disconnected'. The 'How to Install' section contains a note about licensing and subscription requirements.

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

Install Operator

Install your Operator by subscribing to one of the update channels to keep the Operator up to date. The strategy determines either manual or automatic updates.

Update channel *

- release-2.0
- release-2.1
- release-2.2

Installation mode *

- All namespaces on the cluster (default)
This mode is not supported by this Operator
- A specific namespace on the cluster
Operator will be available in a single Namespace only

Installed Namespace *

- Operator recommended Namespace: **PR open-cluster-management**

 **Namespace creation**

Namespace **open-cluster-management** does not exist and will be created.

- Select a Namespace

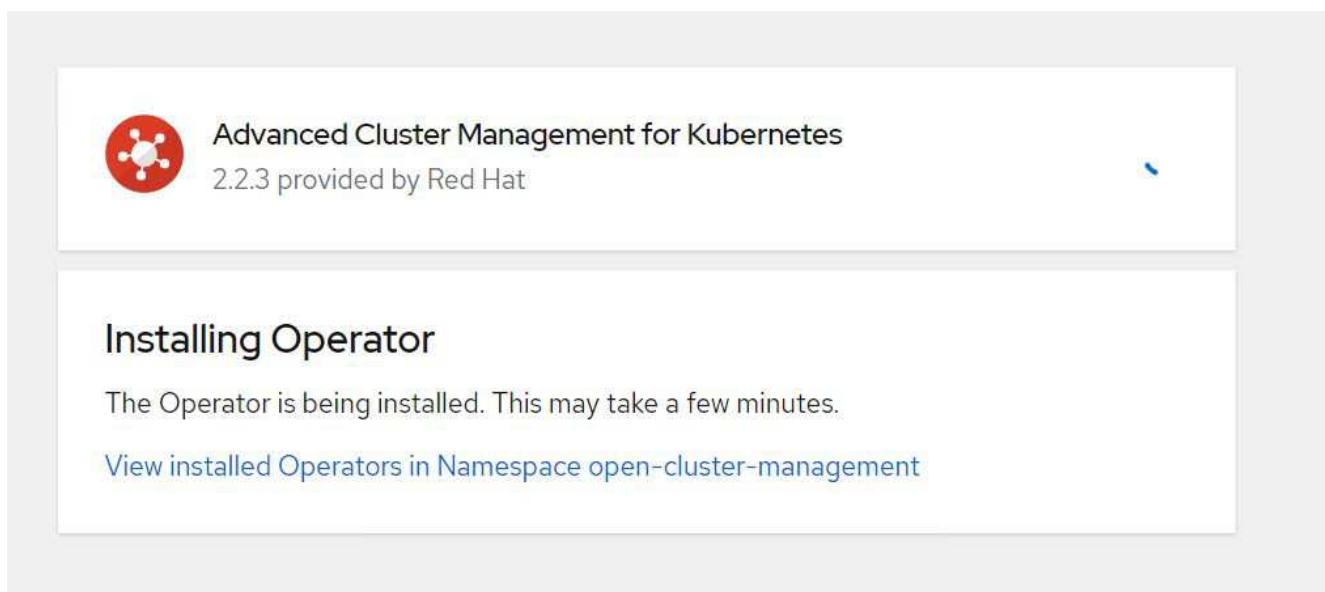
Approval strategy *

- Automatic
- Manual

Install

Cancel

5. Wait for the operator installation to complete.



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



Advanced Cluster Management for Kubernetes

2.2.3 provided by Red Hat



Installed operator - operand required

The Operator has installed successfully. Create the required custom resource to be able to use this Operator.

MCH MultiClusterHub ! Required

Advanced provisioning and management of OpenShift and Kubernetes clusters

[Create MultiClusterHub](#)

[View installed Operators in Namespace open-cluster-management](#)

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

Project: open-cluster-management ▾

Advanced Cluster Management for Kubernetes > Create MultiClusterHub

Create MultiClusterHub

Create by completing the form. Default values may be provided by the Operator authors.

Configure via: Form view YAML view

i Note: Some fields may not be represented in this form view. Please select "YAML view" for full control.

MultiClusterHub
provided by Red Hat

MultiClusterHub defines the configuration for an instance of the MultiCluster Hub

Name *

multiclus

Labels

app=frontend

» Advanced configuration

[Create](#)

[Cancel](#)

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.

Installed Operators

Installed Operators are represented by ClusterServiceVersions within this Namespace. For more information, see the [Understanding Operators documentation](#). Or create an Operator and ClusterServiceVersion using the [Operator SDK](#).

Name	Managed Namespaces	Status	Provided APIs	⋮
 Advanced Cluster Management for Kubernetes 2.2.3 provided by Red Hat	NS open-cluster-management	✓ Succeeded Up to date	MultiClusterHub ClusterManager ClusterDeployment ClusterState View 25 more...	

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

Installed Operators > Operator details

 Advanced Cluster Management for Kubernetes
2.2.3 provided by Red Hat

Actions ▾

Details YAML Subscription Events All instances MultiClusterHub ClusterManager ClusterDeployment ClusterState

MultiClusterHubs

Create MultiClusterHub

Name	Kind	Status	Labels	⋮
MCH multiclusterhub	MultiClusterHub	Phase: ✓ Running	No labels	

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.

Routes

Create Route

Name	Status	Location	Service	⋮
 multicloud-console	✓ Accepted	https://multicloud-console.apps.ocp-vmware2.cie.netapp.com	 management-ingress	

Cluster Lifecycle Management

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

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

Select a provider and enter basic information

Provider * 

 aws Amazon Web Services

Connection name * 

nik-hcl-aws

Namespace * 

default

Configure your provider connection

Base DNS domain 

cie.netapp.com

AWS access key ID * 

AKIATCFBZDOIASD5AH

AWS secret access key * 

.....

Red Hat OpenShift pull secret * 

```
FuS3pNbktVaHplNFc2MkZsbmtBVGN6TktmUlZXcHcxOW9teEZwQ0lYZ1d3cjJobGxJeDBQN0xiZE0yeGM5Q0ZwZk5RR2JUanlxNnNUM21RbOFJb
UFjNCIBYlpEWVZEOHitNkxTMDZPUVpoWFRHcGwtREIDQ2RSYlJRaTlxblDLT2oyQ3pVeUJfNlwicENSa2YyOUSyLWZGSFVfNA==,"email":"Nikhil.k
ulkarni@netapp.com"}, "registry.redhat.io":
```

SSH private key * 

-----BEGIN OPENSSH PRIVATE KEY-----

```
b3BlbnNzaC1rZXktdjEAAAABG5vbmuAAAAEbasdadssad9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
QyNTUxQOAAACCLcwLgAvSiHAEp+DevIRNzaG2zkNreMIZ/UHyf0UWvAAAAJh/wa6xf8Gu
```

SSH public key * 

```
ssh-ed25519 AAAAC3NzaC1IzDIINTE5AAAAItzAuAC746agdh2IcB4/4N6/VE3NobbOQ2t4zVn9QfJ/RRa8A root@nik-rhel8
```

- 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.

^ Configuration

Cluster name * [?](#)

^ Distribution

Select the type of Kubernetes distribution to use for your cluster.

Red Hat OpenShift

Select an infrastructure provider to host your Red Hat OpenShift cluster.

AWS Amazon Web Services

Google Cloud

Microsoft Azure

VMware vSphere

Bare Metal

Release image * [?](#)

Provider connection * [?](#)

[Add a connection](#)

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:

- Navigation to Clusters and click Add a Cluster > Import an Existing Cluster.
- Enter the name of the cluster and click Save Import and Generate Code. A command to add the existing cluster is displayed.
- 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.

Name *

Additional labels

Once you click on "Save import and generate code", the information you entered will be used to generate the code and cannot be modified anymore. If you wish to change any information, you will have to delete and re-import this cluster.

Code generated successfully Import saved

Run a command

1. Copy this command

Click the button to have the command automatically copied to your clipboard.

[Copy command](#)

2. Run this command with `kubectl` configured for your targeted cluster to start the import

Log in to the existing cluster in your terminal and run the command.

[View cluster](#)

[Import another](#)

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

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.

Create an application

YAML: Off

Cancel

Save

Name* ⓘ

Namespace* ⓘ

Repository location for resources

Repository types

Select the type of repository where resources that you want to deploy are located

Git

URL* ⓘ

Branch ⓘ

Path ⓘ

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

Applications

Overview

Advanced configuration

⟳ Refresh every 15s

Last update: 7:36:23 PM

Create application

Search

Name

Namespace

Clusters

②

Resource

②

Time window

②

Created

demo-app

default

Local

Git

8 days ago

⋮

1-1 of 1

<<

<

1

of 1

>

>>

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

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.

Create policy ⓘ YAML: off

Name *

policy-complianceoperator

Namespace * ⓘ

default

Specifications * ⓘ

1x ComplianceOperator

Cluster selector ⓘ

1x local-cluster: "true"

Standards ⓘ

1x NIST-CSF

Categories ⓘ

1x PR.IP Information Protection Processes and Procedures

Controls ⓘ

1x PR.IP-1 Baseline Configuration

 Enforce if supported ⓘ Disable policy ⓘ

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

Summary 1 | Standards 

NIST-CSF



No violations found
Based on the industry standards, there are no cluster or policy violations.

Policies Cluster violations

 Find policies

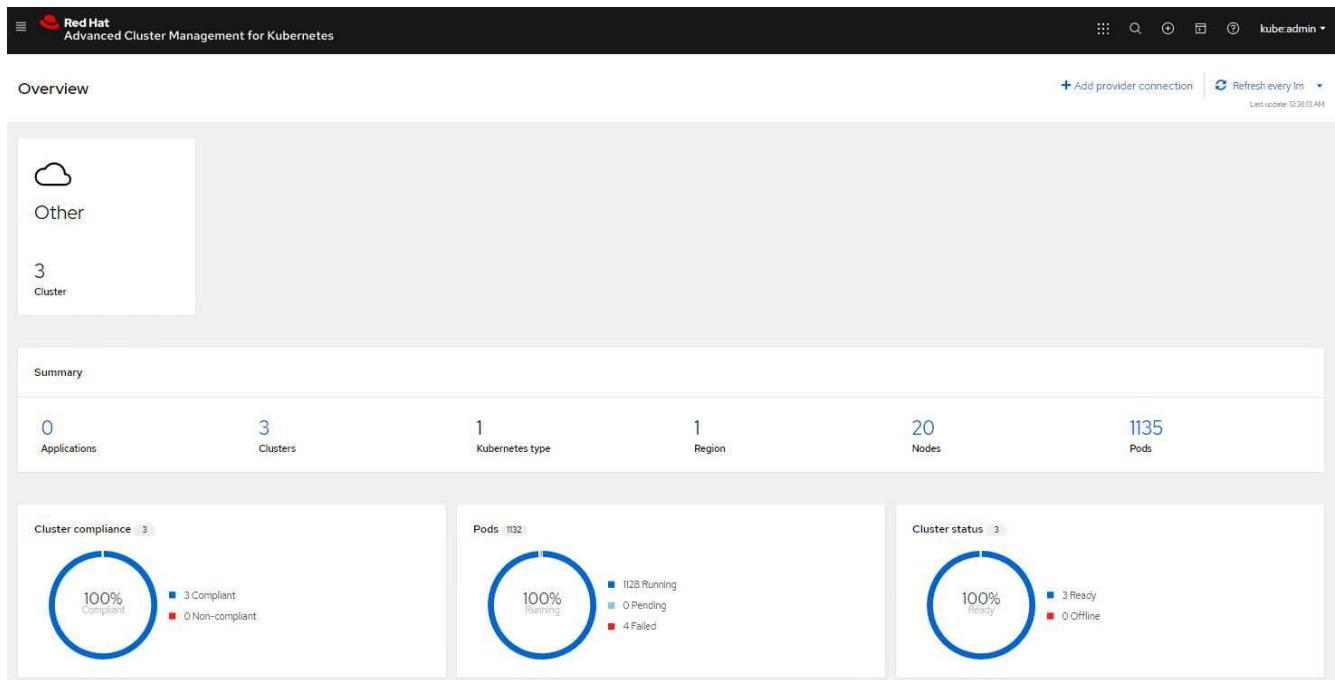
Policy name	Namespace	Remediation	Cluster violations	Standards	Categories	Controls	Created
policy-complianceoperator	default	inform	0/1	NIST-CSF	PRIP Information Protection Processes and Procedures	PRIP-1 Baseline Configuration	32 minutes ago

1-1 of 1    1 of 1  

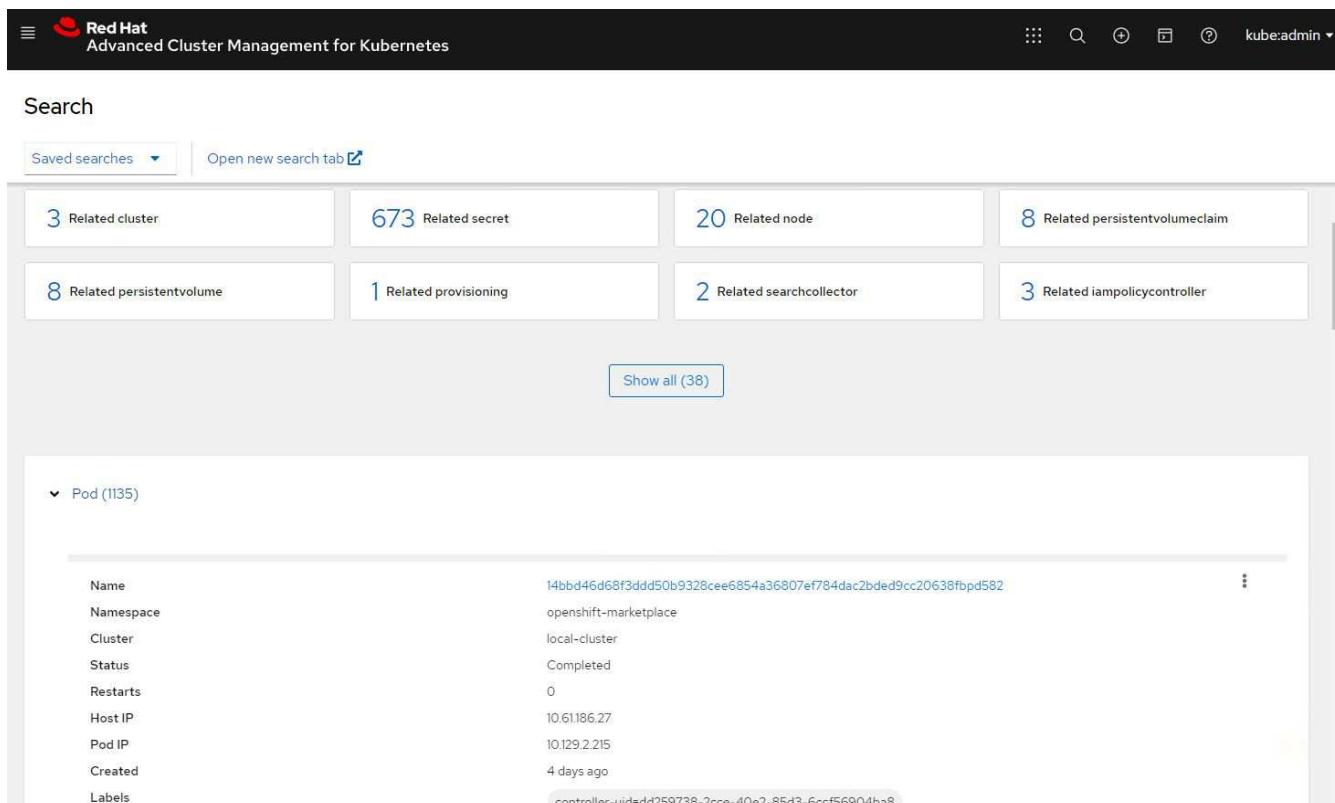
Observability

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.

Search

Saved searches ▼ Open new search tab ↗

3 Related cluster 1k Related pod 12 Related service

Show all (3)

▼ Node (20)

Name	Cluster	Role	Architecture	OS image	CPU	Created	Labels
ocp-master-1-ocp-bare-metal.cie.netapp.com	ocp-bare-metal	master; worker	amd64	Red Hat Enterprise Linux CoreOS 47.83.202103292105-0 (Octpa)	48	a month ago	beta.kubernetes.io/arch=amd64 beta.kubernetes.io/os=linux kubernetes.io/arch=amd64 5 more
ocp-master-2-ocp-bare-metal.cie.netapp.com	ocp-bare-metal	master; worker	amd64	Red Hat Enterprise Linux CoreOS 47.83.202103292105-0 (Octpa)	48	a month ago	beta.kubernetes.io/arch=amd64 beta.kubernetes.io/os=linux kubernetes.io/arch=amd64 5 more
ocp-master-3-ocp-bare-metal.cie.netapp.com	ocp-bare-metal	master; worker	amd64	Red Hat Enterprise Linux CoreOS 47.83.202103292105-0 (Octpa)	48	a month ago	beta.kubernetes.io/arch=amd64 beta.kubernetes.io/os=linux kubernetes.io/arch=amd64 5 more

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

Search

Saved searches ▼ Open new search tab ↗

3k Related secret 787 Related pod 15 Related persistentvolumeclaim 17 Related node 1 Related application

15 Related persistentvolume 1 Related searchcollector 8 Related clusterclaim 3 Related resourcequota 5 Related identity

Show all (159)

▼ Cluster (2)

Name	Available	Hub accepted	Joined	Nodes	Kubernetes version	CPU	Memory	Console URL	Labels
local-cluster	True	True	True	8	v1.20.0+c8905da	84	418501Mi	Launch	cloud=VSphere clusterID=14863d9-69d5-4ae4-98ee-8dff886463c3 installer.name=multiclusterhub 4 more
ocp-vmw	True	True	True	9	v1.20.0+df9c838	28	111981Mi	Launch	cloud=VSphere clusterID=9d76ac4e-4aae-4d45-a2e8-11b6b54282fe name=ocp-vmw 1 more

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.

Create resource

Cancel

Create

Clusters | Select the clusters where the resource(s) will be deployed.

2 x

local-cluster,  ocp-vmw

Resource configuration | Enter the configuration manifest for the resource(s).

YAML

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

Data protection for Container Apps and VMs using Trident protect

This solution shows how to use Trident Protect to perform Data protection operations for Containers and VMs.

1. For details about creating snapshots and backups and restoring from them for container applications in OpenShift Container platform, refer [here](#).
2. For details about creating and restoring from a backup for VMs in OpenShift Virtualization deployed on the OpenShift Container platform, refer [here](#).

Data protection for Container Apps and VMs using 3rd party tools

This solution shows how to use Velero that is integrated with the OADP operator in Red Hat OpenShift Container platform to perform Data protection operations for Containers and VMs.

1. For details about creating and restoring from a backup for container applications in OpenShift Container platform, refer [here](#).
2. For details about creating and restoring from a backup for VMs in OpenShift Virtualization deployed on the OpenShift Container platform, refer [here](#).

Additional resources to learn about Red Hat OpenShift Virtualization integration with NetApp storage

Access additional resources that offer more information on supporting deployment, management, and optimization of Red Hat OpenShift Virtualization with ONTAP across various platforms and technologies.

- NetApp Documentation

<https://docs.netapp.com/>

- Trident Documentation

<https://docs.netapp.com/us-en/trident/index.html>

- 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/>

Copyright information

Copyright © 2026 NetApp, Inc. All Rights Reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

LIMITED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data -Noncommercial Items at DFARS 252.227-7013 (FEB 2014) and FAR 52.227-19 (DEC 2007).

Data contained herein pertains to a commercial product and/or commercial service (as defined in FAR 2.101) and is proprietary to NetApp, Inc. All NetApp technical data and computer software provided under this Agreement is commercial in nature and developed solely at private expense. The U.S. Government has a non-exclusive, non-transferrable, nonsublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b) (FEB 2014).

Trademark information

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.