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

To deploy Jenkins with persistent storage, complete the following steps:

1. In the upper left corner, change the role from Administrator to Developer. Click +Add and select From Catalog. In the Filter by Keyword bar, search for jenkins. Select Jenkins Service with Persistent Storage.
2. Click **Instantiate Template**.

---

**Jenkins**

Provided by Red Hat, Inc.

**Instantiate Template**

<table>
<thead>
<tr>
<th>Provider</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red Hat, Inc.</td>
<td>Jenkins service, with persistent storage. NOTE: You must have persistent volumes available in your cluster to use this template.</td>
</tr>
</tbody>
</table>

**Support**

Get support

**Documentation**

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
4. The Jenkins pods take approximately 10 to 12 minutes to enter the Ready state.
5. After the pods are instantiated, navigate to Networking > Routes. To open the Jenkins webpage, click the URL provided for the jenkins route.

6. Because OpenShift OAuth was used while creating the Jenkins app, click Log in with OpenShift.
7. Authorize the Jenkins service account to access the OpenShift users.

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

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

11. Select the Pipeline tab. From the Try Sample Pipeline drop-down menu, select Github + Maven. The code is automatically populated. Click Save.
12. Click Build Now to trigger the development through the preparation, build, and testing phase. It can take several minutes to complete the whole build process and display the results of the build.
13. Whenever there are any code changes, the pipeline can be rebuilt to patch the new version of software enabling continuous integration and continuous delivery. Click Recent Changes to track the changes from the previous version.
Configure Multi-tenancy on Red Hat OpenShift with NetApp ONTAP

Configuring multi-tenancy on Red Hat OpenShift with NetApp ONTAP

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

To overcome these problems, one can consider running all the applications or workloads in a single Red Hat OpenShift cluster. But in such an architecture, resource isolation and application security vulnerabilities are one of the major challenges. Any security vulnerability in one workload could naturally spill over into another workload, thus increasing the impact zone. In addition, any abrupt uncontrolled resource utilization by one application can affect the performance of another application, because there is no resource allocation policy by default.
Therefore, organizations look out for solutions that pick up the best in both worlds, for example, by allowing them to run all their workloads in a single cluster and yet offering the benefits of a dedicated cluster for each workload.

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

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

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

Next: Architecture.

**Architecture**

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

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

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

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

Red Hat OpenShift – Cluster resources

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

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

NetApp ONTAP

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

**Astra Trident**

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

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

Next: Prerequisites.

Configuration

Prerequisites

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

Next: Cluster Administrator Tasks.

Configuration: cluster-admin tasks

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

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

3. Create the developer role for project-1.

```yaml
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]
- verbs: ['*']
  apiGroups: ['']
  resources: [bindings, configmaps, endpoints, events, persistentvolumenclaims, pods, pods/log, pods/attach, podtemplates, replicationcontrollers]
```
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.

```bash
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:
        - resourcequotas
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.

```yaml
$ 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
```

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

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

9. Create RoleBindings for developers binding the developer-project-1 role to the corresponding group (ocp-project-1) in project-1.
cat << EOF | oc create -f -
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: project-1-developer
  namespace: project-1
subjects:
  - kind: Group
    apiGroup: rbac.authorization.k8s.io
    name: ocp-project-1
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: developer-project-1
EOF

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

Next: Storage Administrator Tasks.

Configuration: Storage-admin tasks

The following resources must be configured by a storage administrator:

1. Log into the NetApp ONTAP cluster as admin.

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

4. Create the backend for project-1 and map it to the SVM dedicated to the project. NetApp recommends using the SVM’s vsadmin account to connect the backend to SVM instead of using the ONTAP cluster administrator.
cat << EOF | tridentctl -n trident create backend -f
{
   "version": 1,
   "storageDriverName": "ontap-nas",
   "backendName": "nfs_project_1",
   "managementLIF": "172.21.224.210",
   "dataLIF": "10.61.181.224",
   "svm": "project-1-svm",
   "username": "vsadmin",
   "password": "NetApp123"
}
EOF

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

We assume that Trident is installed in the trident project.

5. Similarly create the Trident backend for project-2 and map it to the SVM dedicated to project-2.

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

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

7. Likewise, create a storage class for project-2 and configure it to use the storage pools from backend dedicated to project-2.

8. Create a ResourceQuota to restrict resources in project-1 requesting storage from storageclasses dedicated to other projects.
9. Similarly, create a ResourceQuota to restrict resources in project-2 requesting storage from storageclasses dedicated to other projects.

Next: Validation.

Validation

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

Validate access to create PVCs or pods in assigned project

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

   `oc create ns sub-project-1`

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

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

4. Check the PV associated with the PVC.

   oc get pv

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

   volume show -vserver project-1-svm

6. Create a pod in project-1 and mount the PVC created in previous step.
7. Check if the pod is running and whether it mounted the volume.

```bash
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.
3. Create a PVC in project-2.

```yaml
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.

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

5. Create a pod in project-2.

```bash
```
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.
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.
6. Create the developer role for project-3.

```bash
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
EOF
```
- 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.

```bash
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.
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.

```bash
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

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

To address this challenge, Red Hat introduced OpenShift Virtualization (formerly known as Container Native Virtualization) starting from OpenShift version 4.6. The OpenShift Virtualization feature enables you to run and manage virtual machines alongside containers on the same OpenShift Container Platform installation, providing hybrid management capability to automate deployment and management of VMs through operators. In addition to creating VMs in OpenShift, with OpenShift Virtualization, Red Hat also supports importing VMs from VMware vSphere, Red Hat Virtualization, and Red Hat OpenStack Platform deployments.
Certain features like live VM migration, VM disk cloning, VM snapshots and so on are also supported by OpenShift Virtualization with assistance from Astra Trident when backed by NetApp ONTAP. Examples of each of these workflows are discussed later in this document in their respective sections.

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

Next: Deployment Prerequisites.

Deployment

Deploy Red Hat OpenShift Virtualization with NetApp ONTAP

Prerequisites

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

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

Optionally, you can also specify a subset of the OpenShift cluster nodes to host the OpenShift Virtualization operators, controllers, and VMs by configuring node placement rules. To configure node placement rules for OpenShift Virtualization, follow the documentation here.
For the storage backing OpenShift Virtualization, NetApp recommends having a dedicated StorageClass that requests storage from a particular Trident backend, which in turn is backed by a dedicated SVM. This maintains a level of multitenancy with regard to the data being served for VM-based workloads on the OpenShift cluster.

Next: Deploy via operator.

**Deploy Red Hat OpenShift Virtualization with NetApp ONTAP**

To install OpenShift Virtualization, complete the following steps:

1. Log into the Red Hat OpenShift bare-metal cluster with cluster-admin access.
2. Select Administrator from the Perspective drop down.
4. Select the OpenShift Virtualization tile and click Install.
5. On the Install Operator screen, leave all default parameters and click Install.
6. Wait for the operator installation to complete.

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

8. On the Create HyperConverged screen, click Create, accepting all default parameters. This step starts the installation of OpenShift Virtualization.
9. After all the pods move to the Running state in the openshift-cnv namespace and the OpenShift Virtualization operator is in the Succeeded state, the operator is ready to use. VMs can now be created on the OpenShift cluster.

Next: Workflows: Create VM.

Workflows

Workflows: Red Hat OpenShift Virtualization with NetApp ONTAP
Create VM

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

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

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

5. In the Review and Create pane, select the project you want to create the VM in and furnish the VM details. Make sure that the boot source is selected to be Clone and boot from CD-ROM with the appropriate PVC assigned for the selected OS.
6. If you wish to customize the virtual machine, click Customize Virtual Machine and modify the required parameters.

7. Click Create Virtual Machine to create the virtual machine; this spins up a corresponding pod in the background.

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

Next: Workflows: VM Live Migration.

Workflows: Red Hat OpenShift Virtualization with NetApp ONTAP

VM Live Migration

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

1. Navigate to Workloads > Virtualization > Virtual Machines and click Create > With Wizard.

2. Select the desired the operating system and click Next. Let us assume the selected OS already had a boot source configured with it.

3. In the Review and Create pane, select the project you want to create the VM in and furnish the VM details. Make sure that the boot source is selected to be Clone and boot from CD-ROM with the appropriate PVC assigned for the selected OS.

4. Click Customize Virtual Machine and then click Storage.

5. Click the ellipsis next to rootdisk, and make sure that the storageclass provisioned using Trident is selected. Expand Advanced and select Shared Access (RWX) for Access Mode. Then click Save.
6. Click Review and confirm and then click Create Virtual Machine.

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

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

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


Workflows: Red Hat OpenShift Virtualization with NetApp ONTAP

VM cloning

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

There are certain restrictions with CSI volume cloning to consider:
1. Source PVC and destination PVC must be in the same project.
2. Cloning is supported within the same storage class.
3. Cloning can be performed only when source and destination volumes use the same VolumeMode setting;
for example, a block volume can only be cloned to another block volume.

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

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

**By Shutting down the source VM**

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

1. Navigate to Workloads > Virtualization > Virtual Machines and click the ellipsis next to the virtual machine you wish to clone.
2. Click Clone Virtual Machine and provide the details for the new VM.
3. Click Clone Virtual Machine; this shuts down the source VM and initiates the creation of the clone VM.

4. After this step is completed, you can access and verify the content of the cloned VM.
By keeping the source VM live

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

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

3. Then click Clone. This creates a PVC for the new VM.

4. Navigate to Workloads > Virtualization > Virtual Machines and click Create > With YAML.

5. In the spec > template > spec > volumes section, attach the cloned PVC instead of the container disk. Provide all other details for the new VM according to your requirements.
6. Click Create to create the new VM.
7. After the VM is created successfully, access and verify that the new VM is a clone of the source VM.

Next: Workflows: Create VM from a Snapshot.

Workflows: Red Hat OpenShift Virtualization with NetApp ONTAP

Create VM from a Snapshot

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

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

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

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

4. Select the PVC that you want to create the Snapshot for, enter the name of the Snapshot or accept the default, and select the appropriate VolumeSnapshotClass. Then click Create.

Create VolumeSnapshot

PersistentVolumeClaim *

PVC  rhel8-short-frog-rootdisk-28dwb

Name *

rhel8-short-frog-rootdisk-28dwb-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 Restore as new PVC.

2. Enter the details of the new PVC and click Restore. This creates a new PVC.

3. Next, create a new VM from this PVC. Navigate to Workloads > Virtualization > Virtual Machines and click Create > With YAML.

4. In the spec > template > spec > volumes section, specify the new PVC created from Snapshot instead of
from the container disk. Provide all other details for the new VM according to your requirements.

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

5. Click Create to create the new VM.

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

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

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

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

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

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

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

Next: Deployment Prerequisites.

### Deployment

**Deploy Advanced Cluster Management for Kubernetes**

**Prerequisites**

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

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

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

Next: Installation.

**Deploy Advanced Cluster Management for Kubernetes**

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

1. Choose an OpenShift cluster as the hub cluster and log into it with cluster-admin privileges.

Advanced Cluster Management for Kubernetes
2.2.3 provided by Red Hat

Install

Red Hat Advanced Cluster Management for Kubernetes provides the multicloud hub, a central management console for managing multiple Kubernetes-based clusters across data centers, public clouds, and private clouds. You can use the hub to create Red Hat OpenShift Container Platform clusters on selected providers, or import existing Kubernetes-based clusters. After the clusters are managed, you can set compliance requirements to ensure that the clusters maintain the specified security requirements. You can also deploy business applications across your clusters.

Red Hat Advanced Cluster Management for Kubernetes also provides the following operators:

- Multicluster subscriptions: An operator that provides application management capabilities including subscribing to resources from a channel and deploying those resources on MCH-managed Kubernetes clusters based on placement rules.
- Hive for Red Hat OpenShift: An operator that provides APIs for provisioning and performing initial configuration of OpenShift clusters. These operators are used by the multicloud hub to provide its provisioning and application-management capabilities.

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.
5. Wait for the operator installation to complete.

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

![Create MultiClusterHub](image)

View installed Operators in Namespace open-cluster-management

8. After all the pods move to the Running state in the open-cluster-management namespace and the operator moves to the Succeeded state, Advanced Cluster Management for Kubernetes is installed.
9. It takes some time to complete the hub installation, and, after it is done, the MultiCluster hub moves to Running state.

10. It creates a route in the open-cluster-management namespace. Connect to the URL in the route to access the Advanced Cluster Management console.
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 Add a Connection, provide all the details corresponding to the selected provider type and click Add.
   b. To create a new cluster, navigate to Clusters and click Add a Cluster > Create a Cluster. Provide the details for the cluster and the corresponding provider and click Create.
c. After the cluster is created, it appears in the cluster list with the status Ready.

3. To import an existing cluster, complete the following steps:
   
a. Navigate to Clusters and click Add a Cluster > Import an Existing Cluster.

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

c. Click Copy Command and run the command on the cluster to be added to the hub cluster. This initiates the installation of the necessary agents on the cluster, and, after this process is complete, the cluster appears in the cluster list with status Ready.
4. After you create and import multiple clusters, you can monitor and manage them from a single console.

Next: Features - Application Lifecycle Management.

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

Application lifecycle management

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

1. Navigate to Manage Applications from the sidebar and click Create Application. Provide the details of the application you would like to create and click Save.
2. After the application components are installed, the application appears in the list.

3. The application can now be monitored and managed from the console.
Features: Advanced Cluster Management for Kubernetes on Red Hat OpenShift with NetApp

Governance and risk

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

1. Navigate to Governance and Risk from the sidebar.
2. To create compliance policies, click Create Policy, enter the details of the policy standards, and select the clusters that should adhere to this policy. If you want to automatically remediate the violations of this policy, select the checkbox Enforce if Supported and click Create.
3. After all the required policies are configured, any policy or cluster violations can be monitored and remediated from Advanced Cluster Management.
Features: Advanced Cluster Management for Kubernetes on Red Hat OpenShift with NetApp

Observability

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

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

3. All nodes across the clusters are monitored and analyzed based on a variety of data points. Click Nodes to get more insight into the corresponding details.
4. All clusters are monitored and organized based on different cluster resources and parameters. Click Clusters to view cluster details.

Next: Features - Create Resources.

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

Create resources on multiple clusters

Advanced Cluster Management for Kubernetes allows users to create resources on one or more managed clusters simultaneously from the console. As an example, if you have OpenShift clusters at different sites backed with different NetAppONTAP 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

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

- local-cluster,
- ocp-vmw

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

YAML

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: demo-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 16Gi
  storageClassName: ocp-trident
```