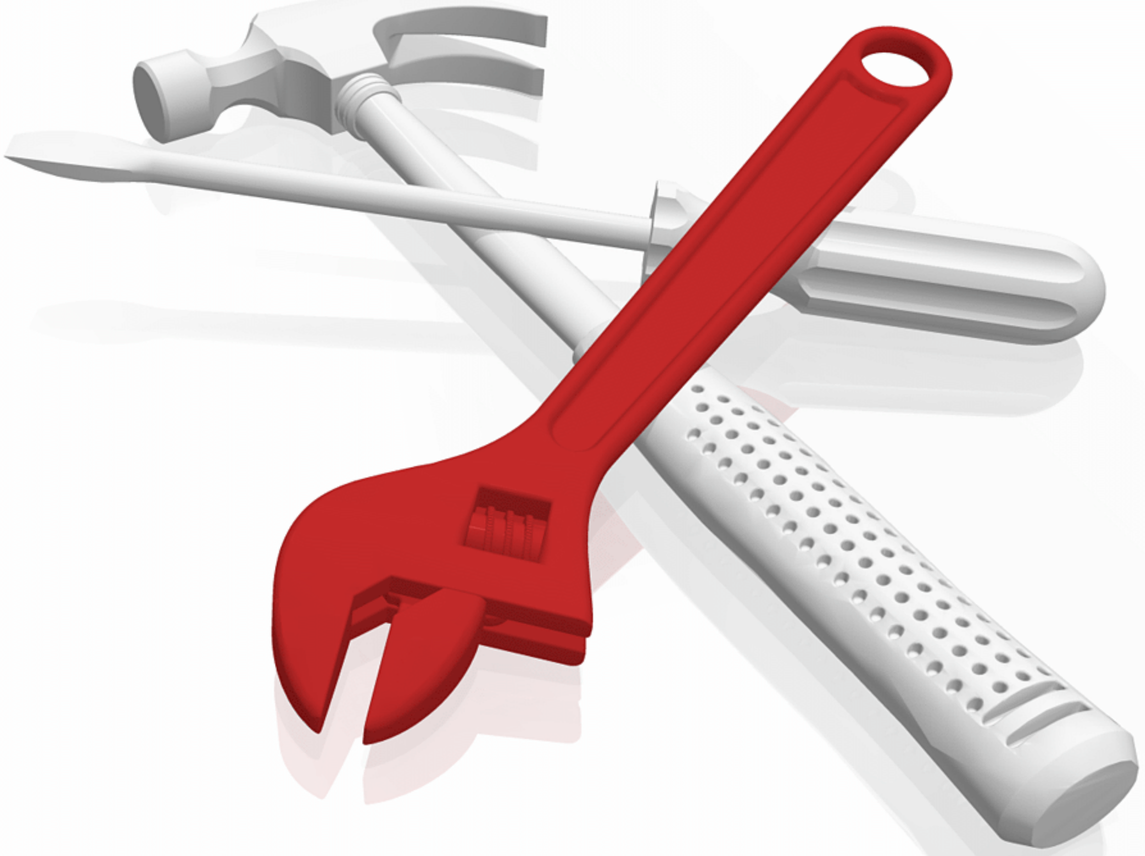
Redbooks

ibm.com/redbooks

Storage Multi-tenancy for Red Hat OpenShift Platform with IBM Storage

IBM Garage Technical Enablement Series

Gauthier Siri



Storage

Redpaper

IBM Redbooks

Storage Multi-tenancy for Red Hat OpenShift Platform with IBM Storage

June 2021

REDP-5638-00

First Edition (June 2021)

This edition applies to IBM Storwize V7000 Gen2 Version 8.3.1, IBM Block Storage CSI Driver Version 1.2, and Red Hat OpenShift Container Platform Version 4.4

This document was created or updated on June 18, 2021.

Contents

Notices v
Trademarks vi

Preface vii
Author. vii
Now you can become a published author, too! vii
Comments welcome. viii
Stay connected to IBM Redbooks viii

Chapter 1. Introduction. 1
1.1 Infrastructure 3
1.2 Required product levels 4
1.3 Technical limits of the solution 5

Chapter 2. Limiting storage resources with IBM Spectrum Virtualize 7
2.1 Creating child pools. 8
  2.1.1 Defining throttling for the child pools. 9
2.2 Creating an ownership group 10
2.3 Creating a user group 13
2.4 Adding users to the group 14
2.5 Creating storage classes on OCP. 15

Chapter 3. Limiting storage resources with OCP. 17
3.1 Using resource quotas and cluster resource quotas. 18
  3.1.1 **Use cases 18**
3.2 Managing storage resources for projects 19
  3.2.1 Creating a ResourceQuota for a project 19
  3.2.2 Viewing the project quota 20
  3.2.3 Avoiding storage over-allocation on new projects. 21
3.3 Managing storage resources across multiple projects 30

Chapter 4. Use cases 33
4.1 Provisioning storage on a Spectrum Virtualize dedicated to an OCP environment . . . 34
4.2 Provisioning storage on a Spectrum Virtualize shared with other applications. 34
4.3 Sharing storage classes across different customers with usage limits per project 35
4.4 Limiting storage usage per project's owner. 35
4.5 Dedicating a storage class to a specific customer's project 36
4.6 Dedicating a storage class to be shared among all the customer's projects. 38
4.7 Conclusion 39

Appendix A. Snapshot quota 41

Notices

Trademarks

Preface

With IBM® Spectrum Virtualize and the Object-Based Access Control, you can implement multi-tenancy and secure storage usage in a Red Hat OpenShift environment.

This IBM Redpaper® publication shows you how to secure the storage usage from the Openshift user to the IBM Spectrum® Virtualize array. You see how to restrict storage usage in a Red Hat Openshift Container Platform to avoid the over-consumption of storage by one or more user. These uses cases can be expanded to the use of this control to provide assistance with billing.

Author

This paper was produced by a specialist working at the IBM Garage™ for Systems in Montpellier, France.

Gauthier Siri is a Storage IT Specialist from IBM France, based in the IBM Systems Center Montpellier. Beginning his career in storage 13 years ago, Gaithier started as an administrator and is now part of a WW pre-sales team, whose goal is to help IBM and Business Partner sales teams demonstrate IBM Storage added value. With a strong block storage background, his natural curiosity led him to new technologies, from automation to containerization, and to support solutions like IBM Spectrum Discover or IBM Storage solution for OpenShift Containerization platform.

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an IBM Redbooks residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, IBM Redbooks
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

Introduction

Through its CSI Drivers, IBM enables the Red Hat OpenShift Container Platform (OCP) to dynamically provision storage on their IBM Storage appliances, block, or file. If not correctly configured, a user can allocate all of the storage that is provided by Red Hat OpenShift Container Platform for their own needs, which leaves other users without the possibility to use the storage.

Another consideration for dynamically provisioning storage is the case when you want to bill different requesters (internal services, external customers, and so on) and limit their usage according to a service level agreement (SLA). For example, a requester wants to provision Flash storage for an application and you allow that user to provision a specific capacity of gold-class storage. Many use cases include a requirement to allow a customer to use, or not use, a storage class within limits.

To accomplish this storage partitioning, the following features are available:

- Spectrum Virtualize Object Base Access Control (OBAC), which allows limiting storage resource that is allocated to the OCP cluster. This feature allows you to control and limit what can be done by the OCP cluster.
- Kubernetes Resource Quotas, which allows the OCP administrator to limit and control OCP user's storage consumption.
- Spectrum Virtualize throttling, which allows you to limit the number of IOPS/Bandwidth on objects. Setting throttling limits on Spectrum Virtualize requires administrator privileges.

Thanks to those features, the storage usage of a containerized environment can be controlled.

Important: The use of OBAC is not mandatory, but it does offer extra security that limits the access of the IBM Block CSI Driver. If OBAC is not used, the IBM Block CSI Driver requires administrator access, and has full access to the storage subsystem. The IBM Block CSI Driver credentials easily can be retrieved by an OCP user with cluster-admin privileges. This configuration might be acceptable if the storage subsystem is dedicated to the OCP cluster; however, if the system hosts other applications and projects, it can be dangerous to provide full access to the OCP administrator.

This chapter includes the following topics:

- 1.1, “Infrastructure” on page 3
- 1.2, “Required product levels” on page 4
- 1.3, “Technical limits of the solution” on page 5

1.1 Infrastructure

To document the multi-tenancy storage use-cases, we built a dedicated infrastructure that consists of the following components:

- A Red Hat OCP cluster version 4.4 where the IBM Block Storage CSI Driver 1.2 was deployed.
- An IBM Spectrum Virtualize appliance (Storwize® V7000 Gen2) version 8.3.1.
- Storage mapping between the Red Hat OCP Cluster and the IBM Spectrum Virtualize is implemented by using iSCSI.

Figure 1-1 shows the infrastructure setup for the use cases.

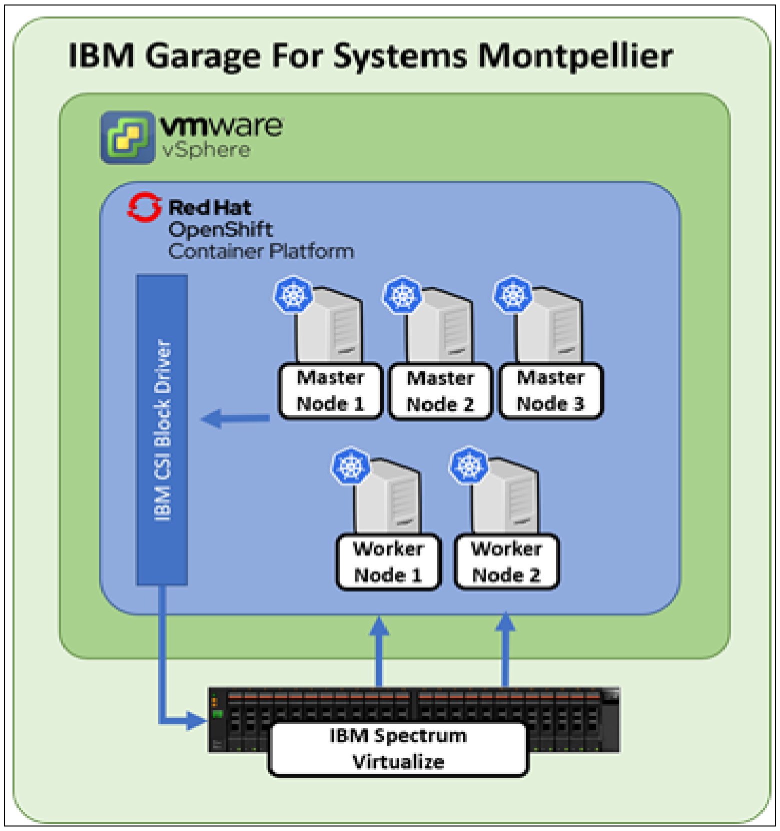


Figure 1-1 Infrastructure overview

In this document, we create different tenants on the IBM Spectrum Virtualize, with their corresponding Kubernetes storage classes. The scope of this document is shown in Figure 1-2.

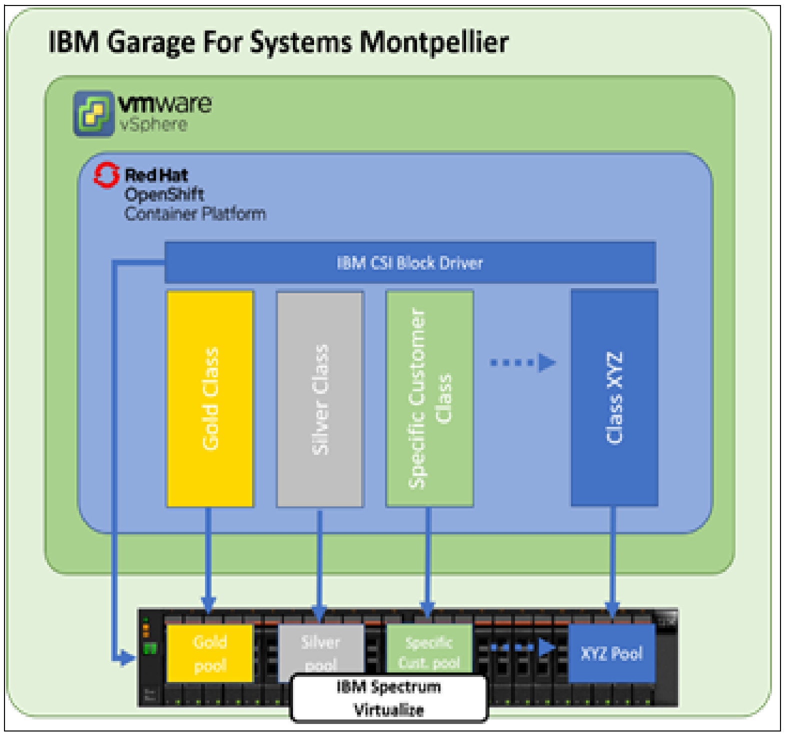


Figure 1-2 Scope of this document

This document is not intended to cover:

- The deployment of Red Hat OpenShift Container Platform or the IBM Block Storage CSI Driver. For more information, see IBM Storage for Red Hat OpenShift Blueprint.
- IBM Spectrum Virtualize installation, features and usage. For more information, see Implementing IBM FlashSystem with IBM Spectrum Virtualize V8.4.

1.2 Required product levels

This solution requires a minimum product level for the following products:

- OBAC is available starting with IBM Spectrum Virtualize 8.3.
- To use Kubernetes volume snapshots, you need:
  - Red Hat OpenShift Container Platform level 4.4 or above
  - IBM Block CSI Driver level 1.2 or above

1.3 Technical limits of the solution

Consider f the following usage limitations:

- As of this writing, a child pool cannot be created in a Data Reduction Pool. This limitation might be lifted in a future code version.
- When OBAC is used:
  - An OCP administrator cannot create a child pool. The Storage Administrator must create and delegate the new child pool.
  - The OCP administrator cannot set throttling limits and must ask the Storage administrator to do it.

2

Limiting storage resources with IBM Spectrum Virtualize

With the Object-Based Access Control (OBAC), you can delegate and limit the usage (creation and deletion) to specific objects. With this feature, you can create administrative groups, with privileges on the following objects:

- Predefined child pools.
- A subset of hosts and host clusters.
- Predefined copy consistency groups (a user also can create their own consistency group).
- Predefined transparent cloud tiering volume groups.

Note: Users controlled by OBAC cannot set up remote replication.

This chapter shows you how to configure the environment. You learn how to restrict the IBM Block CSI Driver user's privileges to avoid impact on the other production systems hosted by this Spectrum Virtualize system.

Setting up Object-Based Access Control typically consists of the following actions:

- Creating:
  - One or more child pools
  - An ownership group
  - A user group associated with the ownership group.
- Adding new or existing users to the user group.

After OBAC is set up, you must configure storage classes and map them to the allocated pools.

This chapter includes the following topics:

- 2.1, "Creating child pools" on page 8
- 2.2, "Creating an ownership group" on page 10
- 2.3, "Creating a user group" on page 13
- 2.4, "Adding users to the group" on page 14
- 2.5, "Creating storage classes on OCP" on page 15

2.1 Creating child pools

Child pools are pool subsets with fully allocated physical capacity. Their capacity must be smaller than the free capacity that is available to the parent pool. Child pools benefit from EasyTier optimization, if it is activated in the parent pool.

Note: As of this writing, child pools cannot be created in Data Reduction Pools. This limitation can be lifted in a future version.

In this OpenShift context, you can imagine different reasons to define many child pools instead of one:

- The Spectrum Virtualize has different pools with different class of storage (Flash drive, SSD drive, spinning drives, and so on), and you want to create different classes of service:
  - A gold class with Flash level performance by using a child pool in a full flash pool
  - A silver class with SSD level performance by using a child pool in a full SSD pool
  - A hybrid class offering dynamic performance optimization by using a child pool in a hybrid Flash/SSD/Spinning drive pool, with EasyTier activated
- You define some dedicated child pools, according to the SLA agreed to for the project/service/customer.

Creating a child pool can be done through the IBM Spectrum Virtualize GUI. Complete the following steps:

1. After you are connected to the GUI as an administrator, select **Pools** → **Pools** (see Figure 2-1).

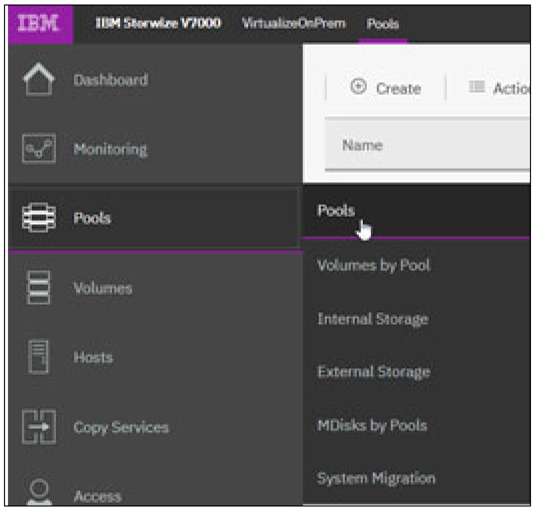
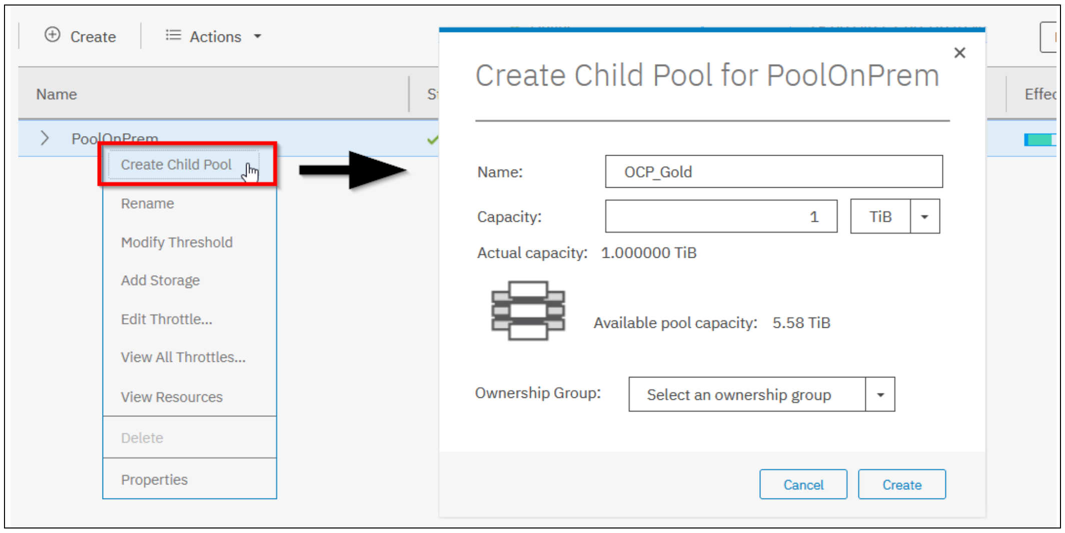


Figure 2-1 Opening the Pools menu

2. From this page, create a child pool by right-clicking a parent pool (top pool) and choose **Create Child Pool.**
3. In the new window, define the name and capacity of the child pool. If the requested ownership group exists, you can directly add this new child pool to the ownership group by choosing the Ownership Group in the combination box.

4. When done, click **Create** (see Figure 2-2).



Create Child Pool for PoolOnPrem

Name: OCP_Gold

Capacity: 1 TiB

Actual capacity: 1.000000 TiB

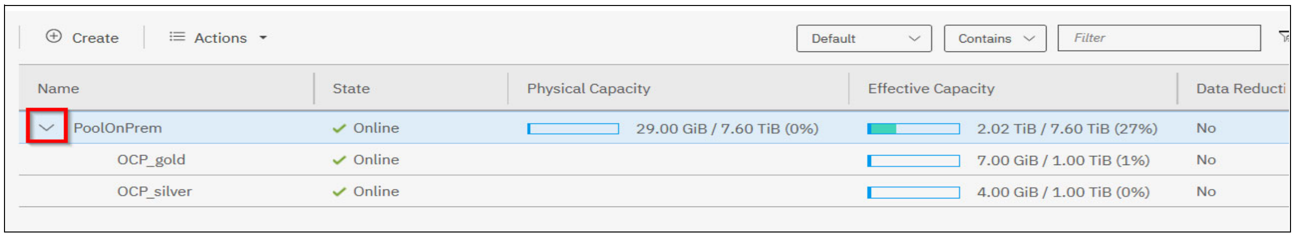
Available pool capacity: 5.58 TiB

Ownership Group: Select an ownership group

Cancel Create

Figure 2-2 Creating a child pool

You can unfold the parent pool to see its child pool and their usage, as shown in Figure 2-3.



Name	State	Physical Capacity	Effective Capacity	Data Reducti
PoolOnPrem	✓ Online	29.00 GiB / 7.60 TiB (0%)	2.02 TiB / 7.60 TiB (27%)	No
OCP_gold	✓ Online		7.00 GiB / 1.00 TiB (1%)	No
OCP_silver	✓ Online		4.00 GiB / 1.00 TiB (0%)	No

Figure 2-3 Pool display

You can create as many child pools as necessary, according to the service class you want to create.

Note: In the demonstration configuration, only one parent pool exists and all of the child pools are created in this parent pool. In a real configuration, you might have different parent pools and create child pools according to your needs.

2.1.1 Defining throttling for the child pools

When defining storage classes, you can define pools and child pools, with the suitable storage drive to match the level of performance you want. You can also refine those performance requirements because of the throttling features embedded in IBM Spectrum Virtualize.

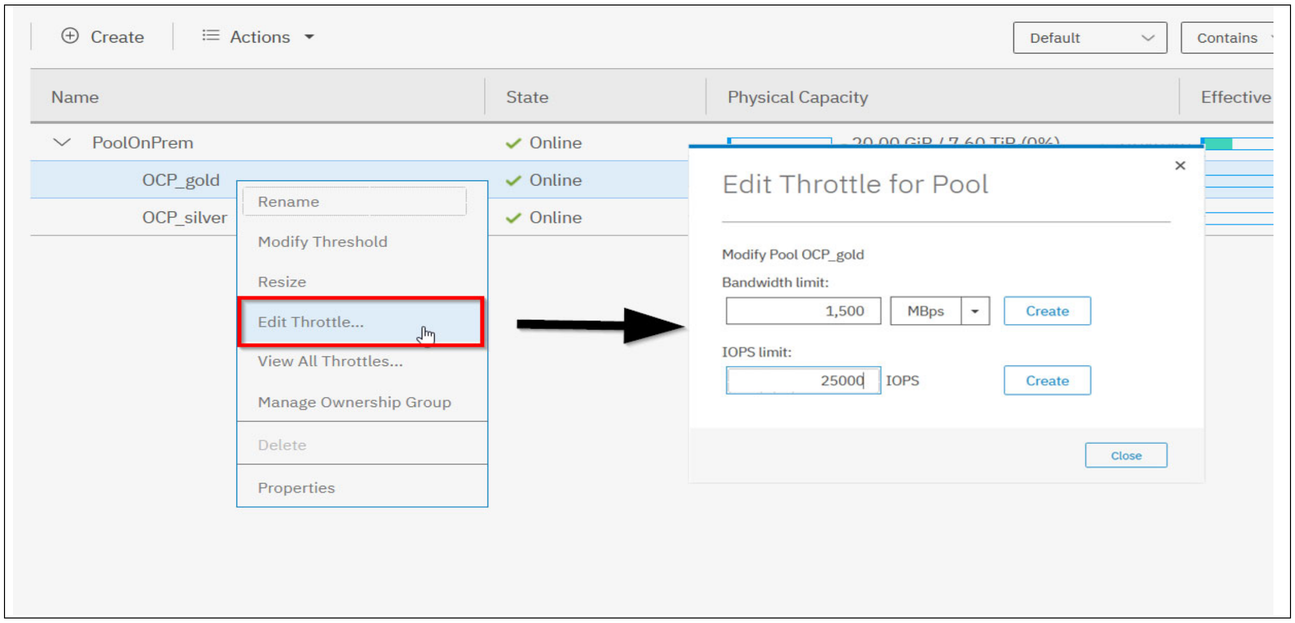
Throttles are a mechanism to control the amount of resources that are used when the system is processing I/Os on supported objects. The system supports throttles on hosts, host clusters, volumes, copy offload operations, and storage pools. If a throttle limit is defined, the system processes the I/O for that object or delays the processing of the I/O to free resources for more critical I/O operations.

Throttles can limit I/O operations per second (IOPS), bandwidth or both.

The throttle limit is a per node limit. For example, if a throttle limit is set to 100 IOPS, each node on the system that can access the volume allows 100 IOPS for that volume. Any I/O operations that exceed the throttle limit are queued at the receiving nodes.

To set the throttle limit, complete the following steps:

1. After you are connected to the GUI as an administrator, click **Pools** → **Pools**.
2. Right-click a pool (or child pool); then, select **Edit Throttle**.
3. In the next window, set the maximum number of IOPS, maximum bandwidth, or both; then, click **Create** for each element to set (see Figure 2-4).



Modify Pool OCP_gold

Bandwidth limit: 1,500 MBps

IOPS limit: 25000 IOPS

Figure 2-4 Editing the throttle pool

If necessary, the storage administrator can also set throttling on the hosts or volumes themselves, independent of the storage classes.

2.2 Creating an ownership group

An ownership group defines a subset of users and objects within the system. You can create ownership groups to further restrict access to specific resources that are defined in the ownership group. Only users with Security Administrator roles can configure and manage ownership groups.

Ownership groups restrict access for users in the ownership group to only those objects that are defined within that ownership group. An owned object can belong to only one ownership group. Users in an ownership group are restricted to viewing and managing objects within their ownership group.

In this OpenShift context example, you create an ownership group that is dedicated to the OCP cluster. The ability allows you to manage the child pools that you created and manage the OCP workers.

The IBM Block CSI driver communicates with Spectrum Virtualize to map and unmap the dynamically provisioned volumes to the OCP worker nodes, when needed.

To create an ownership group, you must be logged as Security Administrator on IBM Spectrum Virtualize. Complete the following steps:

1. From the Spectrum Virtualize GUI, click **Access** → **Ownership Groups** (see Figure 2-5).

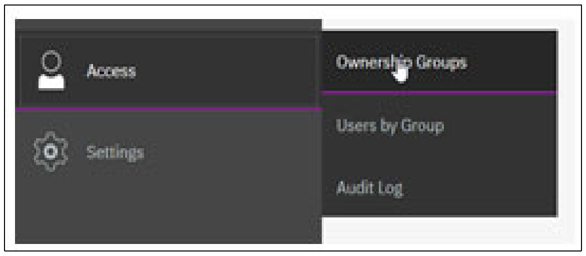


Figure 2-5 Accessing the ownership groups menu

2. Click **Create an Ownership Group** and in the new window, enter the name of the ownership group. Click **Create** (see Figure 2-5).

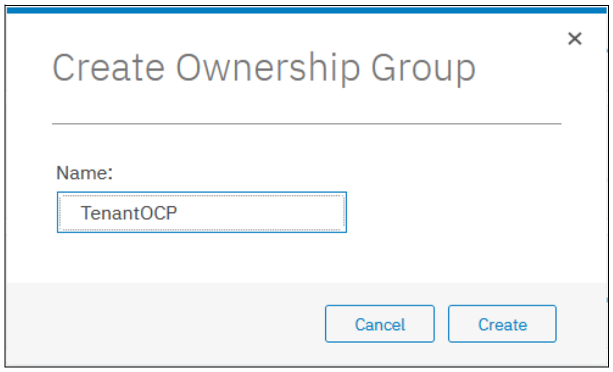
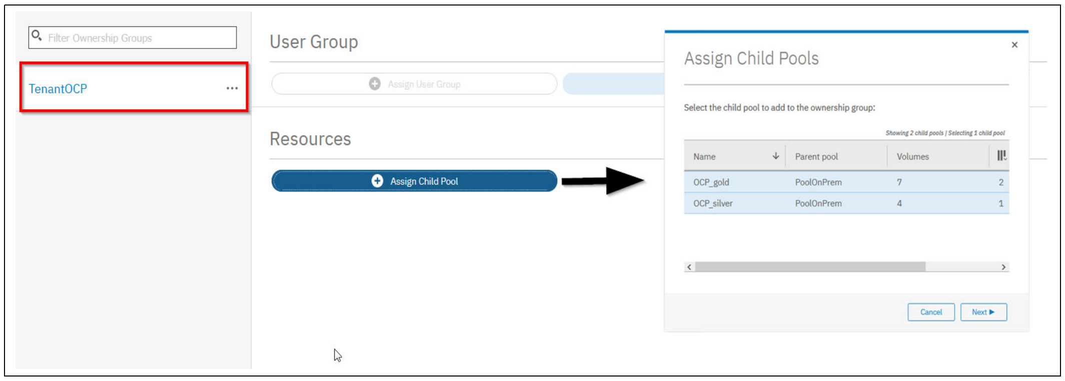


Figure 2-6 Naming the new ownership group

After the group is created, you must add objects to the ownership group.

3. In the left column, select the newly created ownership group; then, click **Assign Child Pool**.
4. In the new window (see Figure 2-7), select the relevant child pools and click **Next**.



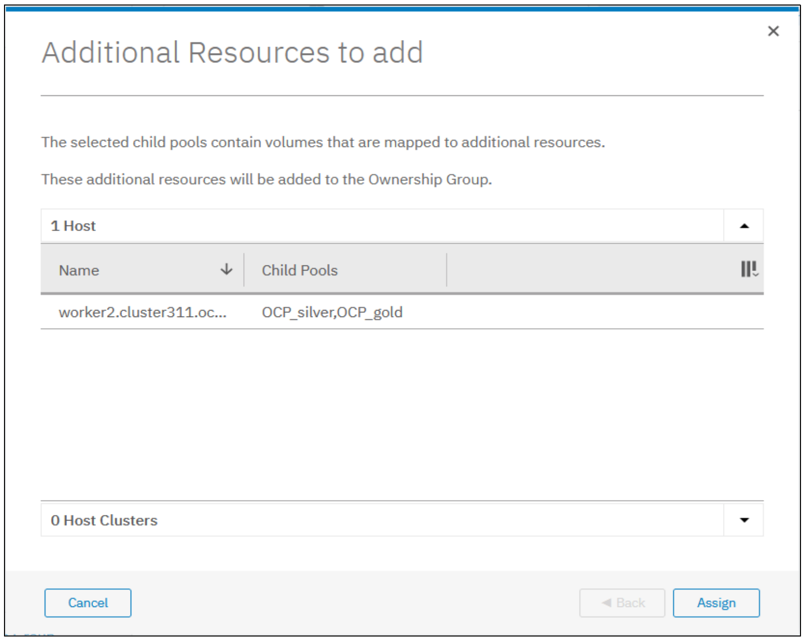
Select the child pool to add to the ownership group:

Showing 2 child pools | Selecting 1 child pool

Name	Parent pool	Volumes	
OCP_gold	PoolOnPrem	7	2
OCP_silver	PoolOnPrem	4	1

Figure 2-7 Selecting the child pools

If the child pools a contain volumes that are mapped to hosts, a message appears that informs you that those hosts also can be added to the ownership group (see Figure 2-8).


Additional Resources to add

The selected child pools contain volumes that are mapped to additional resources.

These additional resources will be added to the Ownership Group.

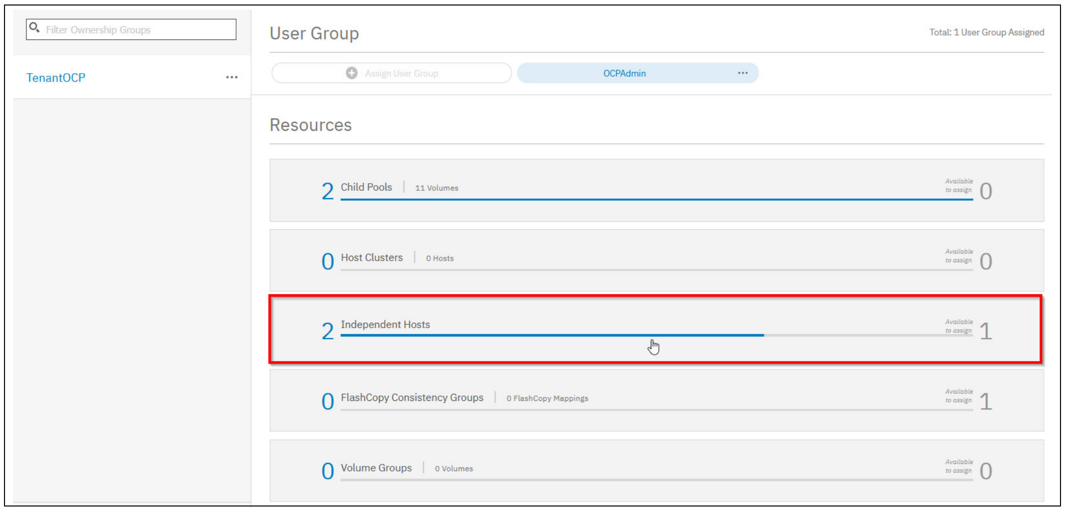
1 Host

Name	Child Pools
worker2.cluster311.oc...	OCP_silver,OCP_gold

0 Host Clusters

Figure 2-8   Additional Resources to add window

5. Click **Assign**.
6. After the child pools are added, you can add objects, such as hosts, to the ownership group. In the ownership group main panel, click **Independent Hosts** (see Figure 2-9).



2	Child Pools	11 Volumes	Available to assign 0
0	Host Clusters	0 Hosts	Available to assign 0
2	Independent Hosts		Available to assign 1
0	FlashCopy Consistency Groups	0 FlashCopy Mappings	Available to assign 1
0	Volume Groups	0 Volumes	Available to assign 0

Figure 2-9   Adding hosts to the ownership group

7. Click **Assign**. Then, select the hosts to add to the ownership group. Choose the host to assign and click **Assign** (see Figure 2-10).

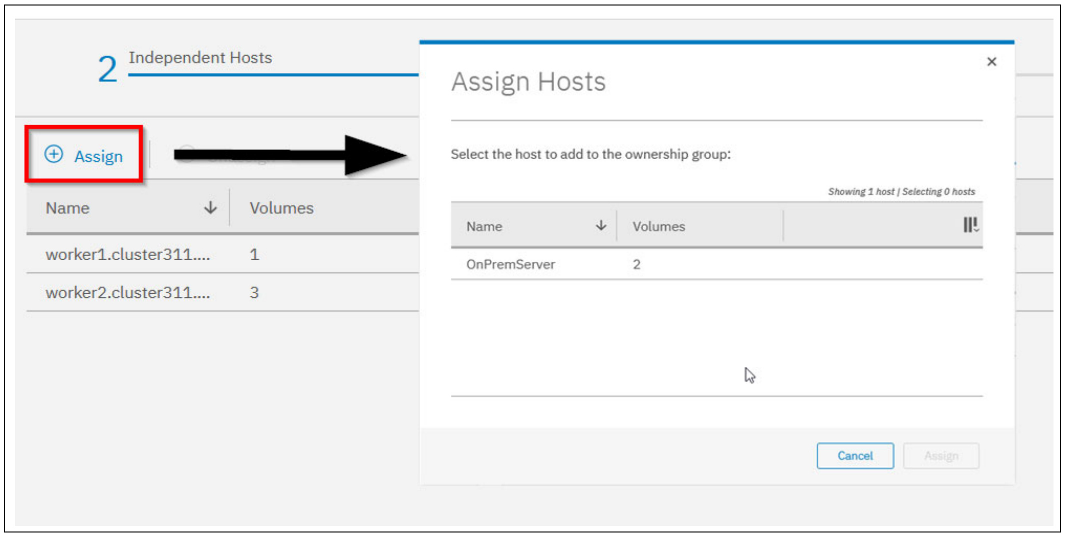


Figure 2-10 Assigning a host to the ownership group

You can add as many objects as necessary to the ownership group and add or resize child pools as the storage needs grow.

2.3 Creating a user group

In IBM Spectrum Virtualize, a user's role is defined by groups. Preconfigured groups are available, such as SecurityAdmin and Administrator, that are global, which giving access to all objects but with different privileges.

You can also create custom groups with a specific role that is attached to a specific ownership group. All the users that belong to this user group have the relevant role in the linked ownership group.

Consider the following rules:

- A user can belong to only one user group.
- A custom user group cannot use the Security Administrator role.
- To create a user group, you must be logged as Security Administrator on IBM Spectrum Virtualize.

To create a user group, complete the following steps:

1. From the Spectrum Virtualize GUI, click **Access** → **Users by Groups** (see Figure 2-11).

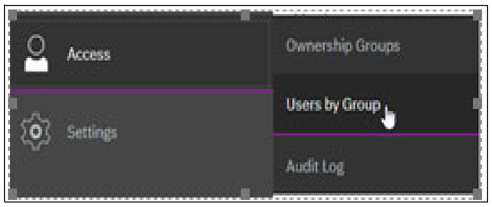
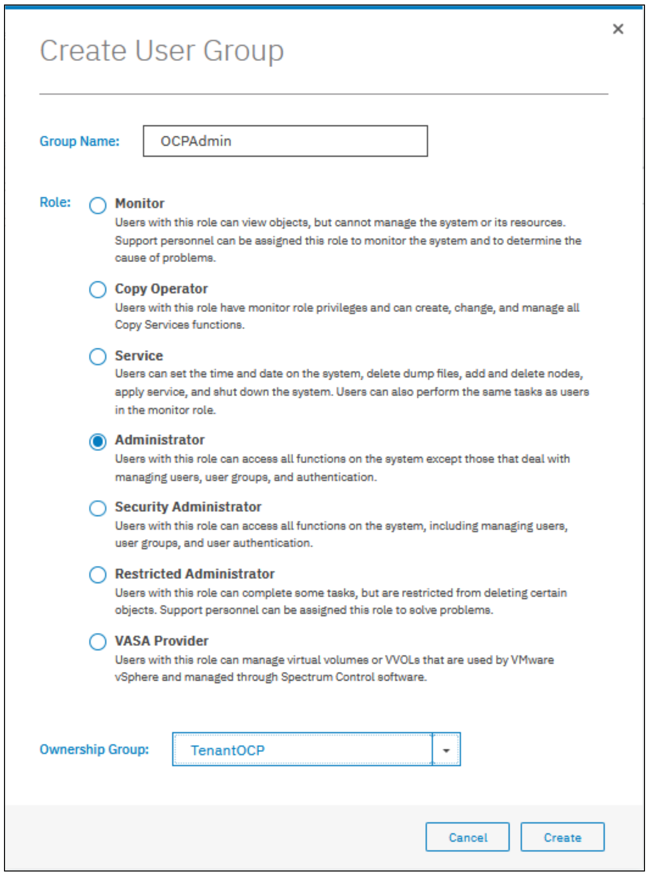


Figure 2-11 Accessing the user by groups menu

2. Click **Create User Group**.
3. In the new window, enter a name for the user group and the Administrator role. If necessary, scroll down the windows and select the relevant, newly created, ownership group. Click **Create** (see Figure 2-12).


Create User Group

Group Name: OCPAdmin

Role:
- ( ) **Monitor**
  Users with this role can view objects, but cannot manage the system or its resources. Support personnel can be assigned this role to monitor the system and to determine the cause of problems.
- ( ) **Copy Operator**
  Users with this role have monitor role privileges and can create, change, and manage all Copy Services functions.
- ( ) **Service**
  Users can set the time and date on the system, delete dump files, add and delete nodes, apply service, and shut down the system. Users can also perform the same tasks as users in the monitor role.
- [✓] **Administrator**
  Users with this role can access all functions on the system except those that deal with managing users, user groups, and authentication.
- ( ) **Security Administrator**
  Users with this role can access all functions on the system, including managing users, user groups, and user authentication.
- ( ) **Restricted Administrator**
  Users with this role can complete some tasks, but are restricted from deleting certain objects. Support personnel can be assigned this role to solve problems.
- ( ) **VASA Provider**
  Users with this role can manage virtual volumes or VVOLs that are used by VMware vSphere and managed through Spectrum Control software.

Ownership Group: TenantOCP

Figure 2-12 Naming the new user group

In this example, all of the users that belong to this group are an administrator of the ownership group TenantOCP. The user that is used by the IBM Block CSI Driver (which is defined during its configuration) must belong to this group.

2.4 Adding users to the group

In this OCP context, the user that is used by the IBM Block CSI Driver to provision and use storage must belong to the user group that was created. Doing so allows you to limit the scope of the OCP Administrator on the IBM Spectrum Virtualize system, which avoids the potential for OCP starving other non-OCP applications.

In this demonstration, the user that is used by the IBM Block CSI Driver is named `ocpadmin`. The `ocpadmin` user's privileges are Administrator, which gives it the minimum required authority to create volumes, map them, perform FlashCopies, and other necessary tasks.

Remember that to create or add a user, you must be logged as a Security Administrator on the IBM Spectrum Virtualize system.

To add `ocpadmin` to the user OCPAdmin group that was created, complete the following steps:

1. From the Spectrum Virtualize GUI, click **Access** → **Users by Groups** and choose **All Users** (see Figure 2-13).

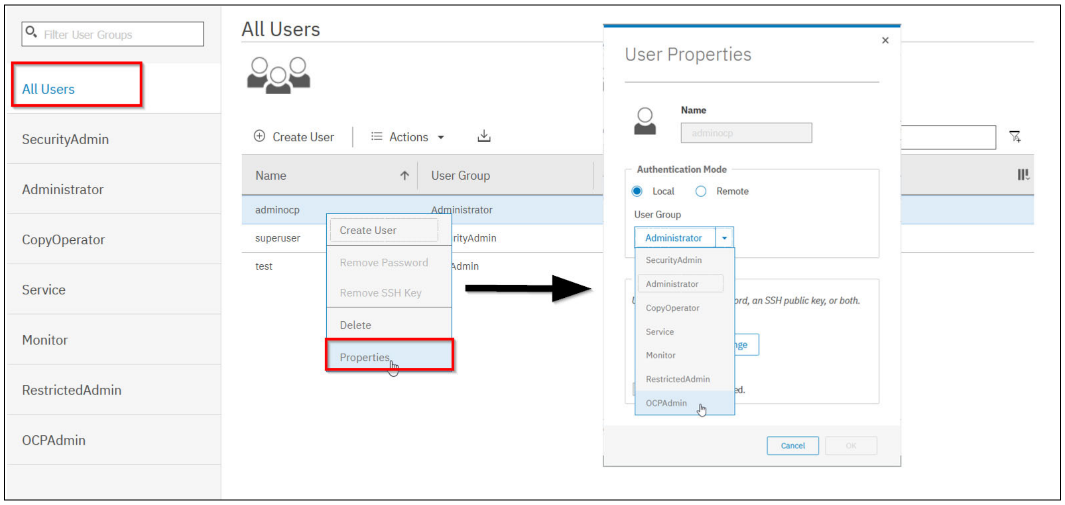


Figure 2-13 Changing the group for the IBM Block CSI Driver user

2. Find the user that is used by the IBM Block CSI Driver, right-click it, and choose **Properties**.
3. In the next window, change its group from Administrator to OCPAdmin; then, click **OK**.

The IBM Block CSI Driver user has now limited privileges and can create volumes in the delegated child pools only and attach them to only the delegated hosts.

2.5 Creating storage classes on OCP

In this example, you modify the IBM Block CSI Driver user's privileges, which limits its access to a specific child pool. You must reflect those changes on OCP and create Storage classes that match those new child pools.

In this demonstration infrastructure, you create two child pools: OCP_gold and OCP_silver. You create the matching storage classes on the OCP cluster.

Complete the following steps:

1. Log on to the OCP cluster with privileges to create storage classes (for example, cluster-admin privileges).
2. Define YAML files describing your storage classes by using the IBM Block CSI Driver and targeting the suitable child pool, as shown in Example 2-1.

Example 2-1 YAML

```
$ cat sc-gold.yml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: v7k-gold
  namespace: ibm-block-csi
provisioner: block.csi.ibm.com
```

```
parameters:
  pool: OCP_gold
  csi.storage.k8s.io/provisioner-secret-name: v7k-secret
  csi.storage.k8s.io/provisioner-secret-namespace: ibm-block-csi
  csi.storage.k8s.io/controller-publish-secret-name: v7k-secret
  csi.storage.k8s.io/controller-publish-secret-namespace: ibm-block-csi
  csi.storage.k8s.io/fstype: ext4
$
$ cat sc-silver.yml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: v7k-silver
  namespace: ibm-block-csi
provisioner: block.csi.ibm.com
parameters:
  pool: OCP_silver
  SpaceEfficiency: thin
  csi.storage.k8s.io/provisioner-secret-name: v7k-secret
  csi.storage.k8s.io/provisioner-secret-namespace: ibm-block-csi
  csi.storage.k8s.io/controller-publish-secret-name: v7k-secret
  csi.storage.k8s.io/controller-publish-secret-namespace: ibm-block-csi
  csi.storage.k8s.io/fstype: ext4
$
f example. (Command)(10pt.)
```

Each YAML file points on one of the child pools. You can also see that the v7k-silver storage class has a SpaceEfficiency parameter set to thin; all volumes that are created by using this storage class are thin provisioned.

For more information about creating storage classes by using the IBM Block CSI Driver, see *IBM Storage for Red Hat OpenShift Blueprint*, REDP-5565.

3

Limiting storage resources with OCP

You now defined control and limitations for the entire OCP cluster. In this chapter, you set up rules to control and limit the OCP users' storage consumption.

This chapter includes the following topics:

- 3.1, “Using resource quotas and cluster resource quotas” on page 18
- 3.2, “Managing storage resources for projects” on page 19
- 3.3, “Managing storage resources across multiple projects” on page 30

3.1 Using resource quotas and cluster resource quotas

By using resource quotas and cluster resource quotas, you can control the allocation of different kinds of resources in the Red Hat OCP Cluster, such as CPU, memory, and the number of pods. With resource quotas, the resource control occurs at the project level. With cluster resource quotas, the resource control occurs across different projects.

By using quotas, you can also control the storage capacity that can be used or the number of volumes that can be used. You can control the usage at a global level or at the storage class level.

By using resource quotas, you can control the following storage elements:

- `requests.storage`

  The sum of storage requests across all persistent volume claims in any state cannot exceed this value.
- `persistentvolumeclaims`

  The total number of persistent volume claims that can exist in the project.
- `<storage-class-name>.storageclass.storage.k8s.io/requests.storage`

  The sum of storage requests across all persistent volume claims in any state that include a matching storage class cannot exceed this value.
- `<storage-class-name>.storageclass.storage.k8s.io/persistentvolumeclaims`

  The total number of persistent volume claims with a matching storage class that can exist in the project.

The global values (`requests.storage` and `persistentvolumeclaims`) prevails over storage class-specific values.

Note: You might always want to create a global quota for each project. Otherwise, any unspecified storage class are uncontrolled and allow unlimited storage provisioning (see "**Case 2**" on page 19).

Consider a Red Hat OCP Cluster with four storage classes:

- `v7k-gold`
- `v7k-silver`
- `v7k-bronze`
- `lower-perf`

3.1.1 Use cases

In this section, we discuss two use cases.

Case 1

In Example 3-1, the Resource Quotas rules that are created are shown.

Example 3-1 Resource Quota rules

```
persistentvolumeclaims: "10"
requests.storage: "7Gi"
v7k-gold.storageclass.storage.k8s.io/requests.storage: "4Gi"
v7k-gold.storageclass.storage.k8s.io/persistentvolumeclaims: "5"
```

```
v7k-silver.storageclass.storage.k8s.io/requests.storage: "5Gi"
v7k-bronze.storageclass.storage.k8s.io/requests.storage: "0"
v7k-bronze.storageclass.storage.k8s.io/persistentvolumeclaims: "0"
```

These rules set the following allowances and limitations:

- The `v7k-bronze` storage class cannot be used to:
  - Request storage (value is explicitly 0)
  - Create claims (value is explicitly 0)
- 4Gi can be requested by using the `v7k-gold` class. 5Gi can be requested by using the `v7k-silver` class. The sum of requested storage cannot exceed the global value (7Gi).
- Five volumes can be claimed by using the `v7k-gold` class. No limit on the number of claimed volumes is set for the `v7k-silver` class. The sum of claimed volumes cannot exceed the global value (10).
- No limit of any kind is set for the storage class `lower-perf`; therefore, it can be used by using the global quotas.

Case 2

In this example, the following Resource Quota rules are created:

```
v7k-gold.storageclass.storage.k8s.io/requests.storage: "0"
v7k-gold.storageclass.storage.k8s.io/persistentvolumeclaims: "0"
```

These rules include the following specifications:

- v7k-gold storage class cannot be used to:
  - Request storage (value is explicitly 0)
  - Create claims (value is explicitly 0)
- No limit of any kind is set for the remaining storage class (global or specific); therefore, it can be used without limitation.

Important: If two quotas control the same resource for a specific project, the lower quota prevails. This rule also applies when a resource quota and a cluster resource quota control the same resource (the lower quota prevails).

3.2 Managing storage resources for projects

The next set of steps are used to manage the storage resources to ensure they are allocated correctly and are available to the intended projects.

3.2.1 Creating a ResourceQuota for a project

A yaml file is created that contains the definition for a resource quota. Is it then applied to the project `my-project`. The file and commands are shown in Example 3-2.

Example 3-2 Yaml file with resource quota definition applied to a project

```
$ cat storage-consumption.yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: storage-consumption
spec:
  hard:
    persistentvolumeclaims: "10"
    requests.storage: "7Gi"
    v7k-gold.storageclass.storage.k8s.io/requests.storage: "3Gi"
    v7k-gold.storageclass.storage.k8s.io/persistentvolumeclaims: "5"
    v7k-silver.storageclass.storage.k8s.io/requests.storage: "5Gi"
    v7k-silver.storageclass.storage.k8s.io/persistentvolumeclaims: "10"
$
$ oc apply -f storage-consumption.yaml -n my-project
$ oc get resourcequotas -n my-project
NAMESPACE     NAME                               CREATED AT
My-project    storage-consumption                2020-08-20T13:11:54Z
$
```

In this case, the `v7k-gold` and `v7k-silver` storage classes are controlled by the global quotas and their specifics quotas. All other storage classes are controlled by the global quotas only.

If one of the quotas is reached while volumes are created and storage is used, the error that is shown in Example 3-3 is raised.

Example 3-3 Error thrown when a quota is reached

```
$ oc apply -f pvc-mypvc-gold3-3gb.yml
Error from server (Forbidden): error when creating "pvc-mypvc-gold3-3gb.yml":
persistentvolumeclaims "mypvc3" is forbidden: exceeded quota:
storage-consumption-test-quota2, requested:
v7k-gold.storageclass.storage.k8s.io/requests.storage=3Gi, used:
v7k-gold.storageclass.storage.k8s.io/requests.storage=2Gi, limited:
v7k-gold.storageclass.storage.k8s.io/requests.storage=3Gi
$
```

3.2.2 Viewing the project quota

As things progress, you might want to monitor the project storage consumption. To do so, you can review the project or the resource quota, as shown in Example 3-4.

Example 3-4 Displaying the project and resource quotas

```
$ oc describe projects.project.openshift.io my-project
Name:           my-project
Created:        2 hours ago
Labels:         <none>
Annotations:    openshift.io/description=
                openshift.io/display-name=
                openshift.io/requester=test
                openshift.io/sa.scc.mcs=s0:c27,c24
                openshift.io/sa.scc.supplemental-groups=1000750000/10000
                openshift.io/sa.scc.uid-range=1000750000/10000
Display Name:   <none>
Description:    <none>
Status:         Active
Node Selector:  <none>
Quota:
    Name:
storage-consumption
    Resource                                                        Used    Hard
    --------                                                        ----    ----
    persistentvolumeclaims                                          3       10
    requests.storage                                                3Gi     7Gi
    v7k-gold.storageclass.storage.k8s.io/persistentvolumeclaims     2       5
    v7k-gold.storageclass.storage.k8s.io/requests.storage           2Gi     3Gi
    v7k-silver.storageclass.storage.k8s.io/persistentvolumeclaims   1       10
    v7k-silver.storageclass.storage.k8s.io/requests.storage         1Gi     5Gi
Resource limits:       <none>
$
$ oc describe resourcequotas storage-consumption -n my-project
Name:                                                           storage-consumption
Namespace:                                                      my-project
Resource                                                        Used  Hard
--------                                                        ----  ----
persistentvolumeclaims                                          3     10
requests.storage                                                3Gi   7Gi
v7k-gold.storageclass.storage.k8s.io/persistentvolumeclaims     2     5
v7k-gold.storageclass.storage.k8s.io/requests.storage           2Gi   3Gi
v7k-silver.storageclass.storage.k8s.io/persistentvolumeclaims   1     10
v7k-silver.storageclass.storage.k8s.io/requests.storage         1Gi   5Gi
$
```

3.2.3 Avoiding storage over-allocation on new projects

When a project is created, no quota is associated to this new project by default. Therefore, the project users can allocate as much storage as they want. By default, every logged in user can create a project. Combining these two behaviors means that by default, any logged in user can create project without quotas, and use as much storage as they want.

The following features can help you to avoid this issue:

- Remove the default privilege to create a project. In this case, a project can be created only by:
  - A Red Hat administrator who creates a project and its resource quota for the requester.
  - Users that were specifically entitled to create projects.
- Define a default quota that is applied to all new projects by using a Project Template. When a user creates a project, a resource quota is also created. This resource quota can then be modified by a Red Hat OCP administrator according to the SLA that is agreed to by the requester.

Removing the privilege to create projects

In Red Hat OpenShift Container Platform, any logged user can create a project by default. To disable this privilege, you must log in with a cluster-admin user.

Check the suitable cluster role binding that links users and groups to a privilege, as shown in Example 3-5.

Example 3-5 review the cluster role binding

```
$ oc describe clusterrolebinding.rbac self-provisioners
Name:         self-provisioners
```

```
Labels:       <none>
Annotations:  rbac.authorization.kubernetes.io/autoupdate: true
Role:
  Kind:  ClusterRole
  Name:  self-provisioner
Subjects:
  Kind   Name                        Namespace
  ----   ----                        ---------
  Group  system:authenticated:oauth
$
```

We now need to remove the self-provisioner cluster role from the group `system:authenticated:oauth`.

There are two cases:

- If the self-provisioners cluster role binding only binds the self-provisioner cluster role to the system:authenticated:oauth group, run the following command:

  ```
  $ oc patch clusterrolebinding.rbac self-provisioners -p '{"subjects": null}'
  clusterrolebinding.rbac.authorization.k8s.io/self-provisioners patched
  $
  ```

  Do not use the process that is described next to remove privileges in this case because it removes the role binding.

- If the self-provisioners cluster role binding binds the self-provisioner cluster role to users, group, and service accounts other than the `system:authenticated:oauth` group, run the following command:

  ```
  $ oc adm policy \
      remove-cluster-role-from-group self-provisioner \
      system:authenticated:oauth

  Warning: Your changes may get lost whenever a master is restarted, unless you
  prevent reconciliation of this rolebinding using the following command: oc
  annotate clusterrolebinding.rbac self-provisioners
  'rbac.authorization.kubernetes.io/autoupdate=false'
  --overwriteclusterrole.rbac.authorization.k8s.io/self-provisioner removed:
  "system:authenticated:oauth"
  $
  ```

  If necessary, you can revert this operation with the following command:

  ```
  oc adm policy add-cluster-role-to-group self-provisioner
  system:authenticated:oauth --rolebinding-name='self-provisioners'
  ```

To complete the operation, you must prevent automatic updates to the role; therefore, revert to the default configuration. Run the following command to disable automatic updates:

```
$ oc patch clusterrolebinding.rbac self-provisioners -p '{ "metadata": {
"annotations": { "rbac.authorization.kubernetes.io/autoupdate": "false" } } }'
$
```

From now on, a standard user cannot create a project unless they are specifically given the privilege to do it. The following command shows the error a user sees in this case:

```
$ oc new-project my-project
Error from server (Forbidden): You may not request a new project via this API.
$
```

Entitling a specific user to create a project

After the project creation privilege is removed, only a cluster-admin user can create a project. A user `my-user-name` can be given the privilege to create a project (self-provisioner) by using the following command:

```
$ oc adm policy add-cluster-role-to-user self-provisioner my-user-name
--rolebinding-name='self-provisioners'
```

After this command is run, the entitled user can create a project, but no quota is applied on this new project. For this reason, consider creating a custom project template to enforce quotas on new projects, as described in “Creating a custom project template enforcing resource quotas” on page 24.

Creating a project with resource quota for a user

After the privilege to create a project is removed from a user, you might want to keep absolute control over project creation. In this case, the OCP administrators must create them on demand. After the project is created, the OCP administrator must give the project administrator privileges to the user and eventually change the requester of the project to match the reality.

This process can be done by using the following commands for project `my-project-name` and user `my-user`:

```
$ oc new-project my-project-name
Now using project "my-project-name" on server
"https://api.cluster311.ocpstorage.icc:6443".
….
to build a new example application in Python. Or use kubectl to deploy a simple
Kubernetes application:
    kubectl create deployment hello-node
--image=gcr.io/hello-minikube-zero-install/hello-node
$ oc adm policy add-role-to-user admin my-user -n my-project-name
--rolebinding-name='admin'
clusterrole.rbac.authorization.k8s.io/admin added: "my-user"
$ oc patch namespaces my-project-name -p
'{"metadata":{"annotations":{"openshift.io/requester": "my-user"}}}'
namespace/my-project-name patched
$
```

If the name space `openshift.io/requester` annotation is correctly updated, any predefined cluster resource quota that is selecting this project annotation applies (see 3.3, “Managing storage resources across multiple projects” on page 30).

Finally, you must create a Resource Quota for this project. Consider the Resource Quota that is defined in 3.2.1, “Creating a ResourceQuota for a project” on page 19“. We can apply it to the newly created project by using the following command:

```
$ oc apply -f storage-consumption.yaml -n my-project
$ oc describe resourcequotas -n my-project-name

Name:                                                           storage-consumption
Namespace:                                                      my-project-name
Resource                                                        Used  Hard
--------                                                        ----  ----
persistentvolumeclaims                                          0     10
requests.storage                                                0     7Gi
v7k-gold.storageclass.storage.k8s.io/persistentvolumeclaims    0     5
```

```
v7k-gold.storageclass.storage.k8s.io/requests.storage         0     3Gi
v7k-silver.storageclass.storage.k8s.io/persistentvolumeclaims  0     10
v7k-silver.storageclass.storage.k8s.io/requests.storage       0     5Gi
$
```

Creating a custom project template enforcing resource quotas

By default, no quota is applied when a project is created. Therefore, the user can provision as much storage as they want. To avoid this situation, you can create a project template to be applied for each new project. A project template applies to all clusters and all users.

In this context, you can create a project template that create a resource quota when each project is created. In this resource quota, you define common default rules that apply to all projects. For example, you might want to forbid any kind of storage provisioning or allow 10 Gi of the lowest tier for testing.

Complete the following steps:

1. As a user with cluster-admin privileges, generate a project template:

   ```
   $ oc adm create-bootstrap-project-template -o yaml > template.yaml
   ```

2. Edit the yaml template file to add the default resource quota to the project template, as shown in Example 3-6.

Example 3-6 Use a YAML file to add a default resource quota

```
$ vi template.yaml
apiVersion: template.openshift.io/v1
kind: Template
metadata:
  creationTimestamp: null
  name: project-request
objects:
- apiVersion: project.openshift.io/v1
  kind: Project
  metadata:
    annotations:
      openshift.io/description: ${PROJECT_DESCRIPTION}
      openshift.io/display-name: ${PROJECT_DISPLAYNAME}
      openshift.io/requester: ${PROJECT_REQUESTING_USER}
    creationTimestamp: null
    name: ${PROJECT_NAME}
  spec: {}
  status: {}
- apiVersion: rbac.authorization.k8s.io/v1
  kind: RoleBinding
  metadata:
    creationTimestamp: null
    name: admin
    namespace: ${PROJECT_NAME}
  roleRef:
    apiGroup: rbac.authorization.k8s.io
    kind: ClusterRole
    name: admin
  subjects:
  - apiGroup: rbac.authorization.k8s.io
    kind: User
    name: ${PROJECT_ADMIN_USER}
```

```
- apiVersion: v1
  kind: ResourceQuota
  metadata:
    name: storage-consumption-${PROJECT_NAME}
  spec:
    hard:
      persistentvolumeclaims: "5"
      requests.storage: 10Gi
      v7k-silver.storageclass.storage.k8s.io/persistentvolumeclaims: "5"
      v7k-silver.storageclass.storage.k8s.io/requests.storage: 5Gi
      v7k-gold.storageclass.storage.k8s.io/persistentvolumeclaims: "0"
      v7k-gold.storageclass.storage.k8s.io/requests.storage: "0"
parameters:
- name: PROJECT_NAME
- name: PROJECT_DISPLAYNAME
- name: PROJECT_DESCRIPTION
- name: PROJECT_ADMIN_USER
- name: PROJECT_REQUESTING_USER
```

In this project template (`project-request`), you added the creation of a ResourceQuota object along with the default objects. This quota is named `storage-consumption-${PROJECT_NAME}` (where PROJECT_NAME is a variable, the value of which is provided by the user during the project creation). It allows the users to use 10 Gi of storage, including 5 Gi of silver storage. However, the users cannot use the gold storage.

Note: To limit the users to a specific storage class, you must specifically forbid or exclude all other storage classes by setting the following quota properties equal to zero:

`<storage-class-name>.storageclass.storage.k8s.io/requests.storage` and
`<storage-class-name>.storageclass.storage.k8s.io/persistentvolumeclaims`

Otherwise, users can still use them within the limits fixed by the global limits (or without limit if no global limits are set).

3. After the project template is edited, create it in the `openshift-config` namespace:

```
$ oc apply -f template.yaml -n openshift-config
template.template.openshift.io/project-request created
$
```

4. Specify the cluster that uses this template when creating \ projects by editing the object `project.config.openshift.io/cluster`. This step can be done by using the CLI or the GUI:

   – From the CLI, with the following command:

```
$ oc edit project.config.openshift.io/cluster
```

– From the GUI by clicking **Administration** → **Cluster Settings** in the Global Configuration tab and clicking **Edit Project Resource** of the object project (Figure 3-1).

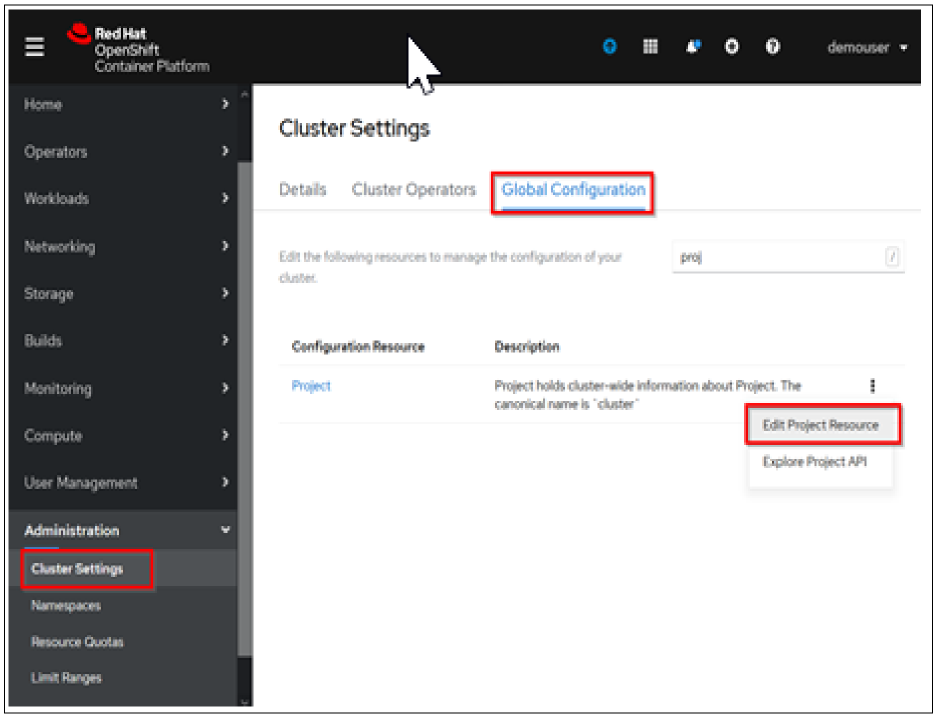


Figure 3-1 Specify the cluster that will use this template

5. Update the spec section of the `project.config.openshift.io/cluster` object to include the `projectRequestTemplate` and name parameters. Also, set the name of your uploaded project template. The default name is `project-request`, but you can choose another name for the project template.

   After it is modified, the `project.config.openshift.io cluster` object should look like Example 3-7

Example 3-7 The projects.config.openshift.io cluster object

```
$ oc get projects.config.openshift.io cluster -o yaml
apiVersion: config.openshift.io/v1
kind: Project
metadata:
  annotations:
    release.openshift.io/create-only: "true"
  creationTimestamp: "2020-03-27T13:31:25Z"
  generation: 2
  name: cluster
  resourceVersion: "60254762"
  selfLink: /apis/config.openshift.io/v1/projects/cluster
  uid: 46357d79-44f8-4d82-b7dd-404cb996abd3
spec:
  projectRequestTemplate:
    name: project-request
$
```

Now when a user creates a project, the resource quota is automatically created and enforced, as shown in Example 3-8.

Example 3-8 Enforcement of the resource quota

```
$ oc new-project the-new-project
Now using project "the-new-project" on server
"https://api.cluster311.ocpstorage.icc:6443".
You can add applications to this project with the 'new-app' command. For example,
try:
    oc new-app django-psql-example
to build a new example application in Python. Or use kubectl to deploy a simple
Kubernetes application:
    kubectl create deployment hello-node
--image=gcr.io/hello-minikube-zero-install/hello-node
$ oc get resourcequota
NAME                                  CREATED AT
storage-consumption-the-new-project   2020-08-24T12:20:02Z
$ oc describe projects the-new-project
Name:           the-new-project
Created:        3 minutes ago
Labels:         <none>
Annotations:    openshift.io/description=
                openshift.io/display-name=
                openshift.io/requester=test
                openshift.io/sa.scc.mcs=s0:c30,c0
                openshift.io/sa.scc.supplemental-groups=1000870000/10000
                openshift.io/sa.scc.uid-range=1000870000/10000
Display Name:   <none>
Description:    <none>
Status:         Active
Node Selector:  <none>
Quota:
           Name:
storage-consumption-the-new-project
                   Resource
Used    Hard
                   --------
----    ----
                   persistentvolumeclaims
0       5
                   requests.storage
0       10Gi
                  v7k-gold.storageclass.storage.k8s.io/persistentvolumeclaims
0       0
                  v7k-gold.storageclass.storage.k8s.io/requests.storage
0       0
                  v7k-silver.storageclass.storage.k8s.io/persistentvolumeclaims
0       5
                  v7k-silver.storageclass.storage.k8s.io/requests.storage
0       5Gi
Resource limits:        <none>
$
```

Modifying a resource quota

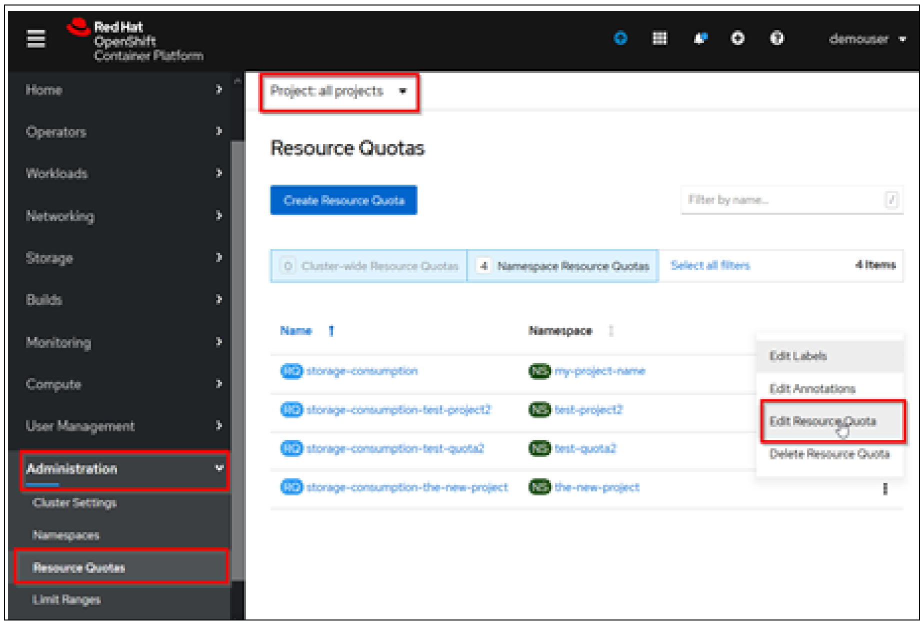
Because of the custom project template, you are now enforcing default quotas automatically on new projects. However, requesters might need more storage or use other, better storage classes.

OCP administrators can easily modify a resource quota by using one of the following methods (see Figure 3-2):

- ► From the CLI, with the following command:

  ```
  $ oc edit resourcequotas quota-to-modify -n project-name
  ```

- ► In the GUI, by clicking **Administration** → **Resource Quotas** and then, clicking **Edit Resource Quota** on the specific object.



Name	Namespace
RQ storage-consumption	NS my-project-name
RQ storage-consumption-test-project2	NS test-project2
RQ storage-consumption-test-quota2	NS test-quota2
RQ storage-consumption-the-new-project	NS the-new-project

Figure 3-2 Edit a resource quota

Next, update the spec section of the Resource Quota according to the requester's needs. The new quota is enforced immediately. As shown in Example 3-9, the requester is allowed to use 5 Gi of gold storage with a maximum of two Persistent Volume Claims.

Example 3-9 Update the resource quota

```
$ oc get resourcequotas -n the-new-project storage-consumption-the-new-project -o
yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  creationTimestamp: "2020-08-24T12:20:02Z"
  name: storage-consumption-the-new-project
  …
spec:
  hard:
    persistentvolumeclaims: "5"
    requests.storage: 10Gi
    v7k-gold.storageclass.storage.k8s.io/persistentvolumeclaims: "2"
    v7k-gold.storageclass.storage.k8s.io/requests.storage: 5Gi
    v7k-silver.storageclass.storage.k8s.io/persistentvolumeclaims: "5"
```

```
    v7k-silver.storageclass.storage.k8s.io/requests.storage: 5Gi
status:
  hard:
    persistentvolumeclaims: "5"
    requests.storage: 10Gi
    v7k-gold.storageclass.storage.k8s.io/persistentvolumeclaims: "0"
    v7k-gold.storageclass.storage.k8s.io/requests.storage: "0"
    v7k-silver.storageclass.storage.k8s.io/persistentvolumeclaims: "5"
    v7k-silver.storageclass.storage.k8s.io/requests.storage: 5Gi
  used:
    persistentvolumeclaims: "0"
    requests.storage: "0"
    v7k-gold.storageclass.storage.k8s.io/persistentvolumeclaims: "0"
    v7k-gold.storageclass.storage.k8s.io/requests.storage: "0"
    v7k-silver.storageclass.storage.k8s.io/persistentvolumeclaims: "0"
    v7k-silver.storageclass.storage.k8s.io/requests.storage: "0"
$
```

The updated quota is immediately applied, as shown in Example 3-10.

Example 3-10 Updated quota is applied

```
$ oc describe projects the-new-project
Name:           the-new-project
Created:        About an hour ago
Labels:         <none>
Annotations:    openshift.io/description=
                openshift.io/display-name=
                openshift.io/requester=test
                openshift.io/sa.scc.mcs=s0:c30,c0
                openshift.io/sa.scc.supplemental-groups=1000870000/10000
                openshift.io/sa.scc.uid-range=1000870000/10000
Display Name:   <none>
Description:    <none>
Status:         Active
Node Selector:  <none>
Quota:
                Name:
storage-consumption-the-new-project
                Resource                                                Used
Hard
                --------                                                ----
----
                persistentvolumeclaims                                  0      5
                requests.storage                                        0
10Gi
                v7k-gold.storageclass.storage.k8s.io/persistentvolumeclaims    0      2
                v7k-gold.storageclass.storage.k8s.io/requests.storage          0      5Gi
                v7k-silver.storageclass.storage.k8s.io/persistentvolumeclaims  0      5
                v7k-silver.storageclass.storage.k8s.io/requests.storage        0      5Gi
Resource limits:        <none>
$
```

3.3 Managing storage resources across multiple projects

Another possibility to limit the storage consumption is to use a cluster resource quota. Cluster resource quotas are shared across multiple projects. If a cluster resource quota defines a maximum of 10 Ti, those 10 Ti are shared for all the projects that are sharing this cluster quota.

When defining a cluster resource quota, you choose which project it is applied to by selecting it through annotations or labels, or both. For example, the cluster resource quota that is shown in Example 3-11 enforces storage quota on all of the projects that are requested by the user my-user-name.

Example 3-11 Setting a cluster resource quota

```
$ cat quota-storage-consumption-for-my-user-name.yml
apiVersion: v1
kind: ClusterResourceQuota
metadata:
  name: storage-consumption-for-my-user-name
spec:
  quota:
    hard:
      persistentvolumeclaims: "10"
      requests.storage: "7Gi"
      v7k-gold.storageclass.storage.k8s.io/requests.storage: "3Gi"
      v7k-gold.storageclass.storage.k8s.io/persistentvolumeclaims: "5"
      v7k-silver.storageclass.storage.k8s.io/requests.storage: "3Gi"
      v7k-silver.storageclass.storage.k8s.io/persistentvolumeclaims: "5"
  selector:
    annotations:
      openshift.io/requester: " my-user-name"
$ apply -f quota-storage-consumption-for-my-user-name.yml
clusterresourcequota.quota.openshift.io/storage-consumption-for-my-user-name
created
$
```

After it is created, such as the resource quota, the cluster resource quota is immediately applied. To view the resource consumption of a cluster resource quota, you perform one of the following actions:

- If you have cluster-admin privileges, run the following command to view the details of a specific cluster resource quota:

  ```
  $ oc describe clusterresourcequota storage-consumption-for-my-user-name
  ```

- If you do not have cluster-admin privileges, run the following command to view the cluster resource quota that is applied to the current project:

  ```
  $ oc describe AppliedClusterResourceQuota
  ```

In both cases, the output looks like the output that is shown in Example 3-12.

Example 3-12 Output of the describe command for a cluster resource quota

```
$ oc describe AppliedClusterResourceQuota
Name:           storage-consumption-for-my-user-name
Created:        10 minutes ago
Labels:         <none>
Annotations:
kubectl.kubernetes.io/last-applied-configuration={"apiVersion":"quota.openshift.io
/v1","kind":"ClusterResourceQuota","metadata":{"annotations":{},"name":"storage-co
nsumption-for-test"},"spec":{"quota":{"hard":{"persistentvolumeclaims":"10","reque
sts.storage":"7Gi","v7k-gold.storageclass.storage.k8s.io/persistentvolumeclaims":"
5","v7k-gold.storageclass.storage.k8s.io/requests.storage":"3Gi","v7k-silver.stora
geclass.storage.k8s.io/persistentvolumeclaims":"5","v7k-silver.storageclass.storag
e.k8s.io/requests.storage":"3Gi"}},"selector":{"annotations":{"openshift.io/reques
ter":"test"}}}}
Namespace Selector: ["test-quota2" "the-new-project" "my-project-name"
"test-project2"]
Label Selector:
AnnotationSelector: map[openshift.io/requester:test]
Resource                                                    Used    Hard
--------                                                    ----    ----
persistentvolumeclaims                                      2       10
requests.storage                                            2Gi     7Gi
v7k-gold.storageclass.storage.k8s.io/persistentvolumeclaims     1       5
v7k-gold.storageclass.storage.k8s.io/requests.storage       1Gi     3Gi
v7k-silver.storageclass.storage.k8s.io/persistentvolumeclaims   1       5
v7k-silver.storageclass.storage.k8s.io/requests.storage     1Gi     3Gi
$
```

4

Use cases

This chapter presents a series of use cases to demonstrate how the IBM Spectrum Virtualize appliance facilitates multi-tenancy. It includes the following topics:

- 4.1, “Provisioning storage on a Spectrum Virtualize dedicated to an OCP environment” on page 34
- 4.2, “Provisioning storage on a Spectrum Virtualize shared with other applications” on page 34
- 4.3, “Sharing storage classes across different customers with usage limits per project” on page 35
- 4.4, “Limiting storage usage per project’s owner” on page 35
- 4.5, “Dedicating a storage class to a specific customer’s project” on page 36
- 4.6, “Dedicating a storage class to be shared among all the customer’s projects” on page 38
- 4.7, “Conclusion” on page 39

4.1 Provisioning storage on a Spectrum Virtualize dedicated to an OCP environment

In this use case, the storage infrastructure is dedicated to OCP. The platform serves different customers or projects. In this example, it is safe to consider the OCP administrator as a Spectrum Virtualize administrator and that the Object-Based Access Control is not mandatory.

As a Spectrum Virtualize administrator, the OCP administrator can perform the following tasks:

- Create child pools and increase their size.
- Use a Data Reduction Pool (DRP) to create storage classes, and, thus, offer deduplicated volumes.
- Set throttling limits on pools, volumes, hosts, and so on.
- Create volume snapshots.

4.2 Provisioning storage on a Spectrum Virtualize shared with other applications

In this example, the storage infrastructure is shared with other workloads (SAP, Oracle, and so on). A subset of the storage array is dedicated to OCP workloads. Depending on how security is organized, the OCP administrator (and the IBM Block CSI driver) might not have full administrator privileges on the storage array.

With Spectrum Virtualize, you use the Object-based Access Control and create a limited administrator for the OCP environment, as described in Chapter 2, “Limiting storage resources with IBM Spectrum Virtualize” on page 7.

With this logic, storage tenants must be created in advance by the administrator and then delegated to the limited administrator, which is provided to the OCP administrator.

This limited administrator can perform the following tasks:

- Provision and use dynamically storage in the delegated pools.
- Create volumes snapshots.

This limited administrator must refer to the Spectrum Virtualize administrator to perform the following tasks:

- Create child pools for more storage classes.
- Increase the size of an added child.
- Set throttling limits.

This limited administrator cannot use DRPs, and, thus, offer deduplicated volumes.

4.3 Sharing storage classes across different customers with usage limits per project

In this use case, the OCP infrastructure features two storage classes: `v7k-gold` and `v7k-silver`. Kubernetes resource quotas are defined for each project, which setting limits (global and at the storage class level). We want to allow the projects' users to provision 10 Gi of gold storage and 20 Gi of silver storage, with a total maximum 25 Gi.

We define a resource quota that is to be applied to each customers' project, as shown in the following example:

```
$ cat quota-storage-consumption.yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: storage-consumption
spec:
  hard:
    requests.storage: "25Gi"
    v7k-gold.storageclass.storage.k8s.io/requests.storage: "10Gi"
    v7k-silver.storageclass.storage.k8s.io/requests.storage: "20Gi"
$ oc apply -f quota-storage-consumption.yaml -n project1
$ oc apply -f quota-storage-consumption.yaml -n project2
$ oc apply -f quota-storage-consumption.yaml -n project3
$
```

We can apply different resource quota with different values to each project. We can also apply default quota, as described in "Creating a custom project template enforcing resource quotas" on page 24.

4.4 Limiting storage usage per project's owner

In this use case, the OCP infrastructure features two storage classes: `v7k-gold` and `v7k-silver`. We do not want to limit storage usage per project, but per user. Although limiting per user is not possible, we can set a quota to a group of projects based on their requester.

We want to define a quota of 100 Gi of gold storage, 200 Gi of silver storage, with a total maximum capacity 250 Gi and a maximum of 100 of PVCs that are shared by all of Customer1's projects.

We define a cluster resource quota applied to projects whose owner is Customer1, as shown in Example 4-1.

Example 4-1 Defining a cluster resource quota

```
$ cat quota-storage-consumption-for-user-customer1.yaml
apiVersion: v1
kind: ClusterResourceQuota
metadata:
  name: storage-consumption-for-customer1
spec:
  quota:
    hard:
       requests.storage: "250Gi"
```

```
      v7k-gold.storageclass.storage.k8s.io/persistentvolumeclaims: "100"
      v7k-gold.storageclass.storage.k8s.io/requests.storage: "100Gi"
      v7k-silver.storageclass.storage.k8s.io/requests.storage: "200Gi"
  selector:
    annotations:
      openshift.io/requester: "customer1"
$ oc apply -f quota-storage-consumption-for-user-customer1.yaml
$
```

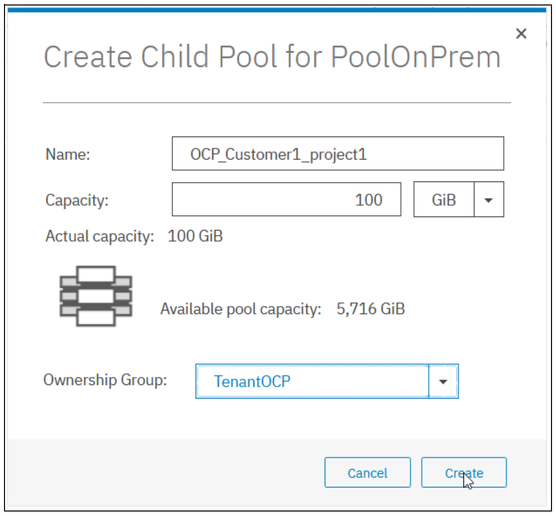
This quota is automatically applied to each project that is created by Customer1. It is also possible to create project-specific resource quota. In this case, the lower limits apply.

4.5 Dedicating a storage class to a specific customer’s project

In this use case, the OCP infrastructure features two storage classes: `v7k-gold` and `v7k-silver`. We want to dedicate a new storage class, with its own capacity (100 Gi) and specificities to our customer Customer1 for his project the-project. Customer1 cannot use any other storage classes.

The OCP project (`the-project`) exists with a default quota `storage-consumption-the-project`, as defined in “Creating a custom project template enforcing resource quotas” on page 24.

The first step is to create a dedicated pool for the customer project. Eventually, some throttling limits must be set. As shown in Figure 4-1, the pool named `OCP_Customer1_project1` is a child pool that is delegated to the OCP tenant.


Create Child Pool for PoolOnPrem

Name: OCP_Customer1_project1

Capacity: 100 GiB

Actual capacity: 100 GiB

Available pool capacity: 5,716 GiB

Ownership Group: TenantOCP

Figure 4-1 Creating a child pool

Note: If Object-based Access Control is used, this pool is a child pool, and the OCP administrator must request its creation.

A storage class must be created that targets this new pool and specifies the space efficiency option, as shown in Example 4-2.

Example 4-2 Creating the storage class

```
$ cat sc-customer1-theproject.yml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: v7k-customer1-theproject
  namespace: ibm-block-csi
provisioner: block.csi.ibm.com
parameters:
  pool: OCP_Customer1_TheProject
  csi.storage.k8s.io/provisioner-secret-name: v7k-secret
  csi.storage.k8s.io/provisioner-secret-namespace: ibm-block-csi
  csi.storage.k8s.io/controller-publish-secret-name: v7k-secret
  csi.storage.k8s.io/controller-publish-secret-namespace: ibm-block-csi
  csi.storage.k8s.io/fstype: ext4
$ oc apply -f sc-customer1-theproject.yml
storageclass.storage.k8s.io/v7k-customer1-theproject created
$
```

We now need to edit the quota (or create a quota because no quota exists) to allow only the use of the storage class `v7k-customer1-theproject`.

Because we cannot allow a specific storage class, we forbid the use of all other storage classes, setting their limits to 0. We specifically forbid the project to use the class `v7k-gold` and `v7k-silver`, as shown in Example 4-3.

Example 4-3 Editing the quota to forbid the use of some storage classes

```
$ oc edit resourcequotas -o yaml storage-consumption-the-project
apiVersion: v1
kind: ResourceQuota
metadata:
  creationTimestamp: "2020-08-25T13:16:49Z"
  name: storage-consumption-the-project
  namespace: the-project
  resourceVersion: "62544742"
  selfLink:
/api/v1/namespaces/the-project/resourcequotas/storage-consumption-the-project
  uid: 3ae58063-b1ff-458c-901a-7a2641a6abea
spec:
  hard:
    v7k-gold.storageclass.storage.k8s.io/persistentvolumeclaims: "0"
    v7k-gold.storageclass.storage.k8s.io/requests.storage: "0"
    v7k-silver.storageclass.storage.k8s.io/persistentvolumeclaims: "0"
    v7k-silver.storageclass.storage.k8s.io/requests.storage: "0"
status:
  ...
$
```

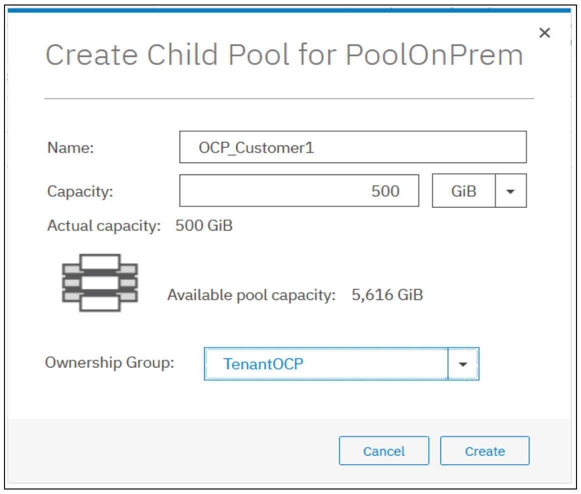
The project's users can use only the remaining “non-forbidden” storage class, which is `v7k-customer1-theproject`.

Important: If you add storage classes, do not forget to add them to the project's quota or the project's users cannot freely use them.

4.6 Dedicating a storage class to be shared among all the customer's projects

In this use case, the OCP infrastructure features two classes: `v7k-gold` and `v7k-silver`. We want to dedicate a new storage class, with its own capacity (500 Gi) and specifications, to Customer1. This storage class is shared among all his projects. Customer1 cannot use any other storage classes.

The first step is to create a dedicated pool for the customer. Eventually, some throttling limits are set. In this example, this pool is named `OCP_Customer1`. It is a child pool that is delegated to the OCP tenant (see Figure 4-2).


Create Child Pool for PoolOnPrem

Name: OCP_Customer1

Capacity: 500 GiB

Actual capacity: 500 GiB

Available pool capacity: 5,616 GiB

Ownership Group: TenantOCP

Figure 4-2 Creating a child pool for PoolOnPrem

Note: If Object-based Access Control is used, this pool is a child pool, and the OCP administrator must request that is be created.

We now create a cluster resource quota and apply it to all the projects that belong to Customer1. Because we cannot allow a specific storage class, we forbid the use of all the others, setting their limits to 0. In our example, we specifically forbid the project to use the class `v7k-gold` and `v7k-silveras`, shown in Example 4-4.

Example 4-4 Creating and applying a resource quota cluster

```
$ cat quota-storage-consumption-for-user-customer1.yaml
apiVersion: v1
kind: ClusterResourceQuota
metadata:
  name: storage-consumption-for-customer1
spec:
  quota:
    hard:
      v7k-gold.storageclass.storage.k8s.io/persistentvolumeclaims: "0"
      v7k-gold.storageclass.storage.k8s.io/requests.storage: "0"
      v7k-silver.storageclass.storage.k8s.io/persistentvolumeclaims: "0"
      v7k-silver.storageclass.storage.k8s.io/requests.storage: "0"
  selector:
    annotations:
      openshift.io/requester: "customer1"
$ oc apply -f quota-storage-consumption-for-user-customer1.yaml
clusterresourcequota.quota.openshift.io/storage-consumption-for-customer1 created
$
```

Important: Be aware of the following situations:

- If you later add storage classes, do not forget to add them to the project’s quota or the project’s users cannot freely use them.
- If Customer1’s projects include defined resource quotas (for example, by a default project template), the lower limits apply. You might want to delete the resource quotas if they are irrelevant.

4.7 Conclusion

In this publication, you saw that it is possible to restrict storage usage in a Red Hat OpenShift Container Platform by using resource quotas to avoid a user over-using storage.

You also saw that, because of IBM Spectrum Virtualize and the Object-Based Access Control, you can implement multi-tenancy to secure storage usage. Combining these features, you can secure the storage usage from the OpenShift user to the IBM Spectrum Virtualize array.

This document provides examples of cases where such control is useful; however, more use-cases might exist where these techniques and features can help to provide billing data.

A

Snapshot quota

It is possible to set quota limiting the numbers of snapshots, per project (`ResourceQuota`) or per group of projects (`ClusterResourceQuota`). To do so, you can define a resource quota (or cluster resource quota), as shown in Example A-1.

Example A-1 Defining a source quota

```
$ cat quota-storage-consumption.yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: storage-consumption
spec:
  hard:
    count/volumesnapshots.snapshot.storage.k8s.io: "2"
    persistentvolumeclaims: "10"
    requests.storage: 7Gi
    v7k-gold.storageclass.storage.k8s.io/persistentvolumeclaims: "5"
    v7k-gold.storageclass.storage.k8s.io/requests.storage: 3Gi
    v7k-silver.storageclass.storage.k8s.io/persistentvolumeclaims: "10"
    v7k-silver.storageclass.storage.k8s.io/requests.storage: 5Gi
$ oc apply -f quota-storage-consumption.yaml
resourcequota.quota.openshift.io/storage-consumption created
$ oc describe resourcequotas storage-consumption
Name:                                                           storage-consumption
Namespace:                                                      the-project
Resource                                                        Used  Hard
--------                                                        ----  ----
count/volumesnapshots.snapshot.storage.k8s.io                   0     2
persistentvolumeclaims                                          2     10
requests.storage                                                2Gi   7Gi
v7k-gold.storageclass.storage.k8s.io/persistentvolumeclaims     1     5
v7k-gold.storageclass.storage.k8s.io/requests.storage           1Gi   3Gi
v7k-silver.storageclass.storage.k8s.io/persistentvolumeclaims   1     10
v7k-silver.storageclass.storage.k8s.io/requests.storage         1Gi   5Gi
```

In this example, the project named `the-project` has a limit of two snapshots. After the limit is reached, creating a snapshot is impossible. Unlike storage quotas, you cannot limit the number of snapshots for a specific Volume Snapshot Class or the snapshot capacity.

REDP-5638-00

ISBN 073845978X