



## **Solution Validation and Use Cases**

NetApp Solutions

NetApp

August 18, 2021

This PDF was generated from [https://docs.netapp.com/us-en/netapp-solutions/containers/rh-os-n\\_use\\_case\\_pipeline.html](https://docs.netapp.com/us-en/netapp-solutions/containers/rh-os-n_use_case_pipeline.html) on August 18, 2021. Always check docs.netapp.com for the latest.

# Table of Contents

Solution Validation and Use Cases: Red Hat OpenShift with NetApp .....	1
Deploy a Jenkins CI/CD Pipeline with Persistent Storage: Red Hat OpenShift with NetApp .....	1
Configure Multi-tenancy on Red Hat OpenShift with NetApp ONTAP .....	11
Red Hat OpenShift Virtualization with NetApp ONTAP .....	32
Advanced Cluster Management for Kubernetes on Red Hat OpenShift with NetApp .....	51

# Solution Validation and Use Cases: Red Hat OpenShift with NetApp

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

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

Next: [Videos and Demos](#).

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

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

### Create the resources required for Jenkins deployment

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

1. Create a new project named Jenkins.

# 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
All Items

Languages

Group By: None ▾

Middleware

CI/CD

Other

Type

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

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.

x

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.
<a href="#">Get support ↗</a>	
Created At	May 26, 3:58 am
	<b>Documentation</b>
	<a href="https://docs.okd.io/latest/using_images/other_images/jenkins.html" style="color: #0070C0;">https://docs.okd.io/latest/using_images/other_images/jenkins.html ↗</a>

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 \*

Jenkins Service Name

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

Jenkins JNLP Service Name

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

Enable OAuth in Jenkins

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

Memory Limit

Maximum amount of memory the container can use.

Volume Capacity \*

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

Jenkins ImageStream Namespace

The OpenShift Namespace where the Jenkins ImageStream resides.

Disable memory intensive administrative monitors

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

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

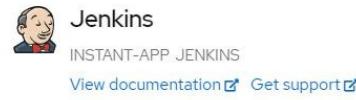
Fatal Error Log File

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

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 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 10–12 minutes to enter the Ready state.

Project: jenkins ▾

## Pods

Create Pod

Filter by name...

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	⋮
P jenkins-1-c77n9	NS jenkins	Running	1/1	RC 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.

Project: jenkins ▾

## Routes

Create Route

Filter by name...

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

Name	Namespace	Status	Location	Service	⋮
RT jenkins	NS jenkins	Accepted	<a href="https://jenkins-jenkins.apps.rhv-ocp-cluster.cie.netapp.com">https://jenkins-jenkins.apps.rhv-ocp-cluster.cie.netapp.com</a> ↗	S 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 M3

Install automatically

Install from Apache

Version 3.6.3

Delete Installer

Add Installer

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's a navigation bar with the Jenkins logo, a search bar, and user information (kube:admin). Below the bar, a sidebar on the left lists various Jenkins features: New Item, People, Build History, Manage Jenkins, My Views, Open Blue Ocean, Lockable Resources, Credentials, and New View. The main content area has a heading 'Welcome to Jenkins!' and a message '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' dialog. The input field contains 'sample-demo'. Below the input field, there's a note '» Required field'. A list of project types is shown:

- Freestyle project**: 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**: 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**: Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Bitbucket Team/Project**: Scans a Bitbucket Cloud Team (or Bitbucket Server Project) for all repositories matching some defined markers.
- 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**: Scans a GitHub organization (or user account) for all repositories matching some defined markers.
- Multibranch Pipeline**: 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](#)

**Build History** [trend](#)

build	last run	changes
#1	May 27, 2020 3:53 PM	No Changes

[Atom feed for all](#) [Atom feed for failures](#)

## 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.

Next: Videos and Demos.

## Configure Multi-tenancy on Red Hat OpenShift with NetApp ONTAP

### Configuring Multitenancy on Red Hat OpenShift with NetApp: Red Hat OpenShift with NetApp

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

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

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

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

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

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

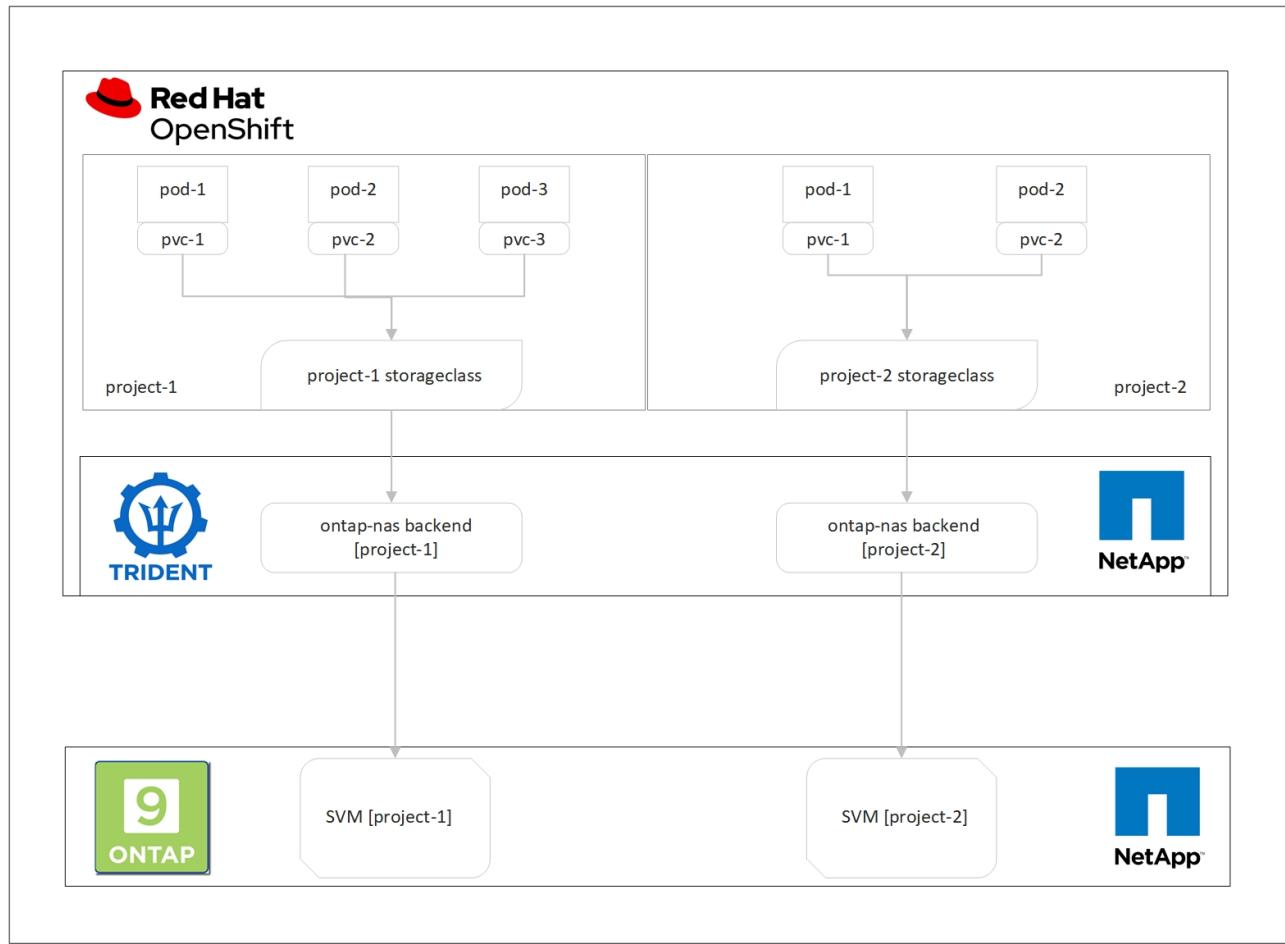
[Next: Architecture.](#)

## Architecture

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

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

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



## Technology requirements

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

## Red Hat OpenShift – Cluster resources

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

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

## NetApp ONTAP

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

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

## Astra Trident

After we have different SVMs for different projects created on NetApp ONTAP, we need to map each SVM to a different Trident backend.

The backend configuration on Trident drives the allocation of persistent storage to OpenShift cluster resources and it requires the details of the SVM to be mapped to, protocol driver for the backend at the minimum. Optionally, it allows us to define how the volumes are provisioned on the storage and to set limits for the size of volumes or usage of aggregates etc. Details of defining the Trident backend for NetApp ONTAP can be found [here](#).

## Red Hat OpenShift – storage resources

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

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

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

[Next: Configuration.](#)

## Configuration

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

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

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

[Next: Prerequisites.](#)

## Configuration

### Pre-requisites

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

[Next: Cluster Administrator Tasks.](#)

### Configuration: cluster-admin tasks

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

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

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

### 3. Create the developer role for project-1.

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

```

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

```



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

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

```

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

```

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

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

#### 8. Configure ClusterRoleBindings for storage admins.

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



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

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

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

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

[Next: Storage Administrator Tasks.](#)

### Configuration: Storage-admin tasks

The following resources must be configured by a storage administrator:

1. Log into the NetApp ONTAP cluster as admin.
2. Navigate to [Storage → Storage VMs](#) and click [Add](#). Create two SVMs, one for project-1 and the other for project-2, providing the required details. Also create an vsadmin account to manage the SVM and its resources.

# 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

SUBNET MASK

GATEWAY

BROADCAST DOMAIN

10.61.181.224

24

Add optional  
gateway

Default-4



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

```

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

```



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



We assume that Trident is installed in trident project.

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

```

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

```

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

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

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

[Next: Validation.](#)

## Validation

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

### Validate access to create PVCs/pods in assigned project

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

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

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

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

4. Check the PV associated with the PVC.

```
oc get pv
```

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

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

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

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

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

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

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

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

```

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

```

### 3. Create a PVC in project-2.

```

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

```

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

```

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

```

### 5. Create a pod in project-2.

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

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

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

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

3. Validate access to view projects.

```
oc get ns
```

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

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

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

```
oc get sc
```

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

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

[Next: Scaling.](#)

## Scaling: Adding more projects

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

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

# 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/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

10.61.181.228

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 sync'd with the OpenShift cluster.

```
oc get groups
```

## 6. Create the developer role for project-3.

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

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



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

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

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

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

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



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



We assume that Trident is installed in the trident project.

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

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

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

```

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

```

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

```

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

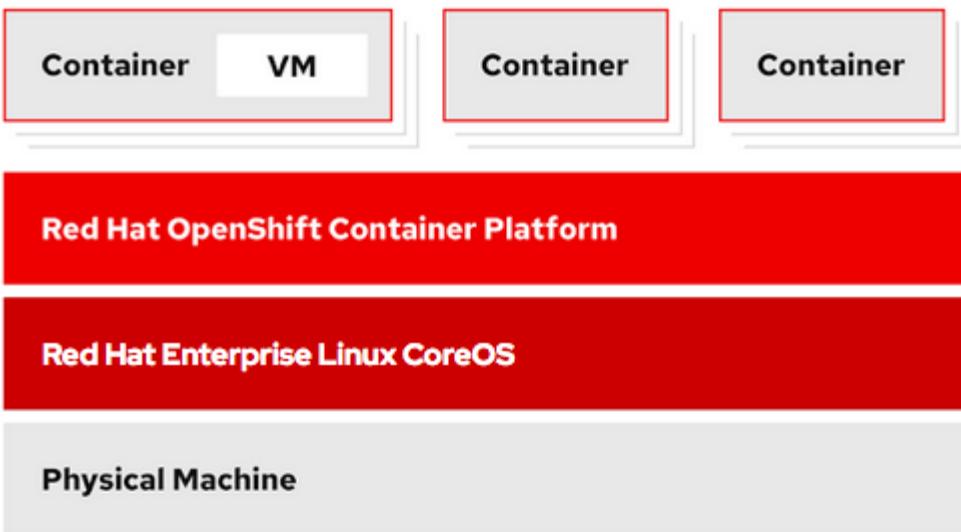
```

## Red Hat OpenShift Virtualization with NetApp ONTAP

### Red Hat OpenShift Virtualization with NetApp ONTAP

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

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



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

To learn more about Red Hat OpenShift Virtualization, see the documentation [here](#).

[Next: Deployment Prerequisites.](#)

## Deployment

### Deploy Red Hat OpenShift Virtualization with NetApp ONTAP

#### Prerequisites:

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

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

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

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

Next: Deploy via operator.

## Deploy Red Hat OpenShift Virtualization with NetApp ONTAP

To install OpenShift Virtualization, complete the following steps:

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



4. Select the OpenShift Virtualization tile and click Install.



## OpenShift Virtualization

2.6.2 provided by Red Hat



Install

### Latest version

2.6.2

### Capability level

- Basic Install
- Seamless Upgrades
- Full Lifecycle
- Deep Insights
- Auto Pilot

### Provider type

Red Hat

### Provider

Red Hat

## Requirements

Your cluster must be installed on bare metal infrastructure with Red Hat Enterprise Linux CoreOS workers.

## Details

**OpenShift Virtualization** extends Red Hat OpenShift Container Platform, allowing you to host and manage virtualized workloads on the same platform as container-based workloads. From the OpenShift Container Platform web console, you can import a VMware virtual machine from vSphere, create new or clone existing VMs, perform live migrations between nodes, and more. You can use OpenShift Virtualization to manage both Linux and Windows VMs.

The technology behind OpenShift Virtualization is developed in the [KubeVirt](#) open source community. The KubeVirt project extends [Kubernetes](#) by adding additional virtualization resource types through [Custom Resource Definitions](#) (CRDs). Administrators can use Custom Resource Definitions to manage [VirtualMachine](#) resources alongside all other resources that Kubernetes provides.

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

Update channel \*

2.1  
 2.2  
 2.3  
 2.4  
 stable

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: openshift-cnv

Namespace creation  
Namespace `openshift-cnv` does not exist and will be created.

Select a Namespace

Approval strategy \*

Automatic  
 Manual

OpenShift Virtualization  
provided by Red Hat

Provided APIs

OpenShift Virtualization Deployment Required  
Represents the deployment of OpenShift Virtualization

6. Wait for the operator installation to complete.



**OpenShift Virtualization**  
2.6.2 provided by Red Hat

A progress bar at the bottom is mostly filled.

## Installing Operator

The Operator is being installed. This may take a few minutes.

[View installed Operators in Namespace openshift-cnv](#)

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



**OpenShift Virtualization**  
2.6.2 provided by Red Hat

A green checkmark icon is present on the right side.

## Installed operator - operand required

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

 **HyperConverged**  **Required**

Creates and maintains an OpenShift Virtualization Deployment

[Create HyperConverged](#)

[View installed Operators in Namespace openshift-cnv](#)

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

Name \*

kubevirt-hyperconverged

Labels

app=frontend

Infra

infra HyperConvergedConfig influences the pod configuration (currently only placement) for all the infra components needed on the virtualization enabled cluster but not necessarily directly on each node running VMs/VMIs.

Workloads

workloads HyperConvergedConfig influences the pod configuration (currently only placement) of components which need to be running on a node where virtualization workloads should be able to run. Changes to Workloads HyperConvergedConfig can be applied only without existing workload.

Bare Metal Platform



true

BareMetalPlatform indicates whether the infrastructure is baremetal.

Feature Gates

featureGates is a map of feature gate flags. Setting a flag to `true` will enable the feature. Setting `false` or removing the feature gate, disables the feature.

Local Storage Class Name

LocalStorageClassName the name of the local storage class.

Create

Cancel

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

Project: openshift-cnv ▾

## 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	Last updated	Provided APIs
 <a href="#">OpenShift Virtualization</a> 2.6.2 provided by Red Hat	 <a href="#">openshift-cnv</a>	<span>✓ Succeeded</span> Up to date	 May 18, 8:02 pm	<a href="#">OpenShift Virtualization Deployment</a> <a href="#">HostPathProvisioner deployment</a>

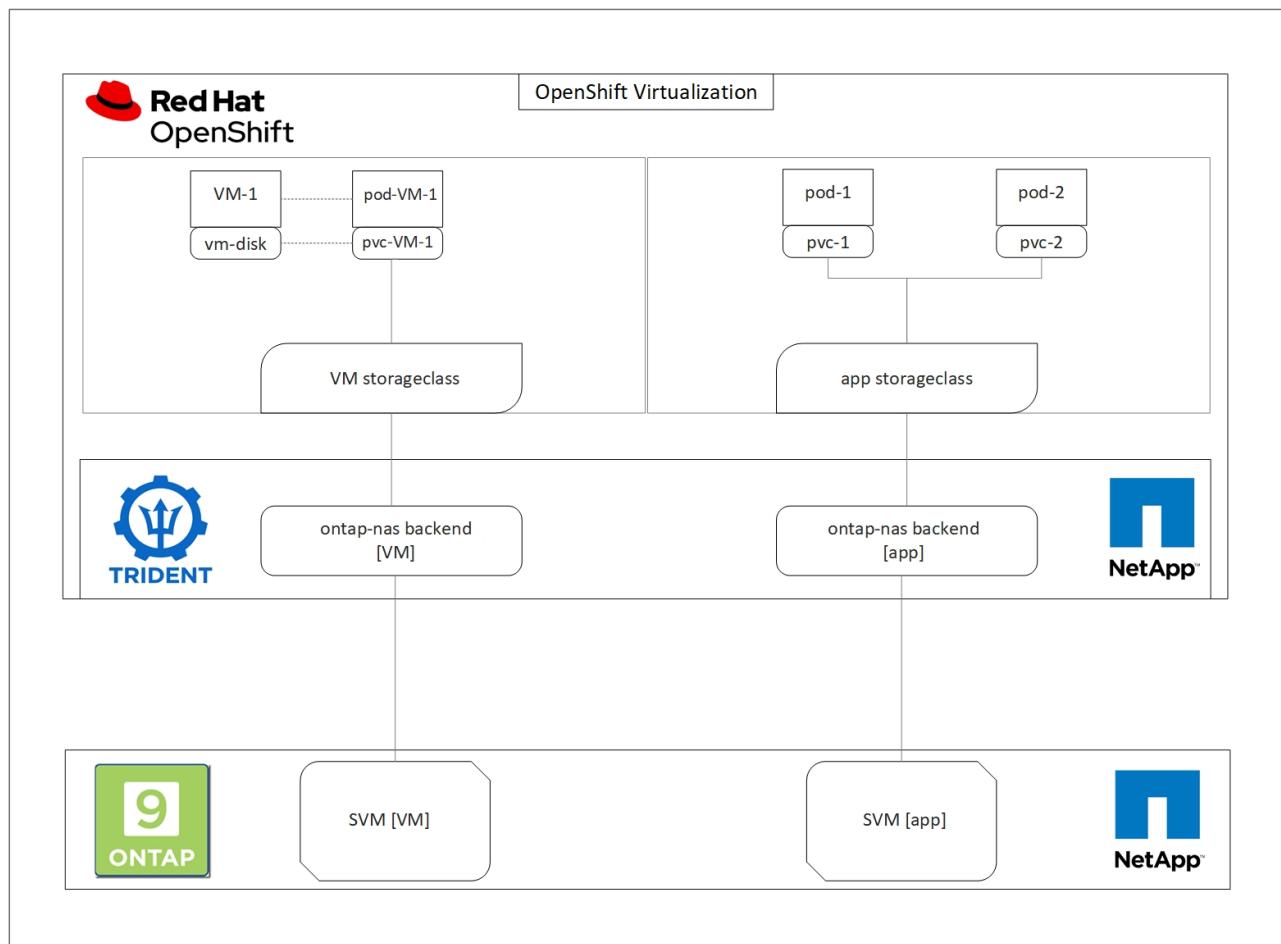
Next: [Workflows: Create VM.](#)

## Workflows

### Workflows: Red Hat OpenShift Virtualization with NetApp ONTAP

## Create VM

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



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

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

## Boot source

This template does not have a boot source. Provide a custom boot source for this **CentOS 8.0+** VM virtual machine.

### Boot source type \*

Import via URL (creates PVC)

### Import URL \*

<https://access.cdn.redhat.com/content/origin/files/sha256/58/588167f828001e57688ec4b9b31c11a59d532489f527488ebc89ac5e952...>

Example: For RHEL, visit the [RHEL download page](#) (requires login) and copy the download link URL of the KVM guest image

Mount this as a CD-ROM boot source ?

### Persistent Volume Claim size \*

5 GiB ▾

Ensure your PVC size covers the requirements of the uncompressed image and any other space requirements. More storage can be added later.

### Advanced

### Storage class \*

basic (default)

### Access mode \*

Single User (RWO)

### Volume mode \*

Filesystem

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

1 Select template

**2 Review and create**

**Review and create**  
You are creating a virtual machine from the Red Hat Enterprise Linux 8.0+ VM template.

**Project \***  
PR default

**Virtual Machine Name \*** ⓘ  
rhe18-light-bat

**Flavor \***  
Small:1CPU | 2 GiB Memory

**Storage**      **Workload profile** ⓘ  
40 GiB      server

**Boot source**  
Clone and boot from CD-ROM  
**PVC** rhe18

ⓘ A new disk has been added to support the CD-ROM boot source. Edit this disk by customizing the virtual machine.  
▼ Disk details

rootdisk-install - Blank - 20GiB - virtio - default Storage class

Start this virtual machine after creation

**Create virtual machine**    **Customize virtual machine**    Back    Cancel

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

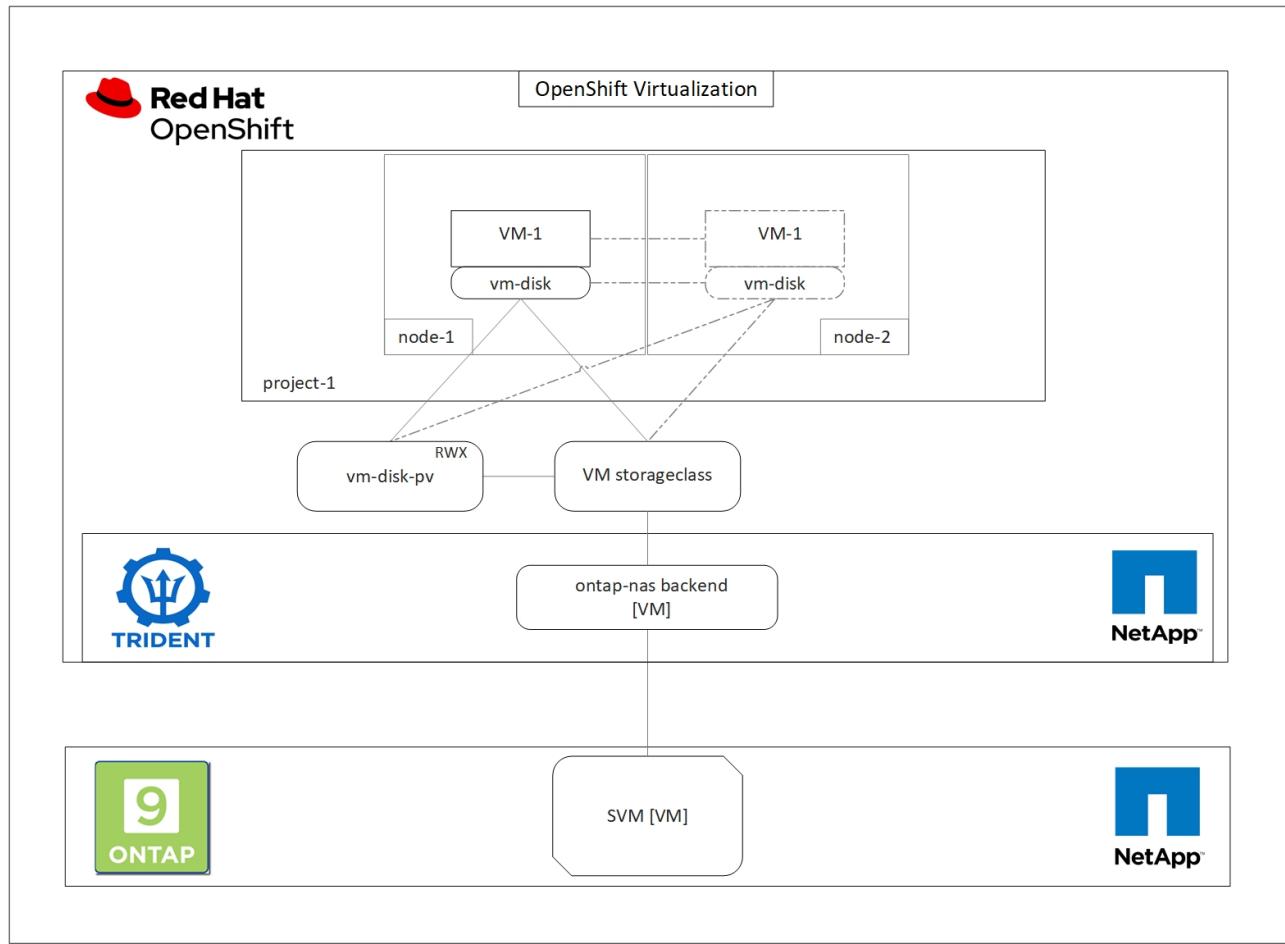
When a boot source is configured for a template or an operating system from an URL or from a registry, it creates a PVC in the `openshift-virtualization-os-images` project and downloads the KVM guest image to the PVC. You must make sure that template PVCs have enough provisioned space to accommodate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

Next: [Workflows: VM Live Migration](#).

## Workflows: Red Hat OpenShift Virtualization with NetApp ONTAP

### VM Live Migration

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



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

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

## Edit Disk

Type: Disk

Interface \*

virtio

Storage Class

basic (default)

▼ Advanced

Volume Mode

Filesystem

Volume Mode is set by Source PVC

Access Mode

Shared Access (RWX) - Not recommended for basic storage class

**ⓘ Access and Volume modes should follow storage feature matrix**

[Learn more ↗](#)

Cancel Save

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

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

1. Navigate to [Workloads > Virtualization > Virtual Machines](#).

2. For the VM you wish to migrate, click the ellipsis, and then click Migrate the Virtual Machine.

3. Click Migrate when the message pops up to confirm.



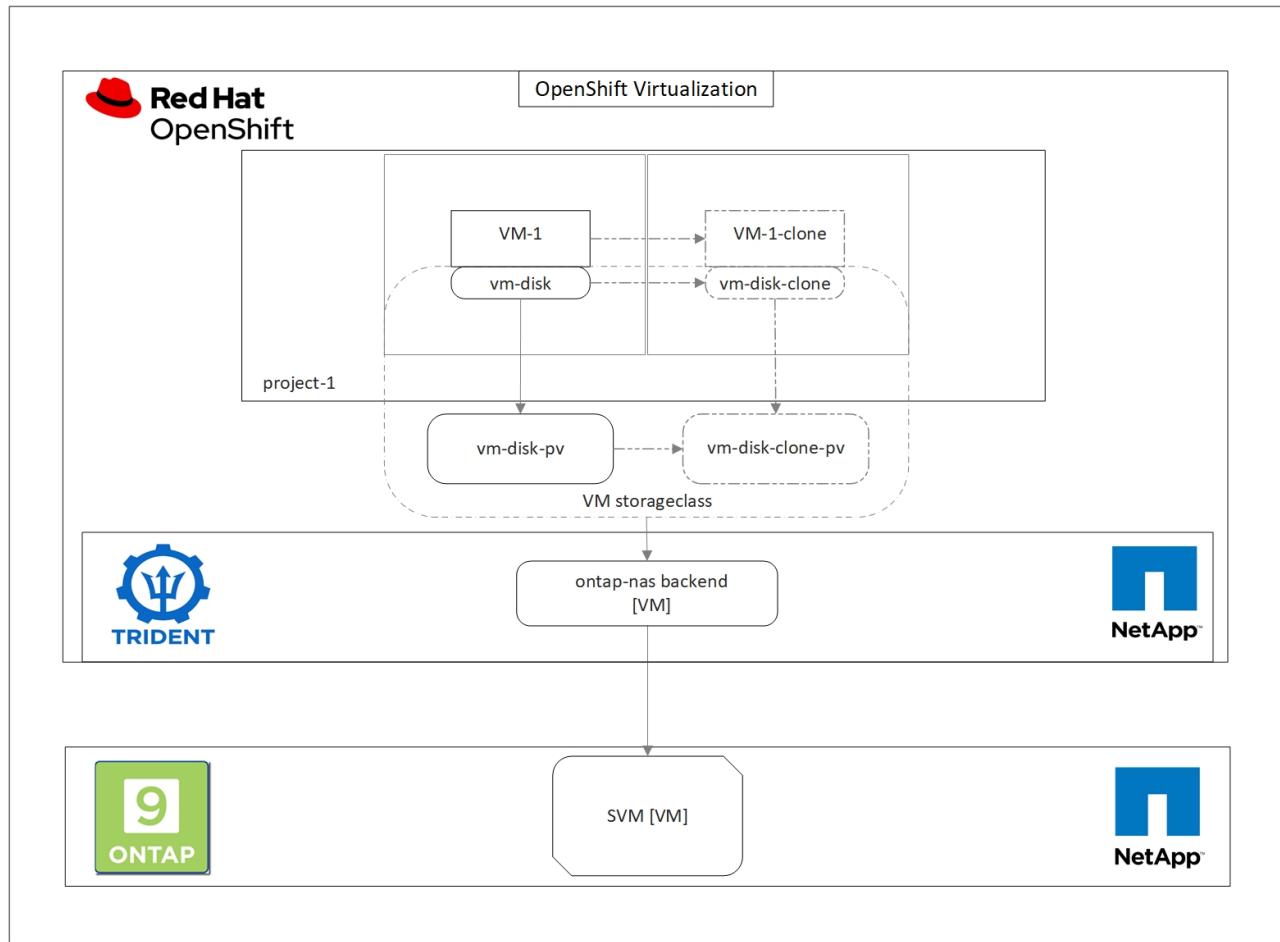
A VM instance in an OpenShift cluster automatically migrates to another node when the original node is placed into maintenance mode if the evictionStrategy is set to LiveMigrate.

[Next: Workflows: VM Cloning.](#)

## Workflows: Red Hat OpenShift Virtualization with NetApp ONTAP

### VM cloning

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



There are certain restrictions with CSI volume cloning to consider:

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

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

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

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

### **By Shutting down the source VM**

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

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

# Clone Virtual Machine

Name *	<input type="text" value="rhel8-short-frog-clone"/>											
Description	<input type="text"/>											
Namespace *	<input type="text" value="default"/>											
<input checked="" type="checkbox"/> Start virtual machine on clone												
Configuration	<table><tr><td>Operating System</td></tr><tr><td>Red Hat Enterprise Linux 8.0 or higher</td></tr><tr><td>Flavor</td></tr><tr><td>Small: 1 CPU   2 GiB Memory</td></tr><tr><td>Workload Profile</td></tr><tr><td>server</td></tr><tr><td>NICs</td></tr><tr><td>default - virtio</td></tr><tr><td>Disks</td></tr><tr><td>cloudinitdisk - cloud-init disk</td></tr><tr><td>rootdisk - 20Gi - basic</td></tr></table>	Operating System	Red Hat Enterprise Linux 8.0 or higher	Flavor	Small: 1 CPU   2 GiB Memory	Workload Profile	server	NICs	default - virtio	Disks	cloudinitdisk - cloud-init disk	rootdisk - 20Gi - basic
Operating System												
Red Hat Enterprise Linux 8.0 or higher												
Flavor												
Small: 1 CPU   2 GiB Memory												
Workload Profile												
server												
NICs												
default - virtio												
Disks												
cloudinitdisk - cloud-init disk												
rootdisk - 20Gi - basic												



The VM rhel8-short-frog is still running. It will be powered off while cloning.

[Cancel](#)

[Clone Virtual Machine](#)

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

## By keeping the source VM live

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

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

## Clone

Name \*

rhel8-short-frog-rootdisk-28dvb-clone

Access Mode \*

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

Size \*

20

GiB



PVC details

Namespace	Requested capacity	Access mode
NS default	20 GiB	Shared Access (RWX)
Storage Class	Used capacity	Volume mode
SC basic	2.2 GiB	Filesystem

Cancel

Clone

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

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

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

Next: [Workflows: Create VM from a Snapshot](#).

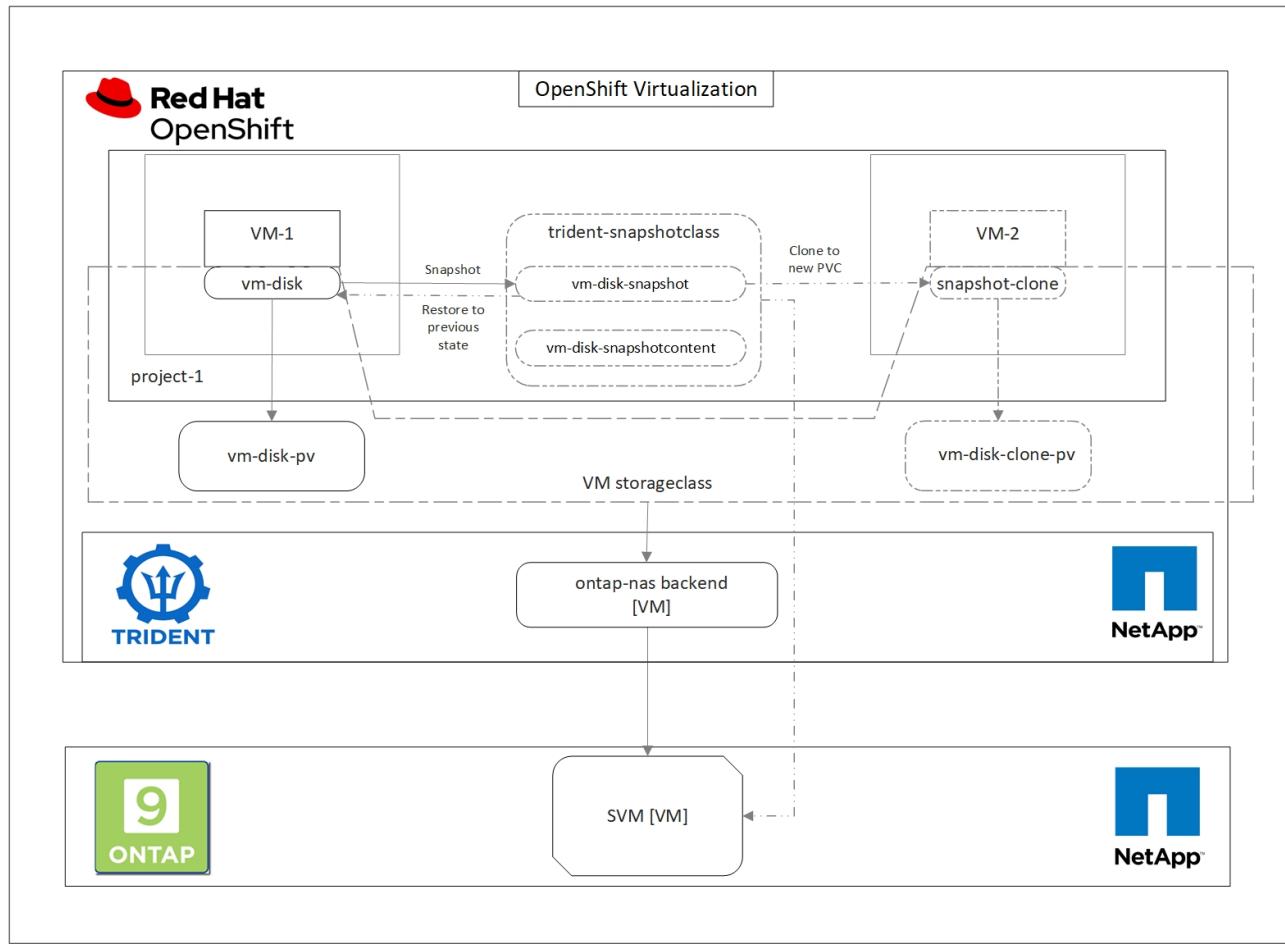
## Workflows: Red Hat OpenShift Virtualization with NetApp ONTAP

### Create VM from a Snapshot

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

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

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



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

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

[View shortcuts](#)

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

[Create](#)

[Cancel](#)

[Download](#)

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

## Create VolumeSnapshot

[Edit YAML](#)

**PersistentVolumeClaim \***

[PVC](#) rhel8-short-frog-rootdisk-28dvh

**Name \***

rhel8-short-frog-rootdisk-28dvh-snapshot

**Snapshot Class \***

[VSC](#) trident-snapshot-class

[Create](#)

[Cancel](#)

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

#### Create a new VM from the snapshot

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

## Restore as new PVC

When restore action for snapshot **rhel8-short-frog-rootdisk-28dvb-snapshot** is finished a new crash-consistent PVC copy will be created.

Name \*

rhel8-short-frog-rootdisk-28dvb-snapshot-restore

Storage Class \*

SC basic

Access Mode \*

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

Size \*

20

GiB



VolumeSnapshot details

Created at

May 21, 12:46 am

Namespace

default

Status

Ready

API version

snapshot.storage.k8s.io/v1

Size

20 GiB

3. Next, create a new VM from this PVC. Navigate to [Workloads > Virtualization > Virtual Machines](#) and click [Create → With YAML](#).
4. In the `spec > template > spec > volumes` section, specify the new PVC created from Snapshot

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

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

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

## Advanced Cluster Management for Kubernetes on Red Hat OpenShift with NetApp

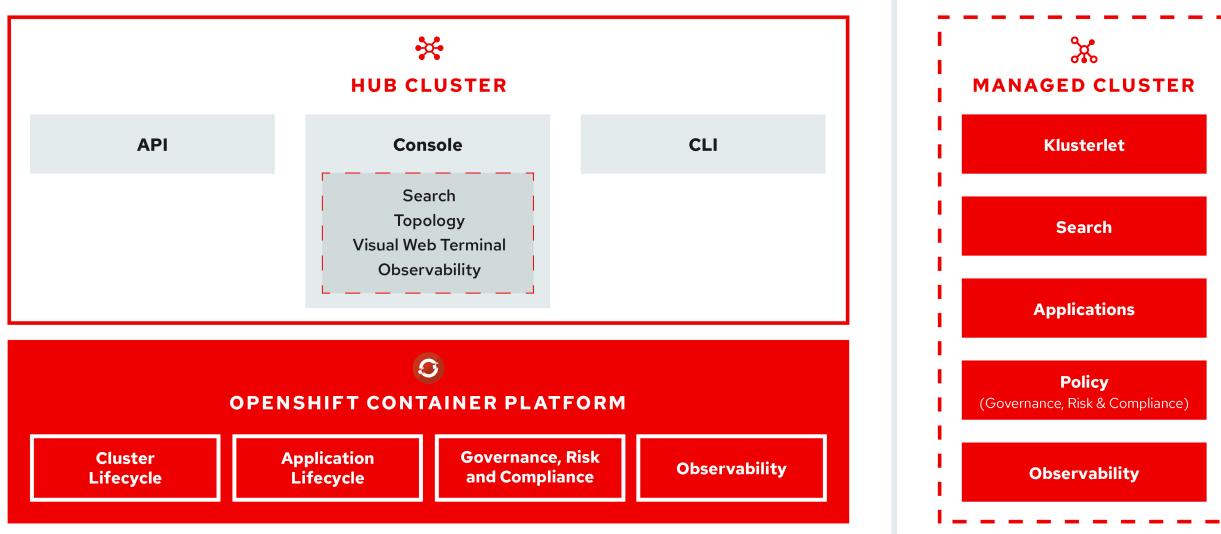
### Advanced Cluster Management for Kubernetes: Red Hat OpenShift with NetApp

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

Red Hat Advanced Cluster Management for Kubernetes allows the users to:

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

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



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

[Next: Deployment Prerequisites.](#)

## Deployment

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

#### Prerequisites

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

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

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

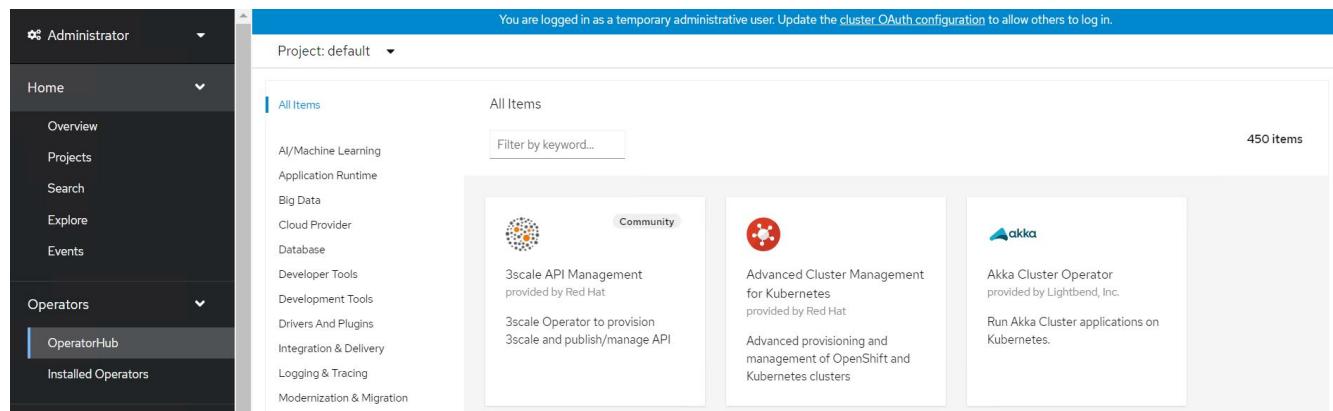
[Next: Installation.](#)

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

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

1. Choose an OpenShift cluster as the hub cluster and log into it with cluster-admin privileges.
2. Navigate to [Operators](#) → [Operators Hub](#) and search for [Advanced Cluster Management for](#)

## Kubernetes.



The screenshot shows the OperatorHub interface. On the left, there's a sidebar with 'Administrator' at the top, followed by 'Home' with sub-options like 'Overview', 'Projects', 'Search', 'Explore', 'Events', and 'Operators'. Under 'Operators', 'OperatorHub' is selected. Below the sidebar is a search bar with 'All Items' and 'Filter by keyword...' options. The main area displays a grid of operator cards. One card for '3scale API Management' is highlighted. Other cards include 'Advanced Cluster Management for Kubernetes' (provided by Red Hat), 'Akka Cluster Operator' (provided by Lightbend, Inc.), and 'akka'. A total of 450 items are shown.

3. Select the [Advanced Cluster Management for Kubernetes](#) and click [Install](#).



**Advanced Cluster Management for Kubernetes**  
2.2.3 provided by Red Hat

**Install**

**Latest version**  
2.2.3

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

**Provider type**  
Red Hat

**Provider**  
Red Hat

**Infrastructure features**  
Disconnected

**How to Install**  
Use of this Red Hat product requires a licensing and subscription agreement.

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:  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.



**Advanced Cluster Management for Kubernetes**  
2.2.3 provided by Red Hat

### Installing Operator

The Operator is being installed. This may take a few minutes.

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

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 \*

multiclusetherub

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.

Project: open-cluster-management ▾

## 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 ClusterSt...  
[progress bar]

### MultiClusterHubs

Create MultiClusterHub

Name	Kind	Status	Labels	⋮
MCH multicloudclusterhub	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.

Project: open-cluster-management ▾

## Routes

Create Route

Filter ▾ Name mul

Name mul

Name	Status	Location	Service	⋮
RT multicloud-console	Accepted	<a href="https://multicloud-console.apps.ocp-vmware2.cie.netapp.com">https://multicloud-console.apps.ocp-vmware2.cie.netapp.com</a>	S management-ingress	

[Next: Features - Cluster Lifecycle Management.](#)

# Features

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

### Cluster Lifecycle Management

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

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

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 \* [?](#)

AKIATCFBZDOIASDSAH

AWS secret access key \* [?](#)

.....

Red Hat OpenShift pull secret \* [?](#)

```
FuS3pNbktVaHplNFc2MkZsbmtBVGN6TktmUlZXcHcxOW9teEZwQ0lYZld3cjJobGxJeDBQNOxlZE0yeGM5QOZwZk5RR2JUanlxNnNUM21Rb0FJbUFjNCIBYlpEWVZE0HltNxTMDZPUVpoWFRHcGwtREIDQ2RSYIJRaTlxblldLT2oyQ3pVeUJfNllwcENSa2YyOUsyLWZGSFVfNA==,"email":"Nikhil.kulkarni@netapp.com"}, "registry.redhat.io":
```

SSH private key \* [?](#)

```
-----BEGIN OPENSSH PRIVATE KEY-----b3BlbnNzaC1rZXktdjEAAAAABG5vbmuUAAAEBasdadssadm9uZQAAAAAAAAAABAAAAMwAAAAtzc2gtZWQyNTUxOQAAACCLcwLgAvSIHAEp+DevIRNzaG2zkNreMIZ/UHyfOUWvAAAAAJh/wa6xf8Gu
```

SSH public key \* [?](#)

```
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIlzAuAC746agdh2lcB4/4N6/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.

<input checked="" type="checkbox"/> AWS Amazon Web Services	<input type="checkbox"/> Google Cloud	<input type="checkbox"/> Microsoft Azure
<input type="checkbox"/> VMware vSphere	<input type="checkbox"/> 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:
- a. Navigate to **Clusters** and click **Add a cluster > Import an existing cluster**.
  - b. Enter the name of the cluster and click **Save import and generate code**. A command to add the existing cluster is displayed.
  - c. Click **Copy command** and run the command on the cluster to be added to the hub cluster. This initiates the installation of the necessary agents on the cluster, and, after this process is complete, the cluster appears in the cluster list with status **Ready**.

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.

[Next: Features - Application Lifecycle Management.](#)

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

### Application lifecycle management

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

1. Navigate to [Manage Applications](#) from the sidebar and click [Create application](#). Provide the details of the application you would like to create and click [Save](#).

Applications /

## 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 ⓘ

- 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	Resource	Time window	Created	⋮
demo-app	default	Local	Git		8 days ago	⋮

1-1 of 1 << < 1 of 1 > >>

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

Next: Features - governance and risk.

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

### Governance and Risk

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

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

# Create policy i



YAML: Off

**Name \***

policy-complianceoperator

**Namespace \*** i

default

**Specifications \*** i

1x ComplianceOperator

**Cluster selector** i

1x local-cluster: "true"

**Standards** i

1x NIST-CSF

**Categories** i

1x PR.IP Information Protection Processes and Procedures

**Controls** i

1x PR.IP-1 Baseline Configuration

 Enforce if supported i Disable policy i

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

## Governance and risk ⓘ

Filter

Refresh every 10s

Last update: 12:54:01 PM

Create policy

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	PR.IP Information Protection Processes and Procedures	PR.IP-1 Baseline Configuration	32 minutes ago

1 - 1 of 1 ▾ << < 1 of 1 > >>

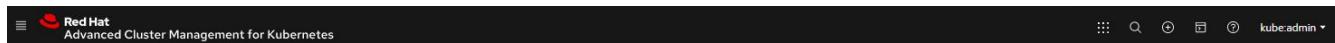
Next: Features - Observability.

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

### Observability

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

1. Navigate to [Observe Environments > Overview](#).



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

This screenshot shows the search interface of the Red Hat Advanced Cluster Management for Kubernetes. At the top, there's a header with the Red Hat logo and the title 'Advanced Cluster Management for Kubernetes'. On the right side of the header, there are several icons and a dropdown menu labeled 'kube:admin'. Below the header, there's a 'Search' section with a 'Saved searches' dropdown and a 'Open new search tab' button. The main content area displays a grid of related resources: 3 Related cluster, 673 Related secret, 20 Related node, 8 Related persistentvolumeclaim, 8 Related persistentvolume, 1 Related provisioning, 2 Related searchcollector, and 3 Related iampolicycontroller. Below this grid, there's a button labeled 'Show all (38)'. Underneath, there's a detailed view of a 'Pod (1135)'. The pod details include: Name (14bbd46d68f3ddd50b9328ce6854a36807ef784dac2bded9cc20638fbpd582), Namespace (openshift-marketplace), Cluster (local-cluster), Status (Completed), Restarts (0), Host IP (10.61.186.27), Pod IP (10.129.2.215), Created (4 days ago), and Labels (controller-uid=dd259738-2cce-40e2-85d3-6ccf56904ba8). There's also a three-dot menu icon on the right side of the pod details.

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=148632d9-69d5-4ae4-98ee-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

Next: Features - Create Resources.

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

### Create resources on multiple clusters

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

## Create resource

[Cancel](#)[Create](#)

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

2 ×

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
```

## **Copyright Information**

Copyright © 2021 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.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277-7103 (October 1988) and FAR 52-227-19 (June 1987).

## **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.