



# Configuration

## NetApp Solutions

NetApp  
August 05, 2022

This PDF was generated from [https://docs.netapp.com/us-en/netapp-solutions/containers/rh-os-n\\_use\\_case\\_multitenancy\\_configuration\\_prerequisites.html](https://docs.netapp.com/us-en/netapp-solutions/containers/rh-os-n_use_case_multitenancy_configuration_prerequisites.html) on August 05, 2022. Always check docs.netapp.com for the latest.

# Table of Contents

- Configuration ..... 1
  - Configuration ..... 1
  - Configuration: cluster-admin tasks ..... 2
  - Configuration: Storage-admin tasks ..... 6
- Validation ..... 9
- Scaling: Adding more projects ..... 14

# Configuration

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

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

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

[Next: Prerequisites.](#)

## Configuration

### Prerequisites

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

[Next: Cluster Administrator Tasks.](#)

# Configuration: cluster-admin tasks

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

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

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

3. Create the developer role for project-1.

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

```
- events
- persistentvolumeclaims
- pods
- pods/log
- pods/attach
- podtemplates
- replicationcontrollers
- services
- limitranges
- namespaces
- componentstatuses
- nodes
- verbs:
  - '*'
apiGroups:
  - trident.netapp.io
resources:
  - trident.snapshots
EOF
```



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

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

```

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

```

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

```

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

```

8. Configure ClusterRoleBindings for storage admins.

```

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

```



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

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

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

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

[Next: Storage Administrator Tasks.](#)

## Configuration: Storage-admin tasks

The following resources must be configured by a storage administrator:

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



# Add Storage VM



STORAGE VM NAME

project-1-svm

## Access Protocol



SMB/CIFS, NFS

iSCSI



Enable SMB/CIFS



Enable NFS



Allow NFS client access

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

EXPORT POLICY

Default

RULES

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

+ Add

DEFAULT LANGUAGE [?](#)

c.utf\_8



NETWORK INTERFACE

Use multiple network interfaces when client traffic is high.

K8s-Ontap-01

IP ADDRESS

10.61.181.224

SUBNET MASK

24

GATEWAY

[Add optional gateway](#)

BROADCAST DOMAIN

Default-4



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

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



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



We assume that Trident is installed in the trident project.

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

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

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

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

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

[Next: Validation.](#)

## Validation

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

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

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

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

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

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

4. Check the PV associated with the PVC.

```
oc get pv
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3. Validate access to view projects.

```
oc get ns
```

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

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

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

```
oc get sc
```

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

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

[Next: Scaling.](#)

## Scaling: Adding more projects

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

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



# Add Storage VM



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 synchronized with the OpenShift cluster.

```
oc get groups
```

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

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

```

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

## Copyright Information

Copyright © 2022 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.