Red Hat
Training and Certification

Student Workbook (ROLE)

ODF 4.7 DO370

Enterprise Kubernetes Storage with Red Hat OpenShift Data Foundation

Edition 3

DO370-ODF4.7-en-3-20221129

Red Hat
Learning Community

Join a community dedicated to learning open source

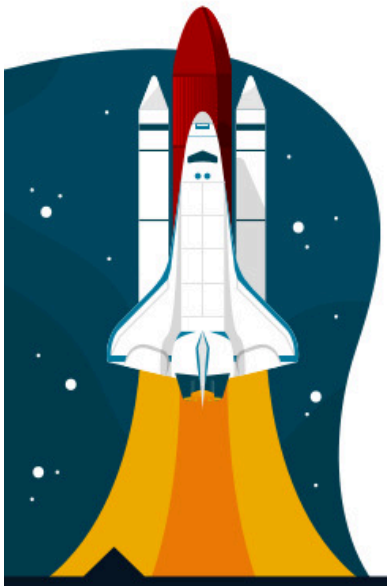
The Red Hat® Learning Community is a collaborative platform for users to accelerate open source skill adoption while working with Red Hat products and experts.

Network with tens of thousands of comunity members

Engage in thousands of active conversations and posts

Join and interact with hundreds of certified training instructors

Unlock badges as you participate and accomplish new goals



This knowledge-sharing platform creates a space where learners can connect, ask questions, and collaborate with other open source practitioners.

Access free Red Hat training videos

Discover the latest Red Hat Training and Certification news

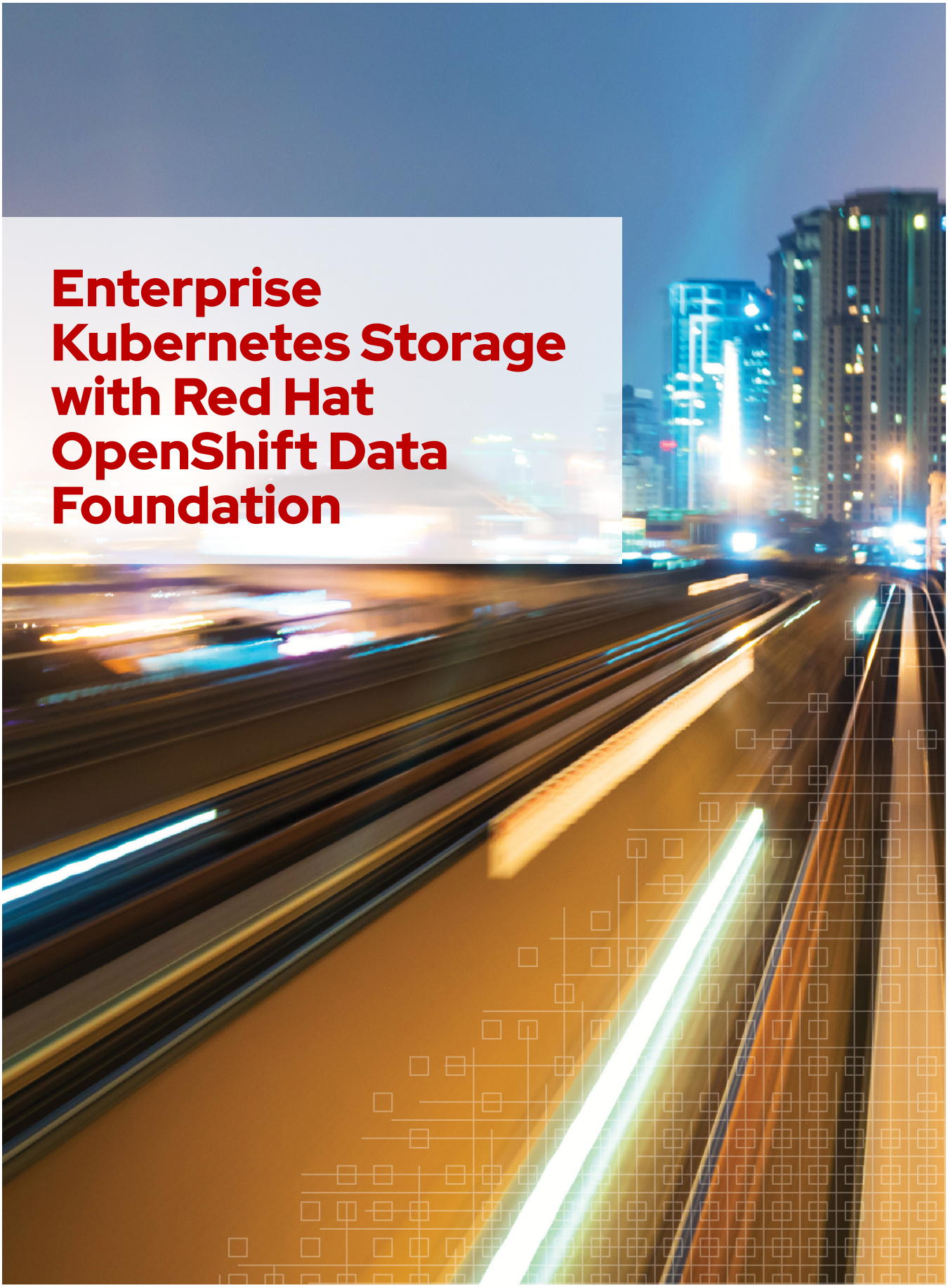
Connect with your instructor - and your classmates - before, after, and during your training course.

Join peers as you explore Red Hat products

Join the conversation learn.redhat.com

Red Hat

Enterprise Kubernetes Storage with Red Hat OpenShift Data Foundation



ODF 4.7 DO370
Enterprise Kubernetes Storage with Red Hat OpenShift Data Foundation
Edition 3 20221129
Publication date 20221129

Authors:	Andrés Hernández, Austin Garrigus, Ivan Chavero, Christopher Caillouet, Michael Phillips, Alejandro Coma
Course Architect:	Fernando Lozano
DevOps Engineers:	Benjamin Chardi, Jim Rigsbee
Editor:	Nicole Muller

Contributors: Chris Tusa, David Sacco, Eliezer Campos Laux, Nicolette Lucas, David O'Brien

Document Conventions

This section describes various conventions and practices that are used throughout all Red Hat Training courses.

Admonitions

Red Hat Training courses use the following admonitions:

References

References describe where to find external documentation that is relevant to a subject.

Note

Notes are tips, shortcuts, or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on something that makes your life easier.

Important

They provide details of information that is easily missed: configuration changes that apply only to the current session, or services that need restarting before an update applies. Ignoring these admonitions will not cause data loss, but might cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring these admonitions will most likely cause data loss.

Inclusive Language

Red Hat Training is currently reviewing its use of language in various areas to help remove any potentially offensive terms. This is an ongoing process and requires alignment with the products and services that are covered in Red Hat Training courses. Red Hat appreciates your patience during this process.

Introduction

DO370: Enterprise Kubernetes Storage with Red Hat OpenShift Data Foundation

This course teaches the essential skills required to design, implement, and manage a Red Hat OpenShift Data Foundation cluster. You also learn to perform day-to-day Kubernetes storage management tasks.

Course Objectives

- Install, configure, and manage Red Hat OpenShift Data Foundation in an OpenShift cluster.
- Provision storage for Red Hat OpenShift cluster services, such as monitoring and registry.
- Provision file and block storage to applications running on the OpenShift cluster.
- Monitor and expand storage capacity and performance.
- Create persistent volume snapshots and clones.
- Provision object storage in S3 buckets for compatible applications.

Audience

- Senior System Administrators who plan, design, and implement production-grade OpenShift clusters.
- Site Reliability Engineers (SREs) who must keep OpenShift clusters and applications running without disruption.

Prerequisites

- Red Hat Certified Specialist in OpenShift Administration certification (EX280) or equivalent knowledge for the roles of Red Hat OpenShift cluster administrator or SRE.
- Red Hat Certified Systems Administrator certification (EX200) or equivalent knowledge of Linux system administration is recommended for all roles.
- Basic knowledge of storage technologies, such as disk types, SAN, and NAS is recommended.

Orientation to the Classroom Environment

The Workstation Machine

In this course, the main computer system used for hands-on learning activities (exercises) is `workstation`.

The `workstation` machine has a standard user account, `student` with the password `student`. No exercise in this course requires that you log in as `root`, but if you must, the `root` password on the `workstation` machine is `redhat`.

It is from the `workstation` machine that you type the `oc` commands to manage the OpenShift cluster, which comes preinstalled as part of your classroom environment.

It is also from the `workstation` machine that you run shell scripts and Ansible Playbooks required to complete the exercises for this course.

If exercises require that you open a web browser to access any application or website, then you are required to use the graphical console of the `workstation` machine and use the Firefox web browser from there.

Note

The first time you start your classroom environment, the OpenShift cluster takes a little longer to become fully available. The `lab` command at the beginning of each exercise checks and waits as required.

If you try to access your cluster using either the `oc` command or the web console without first running a `lab` command, then you might find that your cluster is not yet available. If that happens, then wait a few minutes and try again.

Log in on OpenShift from the Shell

To access your OpenShift cluster from the `workstation` machine, use https://api.ocp4.example.com:6443 as the API URL, for example:

```
[student@workstation ~]$ oc login -u admin -p redhat \
  https://api.ocp4.example.com:6443
```

Besides the `admin` user, who has cluster administrator privileges, your OpenShift cluster also provides a `developer` user, with the password `developer`, who has no special privileges.

Accessing the OpenShift Web Console

If you prefer to use the OpenShift web console, open a Firefox web browser on your `workstation` machine and access the following URL:

https://console-openshift-console.apps.ocp4.example.com

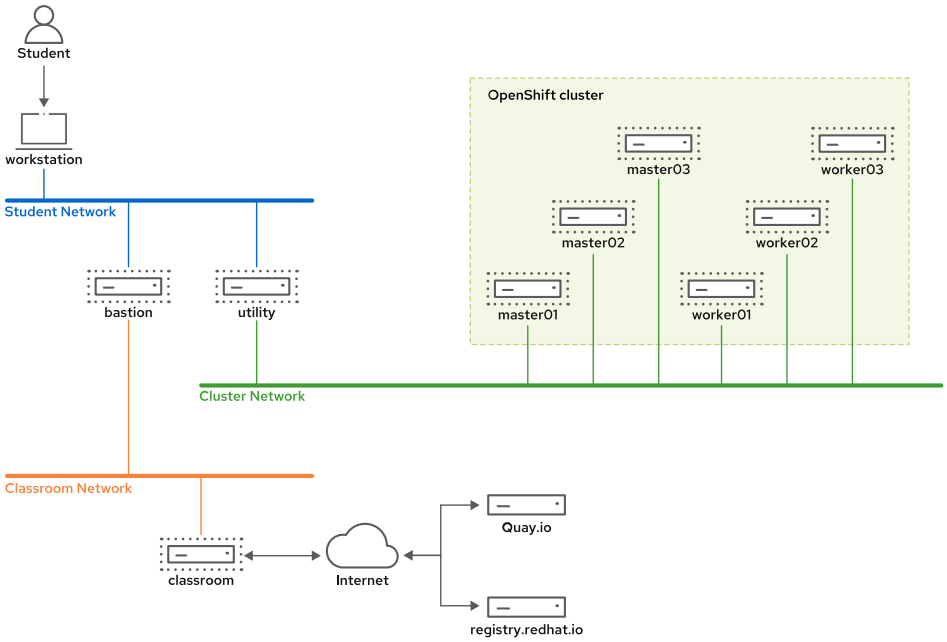
Click `htpasswd_provider` and provide the log in credentials for either the `admin` or `developer` user.

The Classroom Environment

Every student gets a complete remote classroom environment. As part of that environment, every student gets a dedicated OpenShift cluster to perform administration tasks.

The classroom environment runs entirely as virtual machines in a large Red Hat OpenStack Platform cluster, which is shared among many students.

Red Hat Training maintains many OpenStack clusters, in different data centers across the globe, to provide lower latency to students from many countries.



All machines on the Student, Classroom, and Cluster networks run Red Hat Enterprise Linux 8 (RHEL 8), except those machines that are nodes of the OpenShift cluster. These run RHEL CoreOS.

The systems called `bastion`, `utility`, and `classroom` must always be running. They provide infrastructure services required by the classroom environment and its OpenShift cluster. For most exercises, you are not expected to interact with any of these services directly.

Usually, the `lab` commands from exercises access these machines when there is a requirement to setup your environment for the exercise, and require no further action from you.

For the few exercises that require you to access a system other than `workstation`, primarily the `utility` system, you receive explicit instructions and necessary connection information as part of the exercise.

All systems in the *Student Network* are in the `lab.example.com` DNS domain, and all systems in the *Classroom Network* are in the `example.com` DNS domain.

The systems called `master`*XX* and `worker`*XX* are nodes of the OpenShift 4 cluster that is part of your classroom environment.

All systems in the *Cluster Network* are in the `ocp4.example.com` DNS domain.

Classroom Machines

Machine name	IP addresses	ROLE
`workstation.lab.example.com`	172.25.250.9	Graphical workstation used for system administration.
`classroom.example.com`	172.25.254.254	Router linking the Classroom Network to the Internet.
`bastion.lab.example.com`	172.25.250.254	Router linking the Student Network to the Classroom Network.
`utility.lab.example.com`	172.25.250.253	Router linking the Student Network to the Cluster Network and also storage server.
`master01.ocp4.example.com`	192.168.50.10	Control plane node
`master02.ocp4.example.com`	192.168.50.11	Control plane node
`master03.ocp4.example.com`	192.168.50.12	Control plane node
`worker01.ocp4.example.com`	172.25.250.13	Compute node
`worker02.ocp4.example.com`	172.25.250.14	Compute node
`worker03.ocp4.example.com`	172.25.250.15	Compute node

Dependencies on Internet Services

Red Hat OpenShift Container Platform 4 requires access to two container registries to download container images for operators, S2I builders, and other cluster services. These registries are:

- `registry.redhat.io`
- `quay.io`

If either registry is unavailable when starting the classroom environment, then the OpenShift cluster might not start or could enter a degraded state.

If these container registries experience an outage while the classroom environment is up and running, then it might not be possible to complete exercises until the outage is resolved.

The Dedicated OpenShift Cluster

The Red Hat OpenShift Container Platform 4 cluster inside the classroom environment is preinstalled using the Pre-existing Infrastructure installation method. All nodes are treated as bare metal servers, even though they are actually virtual machines in an OpenStack cluster.

OpenShift cloud-provider integration capabilities are not enabled and a few features that depend on that integration, such as machine sets and autoscaling of cluster nodes, are not available.

Your OpenShift cluster is in the state left by running the OpenShift installer with default configurations, excepting a few day-2 customizations:

- The cluster provides a default storage class backed by a Network File System (NFS) storage provider. This means that applications requiring persistent storage volumes perform comparably to running on a cluster installed using the Full-stack Automation installation method.
- There is an HTPasswd Identity Provider (IdP) preconfigured with two users: `admin` and `developer`.

The section *Troubleshooting Access to your OpenShift Cluster* provides information about how to access the `utility` machine.

Restoring Access to your OpenShift Cluster

If you suspect that you cannot log in to your OpenShift cluster as the `admin` user anymore because you incorrectly changed your cluster authentication settings, then run the `lab finish` command from your current exercise and restart the exercise by running its `lab start` command.

If running a `lab` command is not sufficient, then you can follow instructions in the next section to use the `utility` machine to access your OpenShift cluster.

Troubleshooting Access to Your OpenShift Cluster

The `utility` machine was used to run the OpenShift installer inside your classroom environment, and it is a useful resource to troubleshoot cluster issues. You can view the installer manifests and logs in the `/home/lab/ocp4` folder of the `utility` machine.

Logging in to the `utility` server is rarely required to perform exercises. If it looks like your OpenShift cluster is taking too long to start, or is in a degraded state, then you can log in on the `utility` machine as the `lab` user to troubleshoot your classroom environment.

The `student` user on the `workstation` machine is already configured with SSH keys that enable logging in to the `utility` machine without a password.

```
[student@workstation ~]$ ssh lab@utility
```

In the `utility` machine, the `lab` user is preconfigured with a `.kube/config` file that grants access as `system:admin` without first requiring `oc login`.

This allows you to run troubleshooting commands, such as `oc get node`, if they fail from the `workstation` machine.

You should not require SSH access to your OpenShift cluster nodes for regular administration tasks because OpenShift 4 provides the `oc debug` command. The `lab` user on the `utility` server is preconfigured with SSH keys to access all cluster nodes if necessary. For example:

```
[lab@utility ~]$ ssh -i ~/.ssh/lab_rsa core@master01.ocp4.example.com
```

In the preceding example, replace `master01` with the name of the desired cluster node.

Approving Node Certificates on your OpenShift Cluster

Red Hat OpenShift Container Platform clusters are designed to run continuously, 24x7, until they are decommissioned. Unlike a production cluster, the classroom environment contains a cluster that was stopped after installation and will be stopped and restarted several times before you finish this course. This scenario requires special handling that would not be required by a production cluster.

The control plane and compute nodes in an OpenShift cluster frequently communicate with each other. All communication between cluster nodes is protected by mutual authentication based on per-node TLS certificates.

The OpenShift installer handles creating and approving TLS certificate signing requests (CSRs) for the Full-stack Automation installation method. The system administrator manually approves these CSRs for the Preexisting Infrastructure installation method.

All per-node TLS certificates have a short expiration life of 24 hours (the first time) and 30 days (after renewal). When they are about to expire, the affected cluster nodes create new CSRs, and the control plane automatically approves them. If the control plane is offline when the TLS certificate of a node expires, then a cluster administrator is required to approve the pending CSR.

The `utility` machine includes a system service that approves CSRs from the cluster when you start your classroom, to ensure that your cluster is ready when you begin the exercises. If you create or start your classroom and begin an exercise too quickly, then you might find that your cluster is not ready. If so, wait a few minutes while the `utility` machine handles CSRs, and then try again.

Sometimes, the `utility` machine fails to approve all required CSRs, for example, because the cluster took too long to generate all required CSRs requests, and the system service did not wait long enough. It is also possible that some OpenShift cluster nodes did not wait long enough for their CSRs to be approved, issuing new CSRs that superseded previous ones.

If these issues arise, then you will notice that your cluster is taking too long to come up, and your `oc login` or `lab` commands fail. To resolve the problem, you can log in on the `utility` machine, as explained previously, and run the `sign.sh` script to approve any additional and pending CSRs.

```
[lab@utility ~]$ ./sign.sh
```

The `sign.sh` script loops a few times in case your cluster nodes issue new CSRs that supersede the ones it approved.

After you approve, or the system service in the `utility` machine approves all CSRs, then OpenShift must restart some cluster operators. It takes a few moments before your OpenShift cluster is ready to answer requests from clients. To help you handle this scenario, the `utility` machine provides the `wait.sh` script that waits until your OpenShift cluster is ready to accept authentication and API requests from remote clients.

```
[lab@utility ~]$ ./wait.sh
```

Although unlikely, if neither the service on the `utility` machine nor running the `sigh.sh` and `wait.sh` scripts make your OpenShift cluster available to begin exercises, then open a customer support ticket.

Note

You can run troubleshooting commands from the `utility` machine at any time, even if you have control plane nodes that are not ready. Some useful commands include:

- `oc get node` to verify if all of your cluster nodes are ready.
- `oc get csr` to verify if your cluster still has any pending, unapproved CSRs.
- `oc get co` to verify if any of your cluster operators are unavailable, in a degraded state, or progressing through configuration and rolling out pods.

If these fail, you can try destroying and recreating your classroom as a final step before creating a customer support ticket.

Controlling Your Systems

You are assigned remote computers in a Red Hat Online Learning classroom. They are accessed through a web application hosted at rol.redhat.com [http://rol.redhat.com]. You should log in to this site using your Red Hat Customer Portal user credentials.

Controlling the Virtual Machines

The virtual machines in your classroom environment are controlled through a web page. The state of each virtual machine in the classroom is displayed on the page under the **Online Lab** tab.

Machine States

Virtual Machine State	Description
STARTING	The virtual machine is in the process of booting.
STARTED	The virtual machine is running and available (or, when booting, soon will be).
STOPPING	The virtual machine is in the process of shutting down.
STOPPED	The virtual machine is completely shut down. Upon starting, the virtual machine boots into the same state as when it was shut down (the disk will have been preserved).
PUBLISHING	The initial creation of the virtual machine is being performed.
WAITING_TO_START	The virtual machine is waiting for other virtual machines to start.

Depending on the state of a machine, a selection of the following actions is available.

Classroom/Machine Actions

Button or Action	Description
PROVISION LAB	Create the ROL classroom. Creates all of the virtual machines needed for the classroom and starts them. Can take several minutes to complete.
DELETE LAB	Delete the ROL classroom. Destroys all virtual machines in the classroom. **Caution: Any work generated on the disks is lost.**
START LAB	Start all virtual machines in the classroom.
SHUTDOWN LAB	Stop all virtual machines in the classroom.
OPEN CONSOLE	Open a new tab in the browser and connect to the console of the virtual machine. You can log in directly to the virtual machine and run commands. In most cases, you should log in to the `workstation` virtual machine and use `ssh` to connect to the other virtual machines.
ACTION > Start	Start (power on) the virtual machine.
ACTION > Shutdown	Gracefully shut down the virtual machine, preserving the contents of its disk.
ACTION > Power Off	Forcefully shut down the virtual machine, preserving the contents of its disk. This is equivalent to removing the power from a physical machine.
ACTION > Reset	Forcefully shut down the virtual machine and reset the disk to its initial state. **Caution: Any work generated on the disk is lost.**

At the start of an exercise, if instructed to reset a single virtual machine node, click ACTION > Reset for only the specific virtual machine.

At the start of an exercise, if instructed to reset all virtual machines, click ACTION > Reset

If you want to return the classroom environment to its original state at the start of the course, you can click DELETE LAB to remove the entire classroom environment. After the lab has been deleted, you can click PROVISION LAB to provision a new set of classroom systems.

Warning
The DELETE LAB operation cannot be undone. Any work you have completed in the classroom environment up to that point will be lost.

The Autostop Timer

The Red Hat Online Learning enrollment entitles you to a certain amount of computer time. To help conserve allotted computer time, the ROL classroom has an associated countdown timer, which shuts down the classroom environment when the timer expires.

To adjust the timer, click MODIFY to display the New Autostop Time dialog box. Set the number of hours until the classroom should automatically stop. Note that there is a maximum time of ten hours. Click ADJUST TIME to apply this change to the timer settings.

Performing Lab Exercises

Run the `lab` command from the `workstation` machine to prepare your environment before each hands-on exercise, and again to clean up after an exercise. Each hands-on exercise has a unique name within a course.

There are two types of exercises.

- The first type, a *guided exercise*, is a practice exercise that follows a course narrative. If a narrative is followed by a quiz, this usually indicates that the topic did not have an achievable practice exercise.
- The second type, an *end-of-chapter lab*, is a gradable exercise to help verify your learning. When a course includes a comprehensive review, the review exercises are structured as gradable labs. The syntax for running an exercise script is:

```
[student@workstation ~]$ lab action exercise
```

The *action* is a choice of `start`, `grade`, or `finish`. All exercises support `start` and `finish`. Only end-of-chapter labs and comprehensive review labs support `grade`.

start
: The start logic for a script verifies the resources required to begin an exercise. This may include configuring settings, creating resources, checking prerequisite services, and verifying necessary outcomes from previous exercises. Exercise start logic allows you to perform any exercise at any time, even if prerequisite exercises have not been performed.

grade
: End-of-chapter labs help verify what you have learned, after practicing with earlier guided exercises. The grade action directs the `lab` command to display a list of grading criteria, with a PASS or `FAIL` status for each. To achieve a PASS status for all criteria, fix the failures, and then run the grade action again.

finish
: The finish logic for a script deletes exercise resources that are no longer necessary. Cleanup logic allows you to repeatedly perform an exercise, and benefits course performance by ensuring that unneeded objects release their resources.

Listing Available Lab Scripts

To list the available exercises for a course, use tab completion in the `lab` command:

```
[student@workstation ~]$ lab start Tab Tab
internal-gui            workloads-classes       backup-volume
internal-cli            workloads-review        backup-review
services-registry       capacity-monitoring     object-define
services-metrics        capacity-quotas         object-obc
services-review         capacity-extend         object-configure
workloads-ceph          capacity-disk           bucket-monitor
workloads-file          capacity-review         object-review
workloads-block         backup-application      comprehensive-review
```

Troubleshooting Lab Scripts

When you run the `lab` command with a valid exercise and action, it creates one log file in `/tmp/log/labs`, plus the directory if it does not exist. The file, named `exercise`, captures error messages that are normally displayed on your terminal.

```
[student@workstation ~]$ ls -l /tmp/log/labs
-rw-rw-r--. 1 student student 1024 Sep 28 12:00 comprehensive-review
```

Interpreting the Exercise Log Files

Exercise scripts send output to the log file when the scripts fail. Thus, the exercise log is useful for troubleshooting.

Although exercise scripts are always run from `workstation`, they perform tasks on other systems in the course environment. Many course environments, including OpenStack and OpenShift, use a command-line interface (CLI) that is invoked from `workstation` to communicate with server systems by using API calls.

Because script actions typically distribute tasks to multiple systems, additional troubleshooting is necessary to determine where a failed task occurred. Log in to those other systems and use Linux diagnostic skills to read local system log files and determine the root cause of the lab script failure.

Upgrading Lab Package Version

The workstation machine has the course lab package installed. If the instructor requests it, you can upgrade the version to fix a bug in the lab package that came preinstalled.

```
[student@workstation ~]$ lab upgrade DO370
```

Chapter 1

Describing the Architecture and Deploying Red Hat OpenShift Data Foundation Using Internal Mode

Goal

Install an OpenShift Data Foundation cluster on an OpenShift cluster using the internal mode.

Objectives

- Describe the architecture of Red Hat OpenShift Data Foundation.
- Deploy Red Hat OpenShift Data Foundation on-premises using the OpenShift web console
- Deploy Red Hat OpenShift Data Foundation on premises using the CLI.

Sections

- Introduction to the Architecture of Red Hat OpenShift Data Foundation (and Quiz)
- Deploying Red Hat OpenShift Data Foundation from the Web Console (and Guided Exercise)
- Deploying Red Hat OpenShift Data Foundation from the Command Line (and Guided Exercise)

Introduction to the Architecture of Red Hat OpenShift Data Foundation

Objectives

After completing this section, you should be able to describe the architecture of Red Hat OpenShift Data Foundation.

Introducing Red Hat OpenShift Data Foundation

Red Hat OpenShift Data Foundation provides cloud native data services for Red Hat OpenShift Container Platform (RHOCP) or any other infrastructure. OpenShift Data Foundation implementations avoid common storage issues, such as a lack of portability, deployment burden, and vendor lock-in. Running as a Kubernetes service, and following the Kubernetes operator model, OpenShift Data Foundation provides access to file, block, and object storage, based on `Ceph` technology. This enables the platform to run a wide range of workloads, including:

- Data at rest (databases, data warehouses)
- Data in motion (pipelines)
- Data in action (continuous deployment tools, analytics, artificial intelligence (AI), and machine learning (ML))

Furthermore, OpenShift Data Foundation leverages the multi-cloud object gateway (MCG) service, based in the NooBaa project, acting as a data federation service.

The MCG service provides a local object service (S3 API) backed by local storage or cloud-native storage. This storage service provides the same user experience regardless of being on premises or in the cloud. Finally, OpenShift Data Foundation uses the Rook-Ceph storage operator to manage and orchestrate persistent volume claims and dynamic provisioning.

Benefits of Red Hat OpenShift Data Foundation Compared with NFS Storage

OpenShift Data Foundation is a highly available cloud-native storage environment with multi-site availability. It removes the following NFS limitations:

- Heavy administrative workload
  - Manual share provisioning and persistent volume (PV) definition
  - Custom security context requirements
- Does not provide a true RWO mode, which can cause data corruption
- Persistent volume claim (PVC) names cannot be reused or data might be lost
- Not supported for important RHOCP services, such as registry, metrics, and logging

Benefits of Red Hat OpenShift Data Foundation Compared with EBS Storage

OpenShift Data Foundation can provide high availability (HA) across availability zones (AZs) in cloud environments.

Due to the capabilities of `Ceph` storage, OpenShift Data Foundation can replicate and store data using a CRUSH algorithm, avoiding a single point of failure.

Thus, OpenShift Data Foundation can be configured to use storage from different clouds, regions, and AZs.

OpenShift Data Foundation also removes the following limitations of EBS storage:

- Support for RWO access mode only
- No regional HA or portability
- Volume size limits
- Instance attachment limit of 28 objects, including EBS volumes, network interfaces, disk devices, and other objects.

Benefits of Red Hat OpenShift Data Foundation Compared with vSphere Volume

vSphere volumes are backed by VMware vSphere Virtual Machine Disk (VMDK) files. Because vSphere volumes are provided by a traditional virtualization platform, they have the following limitations compared to OpenShift Data Foundation:

- Support for RWO access mode only
- No regional HA or portability
- Incompatible with volume snapshots
- Vendor lock-in

Architecture of OpenShift Data Foundation

OpenShift Data Foundation is comprised of three operators:

- OpenShift Container Storage (OCS) operator. Acts as a meta-operator, installing the Rook-Ceph and NooBaa operators as a requirement.
- Rook-Ceph storage operator. Provides the back-end services for file, block, and object data storage.
- NooBaa operator. Provides the MCG service.

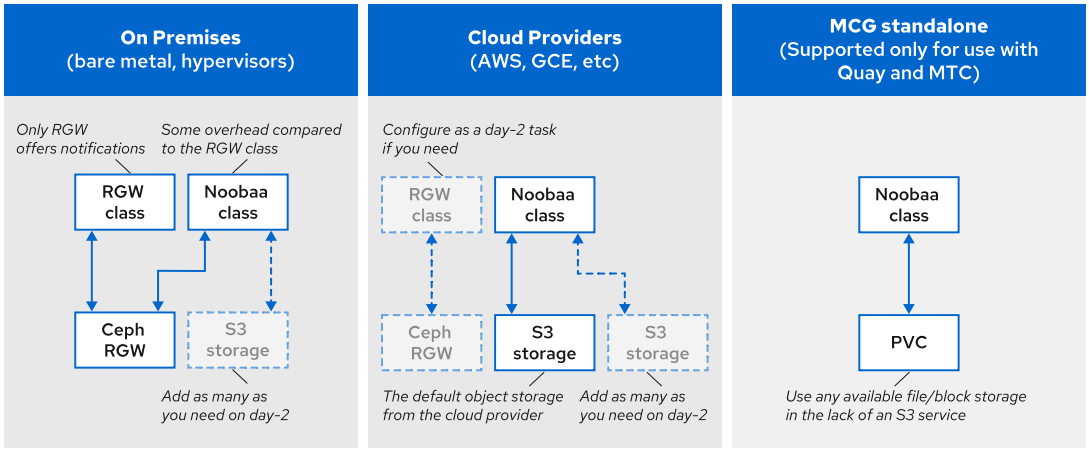


Figure 1.1: Red Hat OpenShift Data Foundation deployment modes and object storage providers

The preceding diagram provides an overview of how different deployment modes affect object storage at deployment time.

- On-premises deployments use physical storage that is passed into Ceph. This storage is used with RGW class object bucket claims (OBCs) or Noobaa class OBCs.
  - S3 storage can be added later to be handled through Noobaa.

- Cloud deployments use S3 storage handled through Noobaa.
  - S3 storage can be added at a later time to be handled through Noobaa.
  - Ceph RGW can also be added later.
- MCG Standalone uses any available PVC handled through Noobaa.

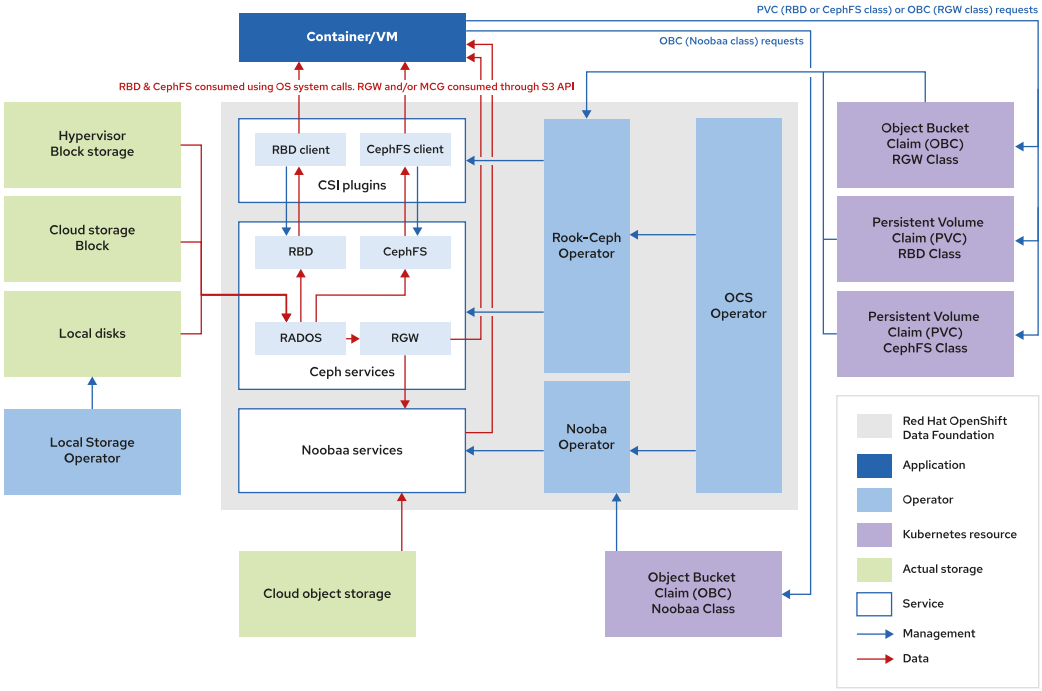


Figure 1.2: Red Hat OpenShift Data Foundation simplified architecture

The preceding diagram shows how each component of OpenShift Data Foundation works with each other piece.

- Block storage is handled through Ceph services within OpenShift Data Foundation.
  - The LSO operator manages the storage provided by the local disks, and the Ceph services handle the block storage volumes provided by the Ceph cluster.
  - The Ceph services can pass data into either Noobaa, for handling as object storage, or into a Kubernetes Container Storage Interface (CSI) plug-in for direct OS-level access by containers or virtual machines (VMs).
- Cloud object storage is handled through Noobaa services within OpenShift Data Foundation. Noobaa provides object storage to your containers or VMs.
  - Noobaa is managed by the Noobaa operator.
- Your applications can request storage in the form of OBCs or PVCs.
  - These OBCs and PVCs are handled by the Noobaa operator and Rook-Ceph operator, respectively. RGW class OBCs are handled by the Rook-Ceph operator.
- The OCS operator provides additional internal management of the Rook-Ceph operator and the Noobaa operator.

Comparing Internal Mode to External Mode Deployments

OpenShift Data Foundation provides deployment flexibility so that you can adopt the most appropriate approach for each environment. Administrators can choose to deploy OpenShift Data Foundation entirely within an RHOCP cluster (internal), or to expose Red Hat Ceph Storage services running outside the cluster (external).

Internal Deployment Approach

Independent of the chosen deployment approach, OpenShift Data Foundation provides the benefits of operator-based deployments. Furthermore, OpenShift Data Foundation can use internal (local) disk devices, if used in combination with the LSO operator.

There are two deployment modalities when deploying OpenShift Data Foundation entirely within RHOCP.

Simple

The Simple deployment uses the local storage devices, in combination with the LSO operator, or portable storage devices such as EBS volumes, vSphere volumes, or SAN volumes.

This approach is useful when:

- Storage requirements are not clear.
- There are no dedicated infrastructure nodes.
- Creating an extra node instance is difficult, such as on bare metal servers.

Optimized

The optimized deployment approach uses dedicated RHOCP infrastructure nodes. This approach is recommended when:

- Storage requirements are clear.
- There are dedicated infrastructure nodes with enough resources available.
- Creating an extra node instance with specific requirements is easy, such as in cloud and virtualized environments.

Infrastructure

OpenShift Data Foundation version 4.7 is available to use on an RHOCP cluster that is running on the following infrastructure:

- Amazon Web Services
- Bare metal
- VMware vSphere
- Red Hat Virtualization 4.4.x or higher (Full-stack automation)
- Microsoft Azure
- Google Cloud (Technology Preview)
- Red Hat OpenStack 13 or higher (Full-stack automation) (Technology Preview)
- IBM Power Systems
- IBM Z and LinuxONE

External Deployment Approach

With the external approach, OpenShift Data Foundation uses independent, external Red Hat Ceph Storage services running outside the cluster.

The external approach is recommended when:

- Storage requirements are significant (600+ storage devices).
- Several RHOCP clusters consume storage services from a common external cluster.

- Another team (such as SRE or storage) manages the external cluster that provides the storage services.

OpenShift Data Foundation version 4.7 can make use of external Red Hat Ceph Storage clusters when the RHOCP cluster is running on the following platforms:

- VMware vSphere
- Bare Metal
- Red Hat OpenStack platform (Technology preview)

References

For more information, refer to the product documentation for *Red Hat OpenShift Data Foundation* at
https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/html-single/planning_your_deployment/index

Quiz

Introduction to the Architecture of Red Hat OpenShift Data Foundation

Choose the correct answers to the following questions:

1. Which three of the following types of data storage are managed using OpenShift Data Foundation? (Choose three.)

a. Files
b. Books
c. Blocks
d. Objects

2. Which two of the following limitations of NFS are removed by using OpenShift Data Foundation? (Choose two.)

a. NFS does not allow for manual interactions to manage allocated storage volumes.
b. NFS does not provide a true RWO mode, which can cause data corruption.
c. NFS is not supported for important RHOCP services, such as registry, metrics, and logging.
d. NFS is not an appropriate backing store for any OpenShift deployed applications.

3. Which two of the following statements about OpenShift Data Foundation are true? (Choose two.)

a. You can manage a variety of storage types required for your applications with a single approach and set of operators.
b. OpenShift Data Foundation provides validity checks to inform users of the proper type of storage for the components of their applications.
c. OpenShift Data Foundation automatically discovers and configures all storage pools on your network and makes these available to the cluster through PVCs.
d. OpenShift Data Foundation enables cluster administrators, engineers, and operators to manage storage with the familiar ease of managing other cluster tasks.

4. Which three of the following operators comprise OpenShift Data Foundation? (Choose three.)

a. librados
b. OpenShift Container Storage
c. Rook-Ceph
d. OpenShift Storage Manager
e. NooBaa

5. **Which three of the following situations are appropriate reasons for using the simple mode approach when performing an OpenShift Data Foundation internal deployment? (Choose three.)**

   a. Storage requirements are not clear.

   b. The environment uses Amazon Web Services for infrastructure.

   c. No dedicated infra nodes are available in the cluster.

   d. Creating a new node instance is difficult or unavailable.

Solution

Introduction to the Architecture of Red Hat OpenShift Data Foundation

Choose the correct answers to the following questions:

1. Which three of the following types of data storage are managed using OpenShift Data Foundation? (Choose three.)

a. Files
b. Books
c. Blocks
d. Objects

2. Which two of the following limitations of NFS are removed by using OpenShift Data Foundation? (Choose two.)

a. NFS does not allow for manual interactions to manage allocated storage volumes.
b. NFS does not provide a true RWO mode, which can cause data corruption.
c. NFS is not supported for important RHOCP services, such as registry, metrics, and logging.
d. NFS is not an appropriate backing store for any OpenShift deployed applications.

3. Which two of the following statements about OpenShift Data Foundation are true? (Choose two.)

a. You can manage a variety of storage types required for your applications with a single approach and set of operators.
b. OpenShift Data Foundation provides validity checks to inform users of the proper type of storage for the components of their applications.
c. OpenShift Data Foundation automatically discovers and configures all storage pools on your network and makes these available to the cluster through PVCs.
d. OpenShift Data Foundation enables cluster administrators, engineers, and operators to manage storage with the familiar ease of managing other cluster tasks.

4. Which three of the following operators comprise OpenShift Data Foundation? (Choose three.)

a. librados
b. OpenShift Container Storage
c. Rook-Ceph
d. OpenShift Storage Manager
e. NooBaa

5. **Which three of the following situations are appropriate reasons for using the simple mode approach when performing an OpenShift Data Foundation internal deployment? (Choose three.)**

   a. Storage requirements are not clear.

   b. The environment uses Amazon Web Services for infrastructure.

   c. No dedicated infra nodes are available in the cluster.

   d. Creating a new node instance is difficult or unavailable.

Deploying Red Hat OpenShift Data Foundation from the Web Console

Objectives

After completing this section, you should be able to deploy Red Hat OpenShift Data Foundation on-premises using the OpenShift web console

Deploying OpenShift Data Foundation from the Graphical User Interface

OpenShift Data Foundation can be deployed by installing the OCS operator from the OperatorHub menu in the RHOCP web console.

If using local volumes as storage, you must first install the LSO operator from OperatorHub. The LSO operator creates the necessary Kubernetes custom resources (CRs) to enable the usage of local volumes attached to the nodes.

The volume modes can be either Block or File system.

To use the local storage with the LSO operator, the nodes must have at least one empty raw block device available.

Thus, the overall installation steps are:

1. Install the LSO operator (optional).
2. Label at least three nodes hosting the data services in the storage cluster (optional).
3. Install the OCS operator (mandatory).
4. Create the storage cluster (mandatory).

Before installing the OCS operator, you must identify three nodes on which the OpenShift Data Foundation services are installed. If those nodes are not labeled with the `cluster.ocs.openshift.io/openshift-storage=` label, OpenShift Data Foundation will label them when creating the storage cluster.

Note

In production environments, taint a node as `infra` to ensure that only Red Hat OpenShift Container Storage resources (or other infrastructure resources) are scheduled on that node. This helps to reduce subscription costs. For more information, refer to the *Managing and Allocating Storage Resources Guide* at https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/html-single/managing_and_allocating_storage_resources/index#how-to-use-dedicated-worker-nodes-for-openshift-container-storage_rhocs

After installing the OCS operator, administrators must create the storage cluster object.

The steps for creating the storage cluster depend on the installation mode. The overall steps for each installation mode follow.

Internal mode

1. Select the installation mode: **Internal**.
2. Select the storage class that uses the storage back end, the requested capacity, and the nodes
3. Enable data encryption, if desired.
4. Review the settings and create the storage cluster object.

Internal - Attached Devices mode

1. Select the installation mode: **Internal - Attached Devices**.
2. Select the nodes to discover available disks, or look in all nodes.
3. After choosing the volume set name, click **Next** and wait until the disk devices are discovered.
4. Enable data encryption, if desired.
5. Review the settings and create the storage cluster object.

External mode

1. Select the installation mode: **External**.
2. Download and run the `ceph-external-cluster-details-exporter.py` script on a Red Hat Ceph Storage node with the `admin` key.
3. Save the output to a `.json` file.
4. Upload the file using the **Browse** button.
5. Review the settings and create the storage cluster object.

Note
For more information on how to create a cluster service in external mode, review the *Deploying OpenShift Container Storage in External Mode Guide* at https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/html-single/deploying_openshift_container_storage_in_external_mode/index#creating-an-openshift-container-storage-cluster-service-for-external-storage_rhocs

Verifying the OpenShift Data Foundation Installation from the Graphical User Interface

To verify the successful installation of OpenShift Data Foundation, confirm the `Running` status of the following pods in the `openshift-storage` namespace or the custom namespace selected during the installation.

Pods Corresponding to an External Mode Installation

Component	Pods	Pod placement
CSI - `cephfs`	`csi-cephfsplugin-*`	1 pod on each worker node

Component	Pods	Pod placement
CSI - `cephfs`	`csi-cephfsplugin-provisioner-*`	2 pods distributed across worker nodes
CSI - `rbd`	`csi-rbdplugin-*`	1 pod on each worker node
CSI - `rbd`	`csi-rbdplugin-provisioner-*`	2 pods distributed across worker nodes
OCS operator	`ocs-operator-*`	1 pod on any worker node
OCS operator	`ocs-metrics-exporter-*`	1 pod on any worker node
Rook-Ceph operator	`rook-ceph-operator-*`	1 pod on any worker node
NooBaa MCG	`noobaa-operator-*`	1 pod on any worker node
NooBaa MCG	`noobaa-core-*`	1 pod on any worker node
NooBaa MCG	`nooba-db-*`	1 pod on any worker node
NooBaa MCG	`noobaa-endpoint-*`	1 pod on any worker node

Internal mode

In addition to the pods described in the "Pods Corresponding to External Mode Installation" table, the internal mode also requires the pods that correspond to Red Hat Ceph Storage.

Additional Pods Corresponding to an Internal Mode Installation

Component	Pods	Notes
Ceph MON	`rook-ceph-mon-*`	• 3 pods distributed across storage nodes • For arbiter, 5 pods are distributed across 3 zones, 2 per data center zones and 1 in arbiter zone
Ceph MGR	`rook-ceph-mgr-*`	• 1 pod on any storage node
Ceph MDS	`rook-ceph-mds-ocs-storagecluster-cephfilesystem-*`	• 2 pods distributed across storage nodes • For arbiter, 2 pods are distributed across 2 data center zones
Ceph RGW	`rook-ceph-rgw-ocs-storagecluster-cephobjectstore-*`	• 1 pod on any storage node • For arbiter, 2 pods are distributed across 2 data center zones
Ceph crash collector	`rook-ceph-crashcollector-*`	• 1 pod on each storage node • For arbiter, 1 pod on each storage node and 1 pod in arbiter zone

Component	Pods	Notes
Ceph OSD	`rook-ceph-osd-*`	• 1 Ceph object storage daemon (Ceph OSD) pod for each device
Ceph OSD	`rook-ceph-osd-prepare-ocs-deviceset-*`	• 1 pod for each device

The web console provides a summarized view of the persistent storage status from the Overview page. This page also provides information about the status of the cluster and object service. After the installation completes, you must ensure that all the data services provided by OpenShift Data Foundation are healthy.

Finally, you must corroborate that OpenShift Data Foundation created the four storage classes:

- `ocs-storagecluster-ceph-rbd`
- `ocs-storagecluster-cephfs`
- `openshift-storage.noobaa.io`
- `ocs-storagecluster-ceph-rgw`

Uninstalling OpenShift Data Foundation

The uninstall method differs depending on whether the installation used internal or external mode. However, there are common considerations for both methods.

Uninstall annotations

To define the OpenShift Data Foundation uninstall behavior, there are two annotations:

- `uninstall.ocs.openshift.io/cleanup-policy`
- `uninstall.ocs.openshift.io/mode`

If the `uninstall.ocs.openshift.io/cleanup-policy` is set to `delete`, the Rook-Ceph storage operator cleans up the physical drives and the `DataDirHostPath`. However, if the `uninstall.ocs.openshift.io/cleanup-policy` is set to `retain`, it keeps the physical drives and `DataDirHostPath`.

If the `uninstall.ocs.openshift.io/mode` is set to `graceful`, the Rook-Ceph and NooBaa operators pause the uninstall process until the PVCs and OBCs are removed manually. However, if the `uninstall.ocs.openshift.io/mode` is set to `forced`, the Rook-Ceph and NooBaa operators proceed with the uninstall process, even if the PVCs and the OBCs still exist.

Note

The Rook-Ceph storage operator removes the PVCs and the NooBaa operator removes the OBCs.

Administrators can apply these annotations by using the following commands:

```
[user@demo ~]$ oc annotate storagecluster ocs-storagecluster \
  uninstall.ocs.openshift.io/cleanup-policy="retain" --overwrite
```

```
[user@demo ~]$ oc annotate storagecluster ocs-storagecluster \
  uninstall.ocs.openshift.io/mode="forced" --overwrite
```

Uninstall prerequisites

Before attempting to uninstall OpenShift Data Foundation, you must:

- Ensure that the Red Hat OpenShift Container Storage cluster is in a healthy state. Otherwise, the process can fail because pods cannot be terminated successfully due to insufficient resources or other problems.
- Ensure that applications or other services are not consuming PVCs or OBCs that use the storage classes provided by OpenShift Data Foundation.
- After removing the dependent resources, delete any other CRs created by using the custom resource definitions provided by OpenShift Data Foundation.

Uninstall process

Regardless of the installation method, common objects must be removed. For example:

- Volume snapshots
- PVCs and OBCs
- Storage cluster
- Clean up pods in `Completed` status
- The `dm-crypt device-mapper` mapping, if encryption was enabled
- The namespace where OpenShift Data Foundation was installed (`openshift-storage` is the default)
- The LSO operator and configurations
- The labels of the storage nodes
- The taints of the storage nodes (if tainted)
- Any PV in the `Released` state
- The MCG storage class `openshift-storage.noobaa.io`

Finally, to complete the removal, administrators must delete the following CRDs:

- `backingstores.noobaa.io`
- `bucketclasses.noobaa.io`
- `bucketclasses.noobaa.io`
- `cephblockpools.ceph.rook.io`
- `cephclusters.ceph.rook.io`
- `cephfilesystems.ceph.rook.io`
- `cephnfses.ceph.rook.io`
- `cephobjectstores.ceph.rook.io`
- `cephobjectstoreusers.ceph.rook.io`
- `noobaas.noobaa.io`
- `ocsinitializations.ocs.openshift.io`
- `storageclusters.ocs.openshift.io`
- `cephclients.ceph.rook.io`
- `cephobjectrealms.ceph.rook.io`
- `cephobjectzonegroups.ceph.rook.io`
- `cephobjectzones.ceph.rook.io`
- `cephrbdmirrors.ceph.rook.io`

References

For more information, refer to the product documentation for *Red Hat OpenShift Data Foundation* at
https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/

For more information on how to remove LSO configurations, refer to the *Removing Local Storage Operator Configurations* section in the *Deploying OpenShift Container Storage Using Bare Metal Infrastructure* guide at
https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/html-single/deploying_openshift_container_storage_using_bare_metal_infrastructure/index#removing-local-storage-operator-configurations_rhocs

For detailed information on how to uninstall Red Hat OpenShift Data Foundation in external mode, visit
https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/html-single/deploying_openshift_container_storage_in_external_mode/index#uninstalling-openshift-container-storage-external-in-external-mode_rhocs
.

For more information on how to uninstall Red Hat OpenShift Data Foundation in internal mode, refer to
https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/html-single/deploying_openshift_container_storage_using_bare_metal_infrastructure/index#uninstalling-openshift-container-storage-in-internal-mode_rhocs

Guided Exercise

Deploying Red Hat OpenShift Data Foundation from the Web Console

In this exercise, you will use the web console to configure your OpenShift cluster to use OpenShift Data Foundation.

Outcomes

You should be able to:

- Label nodes to use as storage nodes.
- Install the operator Local Storage Operator.
- Install the OpenShift Container Storage operator.
- List resources created by the web console.
- Verify the successful installation of OpenShift Data Foundation.

Before You Begin

To perform this exercise, ensure that you have access to a running OpenShift cluster.

Important

If you already completed this exercise or if you completed the *Guided Exercise: Deploying Red Hat OpenShift Data Foundation from the Command Line*, then OpenShift Data Foundation is already installed.

To perform this exercise again, delete and then recreate your lab environment.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable.

```
[student@workstation ~]$ lab start internal-gui
```

Instructions

1. Label the worker nodes to identify them as the nodes used for storage.

   1.1. Log in to your OpenShift cluster as the `admin` user with the `redhat` password.

```
[student@workstation ~]$ oc login -u admin \
  -p redhat https://api.ocp4.example.com:6443
```

1.2. List the worker nodes in the cluster.

```
[student@workstation ~]$ oc get nodes -l node-role.kubernetes.io/worker=
NAME       STATUS   ROLES    AGE   VERSION
worker01   Ready    worker   14d   v1.20.0+a0b09eb
worker02   Ready    worker   14d   v1.20.0+a0b09eb
worker03   Ready    worker   14d   v1.20.0+a0b09eb
```

1.3. Add the `cluster.ocs.openshift.io/openshift-storage=` label to the three worker nodes.

```
[student@workstation ~]$ oc label nodes \
  -l node-role.kubernetes.io/worker= \
  cluster.ocs.openshift.io/openshift-storage=
node/worker01 labeled
node/worker02 labeled
node/worker03 labeled
```

2. Install the Local Storage Operator by using the web console.

2.1. Show the URL for the web console and open it using the web browser.

```
[student@workstation ~]$ oc whoami --show-console
https://console-openshift-console.apps.ocp4.example.com
```

2.2. Using the `htpasswd_provider` identity provider, log in to the cluster with the `admin` username and the `redhat` password.

2.3. Click **Operators** > **OperatorHub**, and then type `local` in the **Filter by keyword** field. Click **Local Storage** from the list of operator results, and then click **Install**.

2.4. Accept all the default settings from the **Operator Installation** page, and then click **Install**.

Note

The namespace `openshift-local-storage` is created.

2.5. After the operator installation completes, click **View Operator**.

2.6. Select the **Local Volume Discovery** tab, and then click **Create Local Volume Discovery**. Select **All nodes** in the **Node Selector** field, and then click **Create**.

Note

The previous step created a `LocalVolumeDiscovery` resource in the `openshift-local-storage` namespace. It might take up to two minutes before the local volume discovery resource discovers all the disk devices on the selected nodes.

2.7. Click **Operators** > **Installed Operators**, and then click **Local Storage**.

2.8. Select the **Local Volume Set** tab, and then click **Create Local Volume Set**. Type `lso-volumeset` in the **Volume Set Name** field. Select **All nodes** in the **Node Selector** field, and then click **Create**.

3. Use the web console to install the OCS operator.

3.1. Click **Operators** > **OperatorHub**, and then type `OCS` in the **Filter by keyword** field. Click **OpenShift Container Storage** from the list of operator results, and then click **Install**.

3.2. Accept all the default settings from the **Operator Installation** page, and then click **Install**.

Note
The namespace `openshift-storage` is created.

4. Configure the OpenShift Container Storage operator to use internal storage.

4.1. After the operator installation completes, click **Create StorageCluster**.

4.2. Select the **Internal - Attached Devices** mode, and then select the `lso-volumeset` Storage Class. Wait until all of the worker nodes are listed, and then click **Next**.

Note
The **Stretch Cluster** check box is disabled and is not required for this exercise.

The **Next** button is disabled until local disks are discovered.

4.3. Ensure that the check box for the **Encryption** field is cleared, and then click **Next**.

4.4. Review the storage cluster configuration, and then click **Create**.

5. Monitor the progress of the pods and PVCs in the `openshift-storage` namespace and examine the resources created by the web console.

5.1. Monitor the progress of the storage cluster and pods in the `openshift-storage` namespace. It takes approximately five minutes before all of the resources are ready.

```
[student@workstation ~]$ watch oc get storagecluster,pods -n openshift-storage
```

Note
Leave the `watch` command running in the terminal window while you proceed with the rest of the exercise.

5.2. While you wait for the operator to be ready, open a second terminal window so that you can view resources created by the web console.

Display the operator groups, subscriptions, and cluster service versions in the `openshift-local-storage` namespace.

- The web console created the `openshift-local-storage` namespace and an operator group for the operator.
- The operator group selects target namespaces in which to generate required roles and role bindings for the operator. The name of the operator group that is displayed in your classroom environment might be different than the one displayed here.
- The subscription resource provides details on how and from where the operator is installed. A successfully created subscription generates a cluster service version resource with details about the specific version of the installed operator. The version that is displayed in your classroom environment might be different than the one displayed here.

```
[student@workstation ~]$ oc get \
  operatorgroups,subscriptions,clusterserviceversions \
  -n openshift-local-storage
NAME                                          AGE
operatorgroup/openshift-local-storage-vzf74   18m

NAME                                    PACKAGE                  SOURCE
 CHANNEL
subscription/local-storage-operator   local-storage-operator   redhat-operators
 4.7

NAME                                             DISPLAY       VERSION  PHASE
clusterserviceversion/local-storage-operator... Local Storage 4.7.0... Succeeded
```

5.3. Display the operator groups, subscriptions, and cluster service versions in the `openshift-storage` namespace.

```
[student@workstation ~]$ watch oc get \
  operatorgroups,subscriptions,clusterserviceversions \
  -n openshift-storage
NAME                                    AGE
operatorgroup/openshift-storage-2dcr9   15m

NAME                          PACKAGE        SOURCE             CHANNEL
subscription/ocs-operator   ocs-operator   redhat-operators   stable-4.7

NAME                                    DISPLAY                       VERSION
 PHASE
clusterserviceversion/ocs-operator...   OpenShift Container Storage   4.7.2
  Succeeded
```

Press `Ctrl+C` to end the `watch` command after the cluster service version displays **Succeeded** in the PHASE column.

5.4. When you created the storage cluster, the web console created several resources, including the local volume discovery resource, which then created a local volume result resource for each targeted node.

List the resources local volume discovery and local volume discovery result in the `openshift-local-storage` namespace.

```
[student@workstation ~]$ oc get \
  localvolumediscovery,localvolumediscoveryresults \
  -n openshift-local-storage
NAME                                          AGE
localvolumediscovery/auto-discover-devices    12m

NAME                                                    AGE
localvolumediscoveryresult/discovery-result-worker01    11m
localvolumediscoveryresult/discovery-result-worker02    11m
localvolumediscoveryresult/discovery-result-worker03    11m
```

5.5. You also created a local volume set resource in the `openshift-local-storage` namespace, and a storage cluster resource in the `openshift-storage` namespace.

Display the local volume set resource.

```
[student@workstation ~]$ oc get localvolumeset -n openshift-local-storage
NAME            STORAGECLASS    PROVISIONED   AGE
lso-volumeset   lso-volumeset   6             10m
```

Display the storage cluster resource.

```
[student@workstation ~]$ oc get storagecluster -n openshift-storage
NAME                 AGE   PHASE   EXTERNAL   CREATED AT             VERSION
ocs-storagecluster   5m    Ready              2021-07-28T01:25:17Z   4.7.0
```

5.6. List the storage classes available in the cluster, and then close the terminal window.

- The `nfs-storage` storage class is part of your classroom configuration, and it existed before this exercise.
- The `lso-volumeset` storage class was created by the local volume set resource.
- The OCS operator created the remaining storage classes.

```
[student@workstation ~]$ oc get storageclasses -o \
  custom-columns='NAME:metadata.name,PROVISIONER:provisioner'
NAME                          PROVISIONER
lso-volumeset                 kubernetes.io/no-provisioner
nfs-storage (default)         nfs-storage-provisioner
ocs-storagecluster-ceph-rbd   openshift-storage.rbd.csi.ceph.com
ocs-storagecluster-ceph-rgw   openshift-storage.ceph.rook.io/bucket
ocs-storagecluster-cephfs     openshift-storage.cephfs.csi.ceph.com
openshift-storage.noobaa.io   openshift-storage.noobaa.io/obc
```

5.7. Return to the first terminal window and verify that the installation is complete.

```
Every 2.0s: oc get storagecluster,pods -n openshift-storage   ...

NAME                 AGE   PHASE   EXTERNAL   CREATED AT             VERSION
ocs-storagecluster   5m    Ready              2021-07-28T01:25:17Z   4.7.0
...output omitted...
```

Press `Ctrl+C` to end the `watch` command after the `ocs-storagecluster` storage cluster displays a status of `Ready` in the PHASE column.

Important

All the exercises in this course depend upon your cluster being correctly configured to use OpenShift Data Foundation.

If you were unable to successfully complete this exercise, then delete and recreate your lab environment. After recreating your lab environment, you can either complete this exercise again or you can let the start script for another exercise configure the OpenShift cluster for you.

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish internal-gui
```

This concludes the guided exercise.

Deploying Red Hat OpenShift Data Foundation from the Command Line

Objectives

After completing this section, you should be able to deploy Red Hat OpenShift Data Foundation on premises using the CLI.

Deploying OpenShift Data Foundation from the Command Line Interface

For automation or infrastructure as code (IaC) purposes, administrators can install OpenShift Data Foundation from the command line by using the `oc` or `kubectl` tools.

As explained elsewhere in this course, the installation of OpenShift Data Foundation consists of installing the OCS operator and creating the storage cluster object. Additionally, if local volumes are attached to the nodes, installing the LSO operator is required.

The Kubernetes operator model allows administrators to install an operator in RHOCP by creating the corresponding operator group and subscription.

The following are examples of the operator group and subscription objects, in YAML format, for you to create in a pre-existing `openshift-local-storage` namespace:

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: openshift-local-storage
  namespace: openshift-local-storage
spec:
  targetNamespaces:
    - openshift-local-storage
```

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: local-storage-operator
  namespace: openshift-local-storage
spec:
  channel: "4.7"
  name: local-storage-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
```

Installing the Local Storage Operator

After you create the operator group and subscription objects, you must create the following custom resources to complete the installation of the LSO:

- The `LocalVolumeDiscovery`, to discover available attached volumes
- The `LocalVolumeSet`, to define the volume mode (file or block)

Examples of the the local volume discovery and local volume set objects in YAML format follow.

```
apiVersion: local.storage.openshift.io/v1alpha1
kind: LocalVolumeDiscovery
metadata:
  name: auto-discover-devices
  namespace: openshift-local-storage
spec
  nodeSelector:
    nodeSelectorTerms:
    - matchExpressions:
      - key: kubernetes.io/hostname
        operator: In
        values
        - worker01
        - worker02
        - worker03
```

The `LocalVolumeDiscovery` CR, `LocalVolumeDiscoveryResults` objects become available within a few seconds. You can retrieve the `LocalVolumeDiscoveryResults` objects with the following command:

```
[user@demo ~]$ oc get localvolumediscoveryresults -n openshift-local-storage
NAME                        AGE
discovery-result-worker01   3s
discovery-result-worker02   2s
discovery-result-worker03   4s
```

When the discovery finishes, you can create a `LocalVolumeSet` by using the devices discovered by the `LocalVolumeDiscovery`.

```
apiVersion: local.storage.openshift.io/v1alpha1
kind: LocalVolumeSet
metadata:
  name: lso-volumeset
  namespace: openshift-local-storage
spec:
  deviceInclusionSpec:
    deviceTypes:
    - disk
    - part
    minSize: 1Gi
  nodeSelector:
    nodeSelectorTerms:
    - matchExpressions:
      - key: kubernetes.io/hostname
        operator: In
        values:
        - worker01
        - worker02
        - worker03
```

```
    storageClassName: lso-volumeset
    maxDeviceCount: 1
    volumeMode: Block
```

The preceding example uses only one device from each worker node to create the local volume set.

Installing the Red Hat OpenShift Container Storage operator

To install the Red Hat OpenShift Container Storage operator, first create the corresponding labels, operator group, and subscription.

Assuming that the namespace name is the default (`openshift-storage`), label it so that it uses the available cluster monitoring solution.

```
[user@demo ~]$ oc label namespace/openshift-storage \
  openshift.io/cluster-monitoring=
```

As mentioned previously, first you must create the operator group and subscription objects, as shown in the following examples.

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: openshift-storage
  namespace: openshift-storage
spec:
  targetNamespaces:
    - openshift-storage
```

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ocs-operator
  namespace: openshift-storage
spec:
  channel: "stable-4.7"
  name: ocs-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
```

After creating the operator group and subscription objects, create the storage cluster. Assuming an installation in internal mode, the following YAML example creates the `StorageCluster` CR.

```
apiVersion: ocs.openshift.io/v1
kind: StorageCluster
metadata:
  name: ocs-storagecluster
  namespace: openshift-storage
spec:
  monDataDirHostPath: /var/lib/rook
  storageDeviceSets:
  - count: 1
    dataPVCTemplate:
```

```
      spec:
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: "1"
        storageClassName: lso-volumeset
        volumeMode: Block
    name: ocs-deviceset-lso-volumeset
    replica: 3
  version: 4.7.0
```

Verifying the Red Hat OpenShift Data Foundation Installation

After the installation is complete, administrators can verify the status of all the objects created by the LSO and OpenShift Data Foundation operators by inspecting each namespace.

The following example is for the LSO operator and assumes the default namespace `openshift-local-storage`.

```
[user@demo ~]$ oc get all -n openshift-local-storage
NAME                                          READY   STATUS    ...
pod/diskmaker-discovery-9ddqx                 1/1     Running   ...
pod/diskmaker-discovery-wlpqj                 1/1     Running   ...
pod/diskmaker-discovery-zhkwk                 1/1     Running   ...
pod/diskmaker-manager-5zx5n                   1/1     Running   ...
pod/diskmaker-manager-gsmbr                   1/1     Running   ...
pod/diskmaker-manager-lwc7n                   1/1     Running   ...
pod/local-storage-operator-687c6d5446-sdlrx   1/1     Running   ...

NAME                                     TYPE        CLUSTER-IP     ...
service/local-storage-operator-metrics   ClusterIP   172.30.125.202 ...

NAME                                 DESIRED   CURRENT   READY   ...
daemonset.apps/diskmaker-discovery   3         3         3       ...
daemonset.apps/diskmaker-manager     3         3         3       ...

NAME                                     READY   UP-TO-DATE   ...
deployment.apps/local-storage-operator   1/1     1            ...

NAME                                                ...     READY  ...
replicaset.apps/local-storage-operator-687c6d5446   ...     1      ...
```

Verify the OpenShift Data Foundation operator status, assuming the default namespace `openshift-storage` and three worker nodes that use local storage.

```
[user@demo ~]$ oc get all -n openshift-storage
NAME                                                  READY   STATUS    ...
pod/csi-cephfsplugin-k59p6                            3/3     Running   ...
pod/csi-cephfsplugin-mlrr8                            3/3     Running   ...
pod/csi-cephfsplugin-provisioner-584644bb58-lz6wd     6/6     Running   ...
pod/csi-cephfsplugin-provisioner-584644bb58-sgm7k     6/6     Running   ...
```

```
pod/csi-cephfsplugin-r8x6z                          3/3     Running   ...
pod/csi-rbdplugin-5m888                             3/3     Running   ...
pod/csi-rbdplugin-ksvh6                             3/3     Running   ...
pod/csi-rbdplugin-provisioner-8667bc7885-8qs4k      6/6     Running   ...
pod/csi-rbdplugin-provisioner-8667bc7885-jzjt5      6/6     Running   ...
pod/csi-rbdplugin-tn6h6                             3/3     Running   ...
pod/noobaa-operator-76bddd78d8-lj5ls                1/1     Running   ...
pod/ocs-metrics-exporter-64fbffb564-7k4vn           1/1     Running   ...
pod/ocs-operator-7644c6f744-rxdxg                   1/1     Running   ...
pod/rook-ceph-operator-585565c4c7-8bl4k             1/1     Running   ...

NAME                                TYPE        CLUSTER-IP      ...
service/csi-cephfsplugin-metrics    ClusterIP   172.30.175.76   ...
service/csi-rbdplugin-metrics       ClusterIP   172.30.19.239   ...

NAME                                DESIRED   CURRENT   READY   ...
daemonset.apps/csi-cephfsplugin     3         3         3       ...
daemonset.apps/csi-rbdplugin        3         3         3       ...

NAME                                          READY   ...
deployment.apps/csi-cephfsplugin-provisioner  2/2     ...
deployment.apps/csi-rbdplugin-provisioner     2/2     ...
deployment.apps/noobaa-operator               1/1     ...
deployment.apps/ocs-metrics-exporter          1/1     ...
deployment.apps/ocs-operator                  1/1     ...
deployment.apps/rook-ceph-operator            1/1     ...

NAME                                                    ...   READY
replicaset.apps/csi-cephfsplugin-provisioner-584644bb58 ...   2
replicaset.apps/csi-rbdplugin-provisioner-8667bc7885    ...   2
replicaset.apps/noobaa-operator-76bddd78d8              ...   1
replicaset.apps/ocs-metrics-exporter-64fbffb564         ...   1
replicaset.apps/ocs-operator-7644c6f744                 ...   1
replicaset.apps/rook-ceph-operator-585565c4c7           ...   1
```

There is extensive information about the status of the cluster data services that can not be easily retrieved via the command-line interface.

Thus, Red Hat recommends verifying the status of the OpenShift Data Foundation installation by using the RHOCP web console.

References

For more information, refer to the product documentation for *Red Hat OpenShift Data Foundation* at
https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/

Guided Exercise

Deploying Red Hat OpenShift Data Foundation from the Command Line

In this exercise, you will configure your OpenShift cluster to use OpenShift Data Foundation.

Outcomes

You should be able to:

- Label nodes to use as storage nodes.
- Install the operator Local Storage Operator.
- Install the OpenShift Container Storage operator.
- Verify the successful installation of OpenShift Data Foundation.

Before You Begin

To perform this exercise, ensure that you have access to a running OpenShift cluster.

Important

If you already completed this exercise or if you completed the *Guided Exercise: Deploying Red Hat OpenShift Data Foundation from the Web Console*, then OpenShift Data Foundation is already installed.

To perform this exercise again, delete and then recreate your lab environment.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable.

```
[student@workstation ~]$ lab start internal-cli
```

Instructions

1. Label the `worker` nodes.

   1.1. Log in to your OpenShift cluster as the `admin` user with the `redhat` password.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  https://api.ocp4.example.com:6443
```

   1.2. List the worker nodes in the cluster.

```
[student@workstation ~]$ oc get nodes -l node-role.kubernetes.io/worker=
NAME       STATUS   ROLES    AGE   VERSION
worker01   Ready    worker   14d   v1.20.0+a0b09eb
worker02   Ready    worker   14d   v1.20.0+a0b09eb
worker03   Ready    worker   14d   v1.20.0+a0b09eb
```

1.3. Add the `cluster.ocs.openshift.io/openshift-storage=` label to the three worker nodes.

```
[student@workstation ~]$ oc label nodes \
  -l node-role.kubernetes.io/worker= \
  cluster.ocs.openshift.io/openshift-storage=
node/worker01 labeled
node/worker02 labeled
node/worker03 labeled
```

2. Install and configure the Local Storage Operator.

2.1. Change to the `~/DO370/labs/internal-cli` directory.

```
[student@workstation ~]$ cd ~/DO370/labs/internal-cli
```

2.2. List the disks available in one of the worker nodes.

- The disk `/dev/vda` has the operating system installed.
- The disks `/dev/vdb` and `/dev/vdc` are available.

```
[student@workstation internal-cli]$ oc debug node/worker01 -- \
  lsblk --paths --nodeps
Starting pod/worker01-debug ...
To use host binaries, run chroot /host
NAME     MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
/dev/vda 252:0    0   80G  0 disk
/dev/vdb 252:16   0  500G  0 disk
/dev/vdc 252:32   0  500G  0 disk

Removing debug pod ...
```

2.3. Create the `openshift-local-storage` namespace and switch to it.

```
[student@workstation internal-cli]$ oc adm new-project openshift-local-storage
Created project openshift-local-storage

[student@workstation internal-cli]$ oc project openshift-local-storage
Now using project "openshift-local-storage" on server "https://
api.ocp4.example.com:6443".
```

2.4. Edit the `lso-operatorgroup.yml` file and modify the placeholder text to match the following content.

```
---
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: openshift-local-storage
  namespace: openshift-local-storage
spec:
  targetNamespaces:
    - openshift-local-storage
```

Note

The `~/DO370/solutions/internal-cli/lso-operatorgroup.yml` file contains the correct configuration and can be used for comparison.

An operator group resource template is available at https://access.redhat.com/documentation/en-us/openshift_container_platform/4.7/html-single/operators/index#olm-understanding-operatorgroups.

2.5. Create the `openshift-local-storage` operator group.

```
[student@workstation internal-cli]$ oc apply -f lso-operatorgroup.yml
operatorgroup.operators.coreos.com/openshift-local-storage created
```

2.6. Edit the `lso-subscription.yml` file and modify the placeholder text to match the following content.

```
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: local-storage-operator
  namespace: openshift-local-storage
spec:
  channel: "4.7"
  installPlanApproval: Automatic
  name: local-storage-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
```

Note

The `~/DO370/solutions/internal-cli/lso-subscription.yml` file contains the correct configuration and can be used for comparison.

A subscription resource template is available at https://access.redhat.com/documentation/en-us/openshift_container_platform/4.7/html-single/operators/index#olm-understanding-olm.

2.7. Create the `local-storage-operator` subscription.

```
[student@workstation internal-cli]$ oc apply -f lso-subscription.yml
subscription.operators.coreos.com/local-storage-operator created
```

2.8. Wait until the installation of the subscription and the cluster service version is complete.

```
[student@workstation internal-cli]$ watch oc get -n openshift-local-storage \
  operatorgroups,subscriptions,clusterserviceversions
NAME                                      AGE
operatorgroup/openshift-local-storage     5m

NAME                                    PACKAGE                  SOURCE
 CHANNEL
subscription/local-storage-operator     local-storage-operator   redhat-operators
 4.7

NAME                                                  DISPLAY          VERSION
  PHASE
clusterserviceversion/local-storage-operator...     Local Storage    4.7.0...
  Succeeded
```

Press `Ctrl+C` to end the `watch` command after the cluster service version displays `Succeeded` in the PHASE column.

2.9. Edit the `localvolumediscovery.yml` file and modify the placeholder text to match the following content.

```
---
apiVersion: local.storage.openshift.io/v1alpha1
kind: LocalVolumeDiscovery
metadata:
  name: auto-discover-devices
  namespace: openshift-local-storage
spec:
  nodeSelector:
    nodeSelectorTerms:
    - matchExpressions:
      - key: kubernetes.io/hostname
        operator: In
        values:
        - worker01
        - worker02
        - worker03
```

Note

The `~/DO370/solutions/internal-cli/localvolumediscovery.yml` file contains the correct configuration and can be used for comparison.

2.10. Create the local volume discovery resource.

```
[student@workstation internal-cli]$ oc apply -f localvolumediscovery.yml
localvolumediscovery.local.storage.openshift.io/auto-discover-devices created
```

2.11. List the local volume discovery results. There is a discovery result for each worker node.

```
[student@workstation internal-cli]$ oc get localvolumediscoveryresults
NAME                        AGE
discovery-result-worker01   4s
discovery-result-worker02   5s
discovery-result-worker03   6s
```

Note

You might need to run the command multiple times until the desired condition is reached. It can take a few seconds before the results are generated.

2.12. Verify that the `/dev/vdb` and `/dev/vdc` disks were discovered on all worker nodes.

```
[student@workstation internal-cli]$ oc get localvolumediscoveryresults \
  -o custom-columns='NAME:metadata.name,PATH:status.discoveredDevices[*].path'
NAME                        PATH
discovery-result-worker01   /dev/vda1,/dev/vda2,...,/dev/vdb,/dev/vdc
discovery-result-worker02   /dev/vda1,/dev/vda2,...,/dev/vdb,/dev/vdc
discovery-result-worker03   /dev/vda1,/dev/vda2,...,/dev/vdb,/dev/vdc
```

Note

This lab limits the number of disk devices that are consumed by OpenShift Data Foundation. You use the remaining disks to increase the storage in another exercise.

2.13. Create the local volume set.

- Include only one of the discovered devices on each of the worker nodes.
- The devices do not need to match a minimum size to be included.
- Specify a storage class name of `lso-volumeset` and use the block volume mode.

Edit the `localvolumeset.yml` file and modify the placeholder text to match the following content.

```
---
apiVersion: local.storage.openshift.io/v1alpha1
kind: LocalVolumeSet
metadata:
  name: lso-volumeset
  namespace: openshift-local-storage
spec:
  deviceInclusionSpec:
    deviceTypes:
    - disk
```

```
    - part
    minSize: 0Ti
  nodeSelector:
    nodeSelectorTerms:
    - matchExpressions:
      - key: kubernetes.io/hostname
        operator: In
        values:
        - worker01
        - worker02
        - worker03
  storageClassName: lso-volumeset
  volumeMode: Block
  maxDeviceCount: 1
```

Note

Notice that the `maxDeviceCount` attribute should be set to `1` so only one device per node is added to the storage cluster.

The `~/DO370/solutions/internal-cli/localvolumeset.yml` file contains the correct configuration and can be used for comparison.

2.14. Create the local volume set resource.

```
[student@workstation internal-cli]$ oc apply -f localvolumeset.yml
localvolumeset.local.storage.openshift.io/lso-volumeset created
```

2.15. Verify that the local volume set has three provisioned devices.

```
[student@workstation internal-cli]$ oc get localvolumesets/lso-volumeset
NAME            STORAGECLASS    PROVISIONED   AGE
lso-volumeset   lso-volumeset   3             2m
```

Note

This lab limits the number of disk devices that are consumed by OpenShift Data Foundation. You use the remaining disks to increase the storage in another exercise.

3. Install and configure the OpenShift Container Storage operator.

3.1. Create the `openshift-storage` namespace and switch to it.

```
[student@workstation internal-cli]$ oc adm new-project openshift-storage
Created project openshift-storage

[student@workstation internal-cli]$ oc project openshift-storage
Now using project "openshift-storage" on server "https://
api.ocp4.example.com:6443".
```

3.2. Edit the `ocs-operatorgroup.yml` file and modify the placeholder text to match the following content.

```
---
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: openshift-storage
  namespace: openshift-storage
spec:
  targetNamespaces:
    - openshift-storage
```

Note

The `~/DO370/solutions/internal-cli/ocs-operatorgroup.yml` file contains the correct configuration and can be used for comparison.

An operator group resource template is available at https://access.redhat.com/documentation/en-us/openshift_container_platform/4.7/html-single/operators/index#olm-understanding-operatorgroups.

3.3. Create the `openshift-storage` operator group.

```
[student@workstation internal-cli]$ oc apply -f ocs-operatorgroup.yml
operatorgroup.operators.coreos.com/openshift-storage created
```

3.4. Edit the `ocs-subscription.yml` file and modify the placeholder text to match the following content.

```
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: ocs-operator
  namespace: openshift-storage
spec:
  channel: "stable-4.7"
  installPlanApproval: Automatic
  name: ocs-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
```

Note

The `~/DO370/solutions/internal-cli/ocs-subscription.yml` file contains the correct configuration and can be used for comparison.

A subscription resource template is available at https://access.redhat.com/documentation/en-us/openshift_container_platform/4.7/html-single/operators/index#olm-understanding-olm.

3.5. Create the `ocs-operator` subscription.

```
[student@workstation internal-cli]$ oc apply -f ocs-subscription.yml
subscription.operators.coreos.com/ocs-operator created
```

3.6. Wait until the installation of the subscription and the cluster service version is complete.

```
[student@workstation internal-cli]$ watch oc get -n openshift-storage \
  operatorgroups,subscriptions,clusterserviceversions
NAME                              AGE
operatorgroup/openshift-storage   71s

NAME                         PACKAGE        SOURCE             CHANNEL
subscription/ocs-operator    ocs-operator   redhat-operators   stable-4.7

NAME                                     DISPLAY                       VERSION
  PHASE
clusterserviceversion/ocs-operator... OpenShift Container Storage   4.7.2
  Succeeded
```

Press `Ctrl+C` to end the `watch` command after the cluster service version displays `Succeeded` in the PHASE column.

3.7. Edit the `storagecluster.yml` file and modify the placeholder text to match the following content.

```
---
apiVersion: ocs.openshift.io/v1
kind: StorageCluster
metadata:
  name: ocs-storagecluster
  namespace: openshift-storage
spec:
  monDataDirHostPath: /var/lib/rook
  storageDeviceSets:
  - count: 1
    dataPVCTemplate:
      spec:
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: "1"
        storageClassName: lso-volumeset
        volumeMode: Block
    name: ocs-deviceset-lso-volumeset
    replica: 3
  version: 4.7.0
```

Note

The `~/DO370/solutions/internal-cli/storagecluster.yml` file contains the correct configuration and can be used for comparison.

3.8. Create the `ocs-storagecluster` storage cluster.

```
[student@workstation internal-cli]$ oc apply -f storagecluster.yml
storagecluster.ocs.openshift.io/ocs-storagecluster created
```

3.9. Wait until the installation and configuration of the OpenShift Container Storage operator is complete.

```
[student@workstation internal-cli]$ watch oc get -n openshift-storage \
  storagecluster,pods
NAME                                 AGE     PHASE   EXTERNAL   CREATED AT
    VERSION
storagecluster/ocs-storagecluster   5m      Ready
 2021-07-29T02:12:29Z   4.7.0

...output omitted...
```

Press `Ctrl+C` to end the `watch` command after the `ocs-storagecluster` storage cluster displays a status of `Ready` in the PHASE column. Be patient because it takes approximately five minutes until the storage cluster is ready.

Important

All other exercises in this course depend upon your cluster being correctly configured to use OpenShift Data Foundation.

If you were unable to complete this exercise, then delete and recreate your lab environment. After recreating your lab environment, you can either attempt this exercise again or you can let the start script for another exercise configure the OpenShift cluster for you.

3.10. Switch to the `default` project.

```
[student@workstation internal-cli]$ oc project default
Now using project "default" on server "https://api.ocp4.example.com:6443".
```

3.11. Change to the `/home/student` directory.

```
[student@workstation internal-cli]$ cd
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish internal-cli
```

This concludes the guided exercise.

Summary

In this chapter, you learned about:

- The architecture and components of Red Hat OpenShift Data Foundation.
- The storage management features provided by OpenShift Data Foundation.
- The available installation methods for OpenShift Data Foundation, including both the CLI and the web console.

Chapter 2

Configuring OpenShift Cluster Services to Use Red Hat OpenShift Data Foundation

Goal

Configure OpenShift Monitoring, Registry, and Logging to use storage provided by OpenShift Data Foundation.

Objectives

- Identify and describe Red Hat OpenShift Data Foundation storage classes
- Configure the internal registry to use storage provided by Red Hat OpenShift Data Foundation
- Configure the OpenShift monitoring subsystem to use block storage from Red Hat OpenShift Data Foundation

Sections

- Introducing Red Hat OpenShift Data Foundation Storage Classes (and Quiz)
- Configuring the Internal Image Registry to Use Red Hat OpenShift Data Foundation (and Guided Exercise)
- Configuring Monitoring to Use Red Hat OpenShift Data Foundation (and Guided Exercise)

Lab

Configuring OpenShift Cluster Services to Use Red Hat OpenShift Data Foundation

Introducing Red Hat OpenShift Data Foundation Storage Classes

Objectives

After completing this section, you should be able to identify and describe Red Hat OpenShift Data Foundation storage classes

Reviewing Kubernetes Storage Concepts

Containerized applications are usually classified as stateless or stateful, depending on whether the data they produce must be preserved.

Ephemeral Storage for Stateless Applications

By default, containers use ephemeral storage. This means that no data persists when the container is terminated.

Ephemeral storage is valid for stateless workloads, such as web servers (Apache Tomcat, Nginx), load balancers (HAProxy, Envoy), serverless applications, and APIs. In stateless workloads, the application does not produce relevant or unique data to be stored as an output.

For example, an HTTP request sent to a web server consistently returns a web page. Similarly, a request to a serverless application consistently returns the output of the same function. In these cases, there is no need to store data.

Persistent Storage for Stateful Applications

When the data produced by a workload must be preserved, is not reproducible, or it is needed to produce later outputs, you must attach persistent storage to your Kubernetes deployments.

For example, banking applications must preserve the traceability of the transactions performed during a user session. If the pod where the application is running is terminated, the data must be retained.

Also, any workload using a database to store data might need to preserve it. For example, an application controlling the stock of a warehouse usually requires a database, which is updated with every transaction. It is vital to preserve the data stored in those databases.

If it is necessary to persist any data produced by an application running on Kubernetes or Red Hat OpenShift Container Platform, administrators must configure their deployments to use a persistent volume (PV) and persistent volume claim (PVC).

Persistent Volumes and Persistent Volume Claims

Administrators can configure their deployments to use PVs when persistent data is a requisite of their applications.

PVs are Kubernetes resources that abstract the users from the details of how the storage is provisioned. To a user who configures a deployment to use a PV, it is irrelevant if an administrator provisions the storage manually or it is dynamically provisioned by a storage class.

PVCs are Kubernetes objects to request storage PVs. A PVC can include specific details such as the size and the access mode (ReadWriteOnce, ReadOnlyMany or ReadWriteMany). For

more information on PVs and PVCs, review the Kubernetes documentation about PVs [https://kubernetes.io/docs/concepts/storage/persistent-volumes/].

Storage Classes

As its name suggests, a storage class resource in Kubernetes describes the storage characteristics, such as quality of service, throughput, or storage technology provided by the backing data services. Kubernetes defines storage classes to provide various cluster storage types addressing different application requirements. Applications consume these storage classes as PVs defined by PVCs.

Each storage class makes use of a storage provisioner that determines the volume plug-in provisioning the PVs. Different storage technologies, such as Ceph File System or Amazon Elastic Block Storage, use different volume provisioners.

The `StorageClass` Kubernetes resource also contains the fields to configure the reclaim policy (to retain, recycle or delete the released volumes), allow volume expansion, and others.

Storage Classes Provided by Red Hat OpenShift Data Foundation

Red Hat OpenShift Data Foundation configures the following storage classes to provide data services that support different storage technologies.

- `ocs-storagecluster-ceph-rbd`. Installed by the Rook-Ceph operator, this class supports block storage devices, primarily for database workloads. Examples include the Red Hat OpenShift Container Platform (OCP) logging and monitoring stack and PostgreSQL.

- `ocs-storagecluster-cephfs`. Installed by the Rook-Ceph operator, this class provides shared and distributed file system data services, primarily used for software development, messaging, and data aggregation workloads. Examples include Jenkins build sources and artifacts, WordPress uploaded content, the RHOCP registry, and messaging using JBoss AMQ.

- `openshift-storage.noobaa.io`. Installed by the NooBaa operator, this class provides the Multicloud Object Gateway (MCG) service. Ultimately, the MCG service provides multicloud object storage as an S3 API endpoint that can abstract the storage and retrieval of data from multiple cloud object stores.

- `ocs-storagecluster-ceph-rgw`. Installed by the Rook-Ceph operator, this class provides on-premises object storage. The Ceph RADOS Gateway (Ceph RGW) object storage interface is a robust S3 API endpoint that scales to tens of petabytes and billions of objects, primarily targeting data-intensive applications. Examples include the storage and access of row, columnar, and semi-structured data with applications like Spark, Presto, Red Hat AMQ Streams (Kafka), and machine learning frameworks like TensorFlow and Pytorch.

Storage Needs of OpenShift Cluster Services

In addition to managing storage for application workloads, OpenShift Data Foundation also satisfies the three main storage needs of production OpenShift and Kubernetes clusters.

These needs are:

- Persisting data from the monitoring stack

- Preserving data from the logging stack

- Providing the OpenShift internal image registery

Persisting data from the monitoring stack is useful for root cause analysis, big data, cost management, or load statistics. OpenShift Data Foundation provides block storage services by using Ceph RADOS Block Device storage (Ceph RBD). Preserving the logging stack data is useful for traceability, troubleshooting, and root cause analysis. Providing an internal registry enables access to images and image workflows using a Red Hat Quay registry, or another container registry, through the Ceph object storage gateway (Ceph RGW).

OpenShift Data Foundation provides file storage services by using Ceph File System storage (Ceph FS).

References

Stateful vs stateless applications
https://www.redhat.com/en/topics/cloud-native-apps/stateful-vs-stateless

For more information about storage classes, review the Kubernetes documentation at
https://kubernetes.io/docs/concepts/storage/storage-classes/
.

For more information, refer to the *Red Hat OpenShift Data Foundation Planning Your Deployment Guide* at
https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/html-single/planning_your_deployment/index#introduction-to-openshift-container-storage-4_rhocs

Quiz

Introducing Red Hat OpenShift Data Foundation Storage Classes

Match the items below to their counterparts in the table.

This class provides multicloud object storage through the MCG.

This class provides on-premises object storage.

This class provides shared and distributed file system data services, primarily used for software development, messaging, and data aggregation workloads.

This class supports block storage devices, primarily for database workloads.

OpenShift Data Foundation Storage Classes	Description
`ocs-storagecluster-ceph-rbd`	
`ocs-storagecluster-cephfs`	
`openshift-storage.noobaa.io`	
`ocs-storagecluster-ceph-rgw`	

Solution

Introducing Red Hat OpenShift Data Foundation Storage Classes

Match the items below to their counterparts in the table.

OpenShift Data Foundation Storage Classes	Description
`ocs-storagecluster-ceph-rbd`	This class supports block storage devices, primarily for database workloads.
`ocs-storagecluster-cephfs`	This class provides shared and distributed file system data services, primarily used for software development, messaging, and data aggregation workloads.
`openshift-storage.noobaa.io`	This class provides multicloud object storage through the MCG.
`ocs-storagecluster-ceph-rgw`	This class provides on-premises object storage.

Configuring the Internal Image Registry to Use Red Hat OpenShift Data Foundation

Objectives

After completing this section, you should be able to configure the internal registry to use storage provided by Red Hat OpenShift Data Foundation

Explaining the Persistent Storage Used by the RHOCP Internal Registry

Red Hat OpenShift Container Platform provides a built-in container image registry to host the images of the workloads running in the cluster. The RHOCP internal registry is managed by its operator and is integrated into the cluster authentication and authorization systems.

Container image data is comprised of two parts:

1. The actual image data that must be stored in a configurable storage location.

2. The image metadata that is stored in the etcd database as standard Kubernetes resources (images and image streams).

You must configure the etcd storage hosting the image metadata when provisioning the cluster. You must also configure the persistent storage used by the internal RHOCP image registry to store and build container images, and to make them available for consumption within the cluster.

Configuring the Persistent Storage for the RHOCP Internal Registry

As mentioned previously, the RHOCP internal registry is managed by its own infrastructure operator. The `configs.imageregistry.operator.openshift.io` resource controls the configuration and desired status of the container image registry.

To configure the storage for the registry, the `configs.imageregistry.operator.openshift.io` resource offers the `storage` parameter. For example, use the following parameters to enable S3 storage:

```
storage:
  s3:
    bucket: <bucket-name>
    region: <region-name>
    regionEndpoint: <region-endpoint-name>
```

Note

These parameters only apply if the `spec.storage.managementState` parameter of the `configs.imageregistry.operator.openshift.io` resource is set to `Managed`.

More information on S3 configuration parameters is available in the section references.

In addition, the `image-registry-private-configuration-user` secret is available in the `openshift-image-registry namespace`. This secret is used to configure storage access and management. The credentials in this secret override the default credentials used by the image registry operator, if configured.

To configure the credentials for S3 storage access, the `image-registry-private-configuration-user` secret must contain two keys:

- `REGISTRY_STORAGE_S3_ACCESSKEY`
- `REGISTRY_STORAGE_S3_SECRETKEY`

The following command creates the `image-registry-private-configuration-user` secret:

```
[user@demo ~]$ oc create secret generic \
  image-registry-private-configuration-user \
  --from-literal=REGISTRY_STORAGE_S3_ACCESSKEY=myaccesskey \
  --from-literal=REGISTRY_STORAGE_S3_SECRETKEY=mysecretkey \
  --namespace openshift-image-registry
```

Note
Kubernetes secrets are base64 encoded, not encrypted.

Choosing the Appropriate Persistent Storage for the RHOCP Internal Registry

You can configure the RHOCP internal registry to use block, file, or object storage.

The different choices for backing the storage of the internal registry mostly depend on if the registry is scaled (configured for high availability) or not.

The following table summarizes the recommended and configurable storage options for the registry:

Storage type	ReadOnlyMany	ReadWriteMany	Registry	Scaled Registry
Block	Yes (1)	No	Configurable	Not configurable
File	Yes (1)	Yes	Configurable	Configurable
Object	Yes	Yes	**Recommended**	**Recommended**

(1) Does not apply to physical disk, virtual machine physical disk, VMware VMDK, loopback over NFS, AWS EBS, or Azure Disk.

Note

Tests have shown issues with using the NFS server on Red Hat Enterprise Linux (RHEL) as the storage back end for RHOCP core services, such as the container image registry. Therefore, Red Hat does not recommend using NFS server in production environments. Other NFS implementations on the market might not have these issues.

References

For the complete list of S3 configuration parameters, refer to the *Image Registry Operator Configuration Parameters for AWS S3* section of the *Red Hat OpenShift Container Platform Registry Guide* at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.7/html-single/registry/index

For detailed instructions on how to configure persistent storage for the RHOCP internal image registry, refer to the *Red Hat OpenShift Container Platform Registry Guide* at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.7/html-single/registry/index

For more information, refer to the *Recommended Configurable Storage Technology* section of the *Red Hat OpenShift Container Platform 4.7 Scalability and Performance Guide* at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.7/html-single/scalability_and_performance/index#recommended-configurable-storage-technology_persistent-storage

For more information, refer to the product documentation for *Red Hat OpenShift Data Foundation* at
https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/

For more information, refer to the *Red Hat OpenShift Container Platform 4.7 Scalability and Performance Guide* at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.7/html-single/scalability_and_performance/index

Guided Exercise

Configuring the Internal Image Registry to Use Red Hat OpenShift Data Foundation

In this exercise, you will configure the internal registry to use object storage provided by Red Hat OpenShift Data Foundation.

Outcomes

You should be able to:

- Create a new object bucket claim (OBC) provided by OpenShift Data Foundation.
- Configure the internal image registry to use the new OBC.
- Verify the persistence of the internal image registry storage.

Before You Begin

To perform this exercise, ensure that you have access to:

- A running OpenShift cluster.
- Installed and configured operators for OpenShift Container Storage (OCS) and the Local Storage Operator (LSO).

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable and that both OCS and the LSO operators are installed and configured correctly. The command also configures the cluster to use trusted certificates, if necessary.

```
[student@workstation ~]$ lab start services-registry
```

Note
It might take up to ten minutes to start the lab in a new classroom environment.

Instructions

1. Identify the storage classes available to the cluster.

   1.1. Log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

1.2. List the available storage classes. In this exercise, you create a new OBC that uses the `openshift-storage.noobaa.io` storage class.

```
[student@workstation ~]$ oc get storageclasses -o name
storageclass.storage.k8s.io/lso-volumeset
storageclass.storage.k8s.io/nfs-storage
storageclass.storage.k8s.io/ocs-storagecluster-ceph-rbd
storageclass.storage.k8s.io/ocs-storagecluster-ceph-rgw
storageclass.storage.k8s.io/ocs-storagecluster-cephfs
storageclass.storage.k8s.io/openshift-storage.noobaa.io
```

2. Create a new OBC named `noobaa-registry` in the `openshift-image-registry` namespace. Use the `openshift-storage.noobaa.io` storage class.

2.1. Change to the `~/DO370/labs/services-registry` directory.

```
[student@workstation ~]$ cd ~/DO370/labs/services-registry
```

2.2. Modify the `obc-registry.yml` file.

- Configure the OBC to use the name `noobaa-registry`.
- Create the OBC in the `openshift-image-registry` namespace.
- Ensure that the bucket name starts with `noobaa-registry` by giving the setting `generateBucketName` a value of `noobaa-registry`.
- Use the `openshift-storage.noobaa.io` storage class.

Ensure that the file matches the following bold lines.

```
---
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: noobaa-registry
  namespace: openshift-image-registry
spec:
  additionalConfig:
    bucketclass: noobaa-default-bucket-class
  generateBucketName: noobaa-registry
  storageClassName: openshift-storage.noobaa.io
```

Note

The `~/DO370/solutions/services-registry/obc-registry.yml` file contains the correct configuration and can be used for comparison.

2.3. Create the new OBC.

```
[student@workstation services-registry]$ oc apply -f obc-registry.yml
objectbucketclaim.objectbucket.io/noobaa-registry created
```

2.4. Verify that the cluster successfully created the `noobaa-registry` OBC. The `noobaa-registry` OBC has a phase of `Bound`.

```
[student@workstation services-registry]$ oc get -n openshift-image-registry \
  objectbucketclaim/noobaa-registry
NAME              STORAGE-CLASS                 PHASE   AGE
noobaa-registry   openshift-storage.noobaa.io   Bound   60s
```

3. Create a new generic secret containing the credentials needed to access and manage the OBC.

3.1. List the secrets in the `openshift-image-registry` namespace. When you created the `noobaa-registry` OBC, a secret named `noobaa-registry` was also created.

```
[student@workstation services-registry]$ oc get secrets -l app=noobaa \
  -n openshift-image-registry
NAME              TYPE     DATA   AGE
noobaa-registry   Opaque   2      1m10s
```

3.2. Extract the `noobaa-registry` secret to your current directory.

```
[student@workstation services-registry]$ oc extract secret/noobaa-registry \
  -n openshift-image-registry
AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY
```

3.3. Use the contents of the `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` files to create the `image-registry-private-configuration-user` secret.

```
[student@workstation services-registry]$ oc create secret generic \
  image-registry-private-configuration-user \
  --from-literal=REGISTRY_STORAGE_S3_ACCESSKEY="$(cat AWS_ACCESS_KEY_ID)" \
  --from-literal=REGISTRY_STORAGE_S3_SECRETKEY="$(cat AWS_SECRET_ACCESS_KEY)" \
  -n openshift-image-registry
secret/image-registry-private-configuration-user created
```

Note

The `Setting up and configuring the registry` section of the OpenShift Container Platform Registry guide contains an example of the `oc create secret` command.

4. Configure the internal registry to use the `noobaa-registry` OBC.

4.1. Identify the bucket name associated with the `noobaa-registry` OBC. The bucket name in your environment is different from the one displayed in this command.

```
[student@workstation services-registry]$ oc get -n openshift-image-registry \
  objectbucketclaim/noobaa-registry -o jsonpath='{.spec.bucketName}{"\n"}'
noobaa-registry-038ca5ee-d9ed-4b20-997a-c72058af2426
```

4.2. Identify the URL for the `s3` route in the `openshift-storage` namespace.

```
[student@workstation services-registry]$ oc get route/s3 -n openshift-storage \
  -o jsonpath='{.spec.host}{"\n"}'
s3-openshift-storage.apps.ocp4.example.com
```

4.3. Edit the `imageregistry-patch.yaml` file and modify the placeholder fields to match the following content.

```
---
apiVersion: imageregistry.operator.openshift.io/v1
kind: Config
metadata:
  name: cluster
spec:
  storage:
    managementState: Managed
    pvc: null
    s3:
      bucket: noobaa-registry-038ca5ee-d9ed-4b20-997a-c72058af2426
      region: us-east-1
      regionEndpoint: https://s3-openshift-storage.apps.ocp4.example.com
```

Note

The `null` value in the `pvc` key indicates that this section will be deleted when the patch is applied.

The `region` parameter is used by the S3 library to create the appropriate AWS URL for the S3 bucket, however, this value is overridden when specifying a custom URL in the `regionEndpoint` parameter.

4.4. Apply a patch to the image registry configuration. This will change the storage type from PVC to OBC.

```
[student@workstation services-registry]$ oc patch configs.imageregistry/cluster \
  --type=merge --patch-file=imageregistry-patch.yaml
config.imageregistry.operator.openshift.io/cluster patched
```

4.5. Confirm that the resource is patched correctly.

```
[student@workstation services-registry]$ oc get configs.imageregistry/cluster \
  -o jsonpath='{.spec.storage}' | jq .
{
  "managementState": "Managed",
  "s3": {
    "bucket": "noobaa-registry-038ca5ee-d9ed-4b20-997a-c72058af2426",
    "encrypt": true,
    "region": "us-east-1",
    "regionEndpoint": "https://s3-openshift-storage.apps.ocp4.example.com",
    "virtualHostedStyle": false
  }
}
```

Note

The `patch` operation deleted the `pvc` key so it is absent from the output.

4.6. Monitor the redeployment of the `image-registry` pod in the `openshift-image-registry` namespace.

```
[student@workstation services-registry]$ watch oc get pods \
  -n openshift-image-registry -l docker-registry=default
NAME                              READY   STATUS    RESTARTS   AGE
image-registry-648cb74f45-fgtj4   1/1     Running   0          90s
```

Press `Ctrl+C` to exit the `watch` command after the new `image-registry` pod is ready.

Note

Because the `image-registry` pod redeploys quickly, it might already be in a `Running` state. The `image-registry` pod displays an age in seconds or minutes instead of in hours or days.

If the `image-registry` pod does not redeploy, run the `oc edit` command and verify that the `spec.storage` fields are correct.

4.7. Query the `image-registry` deployment to verify that it contains the information set in the internal image registry configuration. The first line of the output contains the type of storage used by the image registry.

```
[student@workstation services-registry]$ ./check-imageregistry-s3.sh
REGISTRY_STORAGE s3
REGISTRY_STORAGE_S3_BUCKET noobaa-registry-505bbc9e-e7b7-4d31-a5db-d31dc7cd2dce
REGISTRY_STORAGE_S3_REGION us-east-1
```

```
REGISTRY_STORAGE_S3_REGIONENDPOINT https://s3-openshift-
storage.apps.ocp4.example.com
REGISTRY_STORAGE_S3_ENCRYPT true
REGISTRY_STORAGE_S3_VIRTUALHOSTEDSTYLE false
REGISTRY_STORAGE_S3_USEDUALSTACK true
REGISTRY_STORAGE_S3_CREDENTIALSCONFIGPATH /var/run/secrets/cloud/credentials
```

Note

If the script only returns `REGISTRY_STORAGE filesystem`, then the image registry is still using the PVC storage. Run the `oc edit` command and verify that the `s3` fields are correct.

5. Create a new container image and verify that the source-to-image process successfully pushes the container image to the internal image registry.

5.1. Create a new project named `services-registry`.

```
[student@workstation services-registry]$ oc new-project services-registry
Now using project "services-registry" on server "https://
api.ocp4.example.com:6443".
...output omitted...
```

5.2. Create a new deployment named `hello-world` that uses the `hello-world-nginx` directory of the `https://github.com/RedHatTraining/DO280-apps` git repository.

```
[student@workstation services-registry]$ oc new-app \
  --name hello \
  https://github.com/RedHatTraining/DO280-apps \
  --context-dir hello-world-nginx
...output omitted...
```

5.3. Monitor the logs of the build process. Look for a message indicating that the image was successfully pushed to the internal image registry.

```
[student@workstation services-registry]$ oc logs -f buildconfig/hello
...output omitted...
Successfully pushed image-registry.openshift-image-registry.svc:5000/services-
registry/hello@sha256:...
Push successful
```

5.4. Deploy a job to list the contents of the object bucket.

```
[student@workstation services-registry]$ oc apply -f job-awscli.yaml
job.batch/awscli created
```

5.5. View the logs of the job to show the contents of the object bucket.

```
[student@workstation services-registry]$ oc logs -f job/awscli \
  -n openshift-image-registry
aws s3 ls s3://${BUCKET_NAME} --endpoint-url=https://${BUCKET_HOST} --recursive --
summarize --human-readable
... docker/registry/v2/blobs/sha256/.../data
...output omitted...
... docker/registry/v2/repositories/services-registry/hello/layers/sha256/...
...output omitted...
... docker/registry/v2/repositories/services-registry/hello/manifests/...
...output omitted...

Total Objects: 18
   Total Size: 92.1 MiB
```

6. Verify the persistence of the internal image registry storage.

   6.1. Delete the `image-registry` pod in the `openshift-image-registry` namespace.

```
[student@workstation services-registry]$ oc delete pods \
  -n openshift-image-registry -l docker-registry=default
pod "image-registry-7d75b649b9-6wshl" deleted
```

   6.2. Identify the node on which the `hello` pod is running. The pod runs on `worker01`, `worker02`, or `worker03`.

```
[student@workstation services-registry]$ oc get pods -l deployment=hello -o wide
NAME                     READY   STATUS    ...   NODE       ...
hello-64786cc9df-969c5   1/1     Running   ...   worker02   ...
```

   6.3. Label one of the worker nodes with the `env=qa` label.

```
[student@workstation services-registry]$ oc label node/worker01 env=qa
node/worker01 labeled
```

Note

Do not label the worker node on which the `hello` pod is currently running.

   6.4. List the nodes that have the `env=qa` label.

```
[student@workstation ~]$ oc get nodes -l env=qa
NAME       STATUS   ROLES    AGE     VERSION
worker01   Ready    worker   20h     v1.20.0+a0b09eb
```

   6.5. Modify the `hello` deployment so that it deploys pods to nodes with the `env=qa` label. Use either the `oc edit` command or the following `oc patch` command to make the change.

```
[student@workstation services-registry]$ oc patch deployment/hello --type merge \
  --patch '{"spec":{"template":{"spec":{"nodeSelector":{"env":"qa"}}}}}'
deployment.apps/hello patched
```

6.6. Monitor the `hello` pod, and observe how it redeploys to the node with the `env=qa` label. Press `Ctrl+C` to exit the `watch` command after the new `hello` pod is ready.

```
[student@workstation services-registry]$ watch oc get pods -o wide
```

The `hello` pod runs on a different worker node, verifying that the worker node successfully downloaded the `hello` image from the internal image registry. The ability to download the `hello` image after the `image-registry` pod was deleted proves the persistence of the internal image registry storage.

7. Clean up the exercise environment.

7.1. Delete the `services-registry` project.

```
[student@workstation services-registry]$ oc delete project services-registry
project.project.openshift.io "services-registry" deleted
```

7.2. Remove the `env=qa` label from any worker node that has the label.

```
[student@workstation services-registry]$ oc label nodes env- -l env=qa
node/worker01 labeled
```

Note

You do not need to revert the internal image registry to use storage provided by the `nfs-storage` storage class.

7.3. Delete the job that listed the contents of the S3 bucket.

```
[student@workstation services-registry]$ oc delete job/awscli \
  -n openshift-image-registry
job.batch "awscli" deleted
```

7.4. Return to the `/home/student` directory.

```
[student@workstation services-registry]$ cd
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish services-registry
```

This concludes the guided exercise.

Configuring Monitoring to Use Red Hat OpenShift Data Foundation

Objectives

After completing this section, you should be able to configure the OpenShift monitoring subsystem to use block storage from Red Hat OpenShift Data Foundation

Explaining the RHOCP Monitoring Stack Architecture

The RHOCP monitoring stack provides built-in monitoring and alerts for the core platform components. Available operations include querying metrics, reviewing dashboards, and managing alerting rules. The monitoring stack is integrated into the RHOCP web console.

Furthermore, administrators can also enable monitoring for user-defined projects after the cluster installation.

The monitoring stack provided by RHOCP is based on Prometheus. The following diagram describes the components of the Prometheus platform monitoring stack and the monitoring stack for user-defined projects.

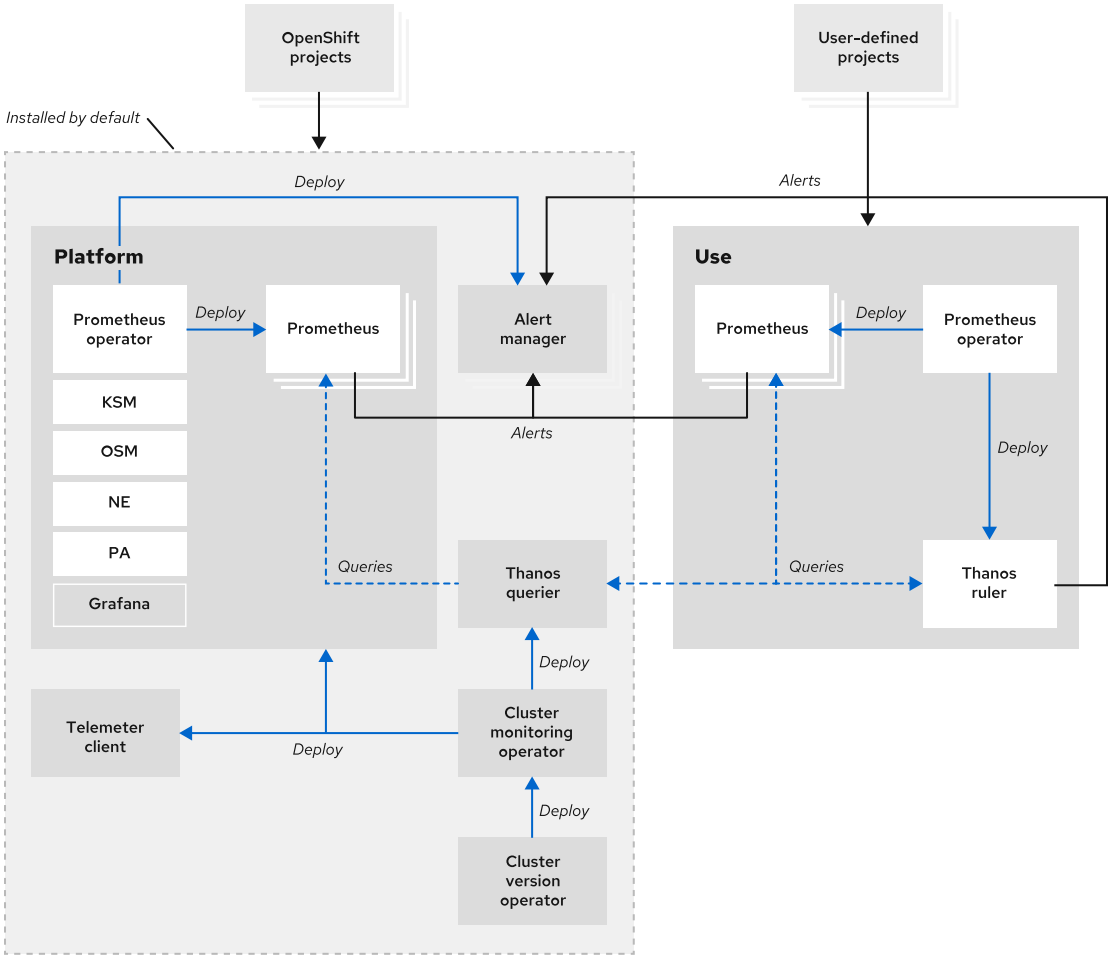


Figure 2.1: Red Hat OpenShift Data Foundation monitoring architecture

As you can see in the preceding diagram, the default monitoring stack that monitors the core platform components is comprised of the following components:

- The Cluster Monitoring operator (CMO), which deploys and manages Prometheus instances, the Thanos Querier, the telemeter client, and the metrics targets. The CMO is deployed by the Cluster Version operator (CVO).
- The Prometheus operator (PO), which creates, configures, and manages platform Prometheus and Alertmanager instances in the `openshift-monitoring` namespace.
- A Prometheus instance for platform monitoring, which contains a time-series database and a rule evaluation engine. This instance also sends alarms to Alertmanager.
- The Prometheus adapter (PA), which translates Kubernetes resource metrics to be used in Prometheus. It also exposes an API to be used for the horizontal pod autoscaler (HPA).
- An Alertmanager instance, which handles alerts received from Prometheus and sends alerts to external systems when configured.
- The `kube-state-metrics` agent (KSM), which converts Kubernetes objects to Prometheus metrics.
- The `openshift-state-metrics` agent (OSM), which expands the metrics provided by the `kube-state-metrics` agent with RHOCP specific resources.
- The `node-exporter` agent (NE), which collects metrics from every node in the cluster. There is one `node-exporter` agent running on each node.
- Thanos querier (TQ), which aggregates and deduplicates Prometheus queries from multiple instances under a single, multitenant interface.
- Grafana, which provides dashboards for analyzing and visualizing core platform metrics.
- The telemeter client, which sends a portion of the data from Prometheus instances to Red Hat for cluster remote health monitoring.

The components of any user-defined Prometheus monitoring stack are:

- The PO, which creates, configures, and manages platform Prometheus and Thanos ruler instances in the `openshift-user-workload-monitoring` namespace.
- A Prometheus instance for monitoring user-defined projects. This instance also sends alarms to Alertmanager.
- A Thanos ruler instance, which is a rule evaluation engine for Prometheus, deployed as a separate process.

Configuring Persistent Storage for the RHOCP Monitoring Stack

By default, the monitoring data storage is set to ephemeral. All the metrics are lost when the pods are restarted or recreated.

You can configure persistent storage volumes (PV) to store the monitoring data. This allows organizations to preserve metrics for post-mortem analysis, big data, cost management, or infrastructure load statistics.

Red Hat recommends using local storage in production environments because of the high IO storage demands. This approach is compatible with using an OpenShift Data Foundation cluster in internal mode.

You can configure either block or file storage for the monitoring stack. The following table summarizes the recommended and configurable storage options:

Storage type	ReadOnlyMany	ReadWriteMany	Monitoring Stack
File	Yes (1)	Yes	Configurable
Object	Yes	Yes	Not Configurable
Block	Yes (1)	No	**Recommended**

(1) Does not apply to physical disk, virtual machine physical disk, VMware VMDK, loopback over NFS, AWS EBS, or Azure Disk.

Prometheus Storage Prerequisites

The following table summarizes Prometheus storage requirements based on the number of nodes and pods in the cluster.

Number of Nodes	Number of Pods	Storage Growth / Day	Storage Growth / 15 Days	RAM Memory Consumption	Network Bandwidth per TSDB
50	1800	6.3 GB	94 GB	6 GB	16 MB
100	3600	13 GB	195 GB	10 GB	26 MB
150	5400	19 GB	283 GB	12 GB	36 MB
200	7200	25 GB	375 GB	14 GB	46 MB

(1) TSDB means time-series database.

Red Hat recommends using at least three infrastructure nodes to host the monitoring stack components. Also, non-volatile memory express (NVMe) drives are preferred over other disk types.

To configure the core platform Prometheus storage, administrators must create a `ConfigMap` resource named `cluster-monitoring-config` in the `openshift-monitoring` namespace. An example of configuration map file follows.

```
apiVersion: v1
kind: ConfigMap
data:
  config.yaml: |
    prometheusOperator:
      baseImage: quay.io/coreos/prometheus-operator
      prometheusConfigReloaderBaseImage: quay.io/coreos/prometheus-config-reloader
      configReloaderBaseImage: quay.io/coreos/configmap-reload
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    prometheusK8s:
      retention: 15d (1)
      baseImage: openshift/prometheus
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      volumeClaimTemplate:
```

```
        spec:
          storageClassName: ocs-storagecluster-ceph-rbd (2)
          resources:
            requests:
              storage: 2000Gi (3)
    alertmanagerMain:
      baseImage: openshift/prometheus-alertmanager
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      volumeClaimTemplate:
        spec:
          storageClassName: ocs-storagecluster-ceph-rbd (4)
          resources:
            requests:
              storage: 20Gi (5)
    nodeExporter:
      baseImage: openshift/prometheus-node-exporter
    kubeRbacProxy:
      baseImage: quay.io/coreos/kube-rbac-proxy
    kubeStateMetrics:
      baseImage: quay.io/coreos/kube-state-metrics
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    grafana:
      baseImage: grafana/grafana
      nodeSelector:
        node-role.kubernetes.io/infra: ""
    auth:
      baseImage: openshift/oauth-proxy
    k8sPrometheusAdapter:
      nodeSelector:
        node-role.kubernetes.io/infra: ""
metadata:
  name: cluster-monitoring-config
namespace: openshift-monitoring
```

(1) Time units are measured in seconds (s), minutes (m), hours (h), or days (d).

(2) The object storage class provided by OpenShift Data Foundation is named `ocs-storagecluster-ceph-rbd`.

(3) Storage values can be a plain integer or a fixed-point integer that uses one of these suffixes: E, P, T, G, M, K. You can also use the power-of-two equivalents: Ei, Pi, Ti, Gi, Mi, Ki.

(4) The object storage class provided by OpenShift Data Foundation is named `ocs-storagecluster-ceph-rbd`.

(5) Storage values can be a plain integer or a fixed-point integer that uses one of these suffixes: E, P, T, G, M, K. You can also use the power-of-two equivalents: Ei, Pi, Ti, Gi, Mi, Ki.

References

For more information, refer to the product documentation for *Red Hat OpenShift Container Platform 4.7* at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.7/html-single/monitoring/index

For more information, refer to the product documentation for *Red Hat OpenShift Data Foundation* at
https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/

Guided Exercise

Configuring Monitoring to Use Red Hat OpenShift Data Foundation

In this exercise, you will configure Prometheus and Alertmanager to use storage provided by Red Hat OpenShift Data Foundation.

Outcomes

You should be able to:

- Identify storage classes available to the cluster.
- Configure Prometheus and Alertmanager to use block storage provided by the `ocs-storagecluster-ceph-rbd` storage class.

Before You Begin

To perform this exercise, ensure that you have access to:

- A running OpenShift cluster.
- Installed and configured operators for OpenShift Container Storage and the Local Storage Operator.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable and that both OCS and LSO are installed and configured correctly.

```
[student@workstation ~]$ lab start services-metrics
```

Instructions

1. Identify the storage classes available to the cluster.

   1.1. Log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

   1.2. List the available storage classes. Red Hat recommends using block storage for metrics. The `ocs-storagecluster-ceph-rbd` storage class provides block storage.

```
[student@workstation ~]$ oc get storageclasses -o name
storageclass.storage.k8s.io/lso-volumeset
storageclass.storage.k8s.io/nfs-storage
storageclass.storage.k8s.io/ocs-storagecluster-ceph-rbd
storageclass.storage.k8s.io/ocs-storagecluster-ceph-rgw
storageclass.storage.k8s.io/ocs-storagecluster-cephfs
storageclass.storage.k8s.io/openshift-storage.noobaa.io
```

2. Verify that both Prometheus and Alertmanager use ephemeral storage.

   2.1. Change to the `~/DO370/labs/services-metrics` directory.

```
[student@workstation ~]$ cd ~/DO370/labs/services-metrics
```

   2.2. Verify that the `prometheus-k8s` stateful set in the `openshift-monitoring` namespace uses ephemeral storage at the `/prometheus` mount point.

```
[student@workstation services-metrics]$ ./check-prometheus.sh

# Prometheus container volume mount spec:
oc get statefulset/prometheus-k8s \
  -n openshift-monitoring \
  -o jsonpath='{.spec.template.spec.containers}' | \
jq '.[] | select(.name == "prometheus") | .volumeMounts[] | select(.name ==
 "prometheus-k8s-db")'
{
  "mountPath": "/prometheus",
  "name": "prometheus-k8s-db"
}

# Prometheus volume mount type:
oc get statefulset/prometheus-k8s \
  -n openshift-monitoring \
  -o jsonpath='{.spec.template.spec.volumes}' | \
  jq '.[] | select(.name == "prometheus-k8s-db")'
{
  "emptyDir": {},
  "name": "prometheus-k8s-db"
}
```

   2.3. Verify the disk space in the `emptyDir` volume mounted in `/prometheus`.

```
[student@workstation ~]$ oc exec -n openshift-monitoring \
  statefulset/prometheus-k8s -c prometheus -- df -h /prometheus
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda4        80G   13G   67G  17% /prometheus
```

   2.4. Verify that the `alertmanager-main` stateful set in the `openshift-monitoring` namespace uses ephemeral storage at the `/alertmanager` mount point.

```
[student@workstation services-metrics]$ ./check-alertmanager.sh

# AlertManager container volume mount spec:
oc get statefulset/alertmanager-main \
  -n openshift-monitoring \
  -o jsonpath='{.spec.template.spec.containers}' | \
jq '.[] | select(.name == "alertmanager") | .volumeMounts[] | select(.name ==
 "alertmanager-main-db")'
{
  "mountPath": "/alertmanager",
  "name": "alertmanager-main-db"
}

# AlertManager volume mount type:
oc get statefulset/alertmanager-main \
  -n openshift-monitoring \
  -o jsonpath='{.spec.template.spec.volumes}' | \
  jq '.[] | select(.name == "alertmanager-main-db")'
{
  "emptyDir": {},
  "name": "alertmanager-main-db"
}
```

2.5. Verify the disk space in the `emptyDir` volume mounted in `/prometheus`.

```
[student@workstation services-metrics]$ oc exec -n openshift-monitoring \
  statefulset/alertmanager-main -c alertmanager -- df -h /alertmanager
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda4        80G   13G   67G  17% /alertmanager
```

3. Create the `cluster-monitoring-config` configuration map in the `openshift-monitoring` namespace.

3.1. Modify the `metrics-storage.yml` file.

- Configure Prometheus to retain metrics information for seven days.
- Configure each `prometheus-k8s` pod to request a 40 GB PVC from the `ocs-storagecluster-ceph-rbd` storage class.
- Configure each `alertmanager-main` pod to request a 20 GB PVC from the `ocs-storagecluster-ceph-rbd` storage class.

Ensure that the file matches the following bold lines, and then save the file.

```
prometheusK8s:
  retention: 7d
  volumeClaimTemplate:
    spec:
      storageClassName: ocs-storagecluster-ceph-rbd
      resources:
        requests:
          storage: 40Gi
```

```
alertmanagerMain:
  volumeClaimTemplate:
    spec:
      storageClassName: ocs-storagecluster-ceph-rbd
      resources:
        requests:
          storage: 20Gi
```

Note

The `~/DO370/solutions/services-metrics/metrics-storage.yml` file contains the correct configuration and can be used for comparison.

3.2. Create the new configuration map.

```
[student@workstation services-metrics]$ oc create -n openshift-monitoring \
  configmap cluster-monitoring-config --from-file config.yaml=metrics-storage.yml
```

4. Verify that both Prometheus and Alertmanager use block storage from OpenShift Data Foundation.

4.1. Monitor the redeployment of `prometheus-k8s` and `alertmanager-main` pods in the `openshift-monitoring` namespace.

```
[student@workstation services-metrics]$ watch oc get statefulsets \
  -n openshift-monitoring
NAME                READY   AGE
alertmanager-main   3/3     2m47s
prometheus-k8s      2/2     2m48s
```

Press `Ctrl+C` to exit the `watch` command when the `prometheus-k8s` stateful set displays a ready status of `2/2` and the `alertmanager-main` stateful set displays a ready status of `3/3`.

4.2. Verify the PVCs that are used to save the monitoring data are created.

```
[student@workstation services-metrics]$ oc get persistentvolumeclaims \
  -n openshift-monitoring
NAME                                          STATUS   VOLUME    CAPACITY   ...
alertmanager-main-db-alertmanager-main-0      Bound    pvc-...   20Gi       ...
alertmanager-main-db-alertmanager-main-1      Bound    pvc-...   20Gi       ...
alertmanager-main-db-alertmanager-main-2      Bound    pvc-...   20Gi       ...
prometheus-k8s-db-prometheus-k8s-0            Bound    pvc-...   40Gi       ...
prometheus-k8s-db-prometheus-k8s-1            Bound    pvc-...   40Gi       ...
```

4.3. Verify that the `prometheus-k8s` stateful set in the `openshift-monitoring` namespace mounts a block device to the `/prometheus` mount point. A file system starting with `/dev/rbd` indicates block storage. Also check that the volume size is now 40 GB.

```
[student@workstation services-metrics]$ oc exec -n openshift-monitoring \
  statefulset/prometheus-k8s -c prometheus -- df -h /prometheus
Filesystem      Size  Used Avail Use% Mounted on
/dev/rbd1        40G   97M   40G   1% /prometheus
```

4.4. Verify that the `alertmanager-main` stateful set in the `openshift-monitoring` namespace mounts a block device to the `/alertmanager` mount point. A file system starting with `/dev/rbd` indicates block storage. Also check that the volume size is now 20 GB.

```
[student@workstation services-metrics]$ oc exec -n openshift-monitoring \
  statefulset/alertmanager-main -c alertmanager -- df -h /alertmanager
Filesystem      Size  Used Avail Use% Mounted on
/dev/rbd2        20G   45M   20G   1% /alertmanager
```

5. Return to the `/home/student` directory.

```
[student@workstation services-metrics]$ cd
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish services-metrics
```

This concludes the guided exercise.

Lab

Configuring OpenShift Cluster Services to Use Red Hat OpenShift Data Foundation

In this end of chapter review, you will configure OpenShift Cluster Services to use Red Hat OpenShift Data Foundation and monitor this configuration.

Outcomes

You should be able to:

- Reset metering and the associated default no-storage settings.
- Configure a simple registry that uses object storage.
- Verify functionality by adding an image to the registry.

Before You Begin

To perform this exercise, ensure that you have access to a running OpenShift cluster with installed and configured operators for both OpenShift Container Storage (OCS) and Local Storage Operator (LSO).

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start services-review
```

Instructions

1. Identify the storage classes available to the cluster.
   - Change to the `~/DO370/labs/services-review` directory.
2. Create a new object bucket claim named `noobaa-review` in the `openshift-image-registry` namespace.
   - The object bucket claim uses the `openshift-storage.noobaa.io` storage class.
   - You can use the `~/DO370/labs/services-review/obc-review.yml` file as a template.
3. Create a new generic secret containing the credentials needed to access and manage the OBC.
4. Configure the internal registry to use the `noobaa-review` OBC.
5. Create a new project named `services-review` and verify that the source-to-image process successfully pushes the container image to the internal image registry.

- Deploy the `hello-world-nginx` directory from `https://github.com/RedHatTraining/DO280-apps` as a new app named `hello-world`.

6. Change to the `/home/student` directory.

Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade services-review
```

Finish

As the `student` user on the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish services-review
```

This concludes the lab.

Solution

Configuring OpenShift Cluster Services to Use Red Hat OpenShift Data Foundation

In this end of chapter review, you will configure OpenShift Cluster Services to use Red Hat OpenShift Data Foundation and monitor this configuration.

Outcomes

You should be able to:

- Reset metering and the associated default no-storage settings.
- Configure a simple registry that uses object storage.
- Verify functionality by adding an image to the registry.

Before You Begin

To perform this exercise, ensure that you have access to a running OpenShift cluster with installed and configured operators for both OpenShift Container Storage (OCS) and Local Storage Operator (LSO).

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start services-review
```

Instructions

1. Identify the storage classes available to the cluster.

   - Change to the `~/DO370/labs/services-review` directory.

   1.1. Log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  https://api.ocp4.example.com:6443
```

   1.2. Change to the `~/DO370/labs/services-review` directory.

```
[student@workstation ~]$ cd ~/DO370/labs/services-review
```

   1.3. List the available storage classes. In this lab, you create a new object bucket claim using the `openshift-storage.noobaa.io` storage class.

```
[student@workstation services-review]$ oc get storageclasses -o name
storageclass.storage.k8s.io/lso-volumeset
storageclass.storage.k8s.io/nfs-storage
storageclass.storage.k8s.io/ocs-storagecluster-ceph-rbd
storageclass.storage.k8s.io/ocs-storagecluster-ceph-rgw
storageclass.storage.k8s.io/ocs-storagecluster-cephfs
storageclass.storage.k8s.io/openshift-storage.noobaa.io
```

2. Create a new object bucket claim named `noobaa-review` in the `openshift-image-registry` namespace.

   - The object bucket claim uses the `openshift-storage.noobaa.io` storage class.
   - You can use the `~/DO370/labs/services-review/obc-review.yml` file as a template.

   2.1. Edit the `obc-review.yml` file and modify the placeholder text to match the following content.

   - Configure the OBC to use the name `noobaa-review`.
   - Create the OBC in the `openshift-image-registry` namespace.
   - Ensure that the bucket name starts with `noobaa-review` by giving the setting `generateBucketName` a value of `noobaa-review`.
   - Use the `openshift-storage.noobaa.io` storage class.

```
---
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: noobaa-review
  namespace: openshift-image-registry
spec:
  additionalConfig:
    bucketclass: noobaa-default-bucket-class
  generateBucketName: noobaa-review
  storageClassName: openshift-storage.noobaa.io
```

   2.2. Create the new OBC.

```
[student@workstation services-review]$ oc apply -f obc-review.yml
objectbucketclaim.objectbucket.io/noobaa-review created
```

   2.3. Verify that the cluster successfully created the `noobaa-review` OBC. The `noobaa-review` OBC has a phase of `Bound`.

```
[student@workstation services-review]$ oc get objectbucketclaim/noobaa-review \
  -n openshift-image-registry
NAME            STORAGE-CLASS                 PHASE   AGE
noobaa-review   openshift-storage.noobaa.io   Bound   62s
```

3. Create a new generic secret containing the credentials needed to access and manage the OBC.

3.1. List the secrets in the `openshift-image-registry` namespace. When you created the `noobaa-review` OBC, a secret named `noobaa-review` was also created.

```
[student@workstation services-review]$ oc get secret/noobaa-review \
  -n openshift-image-registry
NAME              TYPE     DATA   AGE
noobaa-review   Opaque   2      1m10s
```

3.2. Extract the `noobaa-review` secret to your current directory.

```
[student@workstation services-review]$ oc extract secret/noobaa-review \
  -n openshift-image-registry
AWS_ACCESS_KEY_ID
AWS_SECRET_ACCESS_KEY
```

3.3. Use the contents of the `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` files to create the `image-registry-private-configuration-user` secret.

```
[student@workstation services-review]$ oc create secret generic \
  image-registry-private-configuration-user \
  --from-literal=REGISTRY_STORAGE_S3_ACCESSKEY="$(cat AWS_ACCESS_KEY_ID)" \
  --from-literal=REGISTRY_STORAGE_S3_SECRETKEY="$(cat AWS_SECRET_ACCESS_KEY)" \
  -n openshift-image-registry
secret/image-registry-private-configuration-user created
```

4. Configure the internal registry to use the `noobaa-review` OBC.

4.1. Identify the bucket name associated with the `noobaa-review` OBC. The bucket name in your environment is different from the one displayed in this command.

```
[student@workstation services-review]$ oc get objectbucketclaim/noobaa-review \
  -n openshift-image-registry -o jsonpath='{.spec.bucketName}{"\n"}'
noobaa-review-038ca5ee-d9ed-4b20-997a-c72058af2426
```

4.2. Identify the URL for the `s3` route in the `openshift-storage` namespace.

```
[student@workstation services-review]$ oc get route/s3 -n openshift-storage \
  -o jsonpath='{.spec.host}{"\n"}'
s3-openshift-storage.apps.ocp4.example.com
```

4.3. Edit the `imageregistry-patch.yaml` file and modify the placeholder fields to match the following content.

```
---
apiVersion: imageregistry.operator.openshift.io/v1
kind: Config
metadata:
  name: cluster
```

```
spec:
  storage:
    managementState: Managed
    pvc: null
    s3:
      bucket: noobaa-review-038ca5ee-d9ed-4b20-997a-c72058af2426
      region: us-east-1
      regionEndpoint: https://s3-openshift-storage.apps.ocp4.example.com
```

Note

The `null` value in the `pvc` key indicates that this section will be deleted when the patch is applied.

The S3 library uses the `region` parameter to create the appropriate AWS URL for the S3 bucket. However, this value is overridden when specifying a custom URL in the `regionEndpoint` parameter.

4.4. Apply a patch to the image registry configuration. This will change the storage type from PVC to OBC.

```
[student@workstation services-review]$ oc patch configs.imageregistry/cluster \
  --type=merge --patch-file=imageregistry-patch.yaml
config.imageregistry.operator.openshift.io/cluster patched
```

4.5. Confirm that the resource is patched correctly.

```
[student@workstation services-review]$ oc get configs.imageregistry/cluster \
  -o jsonpath='{.spec.storage}' | jq .
{
  "managementState": "Managed",
  "s3": {
    "bucket": "noobaa-review-038ca5ee-d9ed-4b20-997a-c72058af2426",
    "encrypt": true,
    "region": "us-east-1",
    "regionEndpoint": "https://s3-openshift-storage.apps.ocp4.example.com",
    "virtualHostedStyle": false
  }
}
```

Note

The `patch` operation deleted the `pvc` key so it is absent from the output.

4.6. Monitor the redeployment of the `image-registry` pod in the `openshift-image-registry` namespace.

```
[student@workstation services-review]$ watch oc get pods \
  -n openshift-image-registry -l docker-registry=default
```

Press `Ctrl+C` to exit the `watch` command after the new `image-registry` pod is ready.

Note

Because the `image-registry` pod redeploys quickly, it might already be in a `Running` state. The `image-registry` pod displays an age in seconds or minutes instead of in hours or days.

If the `image-registry` pod does not redeploy, run the `oc edit` command and verify that the `spec.storage` fields are correct.

4.7. Query the `image-registry` deployment to verify that it contains the information set in the internal image registry configuration. The first line of the output contains the type of storage used by the image registry.

```
[student@workstation services-review]$ ./check-imageregistry-s3.sh
REGISTRY_STORAGE s3
REGISTRY_STORAGE_S3_BUCKET noobaa-review-038ca5ee-d9ed-4b20-997a-c72058af2426
REGISTRY_STORAGE_S3_REGION us-east-1
REGISTRY_STORAGE_S3_REGIONENDPOINT https://s3-openshift-
storage.apps.ocp4.example.com
REGISTRY_STORAGE_S3_ENCRYPT true
REGISTRY_STORAGE_S3_VIRTUALHOSTEDSTYLE false
REGISTRY_STORAGE_S3_USEDUALSTACK true
REGISTRY_STORAGE_S3_CREDENTIALSCONFIGPATH /var/run/secrets/cloud/credentials
```

Note

If the script only returns `REGISTRY_STORAGE filesystem`, then the image registry is still using the PVC storage. Run the `oc edit` command and verify that the `s3` fields are correct.

5. Create a new project named `services-review` and verify that the source-to-image process successfully pushes the container image to the internal image registry.

- Deploy the `hello-world-nginx` directory from `https://github.com/RedHatTraining/DO280-apps` as a new app named `hello-world`.

5.1. Create a new project named `services-review`.

```
[student@workstation services-review]$ oc new-project services-review
Now using project "services-review" on server "https://api.ocp4.example.com:6443".
...output omitted...
```

5.2. Create a new deployment named `hello-world` that uses the `hello-world-nginx` directory in the `https://github.com/RedHatTraining/DO280-apps` git repository.

```
[student@workstation services-review]$ oc new-app \
  --name hello-world \
  https://github.com/RedHatTraining/DO280-apps \
  --context-dir hello-world-nginx
...output omitted...
```

5.3. Monitor the logs of the build process. Look for a message indicating that the image was successfully pushed to the internal image registry.

```
[student@workstation services-review]$ oc logs -f buildconfig/hello-world
...output omitted...
Successfully pushed image-registry.openshift-image-registry.svc:5000/services-
review/hello-world@sha256:...
Push successful
```

6. Change to the `/home/student` directory.

```
[student@workstation backup-review]$ cd
```

Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade services-review
```

Finish

As the `student` user on the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish services-review
```

This concludes the lab.

Summary

In this chapter, you learned:

- Common Kubernetes concepts, such as:
  - Storage Providers
  - Storage Pools
  - Storage Classes
  - Persistent Volumes
  - Persistent Volume Claims
- The differences between dynamic and static storage provisioning.
- How Red Hat OpenShift Data Foundation can provide and manage storage for RHOCP services.
- The configuration process to implement OpenShift Data Foundation-managed storage for common cluster services. such as:
  - Monitoring
  - Logging
  - Image Registry

Chapter 3

Configuring Application Workloads to Use Red Hat OpenShift Data Foundation File and Block Storage

Goal

Select and configure OpenShift Data Foundation storage classes to meet application requirements.

Objectives

- Identify the components and gather information from a Ceph storage implementation for Red Hat OpenShift Data Foundation.
- Configure applications to use file storage provided by Red Hat OpenShift Data Foundation.
- Configure applications to use block storage provided by Red Hat OpenShift Data Foundation.
- Configure a new storage class with custom settings.

Sections

- Identifying Ceph Components for a Red Hat OpenShift Data Foundation Implementation (and Guided Exercise)
- Configuring Applications to Use Red Hat OpenShift Data Foundation File Storage (and Guided Exercise)
- Configuring Applications to Use Red Hat OpenShift Data Foundation Block Storage (and Guided Exercise)
- Configuring Custom Storage Classes (and Guided Exercise)

Lab

Configuring Application Workloads to Use Red Hat OpenShift Data Foundation Block and File Storage

Identifying Ceph Components for a Red Hat OpenShift Data Foundation Implementation

Objectives

After completing this section, you should be able to identify the components and gather information from a Ceph storage implementation for Red Hat OpenShift Data Foundation.

Introducing Ceph Architecture

Ceph Storage provides an enterprise-ready, software-defined storage solution that can grow to multipetabytes scale. Ceph has a modular and distributed architecture that contains the following elements:

- An object storage back end known as RADOS (Reliable Autonomic Distributed Object Store.)
- A variety of access methods to interact with RADOS. RADOS is a self-healing and self-managing software-based object store.

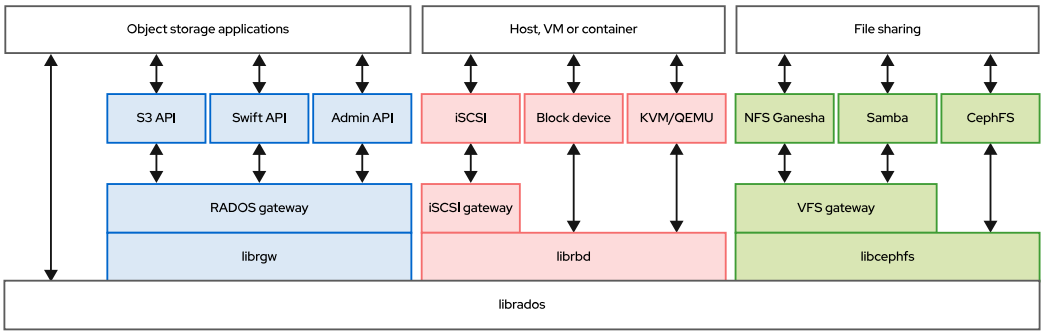


Figure 3.1: Ceph components

Discussing Ceph with OpenShift

Ceph is managed by the Rook-Ceph operator. The Local Storage Operator (LSO) creates persistent volumes that are passed to the Rook-Ceph operator to create the Ceph cluster.

```
[user@demo ~]$ oc get pv
NAME                     CAPACITY   ...   STATUS      ...   STORAGECLASS
local-pv-51858b47        500Gi      ...   Bound       ...   lso-volumeset
local-pv-5a72af91        500Gi      ...   Bound       ...   lso-volumeset
local-pv-83798806        500Gi      ...   Bound       ...   lso-volumeset
local-pv-b00672c8        500Gi      ...   Available   ...   lso-volumeset
local-pv-e03c4e6c        500Gi      ...   Available   ...   lso-volumeset
local-pv-f83b112         500Gi      ...   Available   ...   lso-volumeset
```

Each Ceph element runs inside a pod.

```
[user@demo ~]$ oc get pods -n openshift-storage
...output omitted...
rook-ceph-crashcollector-worker01-8668f6597f-qlknp               1/1  Running
rook-ceph-crashcollector-worker02-598fcf5dc6-wd6jf               1/1  Running
rook-ceph-crashcollector-worker03-f68b85645-7d5cr                1/1  Running
rook-ceph-mds-ocs-storagecluster-cephfilesystem-a-5d64f4ccrsbcw  1/1  Running
rook-ceph-mds-ocs-storagecluster-cephfilesystem-b-7ccb5db468z25  1/1  Running
rook-ceph-mgr-a-7dfc47677-2qgqq                                  1/1  Running
rook-ceph-mon-a-585ffd45d5-7vvlv                                 1/1  Running
rook-ceph-mon-b-5468b495d9-cd7zn                                 1/1  Running
rook-ceph-mon-c-5db66d86d8-t8bwx                                 1/1  Running
rook-ceph-operator-548bcdc79f-xcgjb                              1/1  Running
rook-ceph-osd-0-6cbf78cd77-t7dnr                                 1/1  Running
rook-ceph-osd-1-7976d54f74-b5d98                                 1/1  Running
rook-ceph-osd-2-578d5498bd-6rvjn                                 1/1  Running
rook-ceph-osd-prepare-ocs-deviceset...                           0/1  Completed
rook-ceph-osd-prepare-ocs-deviceset...                           0/1  Completed
rook-ceph-osd-prepare-ocs-deviceset...                           0/1  Completed
rook-ceph-rgw-ocs-storagecluster-cephobjectstore...              1/1  Running
rook-ceph-rgw-ocs-storagecluster-cephobjectstore...              1/1  Running
```

Investigating a Ceph Cluster

There are many ways to verify the health of a Ceph cluster, such as by reviewing the cluster logs, by using Ceph client tools, or by using the `must-gather` tool. These are some of the common ways that you can examine cluster health and obtain troubleshooting information on a Ceph cluster managed by OpenShift Data Foundation.

Inspecting Ceph Components Logs

Reviewing logs is often a quick way to determine the source of a problem. A problem will typically present itself in the form of error messages. With Ceph, many problems are self-healing, but if the cluster does not self-heal you will likely see error messages near the end of the logs.

This section provides information on how to access each Ceph log.

OSD logs

```
[user@demo ~]$ oc logs \
  rook-ceph-osd-0-6cbf78cd77-t7dnr -n openshift-storage
...output omitted...
AddFile(L0 Files): cumulative 0, interval 0
AddFile(Keys): cumulative 0, interval 0
Cumulative compaction: 0.00 GB write, 0.00 MB/s write, 0.00 GB read, 0.00 MB/s
 read, 0.0 seconds
Interval compaction: 0.00 GB write, 0.00 MB/s write, 0.00 GB read, 0.00 MB/s read,
 0.0 seconds
Stalls(count): 0 level0_slowdown, 0 level0_slowdown_with_compaction, 0
 level0_numfiles, 0 level0_numfiles_with_compaction, 0 stop for pending_comp
action_bytes, 0 slowdown for pending_compaction_bytes, 0 memtable_compaction, 0
 memtable_slowdown, interval 0 total count

*\* File Read Latency Histogram By Level [default] *\*
```

Monitor logs

```
[user@demo ~]$ oc logs \
  rook-ceph-mon-a-585ffd45d5-7vvlv -n openshift-storage
...output omitted...
TiB / 1.5 TiB avail; 3.9 KiB/s rd, 37 KiB/s wr, 8 op/s
cluster 2021-07-19 22:00:11.865553 mgr.a (mgr.14203) 2386 : cluster [DBG] pgmap
 v2426: 176 pgs: 176 active+clean; 110 MiB data, 180 MiB used, 1.5
TiB / 1.5 TiB avail; 5.2 KiB/s rd, 39 KiB/s wr, 12 op/s
cluster 2021-07-19 22:00:13.866558 mgr.a (mgr.14203) 2387 : cluster [DBG] pgmap
 v2427: 176 pgs: 176 active+clean; 110 MiB data, 180 MiB used, 1.5
TiB / 1.5 TiB avail; 5.2 KiB/s rd, 18 KiB/s wr, 11 op/s
...output omitted...
```

Rook Operator logs

```
[user@demo ~]$ oc logs \
  rook-ceph-operator-548bcdc79f-xcgjb -n openshift-storage
...output omitted...
2021-07-19 20:39:00.358254 I | operator: looking for secret "rook-ceph-admission-
controller"
2021-07-19 20:39:00.360649 I | operator: secret "rook-ceph-admission-controller"
 not found. proceeding without the admission controller
2021-07-19 20:39:00.361658 I | operator: watching the current namespace for a ceph
 cluster CR
2021-07-19 20:39:00.362034 I | operator: setting up the controller-runtime manager
2021-07-19 20:39:00.363796 I | ceph-cluster-controller: ConfigMap "rook-ceph-
operator-config" changes detected. Updating configurations
2021-07-19 20:39:03.317982 I | ceph-cluster-controller: successfully started
2021-07-19 20:39:03.318073 I | ceph-cluster-controller: hotplug orchestration
 disabled
2021-07-19 20:39:03.318084 I | ceph-crashcollector-controller: successfully
 started
2021-07-19 20:39:03.318152 I | ceph-block-pool-controller: successfully started
...output omitted...
```

CephFS file storage logs

```
[user@demo ~]$ oc logs \
  csi-cephfsplugin-722b2 -n openshift-storage -c csi-cephfsplugin
...output omitted...
I0719 20:39:28.221824   1 cephcsi.go:124] Driver version: release-4.6 and Git
 version: ce3f8de3748499a32ac53f179e7e7554d49d4a7e
I0719 20:39:28.222077   1 cephcsi.go:142] Initial PID limit is set to 1024
I0719 20:39:28.222131   1 cephcsi.go:151] Reconfigured PID limit to -1 (max)
I0719 20:39:28.222138   1 cephcsi.go:170] Starting driver type: cephfs with name:
 openshift-storage.cephfs.csi.ceph.com
I0719 20:39:28.239036   1 volumemounter.go:87] loaded mounter: kernel
E0719 20:39:28.239061   1 volumemounter.go:96] failed to run ceph-fuse exec:
 "ceph-fuse": executable file not found in $PATH
I0719 20:39:28.239595   1 server.go:118] Listening for connections on address:
 &net.UnixAddr{Name:"//csi/csi.sock", Net:"unix"}
```

```
I0719 20:39:28.891828   1 utils.go:159] ID: 1 GRPC call: /csi.v1.Identity/
GetPluginInfo
I0719 20:39:28.892820   1 utils.go:160] ID: 1 GRPC request: {}
...output omitted...
```

RBD plug-in block storage logs

```
[user@demo ~]$ oc logs \
  csi-rbdplugin-86dpc -n openshift-storage \
  -c csi-rbdplugin
...output omitted...
I0719 20:39:28.288333   23592 cephcsi.go:124] Driver version: release-4.6 and Git
 version: ce3f8de3748499a32ac53f179e7e7554d49d4a7e
I0719 20:39:28.288537   23592 cephcsi.go:142] Initial PID limit is set to 1024
I0719 20:39:28.288576   23592 cephcsi.go:151] Reconfigured PID limit to -1 (max)
I0719 20:39:28.288580   23592 cephcsi.go:170] Starting driver type: rbd with name:
 openshift-storage.rbd.csi.ceph.com
I0719 20:39:28.297442   23592 server.go:118] Listening for connections on address:
 &net.UnixAddr{Name:"//csi/csi.sock", Net:"unix"}
I0719 20:39:28.815489   23592 utils.go:159] ID: 1 GRPC call: /csi.v1.Identity/
GetPluginInfo
I0719 20:39:28.816213   23592 utils.go:160] ID: 1 GRPC request: {}
...output omitted...
```

PVC-specific logs

Logs for persistent volume claims (PVC) are handled by a designated pod. There is a pod in charge of each filesystem or block plug-in worker in the Ceph cluster.

Using Ceph Built-in Tools for Collecting Cluster Information

Ceph provides the built in tool `must-gather` and the Ceph CLI. These tools provide a quick way to obtain information and logs for review.

Using the Ceph CLI

To use the Ceph CLI, enter the Rook-Ceph operator and use the `/var/lib/rook/openshift-storage/openshift-storage.config` configuration file to access the cluster:

```
[user@demo ~]$ oc exec -ti \
  pod/rook-ceph-operator-548bcdc79f-xcgjb -n openshift-storage \
  -c rook-ceph-operator -- /bin/bash
bash-4.4$
```

You can obtain many different pieces of information through the CLI, as shown in the following examples.

- Cluster health

```
bash-4.4$ ceph -c /var/lib/rook/openshift-storage/openshift-storage.config health
HEALTH_OK
```

- Cluster status

```
[user@demo ~]$ oc exec -ti \
  pod/rook-ceph-operator-548bcdc79f-xcgjb \
  -n openshift-storage -c rook-ceph-operator -- /bin/bash
bash-4.4$ ceph -c /var/lib/rook/openshift-storage/openshift-storage.config -s
  cluster:
    id:     92aecb6f-1ef7-427b-bff9-71d3307b4454
    health: HEALTH_OK

  services:
    mon: 3 daemons, quorum a,b,c (age 2h)
    mgr: a(active, since 2h)
    mds: ocs-storagecluster-cephfilesystem:1 {0=ocs-storagecluster-cephfilesystem-
b=up:active} 1 up:standby-replay
    osd: 3 osds: 3 up (since 2h), 3 in (since 2h)
    rgw: 2 daemons active (ocs.storagecluster.cephobjectstore.a,
 ocs.storagecluster.cephobjectstore.b)

  data:
    pools:   10 pools, 176 pgs
    objects: 331 objects, 133 MiB
    usage:   3.2 GiB used, 1.5 TiB / 1.5 TiB avail
    pgs:     176 active+clean

  io:
    client:   5.8 KiB/s rd, 8.2 KiB/s wr, 6 op/s rd, 5 op/s wr
```

- Storage usage

```
bash-4.4$ ceph -c /var/lib/rook/openshift-storage/openshift-storage.config df
RAW STORAGE:
    CLASS     SIZE        AVAIL       USED        RAW USED     %RAW USED
    hdd       1.5 TiB     1.5 TiB     245 MiB      3.2 GiB          0.22
    TOTAL     1.5 TiB     1.5 TiB     245 MiB      3.2 GiB          0.22

POOLS:
    POOL                                ID  STORED   ..  USED     .. MAXAVAIL
    ocs-storagecluster-cephblockpool     1   69 MiB  ..  214 MiB  ..   424GiB
    ocs-storagecluster-cephobjectstore...  2      0 B  ..      0 B  ..   424GiB
    ocs-storagecluster-cephfilesystem-...  3  2.7 KiB  ..  1.5 MiB  ..   424GiB
    ocs-storagecluster-cephfilesystem-...  4      0 B  ..      0 B  ..   424GiB
    ocs-storagecluster-cephobjectstore...  5  2.8 KiB  ..  1.9 MiB  ..   424GiB
    ocs-storagecluster-cephobjectstore...  6  3.6 KiB  ..  6.6 MiB  ..   424GiB
    ocs-storagecluster-cephobjectstore...  7      0 B  ..      0 B  ..   424GiB
    ocs-storagecluster-cephobjectstore...  8      0 B  ..      0 B  ..   424GiB
    .rgw.root                           9  4.8 KiB  ..  2.8 MiB  ..   424GiB
    ocs-storagecluster-cephobjectstore... 10    1 KiB  ..  192 KiB  ..   424GiB
```

Using the must-gather Tool

Use the `must-gather` tool for collecting log files and diagnostic information about your cluster.

```
[user@demo ~]$ oc adm must-gather \
  --image=registry.redhat.io/ocs4/ocs-must-gather-rhel8:v4.7 \
  --dest-dir=must-gather
[must-gather      ] OUT Using must-gather plug-in image: registry.redhat.io/ocs4/
ocs-must-gather-rhel8:v4.7
When opening a support case, bugzilla, or issue please include the following
 summary data along with any other requested information.
ClusterID: 89672e34-01c3-46c0-95bc-191aa74c91b3
ClusterVersion: Stable at "4.7.14"
ClusterOperators:
        All healthy and stable


[must-gather      ] OUT namespace/openshift-must-gather-hdv2t created
[must-gather      ] OUT clusterrolebinding.rbac.authorization.k8s.io/must-gather-
c5xq4 created
[must-gather      ] OUT pod for plug-in image registry.redhat.io/ocs4/ocs-must-
gather-rhel8:v4.7 created
[must-gather-kg88p] POD 2021-07-19T22:38:18.754683366Z creating helper pod
[must-gather-kg88p] POD 2021-07-19T22:38:19.444764613Z pod/must-gather-kg88p-
helper created
[must-gather-kg88p] POD 2021-07-19T22:38:19.447672259Z debugging node worker01
[must-gather-kg88p] POD 2021-07-19T22:38:19.447898403Z debugging node worker02
[must-gather-kg88p] POD 2021-07-19T22:38:19.448118974Z debugging node worker03
...output omitted...
```

References

For more information on Ceph architecture, refer to the Red Hat Ceph Storage 4 Architecture Guide at
https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/4/html-single/architecture_guide

For more information on Ceph troubleshooting, refer to the *Troubleshooting OpenShift Container Storage* chapter in the *Red Hat OpenShift Container Storage* documentation at
https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/html-single/troubleshooting_openshift_container_storage/index

Guided Exercise

Identifying Ceph Components for a Red Hat OpenShift Data Foundation Implementation

In this exercise, you will investigate the Ceph cluster by using logging and internal tools.

Outcomes

You should be able to view Ceph related logs and use the `must-gather` tool and Ceph CLI to obtain Ceph cluster information.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster is online and ready.

```
[student@workstation ~]$ lab start workloads-ceph
```

Instructions

1. Log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

2. Access the logs associated with the Ceph pods.

   2.1. View the Ceph pods by using the `oc get pods` command.

```
[student@workstation ~]$ oc get pods -n openshift-storage
...output omitted...
rook-ceph-crashcollector-worker01-8668f6597f-qlknp              1/1  Running
rook-ceph-crashcollector-worker02-598fcf5dc6-wd6jf              1/1  Running
rook-ceph-crashcollector-worker03-f68b85645-7d5cr               1/1  Running
rook-ceph-mds-ocs-storagecluster-cephfilesystem-a-5d64f4ccrsbcw  1/1  Running
rook-ceph-mds-ocs-storagecluster-cephfilesystem-b-7ccb5db468z25  1/1  Running
rook-ceph-mgr-a-7dfc47677-2qgqq                                 1/1  Running
rook-ceph-mon-a-585ffd45d5-7vvlv                                1/1  Running
rook-ceph-mon-b-5468b495d9-cd7zn                                1/1  Running
rook-ceph-mon-c-5db66d86d8-t8bwx                                1/1  Running
rook-ceph-operator-548bcdc79f-xcgjb                             1/1  Running
rook-ceph-osd-0-6cbf78cd77-t7dnr                                1/1  Running
```

```
rook-ceph-osd-1-7976d54f74-b5d98                                  1/1  Running
rook-ceph-osd-2-578d5498bd-6rvjn                                  1/1  Running
rook-ceph-osd-prepare-ocs-deviceset...                            0/1  Completed
rook-ceph-osd-prepare-ocs-deviceset...                            0/1  Completed
rook-ceph-osd-prepare-ocs-deviceset...                            0/1  Completed
rook-ceph-rgw-ocs-storagecluster-cephobjectstore...               1/1  Running
rook-ceph-rgw-ocs-storagecluster-cephobjectstore...               1/1  Running
```

2.2. View the OSD logs by specifying a `rook-ceph-osd` pod.

```
[student@workstation ~]$ oc logs rook-ceph-osd-0-6cbf78cd77-t7dnr \
  -c osd -n openshift-storage
...output omitted...
AddFile(L0 Files): cumulative 0, interval 0
AddFile(Keys): cumulative 0, interval 0
Cumulative compaction: 0.00 GB write, 0.00 MB/s write, 0.00 GB read, 0.00 MB/s
 read, 0.0 seconds
Interval compaction: 0.00 GB write, 0.00 MB/s write, 0.00 GB read, 0.00 MB/s read,
 0.0 seconds
Stalls(count): 0 level0_slowdown, 0 level0_slowdown_with_compaction, 0
 level0_numfiles, 0 level0_numfiles_with_compaction, 0 stop for pending_comp
action_bytes, 0 slowdown for pending_compaction_bytes, 0 memtable_compaction, 0
 memtable_slowdown, interval 0 total count
```

2.3. View the monitor logs by specifying a `rook-ceph-mon` pod.

```
[student@workstation ~]$ oc logs rook-ceph-mon-a-585ffd45d5-7vvlv \
  -c mon -n openshift-storage
...output omitted...
TiB / 1.5 TiB avail; 3.9 KiB/s rd, 37 KiB/s wr, 8 op/s
cluster 2021-07-19 22:00:11.865553 mgr.a (mgr.14203) 2386 : cluster [DBG] pgmap
 v2426: 176 pgs: 176 active+clean; 110 MiB data, 180 MiB used, 1.5
TiB / 1.5 TiB avail; 5.2 KiB/s rd, 39 KiB/s wr, 12 op/s
cluster 2021-07-19 22:00:13.866558 mgr.a (mgr.14203) 2387 : cluster [DBG] pgmap
 v2427: 176 pgs: 176 active+clean; 110 MiB data, 180 MiB used, 1.5
TiB / 1.5 TiB avail; 5.2 KiB/s rd, 18 KiB/s wr, 11 op/s
...output omitted...
```

2.4. View the Rook-Ceph Operator logs by specifying a `rook-ceph-operator` pod.

```
[student@workstation ~]$ oc logs rook-ceph-operator-548bcdc79f-xcgjb \
  -c rook-ceph-operator -n openshift-storage
...output omitted...
2021-07-19 20:39:00.358254 I | operator: looking for secret "rook-ceph-admission-
controller"
2021-07-19 20:39:00.360649 I | operator: secret "rook-ceph-admission-controller"
 not found. proceeding without the admission controller
2021-07-19 20:39:00.361658 I | operator: watching the current namespace for a ceph
 cluster CR
2021-07-19 20:39:00.362034 I | operator: setting up the controller-runtime manager
2021-07-19 20:39:00.363796 I | ceph-cluster-controller: ConfigMap "rook-ceph-
operator-config" changes detected. Updating configurations
2021-07-19 20:39:03.317982 I | ceph-cluster-controller: successfully started
```

```
2021-07-19 20:39:03.318073 I | ceph-cluster-controller: hotplug orchestration
 disabled
2021-07-19 20:39:03.318084 I | ceph-crashcollector-controller: successfully
 started
2021-07-19 20:39:03.318152 I | ceph-block-pool-controller: successfully started
...output omitted...
```

2.5. View the CephFS file storage logs by specifying a `csi-cephfsplugin` pod.

```
[student@workstation ~]$ oc logs csi-cephfsplugin-722b2 \
  -c csi-cephfsplugin -n openshift-storage
...output omitted...
I0719 20:39:28.221824   1 cephcsi.go:124] Driver version: release-4.6 and Git
 version: ce3f8de3748499a32ac53f179e7e7554d49d4a7e
I0719 20:39:28.222077   1 cephcsi.go:142] Initial PID limit is set to 1024
I0719 20:39:28.222131   1 cephcsi.go:151] Reconfigured PID limit to -1 (max)
I0719 20:39:28.222138   1 cephcsi.go:170] Starting driver type: cephfs with name:
 openshift-storage.cephfs.csi.ceph.com
I0719 20:39:28.239036   1 volumemounter.go:87] loaded mounter: kernel
E0719 20:39:28.239061   1 volumemounter.go:96] failed to run ceph-fuse exec:
 "ceph-fuse": executable file not found in $PATH
I0719 20:39:28.239595   1 server.go:118] Listening for connections on address:
 &net.UnixAddr{Name:"//csi/csi.sock", Net:"unix"}
I0719 20:39:28.891828   1 utils.go:159] ID: 1 GRPC call: /csi.v1.Identity/
GetPluginInfo
I0719 20:39:28.892820   1 utils.go:160] ID: 1 GRPC request: {}
...output omitted...
```

2.6. View the RBD plug-in logs by specifying a `csi-rbdplugin` pod.

```
[student@workstation ~]$ oc logs csi-rbdplugin-86dpc \
  -c csi-rbdplugin -n openshift-storage
...output omitted...
I0719 20:39:28.288333   23592 cephcsi.go:124] Driver version: release-4.6 and Git
 version: ce3f8de3748499a32ac53f179e7e7554d49d4a7e
I0719 20:39:28.288537   23592 cephcsi.go:142] Initial PID limit is set to 1024
I0719 20:39:28.288576   23592 cephcsi.go:151] Reconfigured PID limit to -1 (max)
I0719 20:39:28.288580   23592 cephcsi.go:170] Starting driver type: rbd with name:
 openshift-storage.rbd.csi.ceph.com
I0719 20:39:28.297442   23592 server.go:118] Listening for connections on address:
 &net.UnixAddr{Name:"//csi/csi.sock", Net:"unix"}
I0719 20:39:28.815489   23592 utils.go:159] ID: 1 GRPC call: /csi.v1.Identity/
GetPluginInfo
I0719 20:39:28.816213   23592 utils.go:160] ID: 1 GRPC request: {}
...output omitted...
```

3. Use the Ceph CLI to view information about Ceph.

3.1. To access the Ceph CLI, connect to a `rook-ceph-operator` pod using the `oc exec` command and enter a `bash` shell.

```
student@workstation ~]$ oc exec -it \
  pod/rook-ceph-operator-548bcdc79f-xcgjb -n openshift-storage \
  -c rook-ceph-operator -- /bin/bash
bash-4.4$
```

3.2. Obtain the cluster health with the `ceph` command and the `health` operator. The Ceph configuration is specified using the `-c` flag.

```
bash-4.4$ ceph -c /var/lib/rook/openshift-storage/openshift-storage.config health
HEALTH_OK
```

3.3. Obtain the cluster status using the `-s` flag.

```
bash-4.4$ ceph -c /var/lib/rook/openshift-storage/openshift-storage.config -s
  cluster:
    id:     92aecb6f-1ef7-427b-bff9-71d3307b4454
    health: HEALTH_OK

  services:
    mon: 3 daemons, quorum a,b,c (age 2h)
    mgr: a(active, since 2h)
    mds: ocs-storagecluster-cephfilesystem:1 {0=ocs-storagecluster-cephfilesystem-
b=up:active} 1 up:standby-replay
    osd: 3 osds: 3 up (since 2h), 3 in (since 2h)
    rgw: 2 daemons active (ocs.storagecluster.cephobjectstore.a,
 ocs.storagecluster.cephobjectstore.b)

  data:
    pools:   10 pools, 176 pgs
    objects: 331 objects, 133 MiB
    usage:   3.2 GiB used, 1.5 TiB / 1.5 TiB avail
    pgs:     176 active+clean

  io:
    client:   5.8 KiB/s rd, 8.2 KiB/s wr, 6 op/s rd, 5 op/s wr
```

3.4. Obtain the cluster disk usage by using the `df` operator.

```
bash-4.4$ ceph -c /var/lib/rook/openshift-storage/openshift-storage.config df
RAW STORAGE:
    CLASS     SIZE        AVAIL       USED        RAW USED     %RAW USED
    hdd       1.5 TiB     1.5 TiB     245 MiB      3.2 GiB          0.22
    TOTAL     1.5 TiB     1.5 TiB     245 MiB      3.2 GiB          0.22

POOLS:
    POOL                                ID  STORED   ..  USED     .. MAXAVAIL
    ocs-storagecluster-cephblockpool     1   69 MiB  ..  214 MiB  ..   424GiB
    ocs-storagecluster-cephobjectstore...  2      0 B  ..      0 B  ..   424GiB
    ocs-storagecluster-cephfilesystem-...  3  2.7 KiB  ..  1.5 MiB  ..   424GiB
    ocs-storagecluster-cephfilesystem-...  4      0 B  ..      0 B  ..   424GiB
    ocs-storagecluster-cephobjectstore...  5  2.8 KiB  ..  1.9 MiB  ..   424GiB
    ocs-storagecluster-cephobjectstore...  6  3.6 KiB  ..  6.6 MiB  ..   424GiB
```

```
    ocs-storagecluster-cephobjectstore...  7      0 B  ..      0 B  ..   424GiB
    ocs-storagecluster-cephobjectstore...  8      0 B  ..      0 B  ..   424GiB
    .rgw.root                              9  4.8 KiB  ..  2.8 MiB  ..   424GiB
    ocs-storagecluster-cephobjectstore... 10    1 KiB  ..  192 KiB  ..   424GiB
```

4. Use the `must-gather` tool to obtain Ceph cluster informaton.

   4.1. Execute `oc adm must-gather` with the options `--image`, specifying the `must-gather` image, and `--dest-dir`, specifying where the gathered information will be placed.

```
[student@workstation ~]$ oc adm must-gather \
  --image=registry.redhat.io/ocs4/ocs-must-gather-rhel8:v4.7 \
  --dest-dir=must-gather
[must-gather      ] OUT Using must-gather plug-in image: registry.redhat.io/ocs4/
ocs-must-gather-rhel8:v4.7
When opening a support case, bugzilla, or issue please include the following
 summary data along with any other requested information.
ClusterID: 89672e34-01c3-46c0-95bc-191aa74c91b3
ClusterVersion: Stable at "4.7.14"
ClusterOperators:
        All healthy and stable


[must-gather      ] OUT namespace/openshift-must-gather-hdv2t created
[must-gather      ] OUT clusterrolebinding.rbac.authorization.k8s.io/must-gather-
c5xq4 created
[must-gather      ] OUT pod for plug-in image registry.redhat.io/ocs4/ocs-must-
gather-rhel8:v4.7 created
[must-gather-kg88p] POD 2021-07-19T22:38:18.754683366Z creating helper pod
[must-gather-kg88p] POD 2021-07-19T22:38:19.444764613Z pod/must-gather-kg88p-
helper created
[must-gather-kg88p] POD 2021-07-19T22:38:19.447672259Z debugging node worker01
[must-gather-kg88p] POD 2021-07-19T22:38:19.447898403Z debugging node worker02
[must-gather-kg88p] POD 2021-07-19T22:38:19.448118974Z debugging node worker03
...output omitted...
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish workloads-ceph
```

This concludes the guided exercise.

Configuring Applications to Use Red Hat OpenShift Data Foundation File Storage

Objectives

After completing this section, you should be able to configure applications to use file storage provided by Red Hat OpenShift Data Foundation.

Introducing File Storage

File storage solutions provide the familiar directory and subdirectory structure found in many environments. Using file storage is ideal when applications generate or consume reasonable volumes of organized data. Applications that use file-based implementations are prevalent, easily managed, and provide an affordable storage solution.

File-based solutions are a good fit for data backup and archiving, due to their reliability, as well as file sharing and collaboration services. Most data centers provide file storage solutions, such as a network attached storage (NAS) cluster, for these common scenarios.

While both familiar and prevalent, file storage solutions are not ideal for all application scenarios. One particular pitfall of file storage is the poor handling of very large data sets or unstructured data.

File Storage Scenarios

As previously mentioned, NAS solutions can provide file-based storage to applications within the same data center. This approach is common to many application architectures, such as:

- Web server content
- File share services
- FTP storage
- Backup archives

These applications take advantage of data reliability and the ease of file sharing available by using file storage. Additionally, for file storage data, the OS and filesystem handle the locking and caching of the files.

Note

Although OpenShift Data Foundation makes managing the various storage classes an easy configuration setting for applications, nothing validates the architectural storage choices for your deployment. Always consider the proper storage class for each component of your application to obtain the most performant architecture.

For example, Hadoop workloads, comprised of large sets of unstructured data, are not a good candidate for using file storage solutions.

Red Hat OpenShift Data Foundation File Storage Components

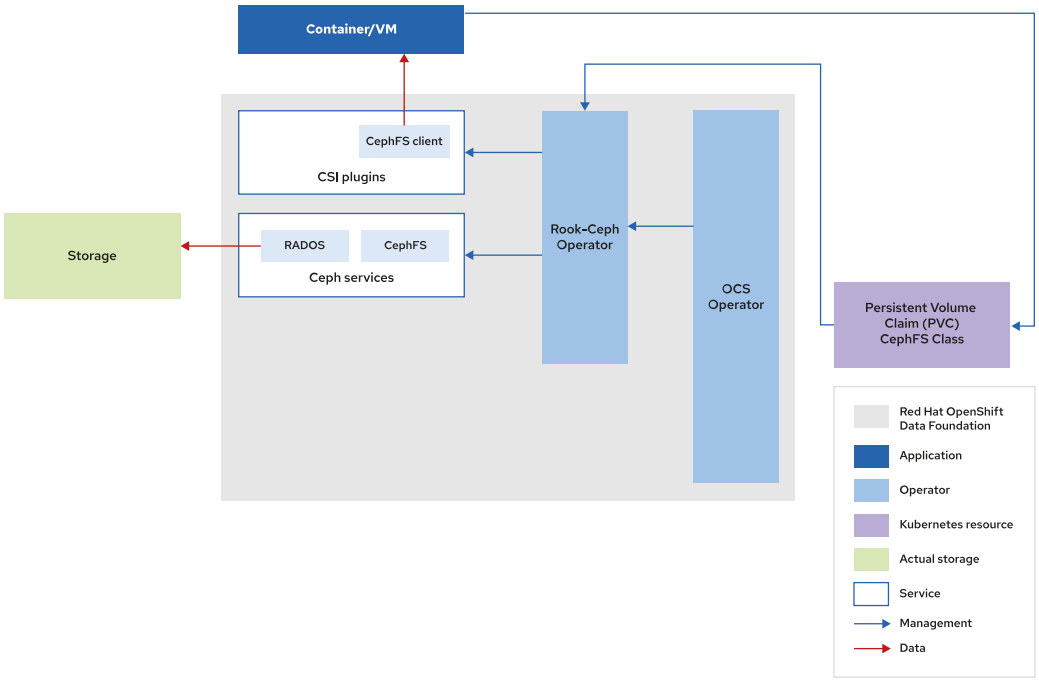


Figure 3.2: Red Hat OpenShift Data Foundation file storage components using CephFS

The preceding image shows the relationship between these components:

- The OpenShift Container Storage (OCS) operator that installs Red Hat OpenShift Data Foundation
- Ceph operator and deployed services
- Container Storage Interface (CSI) for CephFS
- Storage pools fulfilling the PVCs

Configuring a Persistent Volume Claim

The OpenShift Data Foundation file storage class that is used in application workloads is specified in the storage requirements within your custom resource definitions. Configure the file storage class `ocs-storagecluster-cephfs` to allow multiple nodes to access the storage volume by setting the `accessModes` value to `ReadWriteMany`.

The OpenShift Data Foundation File Storage Class

OpenShift Data Foundation provides the file storage class `ocs-storagecluster-cephfs` through the CSI for CephFS.

```
[user@demo ~]$ oc get storageclasses.storage.k8s.io
NAME                    PROVISIONER                             RECLAIM
 VOLBINDMODE VOLEXP

ocs-storagecluster-cephfs openshift-storage.cephfs.csi.ceph.com  Delete
 Immediate   true
```

Use the `oc describe` command to view the details of the `ocs-storagecluster-cephfs` storage class.

```
[user@demo ~]$ oc describe storageclass ocs-storagecluster-cephfs
Name:               ocs-storagecluster-cephfs
IsDefaultClass:     No
Annotations:        description=Provides RWO and RWX Filesystem volumes
Provisioner:        openshift-storage.cephfs.csi.ceph.com
Parameters:         clusterID=openshift-storage,csi.storage.k8s.io/controller-
expand-secret-name=rook-csi-cephfs-provisioner,csi.storage.k8s.io/controller-
expand-secret-namespace=openshift-storage,csi.storage.k8s.io/node-stage-
secret-name=rook-csi-cephfs-node,csi.storage.k8s.io/node-stage-secret-
namespace=openshift-storage,csi.storage.k8s.io/provisioner-secret-name=rook-csi-
cephfs-provisioner,csi.storage.k8s.io/provisioner-secret-namespace=openshift-
storage,fsName=ocs-storagecluster-cephfilesystem
AllowVolumeExpansion:  True
MountOptions:       <none>
ReclaimPolicy:      Delete
VolumeBindingMode:  Immediate
Events:             <none>
```

Example Persistent Volume Claim

The following file shows an example custom resource definition for the `production-application` using a `10Gi` volume provided by the OpenShift Data Foundation `ocs-storagecluster-cephfs` file storage class.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: production-application
spec:
  accessModes:
  - ReadWriteMany
  storageClassName: ocs-storagecluster-cephfs
  resources:
    requests:
      storage: 10Gi
```

Notice that the `accessModes` is set to `ReadWriteMany` to accommodate the common scenario of sharing centralized files with numerous application instances. For example, this setting allows multiple web servers to share a common file storage data set.

References

For more information, refer to the product documentation for *Red Hat OpenShift Data Foundation* at
https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/

Guided Exercise

Configuring Applications to Use Red Hat OpenShift Data Foundation File Storage

In this exercise, you will configure an application to use file storage that is provided by OpenShift Data Foundation.

Outcomes

You should be able to:

- Deploy an application that uses a persistent volume (PV) as backing storage.
- Interact with the persistent storage from the example application.
- Access the PV data with another application.

Before You Begin

To perform this exercise, ensure you have:

- A running OpenShift cluster.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start workloads-file
```

Instructions

1. Deploy the example application in the OpenShift cluster.

   1.1. Log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

   1.2. Create a new project.

```
[student@workstation ~]$ oc new-project image-tool
Now using project "image-tool" on server "https://api.ocp4.example.com:6443".
...output omitted...
```

2. Deploy the application.

   2.1. Change to the `~/DO370/labs/workloads-file` directory.

```
[student@workstation ~]$ cd ~/DO370/labs/workloads-file
```

2.2. Create the resources for the test application.

```
[student@workstation workloads-file]$ oc apply -f serviceaccount.yaml \
  -f deployment.yaml -f service.yaml -f route.yaml
serviceaccount/image-tool created
deployment.apps/image-tool-pvc created
service/image-tool-pvc created
route.route.openshift.io/image-tool-pvc created
```

2.3. Verify that the pod is running.

```
[student@workstation workloads-file]$ oc get pods -l app=image-tool-pvc
NAME                              READY   STATUS    RESTARTS   AGE
image-tool-pvc-79c9bb56dc-9ph72   1/1     Running   0          5s
```

Note

You might need to run the command multiple times until the desired condition is reached.

2.4. Retrieve the hostname for the route.

```
[student@workstation workloads-file]$ oc get route/image-tool-pvc \
  -o jsonpath='{.spec.host}{"\n"}'
image-tool-pvc-image-tool.apps.ocp4.example.com
```

2.5. Open the URL in a web browser.

- https://image-tool-pvc-image-tool.apps.ocp4.example.com/

2.6. View the the deployment logs.

```
[student@workstation workloads-file]$ oc logs deployment/image-tool-pvc | head
```

2.7. Verify the message that says the application is using **ephemeral** storage. This means that the application container does not have a PV mounted in `/var/storage`.

```
[2021-04-19 22:40:08,101] INFO in app: Using EPHEMERAL as backing storage
[2021-04-19 22:40:08,102] INFO in app: Serving files from /var/storage
[2021-04-19 22:40:08 +0000] [1] [INFO] Starting gunicorn 20.0.4
[2021-04-19 22:40:08 +0000] [1] [INFO] Listening at: http://0.0.0.0:5000 (1)
[2021-04-19 22:40:08 +0000] [1] [INFO] Using worker: sync
[2021-04-19 22:40:08 +0000] [15] [INFO] Booting worker with pid: 15
[2021-04-19 22:51:45,452] INFO in app: FILE: Listing directory
...output omitted...
```

3. Configure persistent storage for the application.

3.1. List the storage classes available in the cluster, and identify the storage class used for CephFS.

```
[student@workstation workloads-file]$ oc get storageclasses -o name
storageclass.storage.k8s.io/lso-volumeset
storageclass.storage.k8s.io/nfs-storage
storageclass.storage.k8s.io/ocs-storagecluster-ceph-rbd
storageclass.storage.k8s.io/ocs-storagecluster-ceph-rgw
storageclass.storage.k8s.io/ocs-storagecluster-cephfs
storageclass.storage.k8s.io/openshift-storage.noobaa.io
```

3.2. Edit the `deployment.yaml` file to match the following lines.

```
        volumeMounts:
        - name: image-tool-storage
          mountPath: "/var/storage"
      volumes:
      - name: image-tool-storage
        persistentVolumeClaim:
          claimName: image-tool
```

Warning

Be sure to remove the # character only and do not remove any white spaces that might affect the YAML indentation.

There is a `deployment.yaml` file in the `solutions` directory in case you want to check for syntax errors.

3.3. Edit the `pvc.yaml` file to match the following specification.

- Set the access mode to `ReadWriteMany`
- Set the storage class to `ocs-storagecluster-cephfs`
- Adjust the storage request to `1Gi`

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: image-tool
spec:
  accessModes:
  - ReadWriteMany
  storageClassName: ocs-storagecluster-cephfs
  resources:
    requests:
      storage: 1Gi
```

Note

There is a `pvc.yaml` file in the `solutions` directory in case you want to check for syntax errors.

3.4. Create a PVC resource and apply the new version of the deployment.

```
[student@workstation workloads-file]$ oc apply -f pvc.yaml -f deployment.yaml
persistentvolumeclaim/image-tool created
deployment.apps/image-tool-pvc configured
```

3.5. List the pods, and then wait until there is only one replica with a running status.

```
[student@workstation workloads-file]$ oc get pods
NAME                              READY   STATUS        RESTARTS   AGE
image-tool-pvc-77f6f4465d-lhtw8   1/1     Running       0          15s
```

Note

You might need to run the command multiple times until the desired condition is reached.

3.6. Verify that the image-tool PVC is bound.

```
[student@workstation workloads-file]$ oc get persistentvolumeclaims
NAME         STATUS  VOLUME   CAPACITY  ACCESS MODES  STORAGECLASS
image-tool   Bound   pvc-...  1Gi       RWX           ocs-storagecluster-cephfs
```

3.7. List the mounts and volumes used by the pods in the `image-tool-pvc` deployment.

```
[student@workstation workloads-file]$ oc describe deployment/image-tool-pvc
...output omitted...
  Containers:
   image-tool-pvc:
    Image:        quay.io/redhattraining/image-tool:latest
    Port:         5000/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:
      /var/storage from image-tool-storage (rw)
  Volumes:
   image-tool-storage:
    Type:       PersistentVolumeClaim (a reference to a PersistentVolumeClaim in
 the same namespace)
    ClaimName:  image-tool
    ReadOnly:   false
...output omitted...
```

3.8. Verify that `/var/storage` is mounted in the container.

```
[student@workstation workloads-file]$ oc exec -it deployment/image-tool-pvc -- \
  df -h /var/storage
Filesystem                              Size   Used   Avail   Use%   Mounted on
172.30.200.149:6789,172.30.41.15:...    1.0G      0    1.0G     0%   /var/storage
```

Note

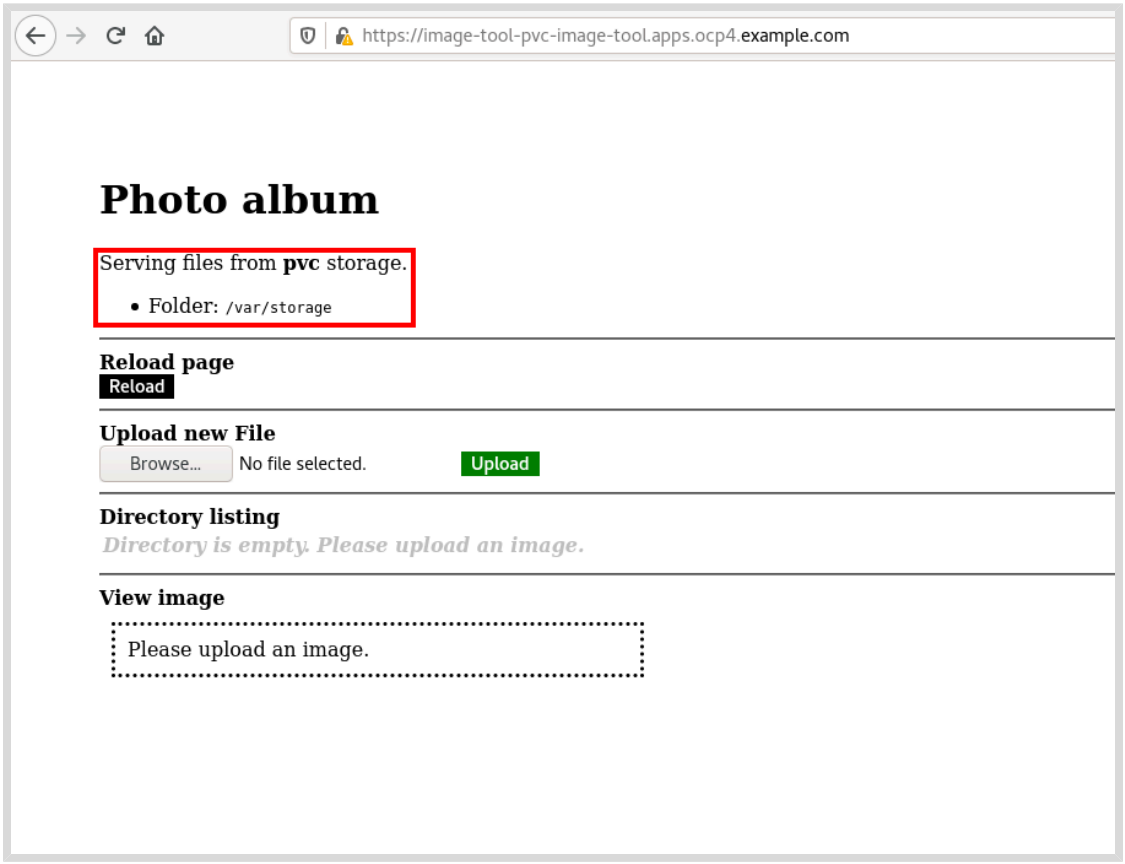
If the command returns an output similar to the following, then you have to check that the `deployment.yaml` file has been edited correctly and the new version of the deployment is running.

```
[student@workstation workloads-file]$ oc exec -it deployment/image-tool-pvc \
  -- df -h /var/storage
Filesystem      Size   Used   Avail   Use%   Mounted on
overlay          80G    17G     64G    21%   /
```

4. Access the new version of the application.

   4.1. Refresh the page in the web browser. The application displays a message specifying what type of storage it is using.

   - https://image-tool-pvc-image-tool.apps.ocp4.example.com/



   4.2. Verify the logs again to see if the application is using persistent storage from a PVC.

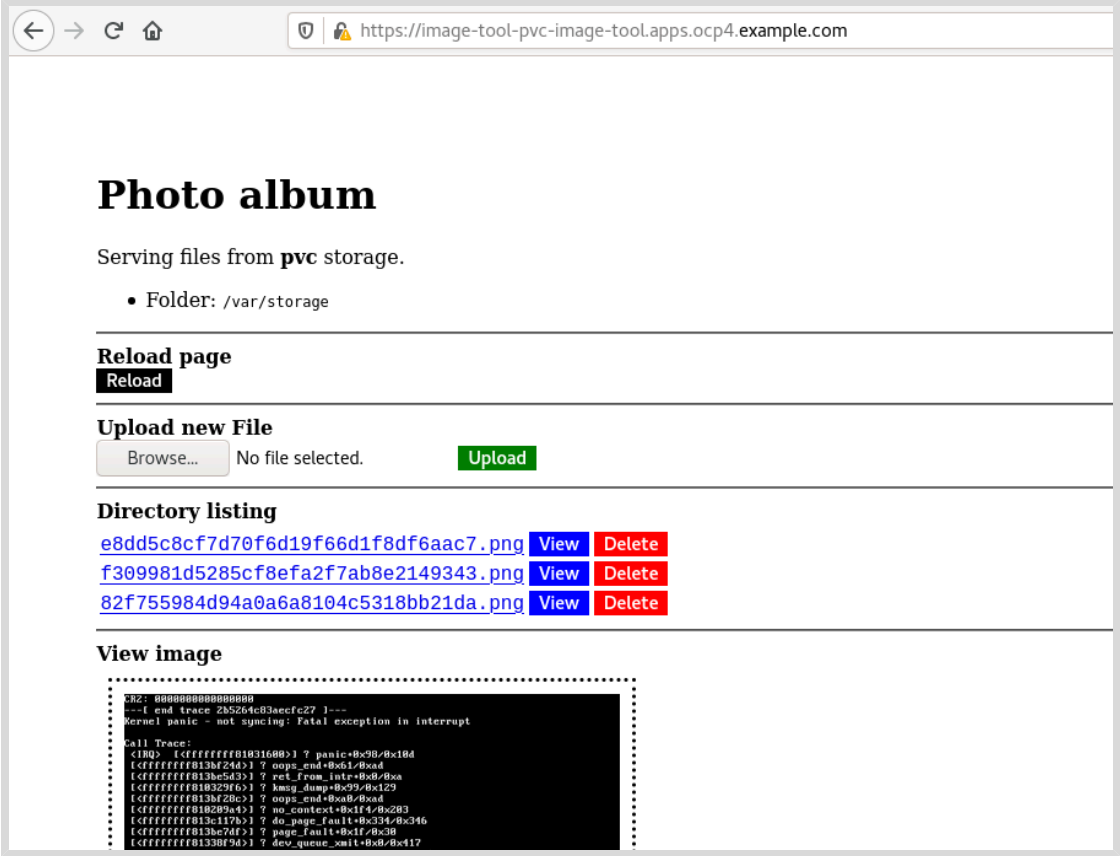
```
[student@workstation workloads-file]$ oc logs deployment/image-tool-pvc | head
[2021-04-19 23:02:08,630] INFO in app: Using PVC as backing storage
[2021-04-19 23:02:08,630] INFO in app: Serving files from /var/storage
[2021-04-19 23:02:08 +0000] [1] [INFO] Starting gunicorn 20.0.4
[2021-04-19 23:02:08 +0000] [1] [INFO] Listening at: http://0.0.0.0:5000 (1)
```

```
[2021-04-19 23:02:08 +0000] [1] [INFO] Using worker: sync
[2021-04-19 23:02:08 +0000] [12] [INFO] Booting worker with pid: 12
[2021-04-19 23:03:00,623] INFO in app: FILE: Listing directory
...output omitted...
```

4.3. Click **Browse**, select an image from the `~/DO370/labs/workloads-file/pictures` directory, and then click **Open**. Click **Upload** to send the file to the application.

Repeat this step for all the images in the `~/DO370/labs/workloads-file/pictures` directory.

4.4. Click **View** to the right of the image name to view it in the lower pane.



4.5. Click **Delete** for any of the uploaded images that you do not want to keep.

4.6. Switch to the terminal window and view the deployment logs.

```
[student@workstation workloads-file]$ oc logs deployment/image-tool-pvc | \
  grep -i FILE
[2021-04-19 23:02:08,630] INFO in app: Serving files from /var/storage
[2021-04-19 23:03:00,623] INFO in app: FILE: Listing directory
[2021-04-19 23:10:51,494] INFO in app: Trying to upload file
[2021-04-19 23:10:51,495] INFO in app: FILE: Write: /var/storage/
f309981d5285cf8efa2f7ab8e2149343.png
[2021-04-19 23:10:51,496] INFO in app: File uploaded
[2021-04-19 23:10:51,603] INFO in app: FILE: Listing directory
```

```
[2021-04-19 23:14:47,947] INFO in app: FILE: Read: /var/storage/
f309981d5285cf8efa2f7ab8e2149343.png
[2021-04-19 23:14:54,961] INFO in app: FILE: Delete:
 f309981d5285cf8efa2f7ab8e2149343.png
[2021-04-19 23:14:54,987] INFO in app: FILE: Listing directory
...output omitted...
```

5. Delete and recreate the application to test persistent storage in the new pods.

5.1. Delete the deployment. The application data is stored in a PV and it will be available when new replica pods are executed.

```
[student@workstation workloads-file]$ oc delete deployment/image-tool-pvc
deployment.apps "image-tool-pvc" deleted

[student@workstation workloads-file]$ oc get pods -l app=image-tool-pvc
No resources found in image-tool namespace.
```

5.2. Verify that the image-tool PVC is still bound. The PVC is not deleted when the deployment pods are terminated.

```
[student@workstation workloads-file]$ oc get persistentvolumeclaims
NAME         STATUS  VOLUME   CAPACITY  ACCESS MODES  STORAGECLASS
image-tool   Bound   pvc-...  1Gi       RWX           ocs-storagecluster-cephfs
```

5.3. Create the deployment again and wait until all the new pods are running.

```
[student@workstation workloads-file]$ oc apply -f deployment.yaml
deployment.apps/image-tool-pvc created

[student@workstation workloads-file]$ oc get pods -l app=image-tool-pvc
NAME                              READY   STATUS    RESTARTS   AGE
image-tool-pvc-77f6f4465d-2qbfl   1/1     Running   0          22s
```

Note

You might need to run the `oc get pods` command multiple times until the desired condition is reached.

5.4. Refresh the page in the web browser and review the message that says the application is using **PVC** storage.

- https://image-tool-pvc-image-tool.apps.ocp4.example.com/

Verify that the images you uploaded previously are shown.

6. Access the files on the persistent storage.

6.1. Enter the pod of the `image-tool-pvc` deployment.

```
[student@workstation workloads-file]$ oc exec -it deployment/image-tool-pvc -- \
  /bin/bash
...output omitted...
(app-root) bash-4.4$
```

6.2. Verify that `/var/storage` is mounted in the container.

```
(app-root) bash-4.4$ df -h /var/storage
Filesystem                              Size   Used   Avail   Use%   Mounted on
172.30.200.149:6789,172.30.41.15:...    1.0G      0    1.0G     0%   /var/storage
```

6.3. List the files in the storage folder.

```
(app-root) sh-4.4$ ls -la /var/storage
total 42
drwxrwxrwx. 2 root       root     3 Apr 19 23:11 .
drwxr-xr-x. 1 root       root    21 Mar 27 00:21 ..
-rw-r--r--. 1 1000640000 root 16773 Apr 19 23:11
 82f755984d94a0a6a8104c5318bb21da.png
-rw-r--r--. 1 1000640000 root 19348 Apr 19 23:11
 e8dd5c8cf7d70f6d19f66d1f8df6aac7.png
-rw-r--r--. 1 1000640000 root  5855 Apr 19 23:10
 f309981d5285cf8efa2f7ab8e2149343.png
```

Note
If the folder is empty, try uploading some images from the application web page.

6.4. Exit the pod.

```
(app-root) bash-4.4$ exit
```

7. Access the files in the PV with a second application.

7.1. Edit the `nginx.yaml` file to match the following lines. The PV is mounted in `/opt/app-root/src/www` and nginx is configured to serve files from that directory. The `image-tool` PVC is the same used by the other application.

```
      volumeMounts:
      - name: persistent-storage
        mountPath: "/opt/app-root/src/www"
    volumes:
    - name: persistent-storage
      persistentVolumeClaim:
        claimName: image-tool
```

Note

Be sure to remove the # character only and do not remove any white spaces that might affect the YAML indentation.

There is a `nginx.yaml` file in the `solutions` directory in case you want to check for syntax errors.

7.2. Deploy a second application that mounts the PV to access the files.

```
[student@workstation workloads-file]$ oc apply -f nginx.yaml
deployment.apps/nginx created
service/nginx created
route.route.openshift.io/nginx created
```

7.3. Verify that the pods are running.

```
[student@workstation workloads-file]$ oc get pods -l app=nginx
NAME                     READY   STATUS    RESTARTS   AGE
nginx-68ff7c859-c7zvw    1/1     Running   0          36s
```

Note

You might need to run the command multiple times until the desired condition is reached.

7.4. List the mounts and volumes used by the pods in the `nginx` deployment.

```
[student@workstation workloads-file]$ oc describe deployment/nginx
...output omitted...
  Containers:
   nginx:
    Image:        quay.io/redhattraining/nginx:autoindex
    Port:         8080/TCP
    Host Port:    0/TCP
    Environment:  <none>
    Mounts:
      /opt/app-root/src/www from persistent-storage (rw)
  Volumes:
   persistent-storage:
    Type:       PersistentVolumeClaim (a reference to a PersistentVolumeClaim in
 the same namespace)
    ClaimName:  image-tool
    ReadOnly:   false
...output omitted...
```

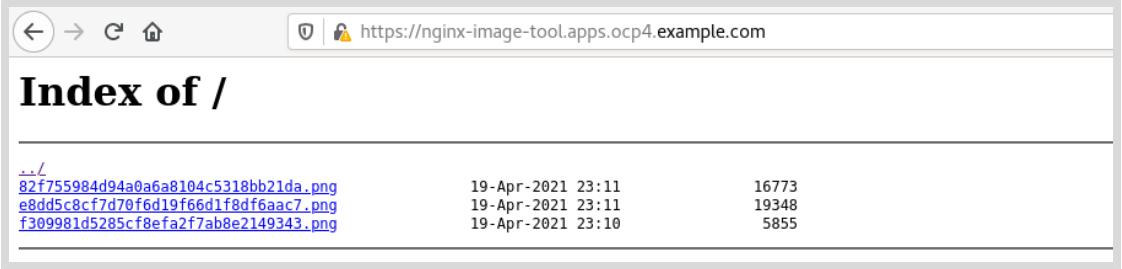
7.5. Retrieve the hostname for the route.

```
[student@workstation workloads-file]$ oc get route/nginx \
  -o jsonpath='{.spec.host}{"\n"}'
nginx-image-tool.apps.ocp4.example.com
```

7.6. Open the URL in a new web browser tab.

- https://nginx-image-tool.apps.ocp4.example.com/

7.7. Check that the files listed in the application are the ones that were previously shown.



7.8. List the files in the `/opt/app-root/src/www` directory.

```
[student@workstation workloads-file]$ oc exec -it deployment/nginx -- \
  ls -l /opt/app-root/src/www
total 42
drwxrwxrwx. 2 root       root     3 Apr 19 23:11 .
drwxr-xr-x. 1 root       root    21 Mar 27 00:21 ..
-rw-r--r--. 1 1000640000 root 16773 Apr 19 23:11
 82f755984d94a0a6a8104c5318bb21da.png
-rw-r--r--. 1 1000640000 root 19348 Apr 19 23:11
 e8dd5c8cf7d70f6d19f66d1f8df6aac7.png
-rw-r--r--. 1 1000640000 root  5855 Apr 19 23:10
 f309981d5285cf8efa2f7ab8e2149343.png
```

7.9. Close the web browser and change to the `/home/student` directory.

```
[student@workstation workloads-file]$ cd
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish workloads-file
```

This concludes the guided exercise.

Configuring Applications to Use Red Hat OpenShift Data Foundation Block Storage

Objectives

After completing this section, you should be able to configure applications to use block storage provided by Red Hat OpenShift Data Foundation.

Block Storage Concepts

Block storage solutions, such as storage area network (SAN) and iSCSI technologies, provide access to raw block devices for application storage. These block devices function as independent storage volumes, such as the physical drives found in servers, and typically require formatting and mounting for application access.

Using block storage is ideal when applications require faster access for optimizing computationally heavy data workloads. Applications using block-level storage implementations gain efficiencies by communicating at the raw device level, instead of relying on OS layer access.

Block-level approaches allow data to be distributed on blocks across the storage volume. Blocks also use basic metadata, including a unique identification number for each block of data, for quick retrieval and reassembly of blocks for reading.

Block Storage Use Cases

SAN and iSCSI technologies provide applications with block-level volumes from network-based storage pools. Using block-level access to storage volumes is a common approach for application architectures, such as:

- SQL Databases (single node access)
- Virtual Machines (multi-node access)
- High-performance data access
- Server-side processing applications
- Multiple block device RAID configurations Application storage that uses several block devices in a RAID configuration benefits from the data integrity and performance that the various arrays provide.

Note

Although OpenShift Data Foundation makes managing the various storage classes an easy configuration setting for applications, nothing validates the architectural storage choices for your deployment. Please consider the proper storage class for each component of your application to attain the most performant architecture.

Red Hat OpenShift Data Foundation Block Storage Components

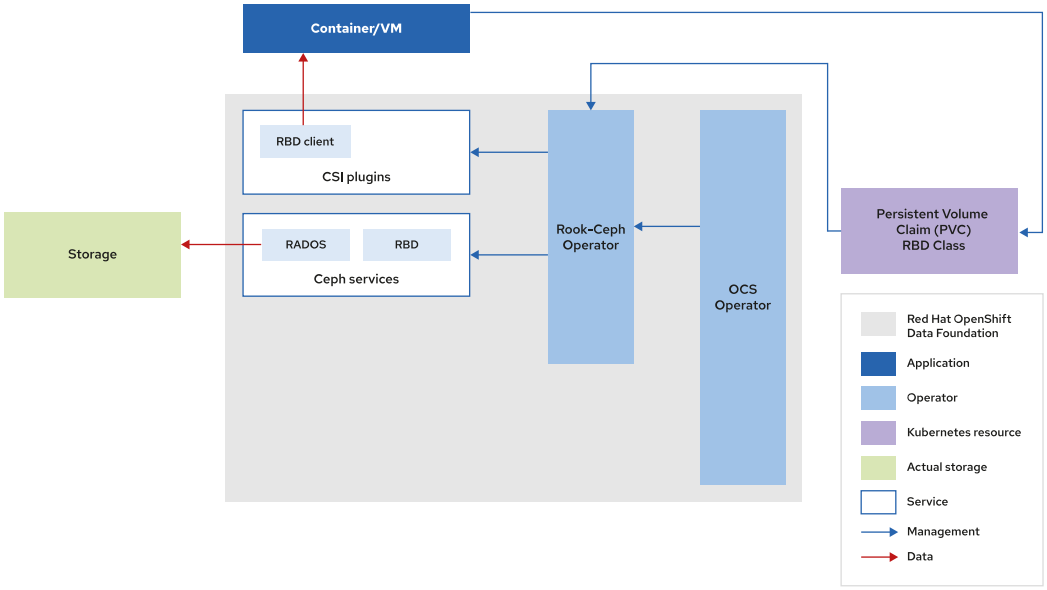


Figure 3.6: Red Hat OpenShift Data Foundation block storage components using RADOS block devices

The preceding image shows the relationship between these components:

- OpenShift Data Foundation OpenShift Container Storage operator
- Ceph operator and deployed services
- Container storage interface (CSI) for RADOS block device (RBD)
- Storage pools fulfilling the PVCs

Configuring a Persistent Volume Claim

The OpenShift Data Foundation block storage class that is used in application workloads is specified in the storage requirements within your custom resource definitions. Configure the file storage class `ocs-storagecluster-cephfs` to allow multiple nodes to access the storage volume by setting the `accessModes` value to `ReadWriteMany`.

The Red Hat OpenShift Data Foundation File Storage Class

OpenShift Data Foundation provides the file storage class `ocs-storagecluster-ceph-rbd` through the CSI for RBD.

```
[user@demo ~]$ oc get storageclasses
NAME                          PROVISIONER                             RECLAIM
 VOLBINDMODE VOLEXP

ocs-storagecluster-ceph-rbd   openshift-storage.rbd.csi.ceph.com  Delete
 Immediate   true
```

Use the `oc describe` command to view the details of the `ocs-storagecluster-ceph-rbd` storage class.

```
[user@demo ~]$ oc describe sc ocs-storagecluster-ceph-rbd
Name:                ocs-storagecluster-ceph-rbd
IsDefaultClass:      No
Annotations:         description=Provides RWO Filesystem volumes, and RWO and RWX
 Block volumes
Provisioner:         openshift-storage.rbd.csi.ceph.com
Parameters:          clusterID=openshift-storage,csi.storage.k8s.io/
                     controller-expand-secret-name=rook-csi-rbd-
provisioner,csi.storage.k8s.io/
                     controller-expand-secret-namespace=openshift-
storage,csi.storage.k8s.io/
                     fstype=ext4,csi.storage.k8s.io/
                     node-stage-secret-name=rook-csi-rbd-node,csi.storage.k8s.io/
                     node-stage-secret-namespace=openshift-
storage,csi.storage.k8s.io/
                     provisioner-secret-name=rook-csi-rbd-
provisioner,csi.storage.k8s.io/
                     provisioner-secret-namespace=openshift-
storage,imageFeatures=layering,imageFormat=2,pool=ocs-storagecluster-cephblockpool
AllowVolumeExpansion:  True
MountOptions:        <none>
ReclaimPolicy:       Delete
VolumeBindingMode:   Immediate
Events:              <none>
```

Example Persistent Volume Claim

The following file shows an example custom resource definition for the `production-application` using a `10Gi` volume provided by the OpenShift Data Foundation `ocs-storagecluster-ceph-rbd` block storage class.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: production-application
spec:
  accessModes:
  - ReadWriteOnce
  storageClassName: ocs-storagecluster-ceph-rbd
  resources:
    requests:
      storage: 10Gi
```

Notice that the `accessModes` is set to `ReadWriteOnce` to provide security and data integrity by allowing access to a single node. For example, this setting is appropriate for the data volume of a SQL database using block storage.

References

For more information, refer to the product documentation for *Red Hat OpenShift Data Foundation* at
https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/

Guided Exercise

Configuring Applications to Use Red Hat OpenShift Data Foundation Block Storage

In this exercise, you will configure an application to use block storage provided by Red Hat OpenShift Data Foundation.

Outcomes

You should be able to:

- Deploy databases that use block storage provided by ODF.

Before You Begin

To perform this exercise, ensure that you have:

- A running OpenShift cluster.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start workloads-block
```

Instructions

1. Log in to your OpenShift cluster as the `admin` user with the `redhat` password and create a new `workloads-block` project.

   1.1. Log in to the OpenShift cluster.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  https://api.ocp4.example.com:6443
Login successful.
...output omitted...
```

   1.2. Create a new `workloads-block` project.

```
[student@workstation ~]$ oc new-project workloads-block
Now using project "workloads-block" on server "https://api.ocp4.example.com:6443".
...output omitted...
```

2. Create a block storage persistent volume claim.

   2.1. Inspect the contents of the `~/DO370/labs/workloads-block/pvc.yaml` file and give the name `famous-quotes-db` to the persistent volume claim. Verify that

the `storageClassName` is the block storage class `ocs-storagecluster-ceph-rbd` provided by ODF.

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: famous-quotes-db
spec:
  accessModes:
  - ReadWriteOnce
  storageClassName: ocs-storagecluster-ceph-rbd
  resources:
    requests:
      storage: 1Gi
```

2.2. Use the `oc apply` command to create the `famous-quotes-db` persistent volume claim.

```
[student@workstation ~]$ oc apply -f ~/DO370/labs/workloads-block/pvc.yaml
persistentvolumeclaim/famous-quotes-db created
```

2.3. Verify that the `famous-quotes-db` persistent volume claim is successfully created.

```
[student@workstation ~]$ oc describe pvc/famous-quotes-db
Name:          famous-quotes-db
Namespace:     workloads-block
StorageClass:  ocs-storagecluster-ceph-rbd
Status:        Bound
...output omitted...
```

3. Deploy a MariaDB database.

3.1. Use the `mariadb-persistent` template to deploy a MariaDB instance.

```
[student@workstation ~]$ oc process mariadb-persistent -n openshift \
  -p DATABASE_SERVICE_NAME=famous-quotes-db     \
  -p MYSQL_USER=myuser -p MYSQL_PASSWORD=r3dh4t \
  -p MYSQL_DATABASE=quotes                      \
  | oc create -f -
secret/famous-quotes-db created
service/famous-quotes-db created
deploymentconfig.apps.openshift.io/famous-quotes-db created
Error from server (AlreadyExists): persistentvolumeclaims "famous-quotes-db"
 already exists
```

Note

This template attempts to create a persistent volume claim, ignore the `(AlreadyExists)`, this is an expected message that means that the `famous-quotes-db` PVC already exists.

4. Deploy the `famous-quotes` application.

4.1. Copy the `~/DO370/labs/workloads-block/famous-quotes.yaml` file to the `student` home directory.

```
[student@workstation ~]$ cp ~/DO370/labs/workloads-block/famous-quotes.yaml ./
```

4.2. Edit the deployment in the `famous-quotes.yaml` file to match the following settings:

```
...output omitted...
      - env:
        - name: QUOTES_DATABASE
          value: quotes
        - name: QUOTES_HOSTNAME
          value: famous-quotes-db
        - name: QUOTES_USER
          value: myuser
...output omitted...
```

4.3. Use the `oc apply` command to create the `famous-quotes` deployment.

```
[student@workstation ~]$ oc apply -f famous-quotes.yaml
```

5. Verify that the application working.

```
[student@workstation ~]$ curl -k \
  https://famous-quotes-workloads-block.apps.ocp4.example.com
<html>
        <head>
        <title>Quotes</title>
    </head>
    <body>

        <h1>Quote List</h1>

            <ul>

                    <li>1: When words fail, music speaks.
- William Shakespeare
</li>
...output omitted...
```

6. Inspect the `famous-quotes-db` deployment configuration to verify the use of the `famous-quotes-db` PVC.

```
[student@workstation ~]$ oc exec dc/famous-quotes-db -- df -h | \
  grep /var/lib/mysql/data
/dev/rbd0       976M  180M  780M  19% /var/lib/mysql/data
```

7. Remove the `workloads-block` project to clean up the environment.

7.1. Use the `oc project` command to change to the default project.

```
[student@workstation workloads-block]$ oc project default
```

7.2. Remove the `workloads-block` project.

```
[student@workstation workloads-block]$ oc delete project workloads-block
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish workloads-block
```

This concludes the guided exercise.

Configuring Custom Storage Classes

Objectives

After completing this section, you should be able to configure a new storage class with custom settings.

Introducing Storage Classes

Storage classes are Kubernetes objects that persistent volumes (PV) and persistent volume claims (PVC) use to obtain storage from the assigned provisioner. Administrators use custom storage classes to define settings and policies for PVs and PVCs, such as Ceph pools, compression, and reclaim policies.

The following fields provide basic storage class configuration:

- `provisioner`: Identifies the source of the storage (for example, Ceph or AWS).
- `parameters`: Enables configuration for the provisioner.
- `reclaimPolicy`: Defines what to do when the storage is released.

Default Storage Classes

The Red Hat OpenShift Data Foundation storage operator uses the `lso-volumeset` storage class. This class is created by the local storage operator (LSO) to make disk devices available to the Rook-Ceph operator as PVs. The `lso-volumeset` class is required for internal deployments.

The OpenShift Data Foundation storage operator creates the following storage classes by default:

- `ocs-storagecluster-ceph-rbd`: This storage class provides block storage on demand. Extending provisioner and reclaimPolicy settings, this class enables the configuration of the file system type, Ceph pool, Ceph image format, and image features. This class uses the `openshift-storage.rbd.csi.ceph.com` provisioner.
- `ocs-storagecluster-ceph-rgw`: This storage class provides block storage with a S3 compatible storage bucket. The Rook-Ceph operator creates a `CephObjectStorage` custom resource named `ocs-storagecluster-cephobjectstore` that provides the block storage for the PVCs.
- `ocs-storagecluster-cephfs`: This storage class provides file storage for the pods by using the `openshift-storage.cephfs.csi.ceph.com` provisioner.
- `openshift-storage.noobaa.io`: This storage class provides NooBaa block storage, an agnostic object storage layer that enables plugging in different types of cloud storage to OpenShift Data Foundation. This storage class uses the `openshift-storage.noobaa.io/obc` provisioner.

The storage classes created by OpenShift Data Foundation are controlled and owned by the storage operator and they can not be deleted or modified except for adding labels or annotations.

Custom Storage Classes

The storage classes that are created by OpenShift Data Foundation might not be sufficient for all use cases. You can define custom storage classes that provide settings more suitable for your needs.

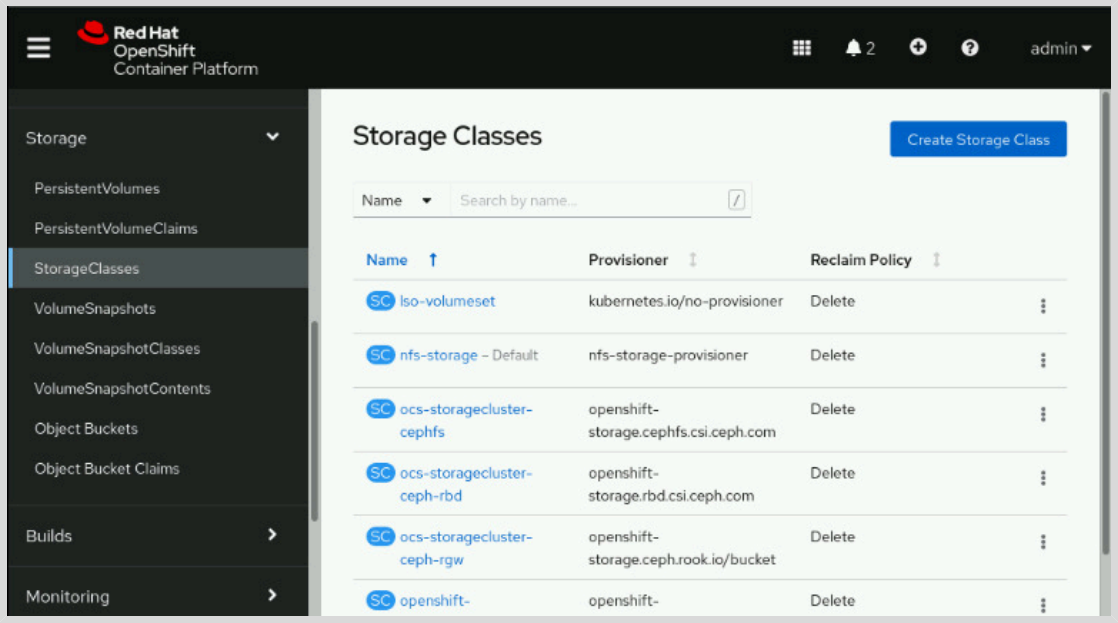
Custom storage classes can define parameters such as file system type, storage pool, compression, reclaim policy, and provisioner, among others.

Defining Custom Storage Classes

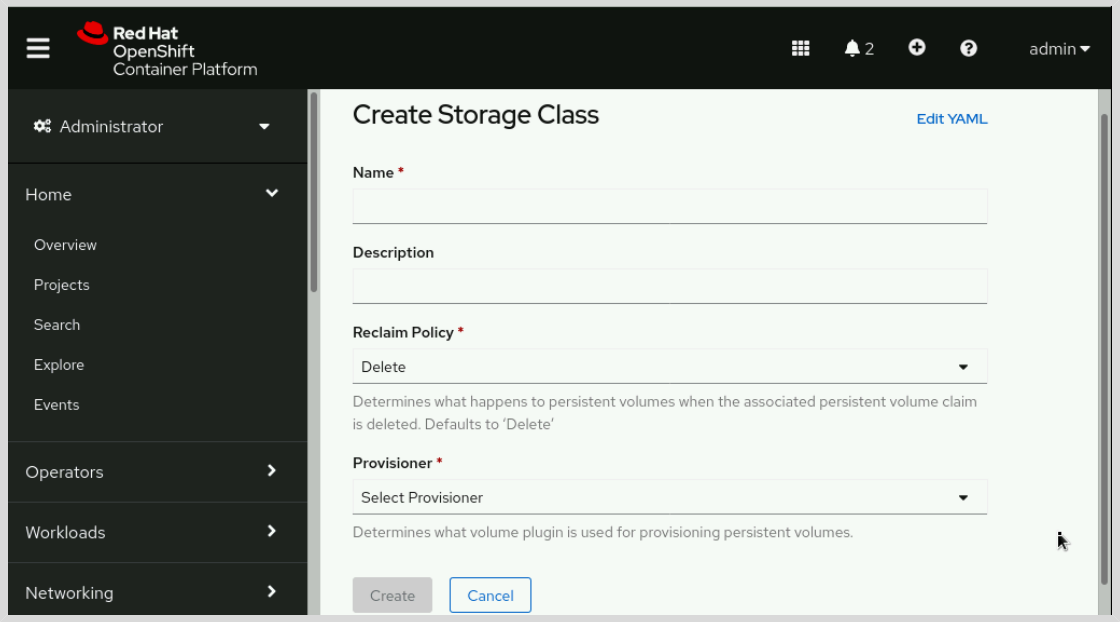
Custom storage classes are defined by using either the web console or the command line interface. Storage class objects are globally scoped, which means that only `cluster-admin` or `storage-admin` users can create them.

To define a custom storage class using the GUI:

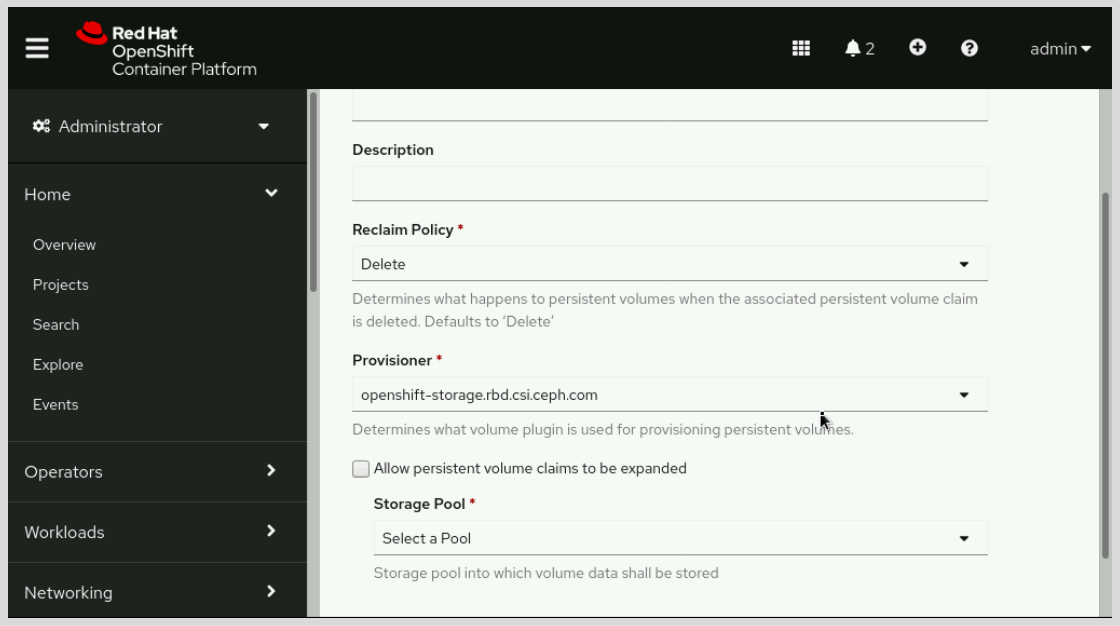
From the web console, navigate to the storage classes section. Click **Storage**, scroll down, and then click the **StorageClasses** link.



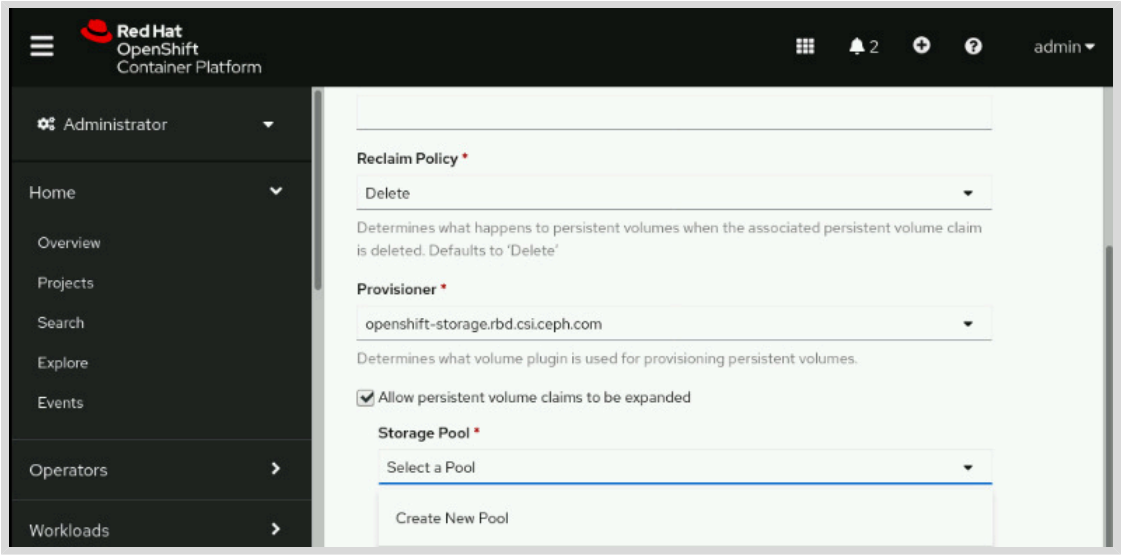
In the **Storage Classes** section: click **Create Storage Class**.



Use the `Provisioner` field to set the volume plug-in for storage classes. There are several volume plug-ins, such as AWS Elastic Block Store, Azure File, Azure Disk, CephFS, and Ceph RBD, among others.



Each provisioner has a set of particular options. In this example, the `openshift-storage.rbd.csi.ceph.com` provisioner enables setting Ceph storage pools. The web console also provides the option to create new Ceph storage pools, allowing you to set the Ceph replication and to enable compression if desired.



To find out more about provisioner options, consult the documentation for the provisioner.

To define a custom storage class using the CLI:

Storage classes are defined by the StorageClass resource. The following manifest describes a storage class that uses the `openshift-storage.rbd.csi.ceph.com`.

```
allowVolumeExpansion: true
kind: StorageClass (1)
apiVersion: storage.k8s.io/v1 (2)
metadata:
  name: ocs-storagecluster-ceph-rbd (3)
parameters: (4)
  clusterID: openshift-storage
  csi.storage.k8s.io/controller-expand-secret-name: rook-csi-rbd-provisioner
  csi.storage.k8s.io/controller-expand-secret-namespace: openshift-storage
  csi.storage.k8s.io/fstype: ext4
  csi.storage.k8s.io/node-stage-secret-name: rook-csi-rbd-node
  csi.storage.k8s.io/node-stage-secret-namespace: openshift-storage
  csi.storage.k8s.io/provisioner-secret-name: rook-csi-rbd-provisioner
  csi.storage.k8s.io/provisioner-secret-namespace: openshift-storage
  imageFeatures: layering
  imageFormat: "2"
  pool: ocs-storagecluster-cephblockpool
provisioner: openshift-storage.rbd.csi.ceph.com (5)
reclaimPolicy: Delete (6)
volumeBindingMode: Immediate (7)
```

(1) API object type.

(2) API version for storage types.

(3) Name of the storage class.

(4) Parameters: These optional attributes are defined by the provisioner

(5) Provisioner for the storage class

6. When a volume is released, the cluster follows a reclaim policy. Currently, it can be: Retained, Recycled, or Deleted.

7. This field controls when the dynamic provisioning should assign the volume; possible values are Immediate or WaitForFirstConsumer.

These parameters and attributes can be modified for the needs of applications and databases for production, test, and development environments.

References

For more information on creating custom storage classes, refer to the *Defining a storage class* section of the *Storage* documentation for _Red Hat OpenShift Container Platform at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.7/html-single/storage/index#defining-storage-classes_dynamic-provisioning

For more information on creating custom storage classes, refer to the *Storage classes and storage pools* chapter of the *Managing and allocating storage resources* documentation for _Red Hat OpenShift Container Platform at
https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/html-single/managing_and_allocating_storage_resources/index#storage-classes-and-storage-pools_rhocs

For more information, refer to the product documentation for *Red Hat OpenShift Data Foundation* at
https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/

Guided Exercise

Configuring Custom Storage Classes

In this exercise, you will create a new storage class with custom settings.

Outcomes

You should be able to:

- List the storage classes available in Red Hat OpenShift Data Foundation.
- Inspect storage class settings.
- Create a new storage class with custom settings.
- Verify the customized storage class values.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the cluster API is reachable and that OpenShift Data Foundation is working correctly.

```
[student@workstation ~]$ lab start workloads-classes
```

Instructions

1. Log in to your OpenShift cluster as the `admin` user and create a project named: `workloads-classes-ge`.

```
[student@workstation ~]$ oc login -u admin -p \
  redhat https://api.ocp4.example.com:6443
Login successful.
...output omitted...
[student@workstation ~]$ oc new-project workloads-classes-ge
Now using project "workloads-classes-ge" on server "https://
api.ocp4.example.com:6443".
...output omitted...
```

2. From the list of storage classes, identify the `ocs-storagecluster-ceph-rbd` class. This class offers Ceph RBD block storage.

```
[student@workstation ~]$ oc get storageclasses
NAME                          PROVISIONER                             ...
localblock                    kubernetes.io/no-provisioner
lso-volumeset                 kubernetes.io/no-provisioner
nfs-storage (default)         nfs-storage-provisioner
ocs-storagecluster-ceph-rbd   openshift-storage.rbd.csi.ceph.com
```

```
ocs-storagecluster-ceph-rgw   openshift-storage.ceph.rook.io/bucket
ocs-storagecluster-cephfs     openshift-storage.cephfs.csi.ceph.com
openshift-storage.noobaa.io   openshift-storage.noobaa.io/obc
```

3. Inspect the `ocs-storagecluster-ceph-rbd` storage class. Verify in the storage class provider parameters that the type of file system offered by this storage class is `ext4`.

```
[student@workstation ~]$ oc describe storageclass/ocs-storagecluster-ceph-rbd
Name:                  ocs-storagecluster-ceph-rbd
IsDefaultClass:        No
Annotations:           <none>
Provisioner:           openshift-storage.rbd.csi.ceph.com
Parameters:            clusterID=openshift-storage,csi.storage.k8s.io/
controller-expand-secret-name=rook-csi-rbd-provisioner,csi.storage.k8s.io/
controller-expand-secret-namespace=openshift-storage,csi.storage.k8s.io/
fstype=ext4,csi.storage.k8s.io/node-stage-secret-name=rook-csi-
rbd-node,csi.storage.k8s.io/node-stage-secret-namespace=openshift-
storage,csi.storage.k8s.io/provisioner-secret-name=rook-csi-rbd-
provisioner,csi.storage.k8s.io/provisioner-secret-namespace=openshift-
storage,imageFeatures=layering,imageFormat=2,pool=ocs-storagecluster-cephblockpool
AllowVolumeExpansion:  True
MountOptions:          <none>
ReclaimPolicy:         Delete
VolumeBindingMode:     Immediate
Events:                <none>
```

4. Copy the `~/DO370/labs/workloads-classes/ocs-storagecluster-ceph-rbd-XFS.yaml` file to the home directory and set the file system to `xfs`.

   4.1. Copy the `~/DO370/labs/workloads-classes/ocs-storagecluster-ceph-rbd-XFS.yaml` file to the student user home directory.

```
[student@workstation ~]$ cp \
  ~/DO370/labs/workloads-classes/ocs-storagecluster-ceph-rbd-XFS.yaml \
  ~/ocs-storagecluster-ceph-rbd-XFS.yaml
```

   4.2. Edit the `~/ocs-storagecluster-ceph-rbd-XFS.yaml` file. Identify the `fstype` parameter and set it to `xfs`.

```
...output omitted...
parameters:
  clusterID: openshift-storage
  csi.storage.k8s.io/controller-expand-secret-name: rook-csi-rbd-provisioner
  csi.storage.k8s.io/controller-expand-secret-namespace: openshift-storage
  csi.storage.k8s.io/fstype: xfs
  csi.storage.k8s.io/node-stage-secret-name: rook-csi-rbd-node
  csi.storage.k8s.io/node-stage-secret-namespace: openshift-storage
  csi.storage.k8s.io/provisioner-secret-name: rook-csi-rbd-provisioner
  csi.storage.k8s.io/provisioner-secret-namespace: openshift-storage
...output omitted...
```

5. Apply the `~/ocs-storagecluster-ceph-rbd-XFS.yaml` file to add the new storage class.

5.1. Apply the `~/ocs-storagecluster-ceph-rbd-XFS.yaml` file.

```
[student@workstation ~]$ oc apply -f ~/ocs-storagecluster-ceph-rbd-XFS.yaml
```

5.2. Verify that the `ocs-storagecluster-ceph-rbd-xfs` storage class is present.

```
[student@workstation ~]$ oc get storageclasses
NAME                              PROVISIONER                             ...
localblock                        kubernetes.io/no-provisioner
lso-volumeset                     kubernetes.io/no-provisioner
nfs-storage (default)             nfs-storage-provisioner
ocs-storagecluster-ceph-rbd       openshift-storage.rbd.csi.ceph.com
ocs-storagecluster-ceph-rbd-xfs   openshift-storage.rbd.csi.ceph.com
ocs-storagecluster-ceph-rgw       openshift-storage.ceph.rook.io/bucket
ocs-storagecluster-cephfs         openshift-storage.cephfs.csi.ceph.com
openshift-storage.noobaa.io       openshift-storage.noobaa.io/obc
```

6. Use an OpenShift template to create a persistent PostgreSQL database.

6.1. Create a PostgreSQL template that allows storage class configuration from the `DO370/labs/workloads-classes/postgresql-persistent-ge.json` file.

```
[student@workstation ~]$ oc create -f \
  DO370/labs/workloads-classes/postgresql-classes-ge.json
template.template.openshift.io/postgresql-persistent-sc created
```

6.2. Use the `postgresql-persistent-sc` template to create a persistent PostgreSQL database by setting the following template options:

- `STORAGECLASS_NAME=ocs-storagecluster-ceph-rbd-xfs` sets the name by using the storage class created in previous steps.
- `VOLUME_CAPACITY=150Mi` sets the initial volume capacity.
- `POSTGRESQL_USER=student` names the PostgreSQL user that manages the database.
- `POSTGRESQL_PASSWORD=redhat` provides the password for the PostgreSQL user.
- `POSTGRESQL_DATABASE=workloads-classes` names the database that will be served by the deployment.
- `DATABASE_SERVICE_NAME=pg-workloads-classes-ge` names the service to expose the database.

```
[student@workstation ~]$ oc new-app --name=pg-workload-classes \
  --template=postgresql-persistent-sc \
  -p STORAGECLASS_NAME=ocs-storagecluster-ceph-rbd-xfs \
  -p VOLUME_CAPACITY=150Mi \
  -p POSTGRESQL_USER=student \
  -p POSTGRESQL_PASSWORD=redhat \
  -p POSTGRESQL_DATABASE=workloads-classes \
  -p DATABASE_SERVICE_NAME=pg-workload-classes-ge
...output omitted...
--> Creating resources ...
    secret "pg-workload-classes-ge" created
    service "pg-workload-classes-ge" created
    persistentvolumeclaim "pg-workload-classes-ge" created
    deploymentconfig.apps.openshift.io "pg-workload-classes-ge" created
--> Success
...output omitted...
```

Note

The preceding commands are also available in the file `~/DO370/labs/workload-classes/pg-workload-classes.sh`.

6.3. Verify that the `pg-workload-classes-ge` PVC was created by using the `ocs-storagecluster-ceph-rbd-xfs` storage class.

```
[student@workstation ~]$ oc get pvc/pg-workload-classes-ge
NAME                     ...  ACCESS MODES   STORAGECLASS                     AGE
pg-workload-classes-ge   ...  RWO            ocs-storagecluster-ceph-rbd-xfs  54s
```

7. Inspect the `pg-workload-classes-ge` deployment configuration and verify that it uses the proper settings.

7.1. Enter the `pg-workload-classes-ge` pod.

```
[student@workstation ~]$ oc rsh dc/pg-workload-classes-ge
sh-4.4$
```

7.2. Verify that the `/var/lib/pgsql/data` is mounted on top of an XFS file system.

```
sh-4.4$ mount | grep /var/lib/pgsql/data
/dev/rbd1 on /var/lib/pgsql/data type xfs ...
```

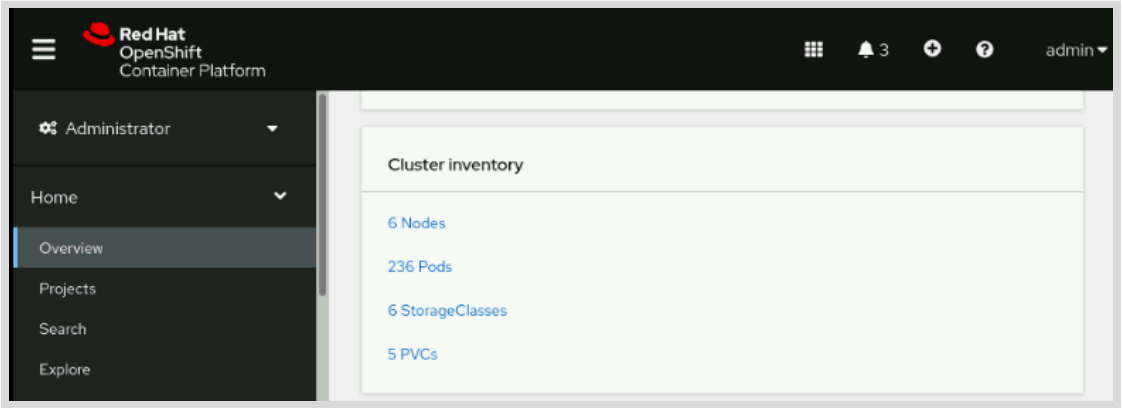
8. From the web console, create a storage class that has a Ceph pool and a limited number of replicas to use for a development PostgreSQL database.

8.1. Show the URL for the web console and open it using the web browser.

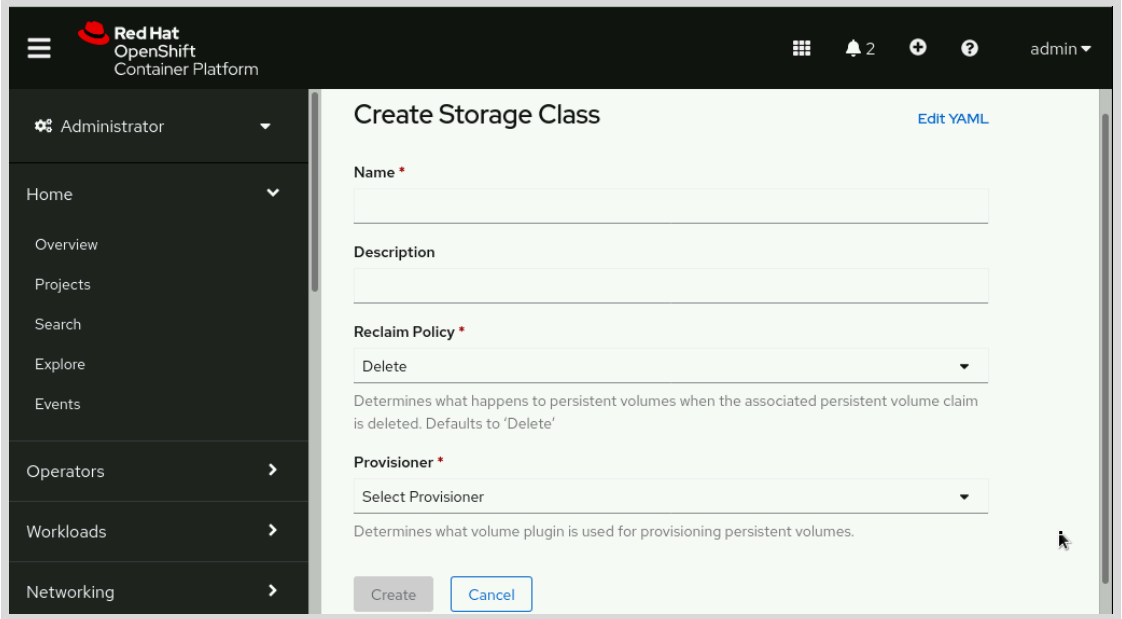
```
[student@workstation ~]$ oc whoami --show-console
https://console-openshift-console.apps.ocp4.example.com
```

8.2. Using the `htpasswd_provider` identity provider, log in to the cluster with the `admin` username and the `redhat` password.

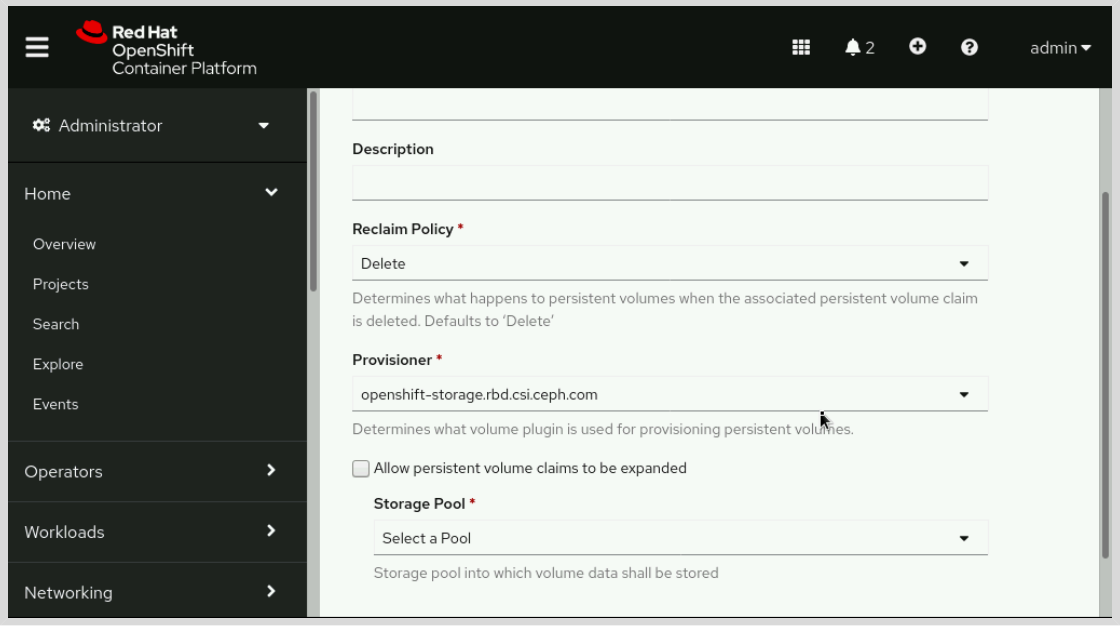
8.3. Enter the storage classes section: Click **Overview** > **Cluster**, scroll down, and then click the **StorageClasses** link.



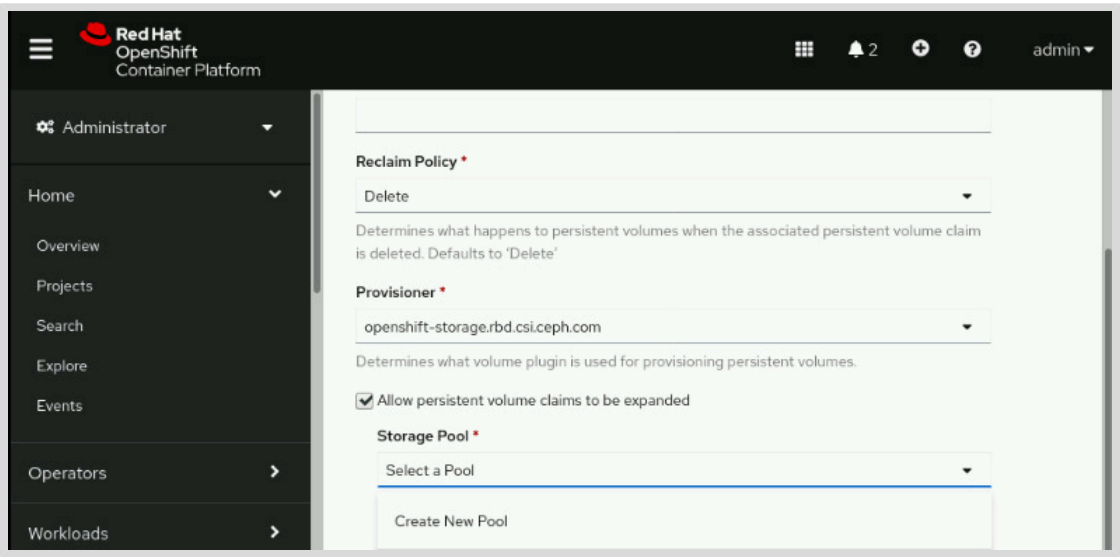
8.4. In the **Storage Class** section: click **Create Storage Class**.



8.5. Select the `openshift-storage.rbd.csi.ceph.com` provisioner from the `Provisioner` field.

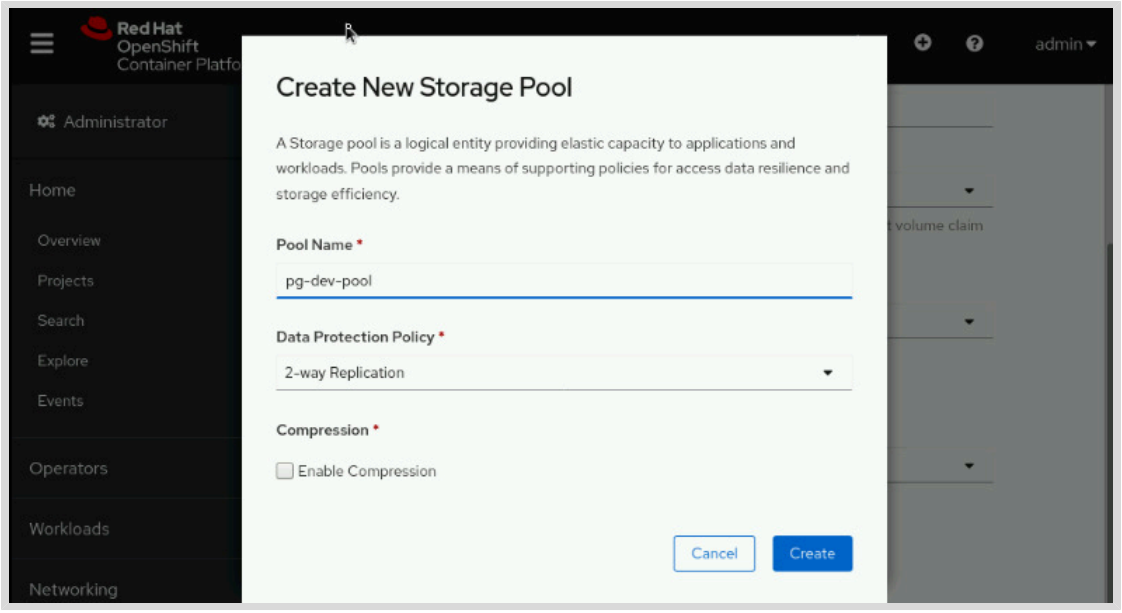


8.6. Click on the `Storage Pool` field and select `Create New Pool` to open the `Create New Storage Pool` panel.



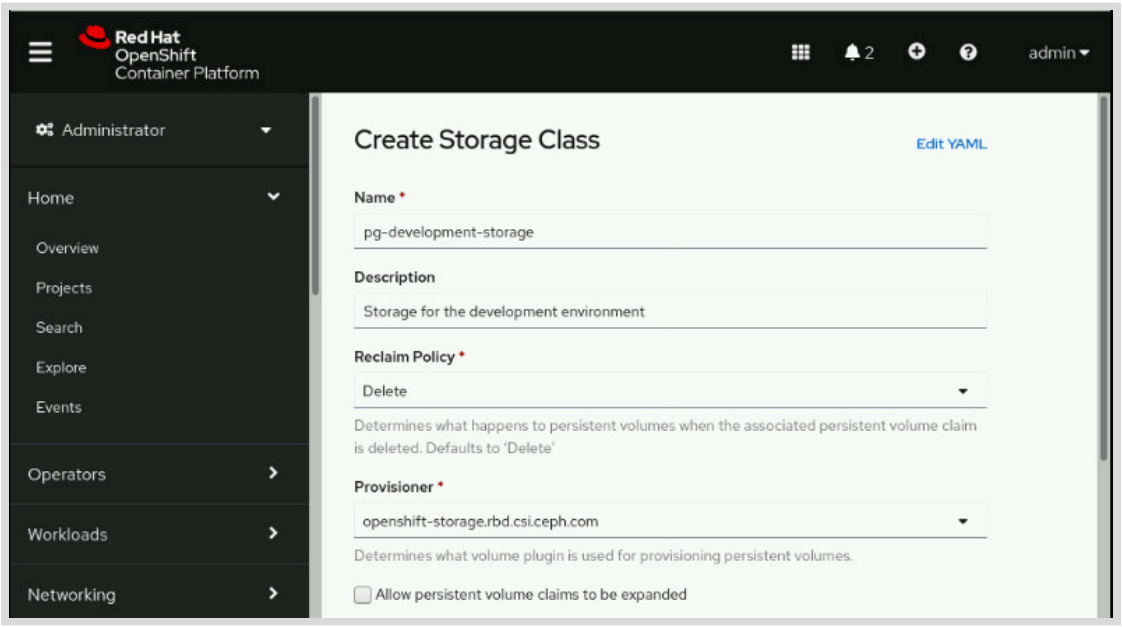
8.7. In the `Pool Name` field type `pg-dev-pool` to set the new pool name.

8.8. Set the `Data Protection Policy` field to `2-way Replication`. This field sets the pool replicas to two so that fewer resources are used for the development environment. To create the new storage pool, click **Create**. Click **Finish** when done.



8.9. On the `Name` field set the name of the new storage class to `pg-development-storage` and type `Storage for the development environment` in the `Description` field. Finally, click the **Create** button to create the storage class.

The Create Storage Class window is displayed. In the `Name` field set the name of the new storage class to `pg-development-storage` and type `Storage for the development environment` in the `Description` field. Finally, click **Create** to create the storage class.



8.10. Open a terminal session and verify that the new storage class is available by running the following command.

```
[student@workstation ~]$ oc get storageclasses
NAME                          PROVISIONER ...
...output omitted...
pg-development-storage        openshift-storage.rbd.csi.ceph.com
...output omitted...
```

9. Create a PostgreSQL deployment that uses the new storage class for a development environment.

9.1. Use the `postgresql-persistent-sc` template to create a persistent PostgreSQL database by setting the following template options:

- `STORAGECLASS_NAME=pg-development-storage` the storage class created in previous steps.
- `VOLUME_CAPACITY=150Mi`, initial volume capacity.
- `POSTGRESQL_USER=developer`, name of the PostgreSQL that manage the database.
- `POSTGRESQL_PASSWORD=devel`, password for the PostgreSQL user.
- `POSTGRESQL_DATABASE=dev-workloads-classes`, name of the database that will be served by the deployment.
- `DATABASE_SERVICE_NAME=dev-pg-workloads-classes-ge`, name of the service to expose the database.

```
[student@workstation ~]$ oc new-app --name=dev-pg-workload-classes \
  --template=postgresql-persistent-sc \
  -p STORAGECLASS_NAME=pg-development-storage \
  -p VOLUME_CAPACITY=150Mi \
  -p POSTGRESQL_USER=developer \
  -p POSTGRESQL_PASSWORD=devel \
  -p POSTGRESQL_DATABASE=dev-workloads-classes \
  -p DATABASE_SERVICE_NAME=dev-pg-workload-classes-ge
...output omitted...
--> Creating resources ...
    secret "dev-pg-workload-classes-ge" created
    service "dev-pg-workload-classes-ge" created
    persistentvolumeclaim "dev-pg-workload-classes-ge" created
    deploymentconfig.apps.openshift.io "dev-pg-workload-classes-ge" created
--> Success
...output omitted...
```

Note

The preceding commands are available in the file `~/DO370/labs/workload-classes/dev-pg-workload-classes.sh`.

9.2. Verify that the PVC created by the `dev-pg-workload-classes` deployment is provided by the `pg-development-storage` storage class.

```
[student@workstation ~]$ oc get pvc
NAME                         STATUS   ...    STORAGECLASS
...output omitted...
dev-pg-workload-classes-ge   Bound    ...    pg-development-storage
```

10. Remove the `workload-classes-ge` project and the `postgresql-persistent-sc` template.

10.1. Remove the `workload-classes-ge` project.

```
[student@workstation ~]$ oc delete project workloads-classes-ge
project.project.openshift.io "workloads-classes-ge" deleted
```

10.2. Remove the `postgresql-persistent-sc` template.

```
[student@workstation ~]$ oc delete template postgresql-persistent-sc -n openshift
template.template.openshift.io "postgresql-persistent-sc" deleted
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish workloads-classes
```

This concludes the guided exercise.

Lab

Configuring Application Workloads to Use Red Hat OpenShift Data Foundation Block and File Storage

In this lab, you will deploy an image application as well as a database that uses file storage and block storage respectively.

Outcomes

You should be able to:

- Create persistent volume claims.
- Configure storage classes for PVCs.
- Add persistent storage to application deployments.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start workloads-review
```

Instructions

1. Log in to the OpenShift cluster and switch to the `workloads-review` project.
2. Deploy the MariaDB database. Use the files provided in the `~/DO370/labs/workloads-review` directory as templates.
   - MariaDB requests a block device with the `ocs-storagecluster-ceph-rbd` storage class, and the `ReadWriteOnce` access mode.
   - The MariaDB deployment mounts the persistent volume claim in the `/var/lib/mysql` directory.
3. Deploy the WordPress application. Use the files provided in the `~/DO370/labs/workloads-review` directory as templates.
   - WordPress requests a file storage with the `ocs-storagecluster-cephfs` storage class, and the `ReadWriteMany` access mode.
   - The WordPress deployment mounts the persistent volume claim in the `/var/www/html` directory.
   - Get the route hostname and verify that the application is working.
4. Change to the `/home/student` directory.

Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until it is successful.

```
[student@workstation ~]$ lab grade workloads-review
```

Finish

As the `student` user on the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish workloads-review
```

This concludes the lab.

Solution

Configuring Application Workloads to Use Red Hat OpenShift Data Foundation Block and File Storage

In this lab, you will deploy an image application as well as a database that uses file storage and block storage respectively.

Outcomes

You should be able to:

- Create persistent volume claims.
- Configure storage classes for PVCs.
- Add persistent storage to application deployments.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start workloads-review
```

Instructions

1. Log in to the OpenShift cluster and switch to the `workloads-review` project.

   1.1. Log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

   1.2. Change to the `workloads-review` project.

```
[student@workstation ~]$ oc project workloads-review
Now using project "workloads-review" on server
"https://api.ocp4.example.com:6443".
```

2. Deploy the MariaDB database. Use the files provided in the `~/DO370/labs/workloads-review` directory as templates.

   - MariaDB requests a block device with the `ocs-storagecluster-ceph-rbd` storage class, and the `ReadWriteOnce` access mode.
   - The MariaDB deployment mounts the persistent volume claim in the `/var/lib/mysql` directory.

2.1. Change to the `~/DO370/labs/workloads-review` directory.

```
[student@workstation ~]$ cd ~/DO370/labs/workloads-review
```

2.2. Edit the `pvc-mariadb.yaml` file and modify the placeholder text to match the following content.

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mariadb
spec:
  accessModes:
  - ReadWriteOnce
  storageClassName: ocs-storagecluster-ceph-rbd
  resources:
    requests:
      storage: 1Gi
```

2.3. Create the persistent volume claim for MariaDB.

```
[student@workstation workloads-review]$ oc apply -f pvc-mariadb.yaml
persistentvolumeclaim/mariadb created
```

2.4. Verify that the persistent volume claim status is set to **Bound**.

```
[student@workstation workloads-review]$ oc describe pvc mariadb | head -n 6
Name:          mariadb
Namespace:     workloads-review
StorageClass:  ocs-storagecluster-ceph-rbd
Status:        Bound
Volume:        pvc-d8e83bc2-f7a6-4c45-91ad-f87e1bfb8edd
Labels:        <none>
```

2.5. Edit the `mariadb.yaml` file and modify the placeholder text to match the following content.

```
...output omitted...
    spec:
      containers:
      - name: mariadb
        image: quay.io/redhattraining/mariadb:10.5
        imagePullPolicy: Always
        ports:
        - containerPort: 3306
        resources: {}
        envFrom:
        - configMapRef:
            name: mariadb
        - secretRef:
            name: mariadb
```

```
        volumeMounts:
        - name: mariadb
          mountPath: "/var/lib/mysql"
      volumes:
      - name: mariadb
        persistentVolumeClaim:
          claimName: mariadb
...output omitted...
```

2.6. Deploy the MariaDB application.

```
[student@workstation workloads-review]$ oc apply -f mariadb.yaml
deployment.apps/mariadb created
service/mariadb created
```

2.7. Wait until the MariaDB pod is running.

```
[student@workstation workloads-review]$ oc get pods -l app=mariadb
NAME                       READY   STATUS    RESTARTS   AGE
mariadb-7759c4d8f4-ktczj   1/1     Running   0          3m
```

2.8. Verify that the persistent volume is mounted on the pod.

```
[student@workstation workloads-review]$ oc exec -it deployment/mariadb -- \
  df -h /var/lib/mysql
Filesystem      Size  Used Avail Use% Mounted on
/dev/rbd2       976M  152M  809M  16% /var/lib/mysql
```

2.9. Inspect the logs of the MariaDB deployment and verify that the application is ready to receive connections.

```
[student@workstation workloads-review]$ oc logs deployment/mariadb | tail
...output omitted...
2022-03-30 19:23:34 0 [Note] Reading of all Master_info entries succeeded
2022-03-30 19:23:34 0 [Note] Added new Master_info '' to hash table
2022-03-30 19:23:34 0 [Note] mysqld: ready for connections.
Version: '10.5.15-MariaDB-1:10.5.15+maria~focal'  socket: '/run/mysqld/
mysqld.sock'  port: 3306  mariadb.org binary distribution
```

3. Deploy the WordPress application. Use the files provided in the `~/DO370/labs/workloads-review` directory as templates.

- WordPress requests a file storage with the `ocs-storagecluster-cephfs` storage class, and the `ReadWriteMany` access mode.
- The WordPress deployment mounts the persistent volume claim in the `/var/www/html` directory.
- Get the route hostname and verify that the application is working.

3.1. Edit the `pvc-wordpress.yaml` file and modify the placeholder text to match the following content.

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wordpress
spec:
  accessModes:
  - ReadWriteMany
  storageClassName: ocs-storagecluster-cephfs
  resources:
    requests:
      storage: 1Gi
```

3.2. Create the persistent volume claim for WordPress.

```
[student@workstation workloads-review]$ oc apply -f pvc-wordpress.yaml
persistentvolumeclaim/wordpress created
```

3.3. Verify that the persistent volume claim status is set to **Bound**.

```
[student@workstation workloads-review]$ oc describe pvc wordpress | head -n 6
Name:          wordpress
Namespace:     workloads-review
StorageClass:  ocs-storagecluster-cephfs
Status:        Bound
Volume:        pvc-3f8b60f2-2a1d-4b8a-bcfd-2c6ecd021dc6
Labels:        <none>
```

3.4. Edit the `wordpress.yaml` file and modify the placeholder text to match the following content.

```
...output omitted...
    spec:
      containers:
      - name: wordpress
        image: quay.io/redhattraining/wordpress:5.7-php7.4-apache
        imagePullPolicy: Always
        ports:
        - containerPort: 80
        resources: {}
        envFrom:
        - configMapRef:
            name: wordpress
        - secretRef:
            name: wordpress
        volumeMounts:
        - name: wordpress
          mountPath: "/var/www/html"
      volumes:
      - name: wordpress
```

```
        persistentVolumeClaim:
          claimName: wordpress
...output omitted...
```

3.5. Deploy the WordPress application.

```
[student@workstation workloads-review]$ oc apply -f wordpress.yaml
deployment.apps/wordpress created
service/wordpress created
```

3.6. Wait until the WordPress pod is running.

```
[student@workstation workloads-review]$ oc get pods -l app=wordpress
NAME                         READY   STATUS    RESTARTS   AGE
wordpress-7c694d65b-rzzw6    1/1     Running   0          3m
```

3.7. Verify that the persistent volume is mounted on the pod.

```
[student@workstation workloads-review]$ oc exec -it deployment/wordpress -- \
  df -h /var/www/html
Filesystem                    Size  Used Avail Use% Mounted on
...:/volumes/csi/csi-vol-...  1.0G   60M  964M   6% /var/www/html
```

3.8. Get the hostname for the WordPress route.

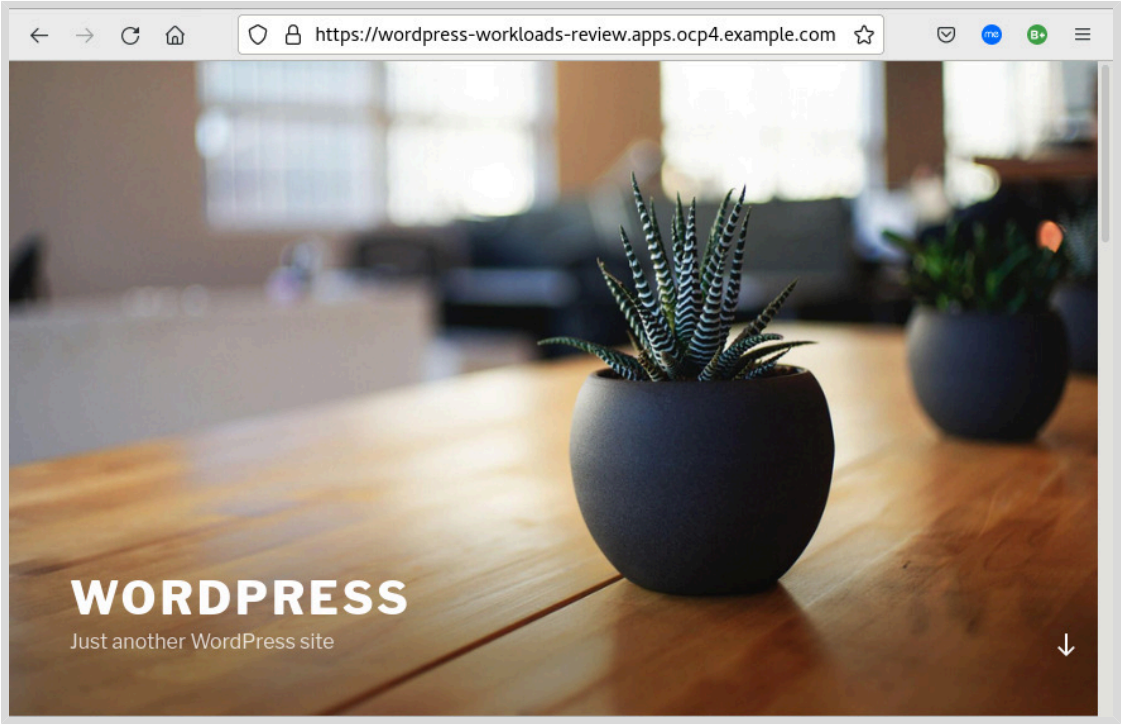
```
[student@workstation workloads-review]$ oc get route wordpress \
  -o jsonpath='{.spec.host}{"\n"}'
wordpress-workloads-review.apps.ocp4.example.com
```

3.9. Verify that the application is running.

```
[student@workstation workloads-review]$ curl -fsSL \
  'https://wordpress-workloads-review.apps.ocp4.example.com/' | \
  egrep '</?title>|site-description'
<title>WordPress &#8211; Just another WordPress site</title>
       <p class="site-description">Just another WordPress site</p>
```

3.10. Open the application web page in the web browser.

- https://wordpress-workloads-review.apps.ocp4.example.com/



4. Change to the `/home/student` directory.

```
[student@workstation workloads-review]$ cd
```

Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until it is successful.

```
[student@workstation ~]$ lab grade workloads-review
```

Finish

As the `student` user on the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish workloads-review
```

This concludes the lab.

Summary

In this chapter, you learned:

- The basic architecture and components of an RHOCP Ceph storage cluster.
- The architecture and components Ceph provides for Red Hat OpenShift Data Foundation functions.
- Application deployment strategies for file and block storage solutions through OpenShift Data Foundation.
- Implementation details of the `ocs-storagecluster-cephfs` storage class for file-based storage solutions.
- Implementation details of the `ocs-storagecluster-ceph-rbd` storage class for block storage solutions.

Note

OpenShift Data Foundation provides simplified management of complex storage solutions for OCP, however, it does not validate proper storage architecture for deployed applications. Although this chapter discusses suggested workloads for each storage class, every application has unique requirements and environmental factors that influence performance.

Some of the variables that impact application storage performance include:

- Availability of storage solutions
- Network and data center latencies
- Disk performance
- Application constraints
- Data types and behaviors:
  - Volume - size or number of entries
  - Velocity - speed of read and write access
  - Variety - uniformity and diversity

Always consider the proper storage solution for your application architecture based on the specific details of your implementation.

Chapter 4

Managing Red Hat OpenShift Data Foundation Block and File Storage Capacity

Goal

Monitor and expand overall OpenShift Data Foundation cluster capacity.

Objectives

- Verify storage health metrics using cluster monitoring.
- Configuring and verify quotas and permissions for the Red Hat OpenShift Data Foundation cluster storage.
- Detect when a storage volume is close to full and expand the volume size without disrupting an application.
- Add additional disks to a Red Hat OpenShift Data Foundation cluster.

Sections

- Monitoring Red Hat OpenShift Data Foundation Cluster Health (and Guided Exercise)
- Configuring Storage Quotas and Permissions (and Guided Exercise)
- Extending Application Storage for Red Hat OpenShift Data Foundation (and Guided Exercise)
- Adding Disks to the Red Hat OpenShift Data Foundation Cluster (and Guided Exercise)

Lab

Managing Red Hat OpenShift Data Foundation Block and File Storage Capacity

Monitoring Red Hat OpenShift Data Foundation Cluster Health

Objectives

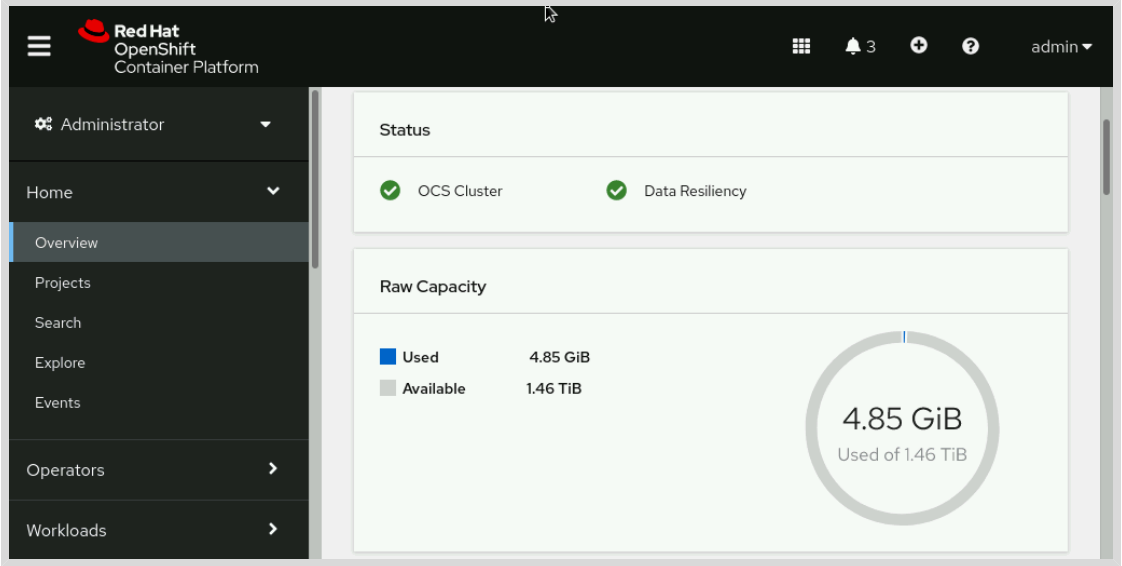
After completing this section, you should be able to verify storage health metrics using cluster monitoring.

Monitoring Metrics

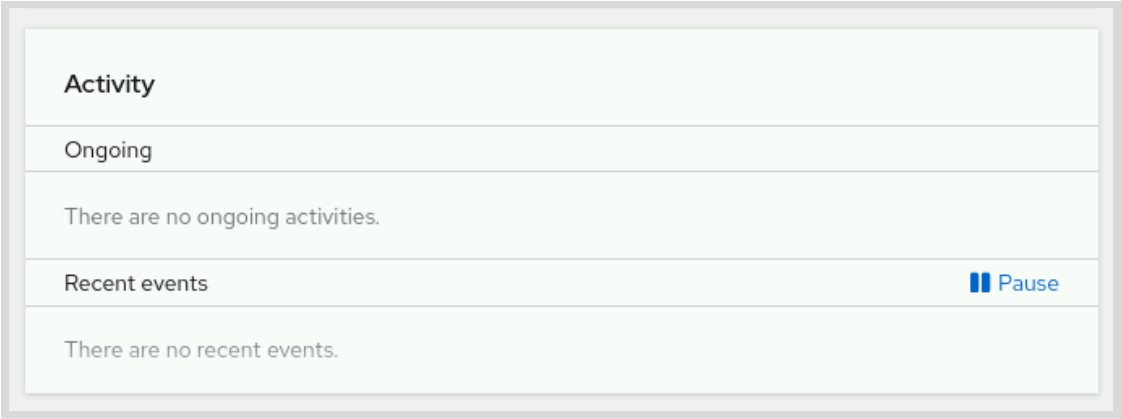
The Red OpenShift Container Platform provides a robust ecosystem in which metrics and monitoring play an important role. These metrics are integrated into the web console under the `Persistent Storage` and `Object Service` selections on the `Overview` menu.

Persistent Storage

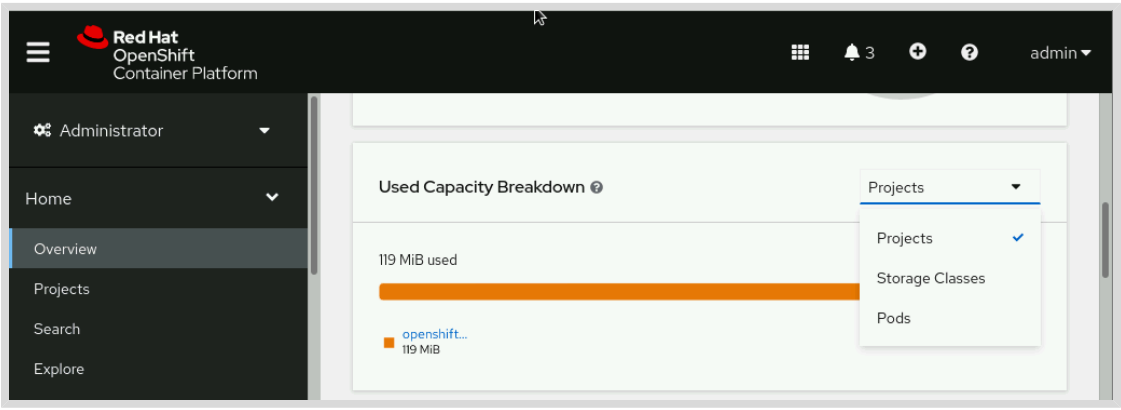
The `Status` section under `Persistent Storage` shows the overall status of the cluster, indicating green for a healthy status and red for a degraded status.



Depending on your resolution, you can find the Activity section at the right or the bottom of the page. This section details events for quick diagnostics when a problem occurs.



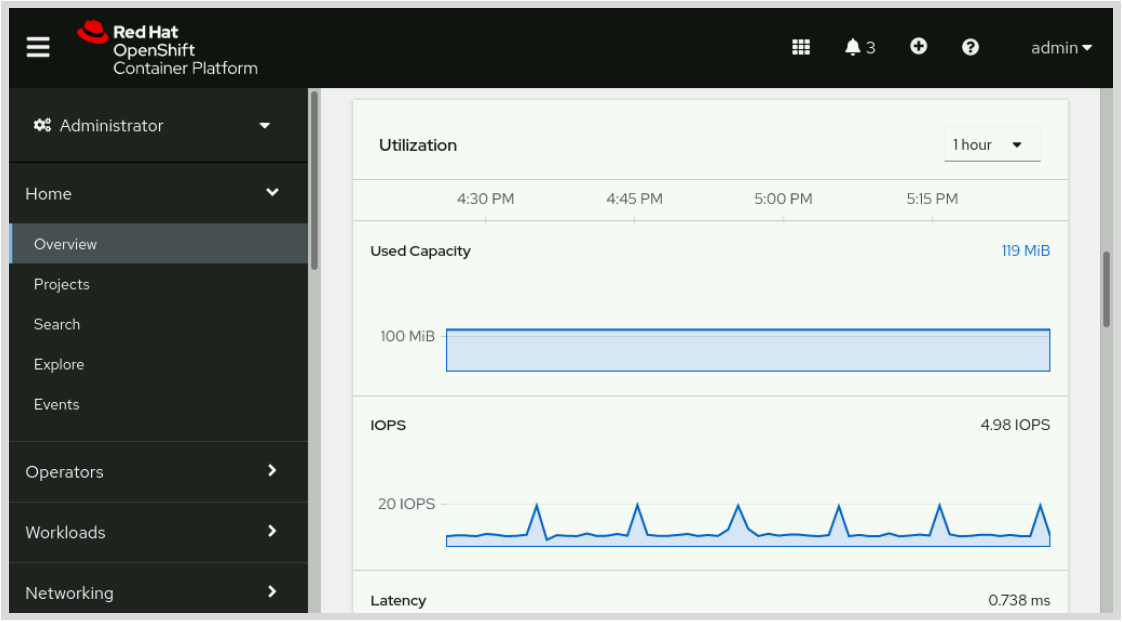
The `Raw Capacity` section shows the overall used and available cluster capacity. The `Used Capacity Breakdown` allows inspecting the capacity by projects, storage classes, and pods.



The `Utilization` section shows the following real-time metrics and graphics:

- Used capacity
- Input-output operations per second (IOPS)
- Latency
- Throughput
- Recovery

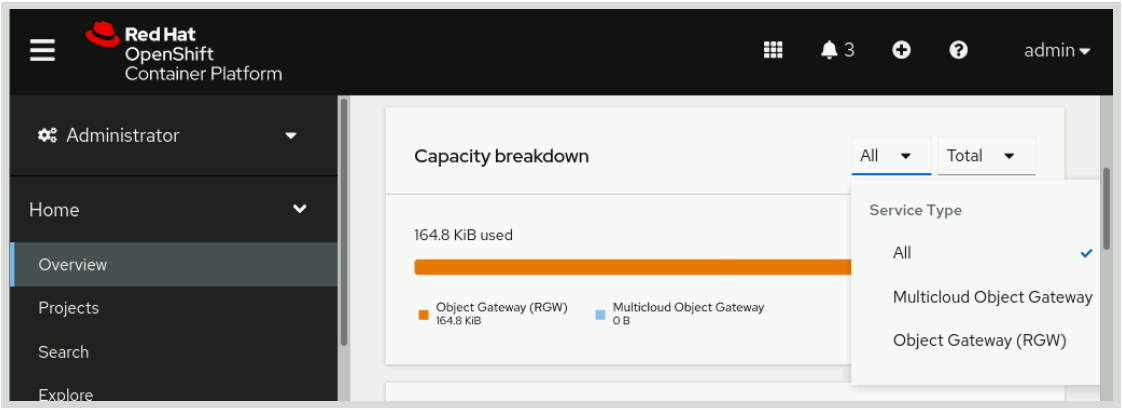
You can filter these graphics to show activity in blocks of 1 hour, 6 hours, or 24 hours to make better comparative assessments of the cluster behavior for improved troubleshooting.



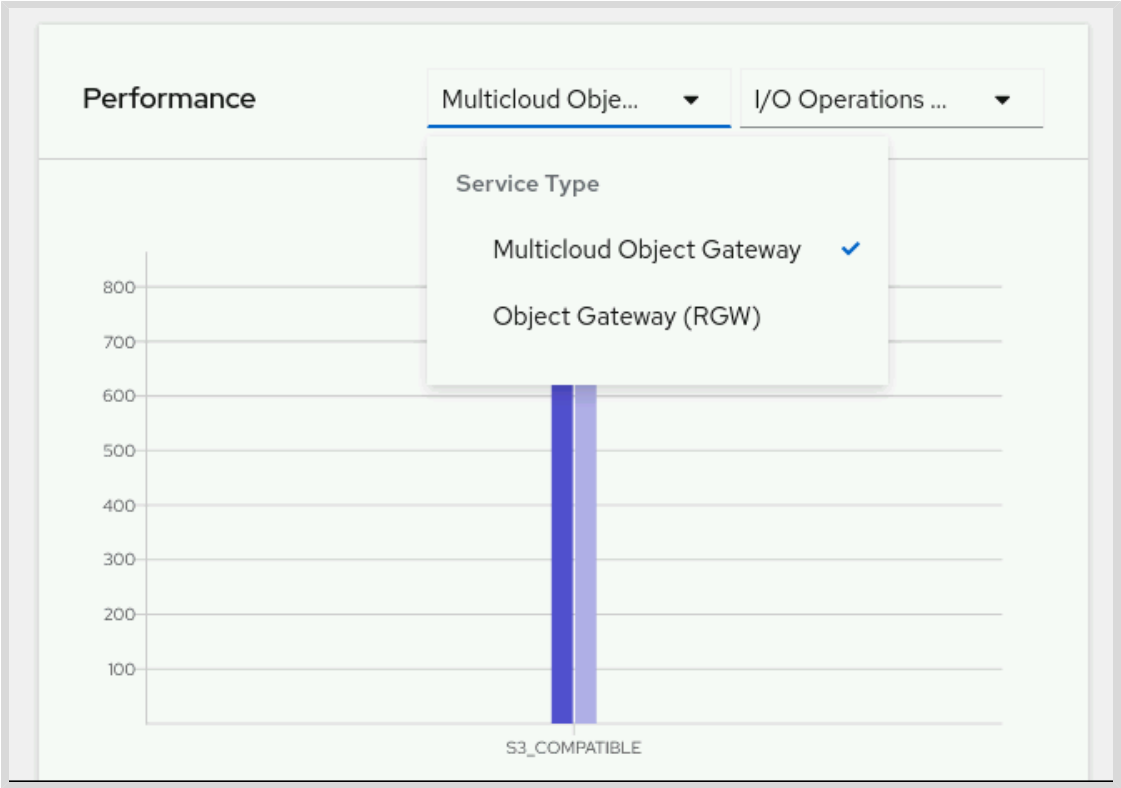
When compression is available, the `Storage Efficiency` section shows the compression ratio of the physically stored data compared to the size of the data received and the physical storage saved after compression.

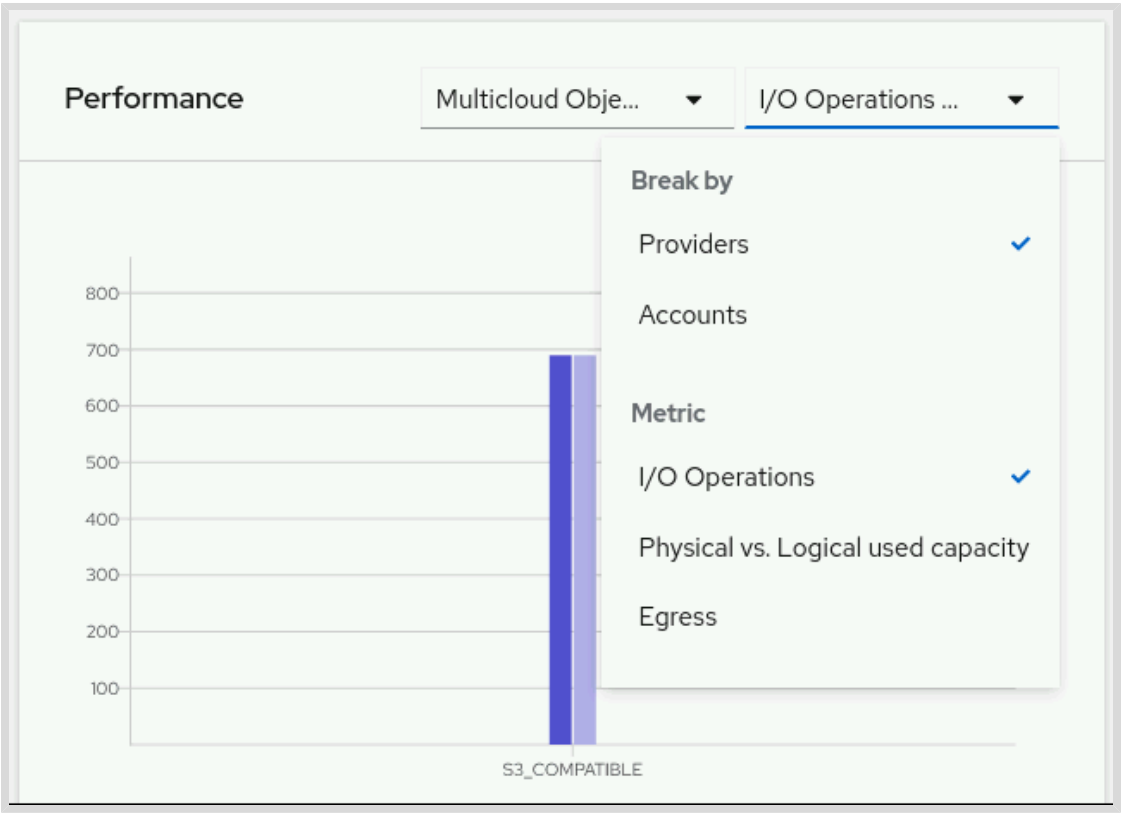
Object Service

A similar set of metrics and graphics is available for `Object Service`, including object service status, data resiliency, performance, and capacity breakdown.



The capacity breakdown for the object service filters by the object gateways enabled in the cluster instead of by projects or pods. Performance is filtered by type of gateway service, providers, accounts, I/O operations, physical versus logical used capacity, and egress.





References

For more information, refer to the monitoring documentation for *Red Hat OpenShift Container Platform* at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.7/html-single/monitoring/index

Guided Exercise

Monitoring Red Hat OpenShift Data Foundation Cluster Health

In this exercise, you will verify storage health metrics.

Outcomes

You should be able to:

- Identify storage metrics in the dashboard
- Verify storage metrics using the dashboard

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the dependencies for this exercise are set up.

```
[student@workstation ~]$ lab start capacity-monitoring
```

Instructions

1. Locate information about persistent storage health using the web console.

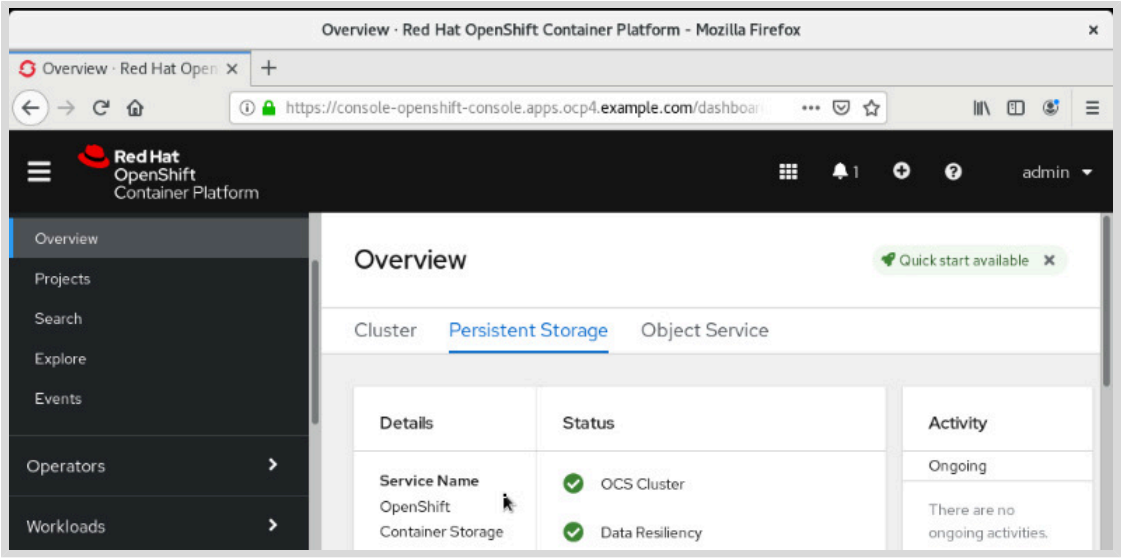
   1.1. Log in to the OpenShift dashboard at https://console-openshift-console.apps.ocp4.example.com as the `admin` user with the password: `redhat`

   1.2. Navigate to **Home** > **Overview** > **Persistent storage**.

2. Verify that the persistent storage is in a healthy state.

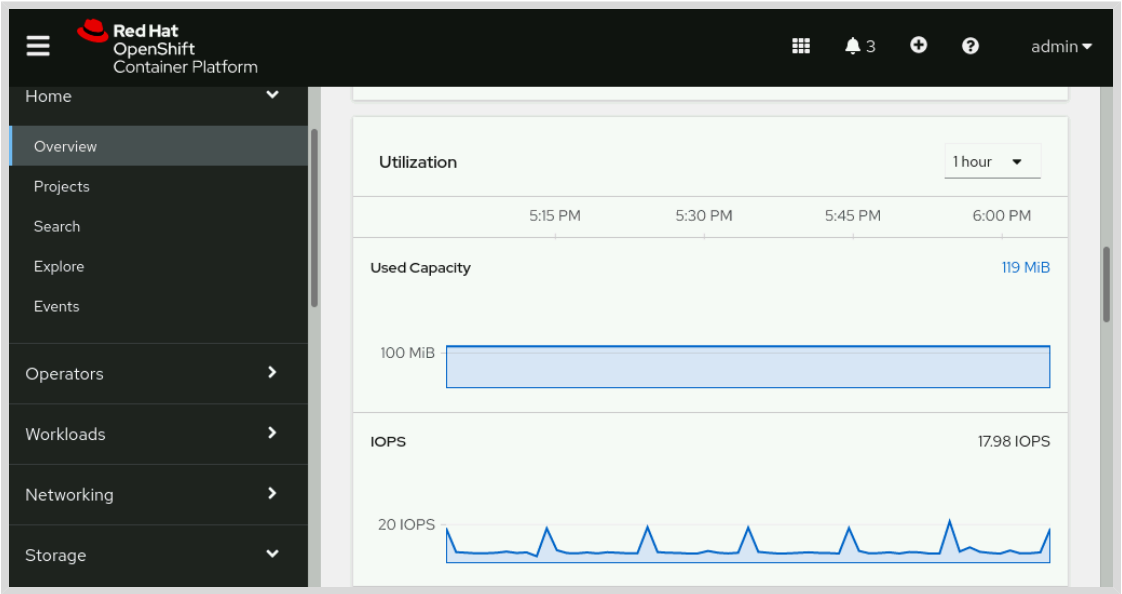
   2.1. Find the dashboard status indicators.

   2.2. Verify that the cluster is in a healthy state by checking the icons and message in the status column. If the cluster is healthy, then the icons are green.



3. Identify persistent storage metrics.

3.1. Find the metrics `Used capacity`, `Input/Output Operations Per Second (IOPS)`, `Latency`, `Throughput` and `Recovery` indicators.



4. Verify persistent storage activity using the CLI.

4.1. Log in to the OpenShift cluster as the `admin` user with the `redhat` password.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

4.2. Change to the `monitoring-ge` project.

```
[student@workstation ~]$ oc project monitoring-ge
Now using project "monitoring-ge" on server "https://api.ocp4.example.com:6443".
```

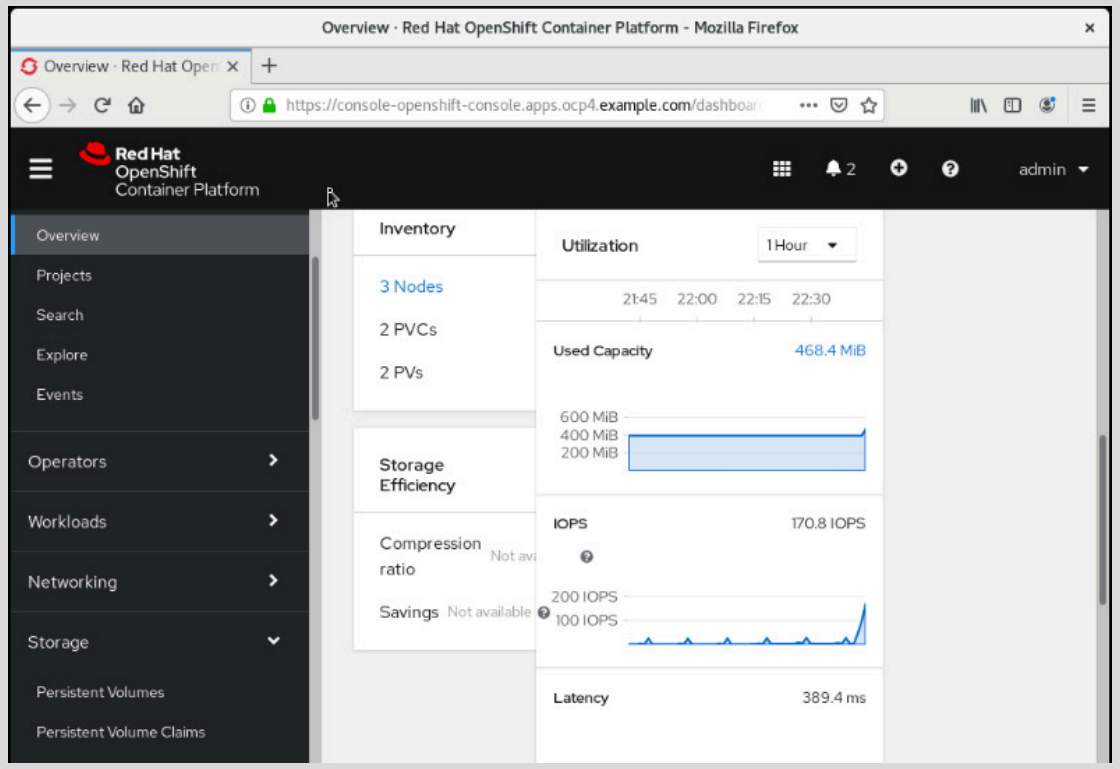
4.3. Open a shell in the `monitoring-ge-pod` pod.

```
[student@workstation ~]$ oc rsh monitoring-ge-pod
sh-4.4#
```

4.4. Create a 100 MB file in an infinite loop to trigger activity in the cluster.

```
sh-4.4# while [ 1 ]; do dd if=/dev/zero of=/mnt/bigfile count=1 bs=100M; done
1+0 records in
1+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 0.258818 s, 405 MB/s
1+0 records in
1+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 0.255261 s, 411 MB/s
...output omitted...
```

4.5. Return to the dashboard and verify that the storage metrics are showing increased activity.



5. Stop the loop and verify the decreased activity in the metrics.

5.1. Press `Ctrl+C` to stop the `while` loop.

```
sh-4.4# while [ 1 ]; do dd if=/dev/zero of=/mnt/bigfile count=1 bs=100M; done
...output omitted...
1+0 records in
1+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 0.263517 s, 398 MB/s
^C0+0 records in
```

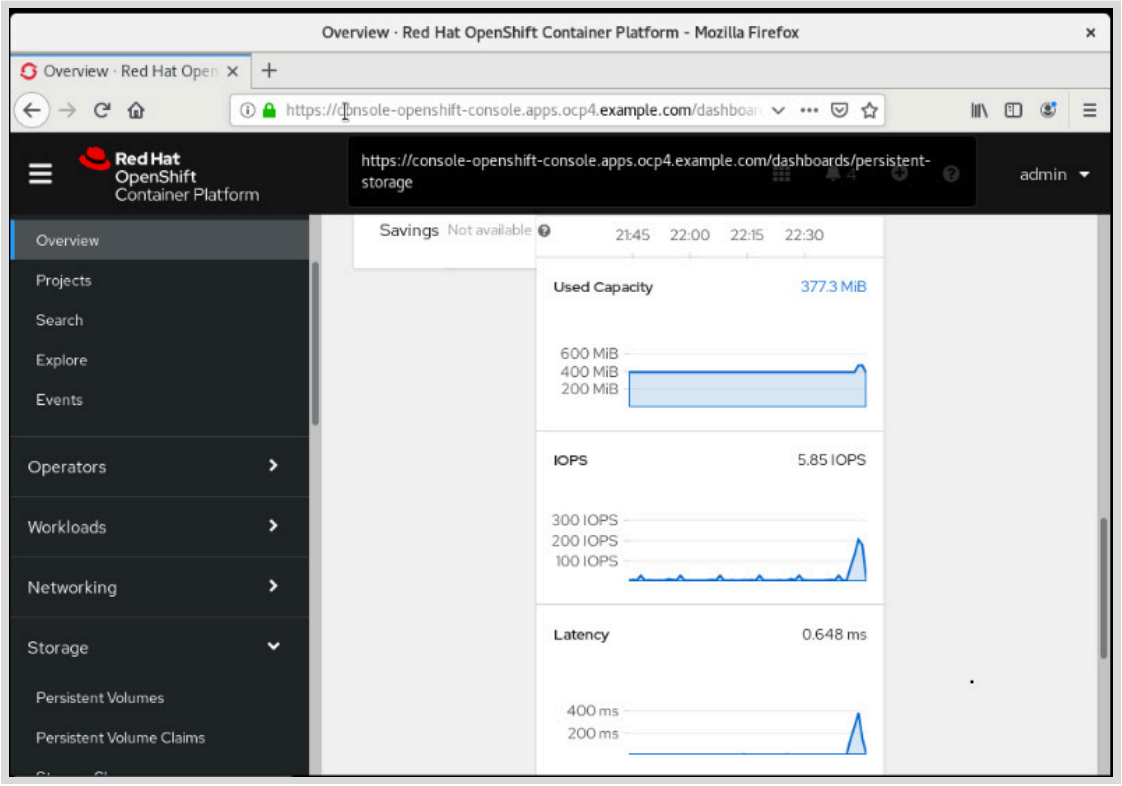
5.2. Remove the `/mnt/bigfile` file.

```
sh-4.4# rm /mnt/bigfile
```

5.3. Exit the pod shell.

```
sh-4.4# exit
exit
[student@workstation ~]$
```

5.4. Return to the dashboard and verify that the storage metrics are showing decreased activity.



Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish capacity-monitoring
```

This concludes the guided exercise.

Configuring Storage Quotas and Permissions

Objectives

After completing this section, you should be able to configuring and verify quotas and permissions for the Red Hat OpenShift Data Foundation cluster storage.

Introducing Storage Cluster Permissions

Storage cluster permissions enable managing storage volume access in the same way that you manage access to many other aspects of OpenShift. The techniques available for managing access to storage through Red Hat OpenShift Data Foundation should be familiar to seasoned OpenShift administrators, operators, and engineers. OpenShift Data Foundation provides a seamless way to manage your OpenShift storage components with the same ease and common approaches that you use to manage other aspects of the cluster.

Introducing Role Based Access Control (RBAC)

Role-based access control (RBAC) is a technique for managing access to resources in a computer system. In Red Hat OpenShift, RBAC determines if a user can perform certain actions within the cluster or project. In deployments using OpenShift Data Foundation to manage cluster storage components, the same approach applies to controlling access to storage classes and clusters.

Using this approach, cluster administrators can allow or deny access to storage classes to restrict the type of workloads or customers consuming that particular resource. This type of control enables the scenario of providing common storage types for development applications but reserving access to production workloads that may require more expensive, faster, or larger storage resources.

RBAC controls storage on the cluster-wide and project levels by determining default settings for cluster storage pools, as well as which users can manage persistent volume claims (PVCs) within projects.

Managing RBAC Using the CLI

Cluster administrators can use the `oc adm policy` command to both add and remove cluster roles and namespace roles.

To give a user a cluster role, use the `add-cluster-role-to-user` subcommand:

```
[user@host ~]$ oc adm policy add-cluster-role-to-user cluster-role username
```

A user with the `cluster-admin` role can fully manage all storage pools, storage classes, and persistent volue claims (PVC) in all projects.

```
[user@host ~]$ oc adm policy add-cluster-role-to-user cluster-admin username
```

Default Roles

OpenShift provides a set of default cluster roles that can be assigned locally or to the entire cluster. You can modify these roles for fine-grained access control to OpenShift resources, but additional steps are required that are outside the scope of this course.

Default Roles	Description
`admin`	Users with this role can manage all project resources, including granting other users access to the project.
`basic-user`	Users with this role have read access to the project.
`cluster-admin`	Users with this role have superuser access to the cluster resources. These users can perform any action on the cluster and have full control of all projects.
`cluster-status`	Users with this role can get cluster status information.
`edit`	Users with this role can create, change, and delete common application resources from the project, such as services and deployments. These users cannot act on management resources, such as limit ranges and quotas, and cannot manage access permissions to the project.
`self-provisioner`	Users with this role can create new projects. This is a cluster role, not a project role.
`view`	Users with this role can view project resources, but cannot modify those resources.

The `admin` role gives a user access to project resources such as quotas and limit ranges, and also the ability to create new applications. The `edit` role gives a user sufficient access to act as a developer inside the project, but working under the constraints configured by a project administrator.

You need the `admin` role when managing certain aspects of storage, such as creating new PVCs within a project.

A user with the `admin` role can also give the same access to others in the project who require this function.

Add the role to a user with the `add-role-to-user` subcommand.

```
[user@host ~]$ oc policy add-role-to-user role-name username -n project
```

For example, to add the user `developer` to the role `admin` in the `custom-app` project, use the following command:

```
[user@host ~]$ oc policy add-role-to-user admin developer -n custom-app
```

Use the `oc adm policy who-can` command to view and verify if a user can now create the `persistentvolumeclaims` resource.

```
[user@host ~]$ oc adm policy who-can create persistentvolumeclaims -n custom-app

Namespace: custom-app
Verb:      create
Resource:  persistentvolumeclaims

Users:  admin
        developer
        system:admin

...output omitted...
```

Limits and Quotas

Limits and Quotas provide mechanisms for cluster administrators to create restrictions on the resources provided by the cluster, such as CPU, memory, and storage.

You can restrict consumption of the storage resources that you manage through OpenShift Data Foundation by using limits and quotas to enable sensible project consumption and provide safeguards against resource exhaustion.

Creating quotas and limits for the storage pools available in your cluster offers several benefits.

- Providing practical restraints for storage consumed by project applications.
- Ensuring that applications do not outgrow the data volume.
- Providing indications of unusual storage growth within a cluster or project.
- Controlling aspects of billing incurred on third party cloud storage providers.

Creating Quotas for Storage Resources

Quotas on storage resources are used to manage the consumption of storage providers within the cluster.

There are two types of quotas available to manage the consumption of resources, one for single projects and another for multiple projects.

To create a quota for a single project, define a resource quota to restrict resource usage for a single namespace.

To create a quota for multiple projects, define a shared cluster resource quota to restrict resource usage across more than one project namespace.

Managing Storage Consumption Per Project with Quotas

You can set resource quotas at the project level to manage storage resource constraints.

Project level quotas set constraints that enforce limitations on the storage resources shown in the following table.

Resource Name	Description
`requests.storage`	The sum of storage requests across all PVCs in any state cannot exceed this value.

Resource Name	Description
`persistentvolumeclaims`	The total number of PVCs that can exist in the project.
`<storage-class-name>.storageclass.storage.k8s requests.storage`	The sum of storage requests, across all PVCs in any state that have a matching storage class, cannot exceed this value.
`<storage-class-name>.storageclass.storage.k8s persistentvolumeclaims`	The total number of PVCs with a matching storage class that can exist in the project.

By using the `<storage-class-name>.storageclass.storage.k8s.io` prefixes, you can specify the resource quota for a particular storage class. This approach allows for specialized storage classes, such as those with faster throughput disks or larger storage pools, to have stricter constraints.

For example the `ResourceQuota` definition limits a project to a maximum of three PVCs:

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: pvc-count
spec:
  hard:
    persistentvolumeclaims: "3"
```

Define the object within a file and then use the `oc create` command to deploy the quota.

Managing Cluster Storage Resources with Quotas

Setting a common quota across many projects is an effective way to enforce a baseline storage allocation for the applications that each project contains.

To view the quotas that are being applied to a project, use the `oc describe` command to list the `AppliedClusterResourceQuota` objects.

```
[user@host ~] oc describe AppliedClusterResourceQuota
```

The preceding command displays the quotas that are applied to the project if any exist:

```
Name:   pvc-count
Namespace:  <none>
Created:  42 hours ago
Labels:   <none>
Annotations:  <none>
Label Selector: <null>
AnnotationSelector: <none>
  Resource               Used  Hard
  --------               ----  ----
  persistentvolumeclaims   1     3
```

Setting Limit Ranges for Storage Resources

Using the `LimitRange` setting to define storage request thresholds provides another mechanism to manage the storage consumption of PVCs within a project. Here is an example of a `LimitRange` definition named `pvc-range` that enforces a `1Gi` minimum and `10Gi` maximum storage allocation for each of the PVCs within a project namespace.

```
apiVersion: "v1"
kind: "LimitRange"
metadata:
  name: "pvc-range"
spec:
  limits:
    - type: "PersistentVolumeClaim"
      min:
        storage: "1Gi"
      max:
        storage: "10Gi"
```

Define the object within a file, and then use the `oc create` command to deploy the `LimitRange` setting.

References

For more information, refer to the product documentation for *Red Hat OpenShift Data Foundation* at
https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/

Guided Exercise

Configuring Storage Quotas and Permissions

In this exercise, you will configure quotas and permissions on a Red Hat OpenShift Data Foundation cluster.

Outcomes

You should be able to:

- Manage storage quotas
- Manage storage permissions

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the dependencies for this exercise are set up.

```
[student@workstation ~]$ lab start capacity-quotas
```

Instructions

1. Log in to the OpenShift cluster as the `admin` user with the `redhat` password and create a new project named `capacity-quotas`.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  https://api.ocp4.example.com:6443
Login successful.

...output omitted...

[student@workstation ~]$ oc new-project capacity-quotas
...output omitted...
```

2. Add the `developer` user to the new project.

```
[student@workstation ~]$ oc adm policy add-role-to-user basic-user developer -n \
  capacity-quotas
```

3. View the users who can create PVCs by using the `oc adm policy` command. Notice that the `developer` user is not shown in this output.

```
[student@workstation ~]$ oc adm policy who-can create persistentvolumeclaims

Namespace: capacity-quotas
Verb:      create
Resource:  persistentvolumeclaims

Users:  admin
        system:admin

...output omitted...
```

4. Because this user needs to create and manage storage, add the additional `admin` role to the `developer` user to grant the ability to create PVCs.

   4.1. Add the `admin` role to the `developer` user.

```
[student@workstation ~]$ oc adm policy add-cluster-role-to-user admin \
  developer
clusterrole.rbac.authorization.k8s.io/admin added: "developer"
```

   4.2. Verify that the `developer` user can now create PVCs.

```
[student@workstation ~]$ oc adm policy who-can create persistentvolumeclaims

Namespace: capacity-quotas
Verb:      create
Resource:  persistentvolumeclaims

Users:  admin
        developer
        system:admin

...output omitted...
```

5. Create a `ResourceQuota` object to limit the total number of PVCs for the project.

   5.1. Change to the `~/DO370/labs/capacity-quotas` directory.

```
[student@workstation ~]$ cd ~/DO370/labs/capacity-quotas
```

   5.2. Edit the file `pvc-quota.yaml` to set the resource quota to a limit of one total PVC.

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: pvc-quota
spec:
  hard:
    persistentvolumeclaims: "1"
```

   5.3. Create the quota using the `oc apply` command.

```
[student@workstation capacity-quotas]$ oc apply -f pvc-quota.yaml
resourcequota/pvc-quota created
```

5.4. View the quotas that are currently applied to the `capacity-quotas` project.

```
[student@workstation capacity-quotas]$ oc get ResourceQuotas
NAME        AGE   REQUEST                 LIMIT
pvc-quota   38m   persistentvolumeclaims: 0/1
```

6. As the new `developer`, user create a PVC.

6.1. Log in to the OpenShift cluster as the `developer` user with the `developer` password and create a PVC.

```
[student@workstation capacity-quotas]$ oc login -u developer -p developer \
  https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

6.2. Edit the file `pvc-test1.yaml` to create a 1 GB PVC on block storage.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-test-1
spec:
  accessModes:
  - ReadWriteOnce
  storageClassName: ocs-storagecluster-ceph-rbd
  resources:
    requests:
      storage: 1Gi
```

6.3. Create the PVC.

```
[student@workstation capacity-quotas]$ oc apply -f pvc-test1.yaml
persistentvolumeclaim/pvc-test-1 created
```

7. Review the quotas that are currently applied to the `capacity-quotas` project.

```
[student@workstation capacity-quotas]$ oc get ResourceQuotas
NAME        AGE   REQUEST                 LIMIT
pvc-quota   38m   persistentvolumeclaims: 1/1
```

8. Attempt to create another PVC by using the `oc apply` command and the `pvc-test1.yaml` file.

8.1. Edit the file `pvc-test2.yaml` to create a `1Gi` block storage PVC.

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-test-2
spec:
  accessModes:
  - ReadWriteOnce
  storageClassName: ocs-storagecluster-ceph-rbd
  resources:
    requests:
      storage: 1Gi
```

8.2. Create the PVC. Notice that the error is due to quota enforcement.

```
[student@workstation capacity-quotas]$ oc apply -f pvc-test2.yaml
Error from server (Forbidden): error when creating "pvc-test2.yaml":
 persistentvolumeclaims "pvc-test-2" is forbidden:
  exceeded quota: pvc-quota, requested: persistentvolumeclaims=1, used:
 persistentvolumeclaims=1, limited:
  persistentvolumeclaims=1
```

9. Remove the `pvc-test-1` PVC.

```
[student@workstation capacity-quotas]$ oc delete persistentvolumeclaims pvc-test-1
persistentvolumeclaim "pvc-test-1" deleted
```

10. Remove the `capacity-quotas` project.

```
[student@workstation capacity-quotas]$ oc delete project capacity-quotas
project.project.openshift.io "capacity-quotas" deleted
```

11. Return to the `/home/student` directory.

```
[student@workstation capacity-quotas]$ cd
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish capacity-quotas
```

This concludes the guided exercise.

Extending Application Storage for Red Hat OpenShift Data Foundation

Objectives

After completing this section, you should be able to detect when a storage volume is close to full and expand the volume size without disrupting an application.

Persistent Volume Claim Expansion

Persistent volume claims (PVC) can be extended to enable applications to expand their storage capacity. A PVC can be expanded only if its storage class has the `allowVolumeExpansion` attribute set to `true` which is the default.

To expand an existing PVC, the storage resource request must be modified. This change triggers a `Resizing` event that will notify the provider to perform a resize of the volume.

In this example a 1 GB file PVC mounted on a running pod `nginx` is resized to 10 GB. The `nginx` pod mounts the PVC in the `/space` directory:

```
[user@demo ~]$ oc exec pod/nginx-58fc88c76d-8rwrb -- \
  df -h | grep /space
172.30.164.48:6789,172.30.62.181:6789,172.30.224.197:6789:/volumes/csi/csi-
vol-6c453868-fc01-11eb-b964-0a580a0b0019/cb1e71b6-a730-4ccf-892e-d21fdcbb2e17   1G
     0   1G   0% /space
```

The `CAPACITY` field shows the storage assigned to the PVC:

```
[user@demo ~]$ oc get pvc
NAME             STATUS   ... CAPACITY   ACCESS MODES   STORAGECLASS
  ...
example-cephfs   Bound    ... 1Gi        RWX            ocs-storagecluster-
cephfs ...
```

PVC size can be increased by editing the resource:

```
[user@demo ~]$ oc edit pvc/example-cephfs
...output omitted...
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  storageClassName: ocs-storagecluster-cephfs
  volumeMode: Filesystem
  volumeName: pvc-8754f06d-1d03-4bd6-be07-29084a5eed7a
...output omitted...
```

The `oc patch` command can also be used to increase the size of the PVC:

```
[user@demo ~]$ oc patch pvc/example-cephfs \
  -p '{"spec":{"resources":{"requests": {"storage": "10Gi"}}}}'
persistentvolumeclaim/example-cephfs patched
```

Note

To increase the size of a block PVC, follow the same procedure.

After a successful resize, the field `Capacity` will be set to the new value. The `Events:` section contains the log of the resize procedure.

```
[user@demo ~]$ oc describe pvc/example-cephfs
...output omitted...
Capacity:      10Gi
...output omitted...
Events:
  Type     Reason              ...    From         Message
  ----     ------                 ----         -------

  Normal   Resizing             ... external-resizer openshift-
storage.cephfs.csi.ceph.com
                 External resizer is resizing volume pvc-8754f06d-1d03-4bd6-
be07-29084a5eed7a
  Warning  ExternalExpanding      ...                  volume_expand

                 Ignoring the PVC: didn't find a plugin capable of expanding the
 volume; waiting for an external controller to process this PVC.
  Normal   VolumeResizeSuccessful ... external-resizer openshift-
storage.cephfs.csi.ceph.com
            Resize volume succeeded
...output omitted...
```

For block devices a `FileSystemResizeRequired` event is also triggered.

```
Normal   Resizing                   1m   external-resizer openshift-
storage.rbd.csi.ceph.com
                                             External resizer is resizing
 volume pvc-74803641-9955-46da-9138
-faf165b25131
  Normal   FileSystemResizeRequired    1m   external-resizer openshift-
storage.rbd.csi.ceph.com
                                             Require file system resize of
 volume on node
  Normal   FileSystemResizeSuccessful  4s    kubelet
                                             MountVolume.NodeExpandVolume
 succeeded for volume "pvc-74803641
-9955-46da-9138-faf165b25131"
```

Pods that mount block PVCs do not experience disruption when a resize is performed. OpenShift Data Foundation ensures that there is stability during PVC expansion.

```
[user@demo ~]$ oc exec pod/nginx-block-5dfdd9cb7c-trzjl -- \
  df -h  | grep /space
/dev/rbd0        10G   45M   10G   1% /space
```

References

For more information, refer to the storage section of the official documentation of the *Red Hat OpenShift Container Platform* at https://docs.openshift.com/container-platform/4.7/storage/expanding-persistent-volumes.html

Guided Exercise

Extending Application Storage for Red Hat OpenShift Data Foundation

In this exercise, you will detect when a storage volume is close to full and expand the volume size without disrupting an application.

Outcomes

You should be able to:

- Identify storage classes that allow volume expansion.
- Verify used and free space inside the volume.
- Extend the size of a volume without disrupting an application.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the dependencies for this exercise are properly set up.

```
[student@workstation ~]$ lab start capacity-extend
```

Instructions

1. Log in to the OpenShift cluster as the `admin` user with the `redhat` password and create a new project called `capacity-extend-ge`.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  https://api.ocp4.example.com:6443
Login successful.

...output omitted...
[student@workstation ~]$ oc new-project capacity-extend-ge
...output omitted...
```

2. Use an OpenShift template to create a persistent PostgreSQL database.

   2.1. Create a PostgreSQL template that allows storageclass configuration by using the `DO370/labs/capacity-extend/postgresql-persistent-ge.json` file. .

```
[student@workstation ~]$ oc create -f \
  DO370/labs/capacity-extend/postgresql-persistent-ge.json
```

   2.2. Verify that the storage class is the block storage offered by OpenShift Data Foundation.

```
[student@workstation ~]$ oc get storageclasses
NAME                          PROVISIONER
 RECLAIMPOLICY   VOLUMEBINDINGMODE   ...
lso-volumeset                 kubernetes.io/no-provisioner           Delete
    WaitForFirstConsumer
nfs-storage (default)         nfs-storage-provisioner                Delete
    Immediate
ocs-storagecluster-ceph-rbd   openshift-storage.rbd.csi.ceph.com     Delete
    Immediate
ocs-storagecluster-ceph-rgw   openshift-storage.ceph.rook.io/bucket  Delete
    Immediate
ocs-storagecluster-cephfs     openshift-storage.cephfs.csi.ceph.com  Delete
    Immediate
openshift-storage.noobaa.io   openshift-storage.noobaa.io/obc        Delete
    Immediate
```

2.3. Use an OpenShift template to create a persistent PostgreSQL database by setting the following template options:

- `STORAGECLASS_NAME=ocs-storagecluster-ceph-rbd`, the name of the PVC.
- `VOLUME_CAPACITY=30Mi`, the initial volume capacity.
- `POSTGRESQL_USER=student`, the name of the PostgreSQL user that manages the database.
- `POSTGRESQL_PASSWORD=redhat`, the password for the PostgreSQL user.
- `POSTGRESQL_DATABASE=capacity-extend-ge`, the name of the database that will be served by the deployment.
- `DATABASE_SERVICE_NAME=pg-capacity-extend-ge`, the name of the service to expose the database.

```
[student@workstation ~]$ oc new-app --name=pg-capacity-extend \
  --template=postgresql-persistent-sc \
  -p STORAGECLASS_NAME=ocs-storagecluster-ceph-rbd \
  -p VOLUME_CAPACITY=30Mi \
  -p POSTGRESQL_USER=student \
  -p POSTGRESQL_PASSWORD=redhat \
  -p POSTGRESQL_DATABASE=capacity-extend-ge \
  -p DATABASE_SERVICE_NAME=pg-capacity-extend-ge

...output omitted...
     The following service(s) have been created in your project: pg-capacity-
extend-ge.

            Username: student
            Password: redhat
       Database Name: capacity-extend-ge
      Connection URL: postgresql://pg-capacity-extend-ge:5432/
...output omitted...
--> Creating resources ...
```

```
    secret "pg-capacity-extend-ge" created
    service "pg-capacity-extend-ge" created
    persistentvolumeclaim "pg-capacity-extend-ge" created
    deploymentconfig.apps.openshift.io "pg-capacity-extend-ge" created
--> Success
...output omitted...
```

Note

As an alternative to typing the command above, run the `~/DO370/labs/capacity-extend/pg-capacity-extend.sh` script

3. Verify that the PostgreSQL database did not deploy successfully by checking the pod status and logs.

3.1. Check the pod status.

```
[student@workstation ~]$ oc get pods
NAME                              READY   STATUS             RESTARTS   AGE
pg-capacity-extend-ge-1-5bjgg     0/1     CrashLoopBackOff   3          77s
pg-capacity-extend-ge-1-deploy    1/1     Running            0          83s
```

3.2. Review the logs for the pod that has the status: `CrashLoopBackOff`. The pod name might not be the same as the one in the previous output. Verify that there is a **No space left on device** error.

```
[student@workstation ~]$ oc logs pod/pg-capacity-extend-ge-1-5bjgg
...output omitted...
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... 2021-06-15 18:40:24.325 UTC [27]
 FATAL:  could not write to file "base/12344/3119": No space left on device
2021-06-15 18:40:24.325 UTC [27] STATEMENT:  CREATE DATABASE template0 IS_TEMPLATE
 = true ALLOW_CONNECTIONS = false;

child process exited with exit code 1
initdb: removing contents of data directory "/var/lib/pgsql/data/userdata"
...output omitted...
```

4. Inspect the PVC that was created by the template.

4.1. Review the list of PVCs in the current project.

```
[student@workstation ~]$ oc get pvc
NAME                    STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS
               AGE
pg-capacity-extend-ge   Bound      ...      30Mi         RWO         ocs-
storagecluster-ceph-rbd   9m59s
```

Notice that the error in previous steps occurred because the PVC size is too small to meet PostgreSQL minimum data directory requirements.

4.2. Extend the size of the PVC.

```
[student@workstation ~]$ oc patch pvc/pg-capacity-extend-ge \
  -p '{"spec":{"resources":{"requests": {"storage": "150M"}}}}'
persistentvolumeclaim/pg-capacity-extend-ge patched
```

4.3. Verify that the PVC is resized successfully.

```
[student@workstation ~]$ oc describe pvc pg-capacity-extend-ge
...output omitted...
Conditions:
  Type                      Status   ...               Reason  Message
  ----                      ------                   ------------------
  FileSystemResizePending   True     ...             Waiting for user to
 (re-)start a pod to finish file system resize of volume on node.
...output omitted...
  Warning  ...  Ignoring the PVC: didn't find a plugin capable of expanding the
 volume; waiting for an external controller to process this PVC.
  Normal   ...  External resizer is resizing volume pvc-de36aad4-
a771-4c68-99e1-5d3eddf8f3e4
  Normal   ...  Require file system resize of volume on node
```

4.4. Restart the deployment to finish the resizing process.

```
[student@workstation ~]$ oc rollout latest \
  deploymentconfig.apps.openshift.io/pg-capacity-extend-ge
deploymentconfig.apps.openshift.io/pg-capacity-extend-ge rolled out
```

4.5. Verify that the deployment successfully restarted.

```
[student@workstation ~]$ oc get pods
NAME                             READY   STATUS      RESTARTS   AGE
pg-capacity-extend-ge-2-deploy   0/1     Completed   0          111s
pg-capacity-extend-ge-2-f5qhn    1/1     Running     0          108s
```

5. Verify that the PVC scaled up to the new size.

5.1. Verify the new size of the PVC.

```
[student@workstation ~]$ oc get pvc
NAME                   STATUS   ...       CAPACITY   ACCESS MODES   STORAGECLASS
                AGE
pg-capacity-extend-ge   Bound    ...       144Mi        RWO        ocs-
storagecluster-ceph-rbd   38m
```

5.2. Verify that the pod mounted the partition with the new PVC size.

```
[student@workstation ~]$ oc rsh \
  deploymentconfig.apps.openshift.io/pg-capacity-extend-ge
sh-4.4$ df -h
Filesystem                          Size  Used Avail Use% Mounted on
...output omitted...
/dev/rbd2                           139M   47M   92M  34% /var/lib/pgsql/data
...output omitted...
```

6. Remove the project and the template from the cluster.

   6.1. Delete the `capacity-extend-ge` project.

```
[student@workstation ~]$ oc delete project capacity-extend-ge
```

   6.2. Delete the `postgresql-persistent-sc` template.

```
[student@workstation ~]$ oc delete template postgresql-persistent-sc -n openshift
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish capacity-extend
```

This concludes the guided exercise.

Adding Disks to the Red Hat OpenShift Data Foundation Cluster

Objectives

After completing this section, you should be able to add additional disks to a Red Hat OpenShift Data Foundation cluster.

Increasing Cluster Capacity

There are multiple ways to increase cluster capacity. For internal deployments, expand capacity by adding disks to nodes or adding new nodes. If you are using storage from a cloud provider or object storage, the expansion depends on the object classes associated with the provider. For internal deployments, you can add disks or nodes, but external deployments vary in their approach to expanding capacity.

Internal Deployments

The Local Storage Operator (LSO) operator discovers disks for each node and creates PVs that are used by the Rook-Ceph operator to create or extend the Ceph cluster.

For internal deployments, cluster capacity can be increased by:

- Adding disks to the nodes
- Adding nodes
- Enabling any existing disks that are unused on the nodes

Enabling Disks

The LSO operator creates `localvolumediscoveryresult` objects from the nodes that are included in the `nodeSelector` field of a `localvolumediscovery` object.

```
[user@demo ~]$  oc describe localvolumediscovery/auto-discover-devices \
  -n openshift-local-storage
...output omitted...
Spec:
  Node Selector:
    Node Selector Terms:
      Match Expressions:
        Key:       kubernetes.io/hostname
        Operator:  In
        Values:
          worker01
          worker02
          worker03
Status:
  Conditions:
    Last Transition Time:  2021-08-12T15:40:18Z
    Message:               successfully running 3 out of 3 discovery daemons
    Status:                True
```

```
    Type:                  Available
  Observed Generation:     1
  Phase:                   Discovering
Events:                    <none>
```

A `localvolumediscoveryresult` object is created for each discovered node and contains the set of disks discovered for the node.

```
[user@demo ~]$ oc get localvolumediscoveryresult \
  -n openshift-local-storage
NAME                          AGE
discovery-result-worker01     18d
discovery-result-worker02     18d
discovery-result-worker03     18d
```

To get the disks available on a node, describe the `localvolumediscoveryresult` object that corresponds to it:

```
[user@demo ~]$ oc describe \
  localvolumediscoveryresult/discovery-result-worker01 \
  -n openshift-local-storage
    ...output omitted...
    Device ID:  /dev/disk/by-id/virtio-aaf40cdd-da3d-4d6d-9
    Fstype:
    Model:
    Path:       /dev/vdc
    Property:   Rotational
    Serial:     aaf40cdd-da3d-4d6d-9
    Size:       536870912000
    Status:
      State:              Available
    Type:                 disk
    Vendor:               0x1af4
  Discovered Time Stamp:  2021-08-17T05:07:40Z
Events:                   <none>
```

The LSO operator should have already created a PV for the device.

```
[user@demo ~]$ oc get pv -n openshift-local-storage
NAME                                    CAPACITY   ...   STATUS ...

local-pv-3e1c7553                       500Gi      ...   Bound     ...
local-pv-59db6771                       500Gi      ...   Available ...
local-pv-6534bb72                       500Gi      ...   Bound     ...
local-pv-8fc03e0d                       500Gi      ...   Available ...
local-pv-a87fae9d                       500Gi      ...   Bound     ...
local-pv-fd21f260                       500Gi      ...   Available ...
...output omitted...
```

To get the PVs created for a node, filter the `oc get pv` command output by the `kubernetes.io/hostname` label:

```
[user@demo ~]$ oc get pv -l kubernetes.io/hostname=worker01
NAME                CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS
local-pv-a87fae9d   500Gi      RWO            Delete           Bound
local-pv-fd21f260   500Gi      RWO            Delete           Available
```

In this example, the PV `local-pv-fd21f260` has an **Available** status. This status means that the Rook-Ceph operator can use this PV as a disk for the internal Ceph storage.

To add available disks to the cluster, create a `LocalVolume` object that includes the available disks, and increase the `storageDeviceSets` setting on the `storagecluster/ocs-storagecluster` object. This triggers the creation of new Ceph OSD pods to provide extended storage.

The `LocalVolume` object `devicePath` attribute needs the available PV device ID. This ID can be obtained by describing the node's `localvolumediscoveryresult`.

```
[user@demo ~]$ oc describe \
  localvolumediscoveryresult/discovery-result-worker01 \
  -n openshift-local-storage
    ...output omitted...
    Device ID:  /dev/disk/by-id/virtio-aaf40cdd-da3d-4d6d-9
    Fstype:
    Model:
    Path:       /dev/vdc
    Property:   Rotational
    Serial:     aaf40cdd-da3d-4d6d-9
    Size:       536870912000
    Status:
      State:              Available
    Type:                 disk
    Vendor:               0x1af4
  Discovered Time Stamp:  2021-08-17T05:07:40Z
Events:                   <none>
```

Describe each of the three nodes in the cluster by using the node's `localvolumediscovery` setting to find available disks and their corresponding identifiers.

The following is a `LocalVolume` manifest that makes the disks available to the `storagecluster` object.

```
apiVersion: local.storage.openshift.io/v1
kind: LocalVolume
metadata:
  name: expanded-local-block
  namespace: openshift-local-storage
spec:
  nodeSelector:
    nodeSelectorTerms:
    - matchExpressions:
        - key: cluster.ocs.openshift.io/openshift-storage
          operator: In
          values:
          - ""
```

```
  storageClassDevices:
    - storageClassName: localblock
      volumeMode: Block
      devicePaths:
        - /dev/disk/by-id/virtio-aaf40cdd-da3d-4d6d-9
        - /dev/disk/by-id/virtio-fe40db23-a129-4dff-9
        - /dev/disk/by-id/virtio-a0d9e043-c4b9-4a6d-8
```

Use the `oc patch` or `oc edit` commands to increase the `storageDeviceSets` attribute and make these changes available to the cluster.

```
[user@demo ~]$ oc patch storagecluster/ocs-storagecluster \
  -n openshift-storage --type json -p '[{"op": "replace", \
  "path": "/spec/storageDeviceSets/0/count", "value": 2}]'
storagecluster.ocs.openshift.io/ocs-storagecluster patched
```

The Rook-Ceph operator triggers the creation of new pods of object storage daemons (OSDs). To verify that the disks are added, list the pods with the label `app=rook-ceph-osd`.

```
[user@demo ~]$ oc get pods -n openshift-storage -l app=rook-ceph-osd
NAME                               READY   STATUS    RESTARTS   AGE
rook-ceph-osd-0-6b45486b8b-shdff   2/2     Running   0          7d12h
rook-ceph-osd-1-58f6449b7b-6lf6c   2/2     Running   0          7d12h
rook-ceph-osd-2-699b87d547-gsmgd   2/2     Running   0          7d12h
rook-ceph-osd-3-7b5c5c66f4-5g5jx   2/2     Running   0          8s
rook-ceph-osd-4-865cd68748-jflq4   2/2     Running   0          7s
rook-ceph-osd-5-9f6b9b479-4g5x2    2/2     Running   0          6s
```

After the OSD pods are up and running, the PVs change their status from `Available` to `Bound`:

```
NAME                                CAPACITY   ...   STATUS ...

local-pv-3e1c7553                   500Gi      ...   Bound     ...
local-pv-59db6771                   500Gi      ...   Bound ...
local-pv-6534bb72                   500Gi      ...   Bound     ...
local-pv-8fc03e0d                   500Gi      ...   Bound ...
local-pv-a87fae9d                   500Gi      ...   Bound     ...
local-pv-fd21f260                   500Gi      ...   Bound ...
...output omitted...
```

This procedure is the same when adding new disks to the nodes.

Adding Nodes

If new nodes are added to the cluster, the storage capacity can be expanded by adding the node name to the `LocalVolumeDiscovery` and `LocalVolumeSet` objects.

```
apiVersion: local.storage.openshift.io/v1alpha1
kind: LocalVolumeDiscovery
metadata:
  name: auto-discover-devices
  namespace: openshift-local-storage
```

```
spec
  nodeSelector:
    nodeSelectorTerms:
    - matchExpressions:
      - key: kubernetes.io/hostname
        operator: In
        values
        - worker01
        - worker02
        - worker03
        - worker04
```

```
apiVersion: local.storage.openshift.io/v1alpha1
kind: LocalVolumeSet
metadata:
  name: lso-volumeset
  namespace: openshift-local-storage
spec:
  deviceInclusionSpec:
    deviceTypes:
    - disk
    - part
    minSize: 1Gi
  nodeSelector:
    nodeSelectorTerms:
    - matchExpressions:
      - key: kubernetes.io/hostname
        operator: In
        values:
        - worker01
        - worker02
        - worker03
        - worker04
  storageClassName: lso-volumeset
  volumeMode: Block
```

Alternatively, nodes can be labeled with the `node-role.kubernetes.io/worker=` `cluster.ocs.openshift.io/openshift-storage=` label to make the storage cluster use the nodes automatically.

```
[user@demo ~]$ oc label nodes \
  -l node-role.kubernetes.io/worker= cluster.ocs.openshift.io/openshift-storage=
```

References

For more information, refer to the scaling storage of the *Red Hat OpenShift Data Foundation* documentation at https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/html-single/scaling_storage/index

Guided Exercise

Adding Disks to the Red Hat OpenShift Data Foundation Cluster

In this exercise, you will add additional disks to an OpenShift Data Foundation cluster.

Outcomes

You should be able to:

- Identify free block device space in cluster nodes.
- Add a block device to cluster.
- Increase the OpenShift Data Foundation cluster capacity.
- Verify the expanded capacity.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the proper dependencies for this exercise are set up.

```
[student@workstation ~]$ lab start capacity-disk
```

Instructions

1. Log in to the OpenShift cluster as the `admin` user with the `redhat` password.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

2. Inspect the storage elements of the cluster and verify if the worker nodes have disks available to add to the cluster.

   2.1. List the Ceph object storage daemons (Ceph OSDs) created by the storage cluster.

```
[student@workstation ~]$ oc get pods -n openshift-storage -l app=rook-ceph-osd
rook-ceph-osd-0-54456cc99d-prhmj                                  1/1
 Running ...
rook-ceph-osd-1-75bb875d6c-9gshf                                  1/1
 Running ...
rook-ceph-osd-2-7c4f7c9cff-458fr                                  1/1
 Running ...
```

2.2. List the workers that were added to the storage cluster during installation.

```
[student@workstation ~]$ oc get localvolumediscoveryresults \
  -n openshift-local-storage
NAME                        AGE
discovery-result-worker01   1d
discovery-result-worker02   1d
discovery-result-worker03   1d
```

2.3. Describe the `discovery-result-worker01` `LocalVolumeDiscoveryResult` and verify that `/dev/vdc` disk status is set to available.

```
[student@workstation ~]$ oc describe \
  localvolumediscoveryresult/discovery-result-worker01 \
  -n openshift-local-storage

...output omitted...
    Device ID:  /dev/disk/by-id/virtio-a8a897e7-40fa-4156-8
    Fstype:
    Model:
    Path:       /dev/vdc
    Property:   Rotational
    Serial:
    Size:       536870912000
    Status:
      State:              Available
    Type:                 disk
...output omitted...
```

Repeat the same procedure for all the nodes to verify that they have disks available. All the nodes must have the `/dev/vdc` device available.

3. Find the device IDs of the available disks and add them to the cluster.

3.1. For each node, find the unique device ID that you retrieved in the previous step (`/dev/vdc`).

```
[student@workstation ~]$ oc describe \
  localvolumediscoveryresult/discovery-result-worker01 \
  -n openshift-local-storage
...output omitted...
    Device ID:  /dev/disk/by-id/virtio-a8a897e7-40fa-4156-8
    Fstype:
    Model:
    Path:       /dev/vdc
    Property:   Rotational
    Serial:
    Size:       536870912000
    Status:
      State:              Available
    Type:                 disk

...output omitted...
```

Alternatively, you can use the `oc debug node/<NodeName>` command to find the device ID by listing the `/dev/disk/by-id` directory.

3.2. Add the disks to the cluster by creating a new `LocalVolume` object that uses the file `~/DO370/labs/capacity-disk/localvolumes-expand.yaml` as starting point. Replace the highlighted lines with the IDs of the devices found in the previous step.

```
apiVersion: local.storage.openshift.io/v1
kind: LocalVolume
metadata:
  name: expanded-local-block
  namespace: openshift-local-storage
spec:
  nodeSelector:
    nodeSelectorTerms:
    - matchExpressions:
        - key: cluster.ocs.openshift.io/openshift-storage
          operator: In
          values:
          - ""
  storageClassDevices:
    - storageClassName: lso-volumeset
      volumeMode: Block
      devicePaths:
        - /dev/disk/by-id/virtio-a8a897e7-40fa-4156-8
        - /dev/disk/by-id/virtio-b493b6fa-5fb0-430f-8
        - /dev/disk/by-id/virtio-de390e6d-a057-4800-a
```

```
[student@workstation ~]$ oc apply -f \
  ~/DO370/labs/capacity-disk/localvolumes-expand.yaml
localvolume.local.storage.openshift.io/expanded-local-block created
```

4. Expand the storage capacity by setting the `StorageDeviceSets count` value to 2.

```
[student@workstation ~]$ oc patch storagecluster/ocs-storagecluster \
  -n openshift-storage --type json -p '[{"op": "replace", \
  "path": "/spec/storageDeviceSets/0/count", "value": 2}]'
storagecluster.ocs.openshift.io/ocs-storagecluster patched
```

Wait a couple of minutes for the changes to be applied.

5. Verify that the devices are added to the cluster.

5.1. List the new Ceph OSDs.

```
[student@workstation ~]$ oc get pods -n openshift-storage -l app=rook-ceph-osd
rook-ceph-osd-3-67d957c977-5vjqt                                  0/1
 Pending ...
rook-ceph-osd-4-6c65b8bdd-kmr7d                                   1/1
 Running ...
rook-ceph-osd-5-7d74799d5b-g2nlf                                  0/1
 Pending ...
...output omitted...
```

5.2. Verify that the status of the devices is changed to `Bound`.

```
[student@workstation ~]$ oc get pv
NAME                                       CAPACITY   ...   STATUS ...
...output omitted...
local-pv-bef80334                          500Gi      ...    Bound    ...
local-pv-c050c922                          500Gi      ...    Bound    ...
local-pv-fb71734e                          500Gi      ...    Bound    ...
...output omitted...
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish capacity-disk
```

This concludes the guided exercise.

Lab

Managing Red Hat OpenShift Data Foundation Block and File Storage Capacity

In this lab, you will expand the OpenShift Data Foundation storage capacity for an existing cluster.

Outcomes

You should be able to:

- Verify cluster storage capacity.
- Expand storage for an application.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the environment is properly setup for this lab.

```
[student@workstation ~]$ lab start capacity-review
```

Instructions

1. As the `admin` user, create a project named `capacity-review`.
2. Use the `postgresql-persistent-sc` template to create a persistent PostgreSQL database deployment named `capacity-review-pg` with a `300 Mi` PVC and set the following parameters:
   - `STORAGECLASS_NAME=ocs-storagecluster-ceph-rbd`
   - `VOLUME_CAPACITY=300Mi`
   - `POSTGRESQL_USER=student`
   - `POSTGRESQL_PASSWORD=redhat`
   - `POSTGRESQL_DATABASE=capacity-review-pg`
   - `DATABASE_SERVICE_NAME=capacity-review-pg`
3. Increase the `capacity-review-pg` deployment PVC capacity to `400 Mi`.
   - Use the following jsonpath with the `oc patch` command to modify the PVC. `{"spec": {"resources":{"requests": {"storage": "400Mi"}}}}`
4. Verify the capacity of the `pvc/pg-capacity-extend-ge` PVC.

Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade capacity-review
```

Finish

As the `student` user on the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish capacity-review
```

This concludes the lab.

Solution

Managing Red Hat OpenShift Data Foundation Block and File Storage Capacity

In this lab, you will expand the OpenShift Data Foundation storage capacity for an existing cluster.

Outcomes

You should be able to:

- Verify cluster storage capacity.
- Expand storage for an application.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the environment is properly setup for this lab.

```
[student@workstation ~]$ lab start capacity-review
```

Instructions

1. As the `admin` user, create a project named `capacity-review`.

   1.1. Log in to the OpenShift cluster as the `admin` user with the `redhat` password. Switch to the project called `capacity-review`.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  https://api.ocp4.example.com:6443
...output omitted...

[student@workstation ~]$ oc project capacity-review
...output omitted...
```

2. Use the `postgresql-persistent-sc` template to create a persistent PostgreSQL database deployment named `capacity-review-pg` with a `300 Mi` PVC and set the following parameters:

   - `STORAGECLASS_NAME=ocs-storagecluster-ceph-rbd`
   - `VOLUME_CAPACITY=300Mi`
   - `POSTGRESQL_USER=student`
   - `POSTGRESQL_PASSWORD=redhat`
   - `POSTGRESQL_DATABASE=capacity-review-pg`
   - `DATABASE_SERVICE_NAME=capacity-review-pg`

2.1. Create the PostgreSQL deployment by setting the following template options:

- `STORAGECLASS_NAME=ocs-storagecluster-ceph-rbd`, the storage class to use
- `VOLUME_CAPACITY=300Mi`, the initial volume capacity
- `POSTGRESQL_USER=student`, the name of the PostgreSQL user that manages the database
- `POSTGRESQL_PASSWORD=redhat`, the password for the PostgreSQL user
- `POSTGRESQL_DATABASE=capacity-review-pg`, the name of the database that will be served by the deployment
- `DATABASE_SERVICE_NAME=capacity-review-pg`, the name of the service to expose the database

```
[student@workstation ~]$ oc new-app --name=capacity-review-pg \
  --template=postgresql-persistent-sc \
  -p STORAGECLASS_NAME=ocs-storagecluster-ceph-rbd \
  -p VOLUME_CAPACITY=300Mi \
  -p POSTGRESQL_USER=student \
  -p POSTGRESQL_PASSWORD=redhat \
  -p POSTGRESQL_DATABASE=capacity-review-pg \
  -p DATABASE_SERVICE_NAME=capacity-review-pg

...output omitted...

          The following service(s) have been created in your project: capacity-
review.
            Username: student
            Password: redhat
       Database Name: capacity-review-pg
      Connection URL: postgresql://capacity-review-pg:5432/

...output omitted...

--> Creating resources ...
    secret "capacity-review-pg" created
    service "capacity-review-pg" created
    persistentvolumeclaim "capacity-review-pg" created
    deploymentconfig.apps.openshift.io "capacity-review-pg" created
--> Success

...output omitted...
```

3. Increase the `capacity-review-pg` deployment PVC capacity to `400 Mi`.

- Use the following jsonpath with the `oc patch` command to modify the PVC. `{"spec": {"resources":{"requests": {"storage": "400Mi"}}}}`

3.1. Use `oc patch` to extend the `pvc/capacity-review-pg` capacity to 400 Mi.

```
[student@workstation ~]$ oc patch pvc/capacity-review-pg \
  -p '{"spec":{"resources":{"requests": {"storage": "400Mi"}}}}'
persistentvolumeclaim/capacity-review-pg patched
```

3.2. Use the `oc rollout` command to finish the resizing process.

```
[student@workstation ~]$ oc rollout latest \
  deploymentconfig.apps.openshift.io/capacity-review-pg
deploymentconfig.apps.openshift.io/capacity-review-pg rolled out
```

4. Verify the capacity of the `pvc/pg-capacity-extend-ge` PVC.

4.1. Verify the new size of the PVC.

```
[student@workstation ~]$ oc get pvc
NAME                STATUS  ...   CAPACITY   ...  STORAGECLASS            ...
capacity-review-pg  Bound   ...   400Mi      ...  storagecluster-ceph-rbd ...
```

4.2. Verify that the pod mounted the partition with the new PVC size.

```
[student@workstation ~]$ oc exec -it deploymentconfig/capacity-review-pg -- \
  df -h /var/lib/pgsql/data
Filesystem      Size  Used Avail Use% Mounted on
/dev/rbd0       362M   48M  311M  14% /var/lib/pgsql/data
```

Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade capacity-review
```

Finish

As the `student` user on the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish capacity-review
```

This concludes the lab.

Summary

In this chapter, you learned how to:

- View important storage metrics.
- Inspect the cluster storage capacity using the web console.
- Manage storage consumption using quotas and permissions.
- Extend application and cluster storage capacity.

Chapter 5

Performing Backup and Restore of Kubernetes Block and File Volumes

Goal

Backup and restore application data using Kubernetes CSI APIs.

Objectives

- Compare traditional backup and containerized applications backup approaches.
- Performing backup and restore of applications resources, images, and data volumes.
- Create volume snapshots for use in restoration and/or duplication of storage volumes.

Sections

- Introducing Backup Concepts (and Quiz)
- Backing up and Restoring Kubernetes Applications (and Guided Exercise)
- Creating Volume Snapshots and Clones (and Guided Exercise)

Lab

Performing Backup and Restore of Kubernetes Block and File Volumes

Introducing Backup Concepts

Objectives

After completing this section, you should be able to compare traditional backup and containerized applications backup approaches.

Application Backup Strategies

Each application handles data in a different way. Some applications require low-level file operations or locking, and other applications only rely on standard file system operations, such as create, read, update, and delete.

Backing up application data is an important task that should be performed regularly. However, each application has a different backup method and some applications even have a set of tools to create and restore backups. One solution is to create a cron job to automate the backup process for your application.

A common backup method for applications that do not use low-level operations is to stop the application and use a tool like `rsync` to copy the data or `tar` to archive it. Database applications have their own specialized tools to create backups. You can take a backup of the database without stopping it by using the following tools:

- `mysqldump`
- `pg_dump`
- `mongodump`

The following methods are alternatives ways to back up applications:

- File system back up with tools such as `dump` or `xfsdump`.
- Volume snapshots from the Logical Volume Manager (LVM) or a storage controller.
- Virtual Machine snapshots or clones.

Container Backup Strategies

The backup and restore strategies are different when running containerized application workloads in OpenShift. Applications run as containers inside pods, and volumes are mounted on each container.

The backup method is different if the application is stateless or stateful, and it also varies depending on which storage type it uses (block, file, or object).

Container Storage Interface (CSI) Volume Snapshot and Cloning

To back up stateless applications you can save the original resource definitions in YAML or JSON format and redeploy your application, if needed. For stateful applications that store data in block or file persistent volumes (PVs), two operations can be performed to backup the data stored in a Kubernetes Container Storage Interface (CSI) volume.

- Persistent volume cloning
- Persistent volume snapshot

If the application uses an object bucket claim, then you can create a `Job` or `CronJob` to sync all the objects from one bucket to another.

OpenShift APIs for Data Protection

Red Hat developed the OpenShift API for Data Protection (OADP) that enables our partners to integrate their data protection solutions with Red Hat OpenShift. The OADP operator is developed in the `Konveyor` upstream project. This operator sets up and installs `Velero` on the OpenShift cluster.

The OADP operator creates a copy of the following items when backing up an application on a given namespace.

- Resource definitions that are stored in in YAML or JSON format.
- Container images that are stored in the internal registry.
- Data that is stored in persistent volumes using CSI volume snapshots and clones.

References

For more information about file system backup, refer to the *Managing File Systems* documentation for *Red Hat Enterprise Linux* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/managing_file_systems/index

For more information about logical volumes, refer to the *Configuring and Managing Logical Volumes* documentation for *Red Hat Enterprise Linux* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_logical_volumes/index

For more information about logical volume snapshots, refer to the *Snapshot Logical Volumes* chapter in the *Configuring and Managing Logical Volumes* documentation for *Red Hat Enterprise Linux* at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_logical_volumes/index#assembly_snapshot-volumes-configuring-and-managing-logical-volumes

For more information about OpenShift volume snapshots and clones, refer to the *Volume Snapshots* and *Volume Cloning* chapters in the *Managing and Allocating Storage Resources* documentation for *Red Hat OpenShift Data Foundation* at https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/html-single/managing_and_allocating_storage_resources/index#volume-snapshots_rhocs

Quiz

Introducing Backup Concepts

Choose the correct answers to the following questions.

1. Which three of the following are types of data storage managed by Red Hat OpenShift Data Foundation? (Choose three.)

a. Block
b. File
c. Swap
d. Object

2. Which two of the following tools can you use to create a backup of the files on a given directory? (Choose two.)

a. `tar`
b. `dd`
c. `rm`
d. `rsync`

3. You can perform CSI volume snapshot and cloning for which two types of persistent volumes? (Choose two.)

a. `emptyDir`
b. Block
c. File
d. Object

4. Which statement is true regarding backing up object buckets?

a. You can create a snapshot or clone of an object bucket because they are similar to persistent volumes.
b. Object buckets cannot be backed up.
c. You can sync the contents from one bucket to another to create a backup.
d. You can mount the object bucket in another pod and copy the contents using `rsync`.

5. Which three of the following methods can you use to back up traditional applications? (Choose three.)

a. File system backup
b. Kubernetes `Job` or `CronJob`
c. LVM volume snapshots
d. Virtual machine clones

Solution

Introducing Backup Concepts

Choose the correct answers to the following questions.

1. Which three of the following are types of data storage managed by Red Hat OpenShift Data Foundation? (Choose three.)

a. Block
b. File
c. Swap
d. Object

2. Which two of the following tools can you use to create a backup of the files on a given directory? (Choose two.)

a. `tar`
b. `dd`
c. `rm`
d. `rsync`

3. You can perform CSI volume snapshot and cloning for which two types of persistent volumes? (Choose two.)

a. `emptyDir`
b. Block
c. File
d. Object

4. Which statement is true regarding backing up object buckets?

a. You can create a snapshot or clone of an object bucket because they are similar to persistent volumes.
b. Object buckets cannot be backed up.
c. You can sync the contents from one bucket to another to create a backup.
d. You can mount the object bucket in another pod and copy the contents using `rsync`.

5. Which three of the following methods can you use to back up traditional applications? (Choose three.)

a. File system backup
b. Kubernetes `Job` or `CronJob`
c. LVM volume snapshots
d. Virtual machine clones

Backing up and Restoring Kubernetes Applications

Objectives

After completing this section, you should be able to performing backup and restore of applications resources, images, and data volumes.

Backing up Application Resources

You can create a backup of the application resource for stateless applications. The following command removes some of the `metadata`, and `status` fields that are not needed in the JSON resource definition.

```
[user@demo ~]$ oc get deployment/example -o json | \
  jq '. | del(
    .metadata.uid,
    .metadata.selfLink,
    .metadata.generation,
    .metadata.resourceVersion,
    .metadata.creationTimestamp,
    .metadata.managedFields,
    .metadata.annotations."deployment.kubernetes.io/revision",
    .metadata.annotations."kubectl.kubernetes.io/last-applied-configuration",
    .status
  )' > deployment-backup.json
```

Backing up Stateful Applications

Stateful applications store their data in block or file persistent volumes. The method for backing up the data is different depending of the application.

- The application must be stopped to back up the data.
- The application has a specialized backup tool like `mysqldump`.

Backing up Application Data from Persistent Volumes

Scale the deployment to zero replicas if the application cannot be paused.

```
[user@demo ~]$ oc scale deployment/my-application --replicas=0
```

Create a job that mounts the persistent volume and copies the data to a backup location.

```
---
apiVersion: batch/v1
kind: Job
metadata:
  name: backup
  namespace: application
```

```
    labels:
      app: backup
  spec:
    backoffLimit: 1
    template:
      metadata:
        labels:
          app: backup
      spec:
        containers:
        - name: backup
          image: registry.access.redhat.com/ubi8/ubi:8.4-209
          command: (1)
          - /bin/bash
          - -vc
          - 'dnf -qy install rsync && rsync -avH /var/application /opt/backup'
          resources: {}
          volumeMounts: (2)
          - name: application-data
            mountPath: /var/application
          - name: backup
            mountPath: /opt/backup
        volumes: (3)
        - name: application-data
          persistentVolumeClaim:
            claimName: pvc-application
        - name: backup
          persistentVolumeClaim:
            claimName: pvc-backup
        restartPolicy: Never
```

(1) Install and execute `rsync` to copy the files from the application data PVC to the backup PVC.

(2) Volume mount definition for the application data and backup PV in the container.

(3) Volume section in the job definition where the application data and backup PVC are referenced.

```
[user@demo ~]$ oc apply -f backup-job.yaml
```

Scale the application back up when the back up is complete to resume operations.

```
[user@demo ~]$ oc scale deployment/my-application --replicas=1
```

Backing up Application Data with a Specialized Tool

Enter into maintenance mode or make the application sync all the data to disk if the application has a specialized backup tool. You don't need to stop the application to create a SQL backup of a database.

```
[user@demo backup-application]$ oc exec -it deployment/mariadb -- /bin/bash
...output omitted...

1000660000@mariadb-6d677c975d-2npxd:~$ mysql -u ${MYSQL_USER}
 -p"${MYSQL_PASSWORD}" ${MYSQL_DATABASE}
...output omitted...

MariaDB [example]> FLUSH TABLES WITH READ LOCK ;
...output omitted...
```

Run a job to back up the database contents to a persistent volume.

```
---
apiVersion: batch/v1
kind: Job
metadata:
  name: mariadb-backup
  namespace: application
  labels:
    app: mariadb-backup
spec:
  backoffLimit: 1
  template:
    metadata:
      labels:
        app: mariadb-backup
    spec:
      containers:
      - name: mariadb-backup
        image: mariadb:10.5
        workingDir: /opt/backup
        command: (1)
        - /bin/bash
        - -vc
        - 'mysqldump -v -h "${MYSQL_HOST}" -P "${MYSQL_PORT}" -u "root" -
p"${MYSQL_ROOT_PASSWORD}" --databases "${MYSQL_DATABASE}" > backup.sql'
        resources: {}
        env: (2)
        - name: MYSQL_HOST
          value: mariadb.application
        - name: MYSQL_PORT
          value: "3306"
        envFrom: (3)
        - configMapRef:
            name: mariadb
        - secretRef:
            name: mariadb
        volumeMounts: (4)
        - name: backup
          mountPath: /opt/backup
      volumes: (5)
      - name: backup
```

```
        persistentVolumeClaim:
          claimName: pvc-backup
      restartPolicy: Never
```

1. Run `mysqldump` to back up the database contents to the backup PV.
2. Define the `mariadb` service endpoint.
3. Pass the database name and credentials as environment variables to the container.
4. Volume mount definition for the backup PV.
5. Volume section in the job definition where the backup PVC is referenced.

```
[user@demo ~]$ oc apply -f backup-job.yaml
```

References

For more information, refer to the product documentation for *Red Hat OpenShift Data Foundation* at
https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/

Guided Exercise

Backing up and Restoring Kubernetes Applications

In this exercise, you will back up and restore a database application.

Outcomes

You should be able to:

- Deploy a database application.
- Back up the data contained in the database.
- Deploy a new instance of the database.
- Restore the backup on the new instance.

Before You Begin

To perform this exercise, ensure that you have:

- A running OpenShift cluster with Red Hat OpenShift Data Foundation installed.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start backup-application
```

Instructions

1. Deploy the example application.

   1.1. Log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

   1.2. Change to the `~/DO370/labs/backup-application` directory.

```
[student@workstation ~]$ cd ~/DO370/labs/backup-application
```

   1.3. Create a new project for the example application.

```
[student@workstation backup-application]$ oc new-project backup-application
Now using project "backup-application" on server "https://
api.ocp4.example.com:6443".

...output omitted...
```

1.4. Deploy the application.

```
[student@workstation backup-application]$ oc apply -f mariadb.yaml
...output omitted...
```

1.5. Wait until the pod is running.

```
[student@workstation backup-application]$ watch oc get deployments,pods
NAME                     READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mariadb  1/1     1            1           39s

NAME                           READY   STATUS    RESTARTS   AGE
pod/mariadb-7df6ccc799-9jqv2   1/1     Running   0          39s
```

Press `Ctrl+C` to end the `watch` command when the pod is running and the ready status is `1/1`.

2. Verify that the data is present in the application.

2.1. Show the data contained in the application.

```
[student@workstation backup-application]$ oc exec -it deployment/mariadb -- \
  /bin/bash
...output omitted...

1000660000@mariadb-6d677c975d-2npxd:~$ mysql -u ${MYSQL_USER} \
  -p"${MYSQL_PASSWORD}" ${MYSQL_DATABASE}
...output omitted...

MariaDB [example]> SELECT * FROM olympics ;
+------+--------+---------------+------------------------+
| YEAR | SEASON | COUNTRY       | CITY                   |
+------+--------+---------------+------------------------+
...output omitted...
| 2010 | Winter | Canada        | Vancouver              |
| 2012 | Summer | United Kingdom | London                |
| 2014 | Winter | Russia        | Sochi                  |
| 2016 | Summer | Brazil        | Rio de Janeiro         |
| 2018 | Winter | South Korea   | Pyeongchang            |
+------+--------+---------------+------------------------+
51 rows in set (0.001 sec)

MariaDB [example]> Quit
1000680000@mariadb-7df6ccc799-9jqv2:~$ exit
```

3. Back up the application.

3.1. Create a PVC to store the backup.

```
[student@workstation backup-application]$ oc apply -f pvc-backup.yaml
persistentvolumeclaim/pvc-backup created
```

3.2. Prepare the database for backup.

The `/opt/prepare-backup.sql` file executes database specific commands that sync all the table information to disk in order to prepare it for backup.

```
[student@workstation backup-application]$ oc exec -it deployment/mariadb -- \
  /bin/bash
...output omitted...

1000680000@mariadb-c545457fb-jmn8p:~$ mysql -v -b -u root \
  -p"${MYSQL_ROOT_PASSWORD}" ${MYSQL_DATABASE} < /opt/prepare-backup.sql
--------------
FLUSH TABLES WITH READ LOCK
--------------
...output omitted...
```

3.3. Open another terminal window and run the backup job.

The `job-backup.yaml` contains a job that connects to the database and executes a command to back up the database to a file.

```
[student@workstation backup-application]$ oc apply -f job-backup.yaml
job.batch/mariadb-backup created
```

3.4. Check the job status and wait until it completes.

```
[student@workstation backup-application]$ oc get jobs,pods -l app=mariadb-backup
NAME                       COMPLETIONS   DURATION   AGE
job.batch/mariadb-backup   1/1           6s         10s

NAME                       READY   STATUS      RESTARTS   AGE
pod/mariadb-backup-4tj6z   0/1     Completed   0          10s
```

3.5. View the logs for the backup job.

```
[student@workstation backup-application]$ oc logs -f job/mariadb-backup

mysqldump -v -h "${MYSQL_HOST}" -P "${MYSQL_PORT}" -u "root" -
p"${MYSQL_ROOT_PASSWORD}" --databases "${MYSQL_DATABASE}" > backup.sql
-- Connecting to mariadb.backup-application...
...output omitted...
-- Disconnecting from mariadb.backup-application...
```

4. Restore the backup.

4.1. Delete the application and PVC.

```
[student@workstation backup-application]$ oc delete deployment/mariadb
deployment.apps "mariadb" deleted

[student@workstation backup-application]$ oc delete pvc/mariadb
persistentvolumeclaim "mariadb" deleted
```

4.2. Deploy another instance of the database and wait until it is ready.

```
[student@workstation backup-application]$ oc apply -f mariadb-replacement.yaml
...output omitted...

[student@workstation backup-application]$ watch oc get deployments,pods \
  -l app=mariadb-new
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mariadb-new    1/1     1            1           60s

NAME                               READY   STATUS    RESTARTS   AGE
pod/mariadb-new-5d6b64d69-svpwv    1/1     Running   0          60s
```

Press `Ctrl+C` to end the `watch` command when the pod is running and the ready status is `1/1`.

4.3. Run the job to restore the data from the backup file.

```
[student@workstation backup-application]$ oc apply -f job-restore.yaml
job.batch/mariadb-restore created
```

4.4. Check the job status and wait until it completes.

```
[student@workstation backup-application]$ oc get jobs,pods -l app=mariadb-restore
NAME                         COMPLETIONS   DURATION   AGE
job.batch/mariadb-restore    1/1           12s        32s

NAME                         READY   STATUS      RESTARTS   AGE
pod/mariadb-restore-584hz    0/1     Completed   0          32s
```

4.5. View the logs for the restore job.

```
[student@workstation backup-application]$ oc logs -f job/mariadb-restore

mysql -v -b -h "${MYSQL_HOST}" -P "${MYSQL_PORT}" -u "root" -
p"${MYSQL_ROOT_PASSWORD}" "${MYSQL_DATABASE}" < backup.sql

...output omitted...

--------------
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */
--------------
```

4.6. Verify that the data is restored in the new database instance.

```
[student@workstation backup-application]$ oc exec -it deployment/mariadb-new -- \
  /bin/bash
...output omitted...

1000680000@mariadb-new-5d6b64d69-svpwv:~$ mysql -u ${MYSQL_USER} \
  -p"${MYSQL_PASSWORD}" ${MYSQL_DATABASE}
...output omitted...

MariaDB [example]> SELECT * FROM olympics ;
+------+--------+---------------+------------------------+
| YEAR | SEASON | COUNTRY       | CITY                   |
+------+--------+---------------+------------------------+
...output omitted...
| 2010 | Winter | Canada        | Vancouver              |
| 2012 | Summer | United Kingdom | London                |
| 2014 | Winter | Russia        | Sochi                  |
| 2016 | Summer | Brazil        | Rio de Janeiro         |
| 2018 | Winter | South Korea   | Pyeongchang            |
+------+--------+---------------+------------------------+
51 rows in set (0.001 sec)

MariaDB [example]> Quit
1000680000@mariadb-new-5d6b64d69-svpwv:~$ exit
```

4.7. Change to the `/home/student` directory.

```
[student@workstation backup-application]$ cd
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish backup-application
```

This concludes the guided exercise.

Creating Volume Snapshots and Clones

Objectives

After completing this section, you should be able to create volume snapshots for use in restoration and/or duplication of storage volumes.

Container Storage Interface

The CSI defines a set of APIs that allow vendors to integrate storage solutions in a Kubernetes cluster. The CSI plug-ins are developed and distributed by vendors to add support for a storage back end, such as Ceph RBD for block devices and CephFS for file systems.

The CSI plug-ins are distributed as container images and they serve as the bridge between the Kubernetes cluster and the storage driver.

Persistent Volume Claims

You can create a volume snapshot or clone from any PVC that is based on the CSI storage classes.

```
[user@demo ~]$ oc get storageclasses | egrep '^NAME|csi'
NAME                          PROVISIONER                             RECLAIMPOLICY
ocs-storagecluster-ceph-rbd   openshift-storage.rbd.csi.ceph.com      Delete
ocs-storagecluster-cephfs     openshift-storage.cephfs.csi.ceph.com   Delete
```

The following PVC uses the `ocs-storagecluster-ceph-rbd` storage class to request a block PV.

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata: (1)
  name: postgresql-data
  namespace: backup-volume
spec:
  accessModes: (2)
  - ReadWriteOnce
  storageClassName: ocs-storagecluster-ceph-rbd (3)
  resources:
    requests:
      storage: 1Gi
```

(1) Name of the PVC and target namespace

(2) Requested access mode for the PV

(3) Storage class for the PVC

Volume Clones

You can create a clone of a PVC by referencing the source PVC in the `dataSource` parameter.

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata: (1)
  name: postgresql-data-clone
  namespace: backup-volume
  labels:
    app: postgresql-data-clone
spec:
  accessModes: (2)
    - ReadWriteOnce
  storageClassName: ocs-storagecluster-ceph-rbd (3)
  dataSource: (4)
    kind: PersistentVolumeClaim
    name: postgresql-data
  resources:
    requests:
      storage: 1Gi
```

(1) Name of the PVC and target namespace

(2) Requested access mode for the PV

(3) Storage class for the PVC

(4) Reference to the source PVC from which to take the clone

After the new PVC is created and bound, you can use it in a container.

```
---
apiVersion: apps/v1
kind: Deployment
metadata: (1)
  name: postgresql-clone
  namespace: backup-volume
  labels:
    app: postgresql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: postgresql
  strategy: {}
  template:
    metadata:
      labels:
        app: postgresql
    spec:
      containers:
      - name: postgres
        image: postgres:10
```

```
...
        volumeMounts: (2)
        - name: postgresql-data
          mountPath: /var/lib/postgresql
      volumes: (3)
      - name: postgresql-data
        persistentVolumeClaim:
          claimName: postgresql-data-clone
```

(1) Name of the deployment and target namespace

(2) Volume mount definition for the application data and backup of PVs in the container

(3) Volume section in the job definition where the application data and backup PVCs are referenced

Volume Snapshots

A volume snapshot is similar to a PVC, and the volume snapshot contents are similar to a PV.

The Ceph RBD and CephFS storage classes provided by Red Hat OpenShift Data Foundation have CSI drivers that provide the snapshot functionality for PVCs.

```
[user@demo ~]$ oc get storageclasses | egrep '^NAME|csi'
NAME                          PROVISIONER                             RECLAIMPOLICY
ocs-storagecluster-ceph-rbd   openshift-storage.rbd.csi.ceph.com      Delete
ocs-storagecluster-cephfs     openshift-storage.cephfs.csi.ceph.com   Delete

[user@demo ~]$ oc get volumesnapshotclasses
NAME                                      DRIVER
ocs-storagecluster-rbdplugin-snapclass    openshift-storage.rbd.csi.ceph.com
ocs-storagecluster-cephfsplugin-snapclass openshift-storage.cephfs.csi.ceph.com
```

You can create a snapshot of a PVC by referencing the source PVC in the `persistentVolumeClaimName` parameter and specifying the volume snapshot class for Ceph RBD or CephFS.

Note

The volume snapshot class driver must match the provisioner associated with the storage class of the source PVC.

```
---
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: postgresql-data (1)
spec:
  volumeSnapshotClassName: ocs-storagecluster-rbdplugin-snapclass (2)
  source:
    persistentVolumeClaimName: postgresql-data (3)
```

(1) Name of the volume snapshot. This resource does not belong to any namespace.

2. Storage class name for the volume snapshot.

3. Source PVC for the volume snapshot.

After the volume snapshot is created, a new PVC can be created. The volume snapshot must be referenced in the `dataSource` parameter of the new PVC.

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata: (1)
  name: postgresql-data-restore
  namespace: backup-volume
  labels:
    app: postgresql-data-snapshot
spec:
  accessModes: (2)
  - ReadWriteOnce
  storageClassName: ocs-storagecluster-ceph-rbd (3)
  dataSource: (4)
    apiGroup: snapshot.storage.k8s.io
    kind: VolumeSnapshot
    name: postgresql-data
  resources:
    requests:
      storage: 1Gi
```

1. Name of the PVC and target namespace

2. Requested access mode for the PV

3. Storage class for the PVC

4. Reference to the source volume snapshot

After the new PVC is created and bound, you can use it in a container.

```
---
apiVersion: apps/v1
kind: Deployment
metadata: (1)
  name: postgresql-snapshot
  namespace: backup-volume
  labels:
    app: postgresql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: postgresql
  strategy: {}
  template:
    metadata:
      labels:
        app: postgresql
```

```
    spec:
      containers:
      - name: postgres
        image: postgres:10
...
        volumeMounts: (2)
        - name: postgresql-data
          mountPath: /var/lib/postgresql
      volumes: (3)
      - name: postgresql-data
        persistentVolumeClaim:
          claimName: postgresql-data-restore
```

(1) Name of the deployment and target namespace

(2) Volume mount definition for the application data and backup PVs in the container

(3) Volume section in the job definition where the application data and backup PVCs are referenced

References

CSI volume cloning
https://docs.openshift.com/container-platform/4.7/storage/container_storage_interface/persistent-storage-csi-cloning.html

CSI volume snapshots
https://docs.openshift.com/container-platform/4.7/storage/container_storage_interface/persistent-storage-csi-snapshots.html

For more information about CSI volumes, refer to the *Using Container Storage Interface* chapter in the *Storage* documentation for *Red Hat OpenShift Container Platform* at
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.7/html-single/storage/index#using-container-storage-interface-csi

For more information about OpenShift volume snapshots and clones, refer to the *Volume Snapshots* and *Volume Cloning* chapters in the *Managing and Allocating Storage Resources* documentation for *Red Hat OpenShift Data Foundation* at
https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/html-single/managing_and_allocating_storage_resources/index#volume-snapshots_rhocs

Guided Exercise

Creating Volume Snapshots and Clones

In this exercise, you will perform backup and restore of a database application by using volume snapshots and clones.

Outcomes

You should be able to:

- Deploy an example application.
- Take a snapshot of the PV that is used by the example application.
- Restore the snapshot, and then deploy a new instance of the application.
- Create a clone of the PV that is used by the example application.
- Update the example application to use the volume clone.

Before You Begin

To perform this exercise, ensure that you have:

- A running OpenShift cluster with Red Hat OpenShift Data Foundation installed.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start backup-volume
```

Instructions

1. Deploy the example application.

   1.1. Log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

   1.2. Change to the `~/DO370/labs/backup-volume` directory.

```
[student@workstation ~]$ cd ~/DO370/labs/backup-volume
```

   1.3. Get the names of the volume snapshot classes currently present in the cluster.

```
[student@workstation ~]$ oc get volumesnapshotclasses
NAME                                      DRIVER
ocs-storagecluster-cephfsplugin-snapclass  openshift-storage.cephfs.csi...
ocs-storagecluster-rbdplugin-snapclass     openshift-storage.rbd.csi...
```

1.4. Change the deletion policy of the volume snapshot classes to preserve the snapshots if the source namespace is deleted.

```
[student@workstation backup-volume]$ oc patch \
  volumesnapshotclass/ocs-storagecluster-cephfsplugin-snapclass \
  --type merge -p '{"deletionPolicy":"Retain"}'
volumesnapshotclass.../ocs-storagecluster-cephfsplugin-snapclass  patched

[student@workstation backup-volume]$ oc patch \
  volumesnapshotclass/ocs-storagecluster-rbdplugin-snapclass \
  --type merge -p '{"deletionPolicy":"Retain"}'
volumesnapshotclass..../ocs-storagecluster-rbdplugin-snapclass  patched
```

1.5. Confirm that the deletion policy changed.

```
[student@workstation backup-volume]$ oc get volumesnapshotclasses
NAME                                DRIVER                      DELETIONPOLICY
ocs-storagecluster-cephfsplugin-...  openshift-storage.cephfs... Retain
ocs-storagecluster-rbdplugin-...     openshift-storage.rbd...    Retain
```

2. Deploy the example application.

2.1. Create a new project for the example application.

```
[student@workstation backup-volume]$ oc new-project backup-volume
Now using project "backup-volume" on server "https://api.ocp4.example.com:6443".

...output omitted...
```

2.2. Deploy the application.

```
[student@workstation backup-volume]$ oc apply -f postgresql.yaml
...output omitted...
```

2.3. Wait until the pod is running.

```
[student@workstation backup-volume]$ watch oc get deployments,pods
NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgresql   1/1     1            1           60s

NAME                              READY   STATUS    RESTARTS   AGE
pod/postgresql-5bd78c5649-7vg2p   1/1     Running   0          60s
```

Press `Ctrl+C` to end the `watch` command when the pod is running and the ready status is `1/1`.

2.4. Show the data contained in the application.

```
[student@workstation backup-volume]$ oc exec -it deployment/postgresql --
 /bin/bash
root@postgresql-f6cf7799c-2s8x2:~# psql -U ${POSTGRES_USER} -w -d ${POSTGRES_DB}
...output omitted...

example=# SELECT * FROM olympics ;
 year | season |    country     |          city
------+--------+----------------+------------------------
...output omitted...
 2010 | Winter | Canada         | Vancouver
 2012 | Summer | United Kingdom | London
 2014 | Winter | Russia         | Sochi
 2016 | Summer | Brazil         | Rio de Janeiro
 2018 | Winter | South Korea    | Pyeongchang
(51 rows)

example=# \q
root@postgresql-f6cf7799c-2s8x2:~# exit
```

3. Make a backup of the application.

3.1. Synchronize all database tables to a disk.

The `job-backup.yaml` file contains a job that connects to the database and executes the statements contained in the `/opt/prepare-backup.sql` file to prepare for the backup.

```
[student@workstation backup-volume]$ oc apply -f job-backup.yaml
job.batch/postgresql-backup created

[student@workstation backup-volume]$ oc logs -f job/postgresql-backup
psql -U ${POSTGRES_USER} -w -d ${POSTGRES_DB} < /opt/prepare-backup.sql
CHECKPOINT
 pg_start_backup
-----------------
 0/2000028
(1 row)

NOTICE:  pg_stop_backup complete, all required WAL segments have been archived
    lsn    |                          labelfile                           |
-----------+--------------------------------------------------------------+
 0/20000F8 | START WAL LOCATION: 0/2000028 (file 000000010000000000000002)+|
           | CHECKPOINT LOCATION: 0/2000060                               +|
           | BACKUP METHOD: streamed                                      +|
           | BACKUP FROM: master                                          +|
           | START TIME: 2021-07-16 21:13:46 CDT                          +|
           | LABEL: PostgreSQL_backup                                     +|
           |                                                               |
(1 row)
```

3.2. Scale the deployment to stop the activity on the PV.

```
[student@workstation backup-volume]$ oc scale deployment/postgresql --replicas 0
deployment.apps/postgresql scaled
```

4. Clone a PV.

4.1. Create a clone of the data volume.

```
[student@workstation backup-volume]$ oc apply -f pvc-clone.yaml
persistentvolumeclaim/postgresql-data-clone created

[student@workstation backup-volume]$ oc get pvc/postgresql-data-clone
NAME                   STATUS   VOLUME                                     ...
postgresql-data-clone  Bound    pvc-4958ef27-a919-462e-86eb-fea0e0532cc5   ...
```

4.2. Verify that the PV is bound to the claim shown in the previous step.

```
[student@workstation backup-volume]$ oc get pv/pvc-4958ef27-a919-462e-86eb-
fea0e0532cc5
NAME                              ...  STATUS   CLAIM
pvc-4958ef27-a919-462e-86eb-...   ...  Bound    backup-volume/postgresql-data-clone
```

4.3. Deploy another instance of the application that uses the PVC that you created from the volume clone.

```
[student@workstation backup-volume]$ oc apply -f postgresql-clone.yaml
deployment.apps/postgresql-clone created
service/postgresql-clone created
```

4.4. Review the data contained in the new instance of the application.

```
[student@workstation backup-volume]$ oc exec -it deployment/postgresql-clone --
 /bin/bash
root@postgresql-snapshot-5f44c79945-l2v9b:/# psql -U ${POSTGRES_USER} -w -d
 ${POSTGRES_DB}
...output omitted...

example=# SELECT * FROM olympics ;
 year | season |    country     |          city
------+--------+----------------+-----------------------
...output omitted...
 2010 | Winter | Canada         | Vancouver
 2012 | Summer | United Kingdom | London
 2014 | Winter | Russia         | Sochi
 2016 | Summer | Brazil         | Rio de Janeiro
 2018 | Winter | South Korea    | Pyeongchang
(51 rows)

example=# \q
root@postgresql-snapshot-5f44c79945-l2v9b:/# exit
```

5. Create a snapshot of a PV.

5.1. Create a volume snapshot of the data volume.

```
[student@workstation backup-volume]$ oc apply -f volumesnapshot-postgres.yaml
volumesnapshot.snapshot.storage.k8s.io/postgresql-data created
```

5.2. List the volume snapshots and get the volume snapshot content of the `postgresql-data` volume.

```
[student@workstation backup-volume]$ oc get volumesnapshots
NAME              ... SOURCEPVC       ... SNAPSHOTCONTENT
postgresql-data ... postgresql-data ... snapcontent-239c97b1-a619-4446-9d38-...
```

5.3. List the volume snapshot contents and look for the one associated with the `postgresql-data` volume snapshot.

```
[student@workstation backup-volume]$ oc get volumesnapshotcontents
NAME                                    ...  DELETIONPOLICY  ...  VOLUMESNAPSHOT
snapcontent-239c97b1-a619-4446-9d38-...  ...  Retain          ...  postgresql-data
```

5.4. Restore the volume snapshot content to a PVC with a different name.

```
[student@workstation backup-volume]$ oc apply -f pvc-restore-postgres.yaml
persistentvolumeclaim/postgresql-data-restore created

[student@workstation backup-volume]$ oc get pvc -l app=postgresql-data-snapshot
NAME                     STATUS   VOLUME                                     ...
postgresql-data-restore   Bound    pvc-dfc4f2b8-1caf-44f0-8311-07378988e982  ...
```

5.5. Get the PV associated with the new PVC that was just restored.

```
[student@workstation backup-volume]$ oc get
 pv/pvc-dfc4f2b8-1caf-44f0-8311-07378988e982
NAME                              ...  CLAIM                                ...
pvc-dfc4f2b8-1caf-44f0-8311-...   ...  backup-volume/postgresql-data-restore  ...
```

5.6. Deploy another instance of the application that uses the PVC you created from the volume snapshot.

```
[student@workstation backup-volume]$ oc apply -f postgresql-snapshot.yaml
deployment.apps/postgresql-snapshot created
service/postgresql-snapshot created
```

5.7. Review the data contained in the new instance of the application.

```
[student@workstation backup-volume]$ oc exec -it deployment/postgresql-snapshot --
 /bin/bash
root@postgresql-snapshot-5f44c79945-l2v9b:/# psql -U ${POSTGRES_USER} -w -d
 ${POSTGRES_DB}
...output omitted...

example=# SELECT * FROM olympics ;
```

```
 year | season |    country     |          city
------+--------+----------------+------------------------
...output omitted...
 2010 | Winter | Canada         | Vancouver
 2012 | Summer | United Kingdom | London
 2014 | Winter | Russia         | Sochi
 2016 | Summer | Brazil         | Rio de Janeiro
 2018 | Winter | South Korea    | Pyeongchang
(51 rows)

example=# \q
root@postgresql-snapshot-5f44c79945-l2v9b:/# exit
```

5.8. Change to the `/home/student` directory.

```
[student@workstation backup-volume]$ cd
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish backup-volume
```

This concludes the guided exercise.

Lab

Performing Backup and Restore of Kubernetes Block and File Volumes

In this lab, you will back up and restore a database application by using volume snapshots and clones.

Outcomes

You should be able to:

- Create a PVC to store the application data files.
- Deploy a database that uses a persistent volume to store its data.
- Create a PVC to store the SQL backup.
- Create a backup of the database and store it in a persistent volume.
- Create a clone of the persistent volume that stores the application data.
- Deploy a new instance of the database that uses the PVC.
- Create a snapshot of the persistent volume that is used by the database.
- Create a new PVC from the volume snapshot.
- Deploy a new instance of the database that uses the new persistent volume.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start backup-review
```

Instructions

1. Create the PVC for the application data.
   - Change to the `backup-review` project.
   - You can use the `~/DO370/labs/backup-review/pvc-data.yaml` file as a template.
2. Deploy the example application using the `~/DO370/labs/backup-review/postgresql.yaml` file.
   - The `postgresql-data` PVC is mounted in the `/var/lib/postgresql` directory.
3. Deploy a job to create a SQL backup of the database and store it in a PV.
   - You can use the `pvc-backup.yaml` and `job-backup.yaml` files in the `~/DO370/labs/backup-review` directory as templates.

- The `postgresql-backup` PVC is mounted in the `/mnt` directory.

4. Create a job to alter the structure of the database.

   - You can use the `~/DO370/labs/backup-review/job-alter.yaml` file as a template.

5. Create a clone of the persistent volume that is used to store the application data, and then deploy a new database instance that uses the new PVC.

   - You can use the `pvc-data-clone.yaml` and `postgresql-clone.yaml` files in the `~/DO370/labs/backup-review` directory as templates.

   - The `postgresql-data-clone` PVC is mounted in the `/var/lib/postgresql` directory.

6. Create a job to insert new rows in the database.

   - You can use the `~/DO370/labs/backup-review/job-insert.yaml` file as template.

7. Create a snapshot of the `postgresql-data-clone` PVC that you created in a previous step. Deploy another instance of the database that uses a new PVC created from this snapshot.

   - You can use the `volumesnapshot.yaml`, `pvc-data-snapshot.yaml`, and `postgresql-snapshot.yaml` files in the `~/DO370/labs/backup-review` directory as templates.

   - The `postgresql-data-snapshot` PVC is mounted in the `/var/lib/postgresql` directory.

8. Change to the `/home/student` directory.

```
[student@workstation backup-review]$ cd
```

Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade backup-review
```

Finish

As the `student` user on the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish backup-review
```

This concludes the lab.

Solution

Performing Backup and Restore of Kubernetes Block and File Volumes

In this lab, you will back up and restore a database application by using volume snapshots and clones.

Outcomes

You should be able to:

- Create a PVC to store the application data files.
- Deploy a database that uses a persistent volume to store its data.
- Create a PVC to store the SQL backup.
- Create a backup of the database and store it in a persistent volume.
- Create a clone of the persistent volume that stores the application data.
- Deploy a new instance of the database that uses the PVC.
- Create a snapshot of the persistent volume that is used by the database.
- Create a new PVC from the volume snapshot.
- Deploy a new instance of the database that uses the new persistent volume.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start backup-review
```

Instructions

1. Create the PVC for the application data.

   - Change to the `backup-review` project.
   - You can use the `~/DO370/labs/backup-review/pvc-data.yaml` file as a template.

   1.1. Log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

1.2. Change to the `backup-review` project.

```
[student@workstation ~]$ oc project backup-review
Now using project "backup-review" on server "https://api.ocp4.example.com:6443".
```

1.3. Change to the `~/DO370/labs/backup-review` directory.

```
[student@workstation ~]$ cd ~/DO370/labs/backup-review
```

1.4. Edit the `pvc-data.yaml` file and modify the placeholder text to match the following content.

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: postgresql-data
  namespace: backup-review
spec:
  accessModes:
  - ReadWriteOnce
  storageClassName: ocs-storagecluster-ceph-rbd
  resources:
    requests:
      storage: 1Gi
```

1.5. Create the PVC.

```
[student@workstation backup-review]$ oc apply -f pvc-data.yaml
```

2. Deploy the example application using the `~/DO370/labs/backup-review/postgresql.yaml` file.

- The `postgresql-data` PVC is mounted in the `/var/lib/postgresql` directory.

2.1. Edit the `postgresql.yaml` file and modify the placeholder text to match the following content.

```
        volumeMounts:
        - name: postgresql-data
          mountPath: /var/lib/postgresql
        - name: init-data
          mountPath: /docker-entrypoint-initdb.d
      volumes:
      - name: postgresql-data
        persistentVolumeClaim:
          claimName: postgresql-data
      - name: init-data
        configMap:
          name: init-data
          defaultMode: 0555
```

2.2. Deploy the example application.

```
[student@workstation backup-review]$ oc apply -f postgresql.yaml
```

2.3. Wait until the pod is running.

```
[student@workstation backup-review]$ watch oc get deployments,pods -l
 app=postgresql
NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgresql   1/1     1            1           61s

NAME                              READY   STATUS    RESTARTS   AGE
pod/postgresql-7dd654f6f-p6hrh    1/1     Running   0          60s
```

Press `Ctrl+C` to end the `watch` command when the pod is running and the ready status is `1/1`.

3. Deploy a job to create a SQL backup of the database and store it in a PV.

- You can use the `pvc-backup.yaml` and `job-backup.yaml` files in the `~/DO370/labs/backup-review` directory as templates.
- The `postgresql-backup` PVC is mounted in the `/mnt` directory.

3.1. Edit the `pvc-backup.yaml` file and modify the placeholder text to match the following content.

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: postgresql-backup
  namespace: backup-review
spec:
  accessModes:
  - ReadWriteMany
  storageClassName: ocs-storagecluster-cephfs
  resources:
    requests:
      storage: 1Gi
```

3.2. Create the PVC that is used to store the backups.

```
[student@workstation backup-review]$ oc apply -f pvc-backup.yaml
```

3.3. Edit the `job-backup.yaml` file and modify the placeholder text to match the following content.

- The `postgresql-backup` PVC is mounted in the `/mnt` directory.

```
        volumeMounts:
        - name: postgresql-backup
          mountPath: /mnt
        - name: postgresql-data
```

```
          mountPath: /var/lib/postgresql
      volumes:
      - name: postgresql-backup
        persistentVolumeClaim:
          claimName: postgresql-backup
      - name: postgresql-data
        emptyDir: {}
```

3.4. Create the job and wait until it completes.

```
[student@workstation backup-review]$ oc apply -f job-backup.yaml

[student@workstation backup-review]$ oc get job/postgresql-backup
NAME                COMPLETIONS   DURATION   AGE
postgresql-backup   1/1           10s        30s
```

3.5. View the logs of the backup job.

```
[student@workstation backup-review]$ oc logs -f job/postgresql-backup
pg_dump -h ${PGHOST} -U ${POSTGRES_USER} -d ${POSTGRES_DB} > /mnt/database.sql ;
 ls -l /mnt/database.sql
-rw-r--r--. 1 root root 9029 Aug 25 20:57 /mnt/database.sql
```

4. Create a job to alter the structure of the database.

- You can use the `~/DO370/labs/backup-review/job-alter.yaml` file as a template.

4.1. Edit the `job-alter.yaml` file and ensure the `PGHOST` environment variable is set to `postgresql` to connect to the proper service.

4.2. Create the job and wait until it completes.

```
[student@workstation backup-review]$ oc apply -f job-alter.yaml

[student@workstation backup-review]$ oc get job/postgresql-alter
NAME              COMPLETIONS   DURATION   AGE
postgresql-alter  1/1           5s         30s
```

4.3. View the job logs.

```
[student@workstation backup-review]$ oc logs -f job/postgresql-alter
psql --echo-queries -h ${PGHOST} -U ${POSTGRES_USER} -d ${POSTGRES_DB} < /opt/
alter.sql
...output omitted...
SELECT * FROM latest ;
year | season |    country     |      city      |              notes
-----+--------+----------------+----------------+--------------------------------
2012 | Summer | United Kingdom | London         |
2014 | Winter | Russia         | Sochi          |
2016 | Summer | Brazil         | Rio de Janeiro |
2018 | Winter | South Korea    | Pyeongchang    |
2020 | Summer | Japan          | Tokyo          | Postponed to the following year
(5 rows)
```

5. Create a clone of the persistent volume that is used to store the application data, and then deploy a new database instance that uses the new PVC.

- You can use the `pvc-data-clone.yaml` and `postgresql-clone.yaml` files in the `~/DO370/labs/backup-review` directory as templates.
- The `postgresql-data-clone` PVC is mounted in the `/var/lib/postgresql` directory.

5.1. Edit the `pvc-data-clone.yaml` file and modify the placeholder text to match the following content.

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: postgresql-data-clone
  namespace: backup-review
  labels:
    app: postgresql-data-clone
spec:
  storageClassName: ocs-storagecluster-ceph-rbd
  accessModes:
    - ReadWriteOnce
  dataSource:
    kind: PersistentVolumeClaim
    name: postgresql-data
  resources:
    requests:
      storage: 1Gi
```

5.2. Create the PVC clone.

```
[student@workstation backup-review]$ oc apply -f pvc-data-clone.yaml
```

5.3. Edit the `postgresql-clone.yaml` file and modify the placeholder text to match the following content.

- The `postgresql-data-clone` PVC is mounted in the `/var/lib/postgresql` directory.

```
        volumeMounts:
        - name: postgresql-data-clone
          mountPath: /var/lib/postgresql
      volumes:
      - name: postgresql-data-clone
        persistentVolumeClaim:
          claimName: postgresql-data-clone
```

5.4. Deploy the new instance of the database that uses the PVC clone.

```
[student@workstation backup-review]$ oc apply -f postgresql-clone.yaml
```

5.5. Wait until the pod is running.

```
[student@workstation backup-review]$ watch oc get deployments,pods -l
 app=postgresql-clone
NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgresql-clone    1/1     1            1           51s

NAME                                    READY   STATUS    RESTARTS   AGE
pod/postgresql-clone-b988bbbcb-k92xv    1/1     Running   0          50s
```

Press `Ctrl+C` to end the `watch` command when the pod is running and the ready status is `1/1`.

6. Create a job to insert new rows in the database.

- You can use the `~/DO370/labs/backup-review/job-insert.yaml` file as template.

6.1. Edit the `job-insert.yaml` file and ensure that the `PGHOST` environment variable is set to `postgresql-clone` to connect to the proper service.

6.2. Create the job and wait until it completes.

```
[student@workstation backup-review]$ oc apply -f job-insert.yaml

[student@workstation backup-review]$ oc get job/postgresql-insert
NAME               COMPLETIONS   DURATION   AGE
postgresql-insert  1/1           5s         30s
```

6.3. View the logs of the job.

```
[student@workstation backup-review]$ oc logs -f job/postgresql-insert
psql --echo-queries -h ${PGHOST} -U ${POSTGRES_USER} -d ${POSTGRES_DB} < /opt/
insert.sql
...output omitted...
SELECT * FROM latest ;
year | season |    country     |      city      |              notes
-----+--------+----------------+----------------+--------------------------------
2016 | Summer | Brazil         | Rio de Janeiro |
2018 | Winter | South Korea    | Pyeongchang    |
2020 | Summer | Japan          | Tokyo          | Postponed to the following year
2022 | Winter | China          | Beijing        | Future edition
2024 | Summer | France         | Paris          | Future edition

(5 rows)
```

7. Create a snapshot of the `postgresql-data-clone` PVC that you created in a previous step. Deploy another instance of the database that uses a new PVC created from this snapshot.

- You can use the `volumesnapshot.yaml`, `pvc-data-snapshot.yaml`, and `postgresql-snapshot.yaml` files in the `~/DO370/labs/backup-review` directory as templates.
- The `postgresql-data-snapshot` PVC is mounted in the `/var/lib/postgresql` directory.

7.1. Edit the `volumesnapshot.yaml` file and modify the placeholder text to match the following content.

```
---
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: postgresql-data-snapshot
  namespace: backup-review
spec:
  volumeSnapshotClassName: ocs-storagecluster-rbdplugin-snapclass
  source:
    persistentVolumeClaimName: postgresql-data-clone
```

7.2. Create the volume snapshot.

```
[student@workstation backup-review]$ oc apply -f volumesnapshot.yaml
```

7.3. Edit the `pvc-data-snapshot.yaml` file and modify the placeholder text to match the following content.

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: postgresql-data-snapshot
  namespace: backup-review
  labels:
    app: postgresql-data-snapshot
spec:
  storageClassName: ocs-storagecluster-ceph-rbd
  accessModes:
  - ReadWriteOnce
  dataSource:
    apiGroup: snapshot.storage.k8s.io
    kind: VolumeSnapshot
    name: postgresql-data-snapshot
  resources:
    requests:
      storage: 1Gi
```

7.4. Create the new PVC using the volume snapshot as the data source.

```
[student@workstation backup-review]$ oc apply -f pvc-data-snapshot.yaml
```

7.5. Edit the `postgresql-snapshot.yaml` file and modify the placeholder text to match the following content.

- The `postgresql-data-snapshot` PVC is mounted in the `/var/lib/postgresql` directory.

```
        volumeMounts:
        - name: postgresql-data-snapshot
          mountPath: /var/lib/postgresql
      volumes:
      - name: postgresql-data-snapshot
        persistentVolumeClaim:
          claimName: postgresql-data-snapshot
```

7.6. Deploy a new instance of the database that uses the PVC created from the volume snapshot.

```
[student@workstation backup-review]$ oc apply -f postgresql-snapshot.yaml
```

7.7. Wait until the pod is running.

```
[student@workstation backup-review]$ oc get deployments,pods -l
 app=postgresql-snapshot
NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgresql-snapshot   1/1     1            1           40s

NAME                                      READY   STATUS    RESTARTS   AGE
pod/postgresql-snapshot-84fbf657fb-ghjtl   1/1     Running   0          41s
```

Press `Ctrl+C` to end the `watch` command when the pod is running and the ready status is `1/1`.

8. Change to the `/home/student` directory.

```
[student@workstation backup-review]$ cd
```

Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade backup-review
```

Finish

As the `student` user on the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish backup-review
```

This concludes the lab.

Summary

In this chapter, you learned about:

- Backing up stateless applications.
- Backing up stateful applications.
- Creating clones of persistent volumes.
- Creating snapshots of persistent volumes.

Chapter 6

Configuring Applications to Use OpenShift Data Foundation Object Storage

Goal

Configure applications to use object storage from OpenShift Data Foundation.

Objectives

- Identify the use cases for object storage and benefits provided by Red Hat OpenShift Data Foundation.
- Create object bucket claims and access the object buckets with S3 clients.
- Configure an application to use S3 compatible object storage provided by Red Hat OpenShift Data Foundation.
- Identify the storage metrics associated with the object buckets.

Sections

- Introducing S3 Object Storage (and Guided Exercise)
- Creating Object Bucket Claims and Accessing Object Storage (and Guided Exercise)
- Configuring Applications to Use Red Hat OpenShift Data Foundation Object Storage (and Guided Exercise)
- Monitoring Red Hat OpenShift Data Foundation Object Buckets (and Guided Exercise)

Lab

Configuring Applications to Use Red Hat OpenShift Data Foundation Object Storage

Introducing S3 Object Storage

Objectives

After completing this section, you should be able to identify the use cases for object storage and benefits provided by Red Hat OpenShift Data Foundation.

Object Storage

Red Hat OpenShift Data Foundation provides object storage for applications to consume. This type of storage has three parts:

Objects
Objects are the storage unit; they have a unique identifier and associated metadata.

Buckets
Buckets contain a set of objects and the policies associated with them.

S3 API server
The S3 server authenticates applications and stores and retrieves objects from the buckets by using API calls.

Object storage is independent of the underlying backing storage, making it effective for cloud-native applications. Organize the objects by adding a prefix to the name that allows you to list, retrieve, or delete a limited set of objects.

```
s3://my-bucket/image.jpg (1)
s3://my-bucket/media/video.mkv (2)
s3://my-bucket/archive/logs/webserver/access_log (3)
```

(1) The object is stored in the root of the bucket.

(2) The prefix for the object is `media`.

(3) The prefix for the object is `archive/logs/webserver`.

Use Cases for Object Storage

Applications can leverage object storage to read or write large amounts of unstructured data, such as text, audio, images, video, or binary content. A directory containing log files or backups can be uploaded to object storage for archiving. The objects stored do not depend on a file system and can be accessed by many applications. The stored content can be exposed in a static website or accessed via the S3 API.

Object storage is frequently used to store:

- Large data sets
- Unstructured data, such as images, audio, or video
- Log files
- Backups for archiving

S3 API

The applications use a client library to access buckets and objects with the S3 API by using the HTTP or HTTPS protocol. The client libraries authenticate with the S3 API by using a set of access keys.

Access key
This is the identifier of the key.

Secret access key
This is the secret part of the key.

Red Hat recommends that you protect both access keys as if they are a password because any application using them will have access to the S3 bucket.

The S3 API implements the following operations for the objects stored in buckets: list, upload, download, rename, and delete. This means that you avoid random file access or locking; the whole object is uploaded or downloaded from the server when accessed.

Large objects can be uploaded in chunks by using a multipart upload that is handled by the S3 client library. However, the whole object is still uploaded to the bucket.

Comparing Object Storage to Other Storage Types

The following table describes the differences between storage types.

Comparison between block, file, and object storage

Type	Block	File	Object
Data access	Block device	File system	S3 API
Random access	Yes	Yes	No
Access modes	`RWO`	`RWX`	`RWX`
Hierarchical	No	Yes	Flat
Storage unit	Block	File	Object
Use cases	• Databases • Virtual machine disks	• Application data	• Unstructured data • Large data sets • Archive storage

Object Storage Provided by Red Hat OpenShift Data Foundation

OpenShift Data Foundation provides object storage using two operators:

- Ceph RADOS Object Gateway (Ceph RGW)
- Multicloud Object Gateway (NooBaa MCG)

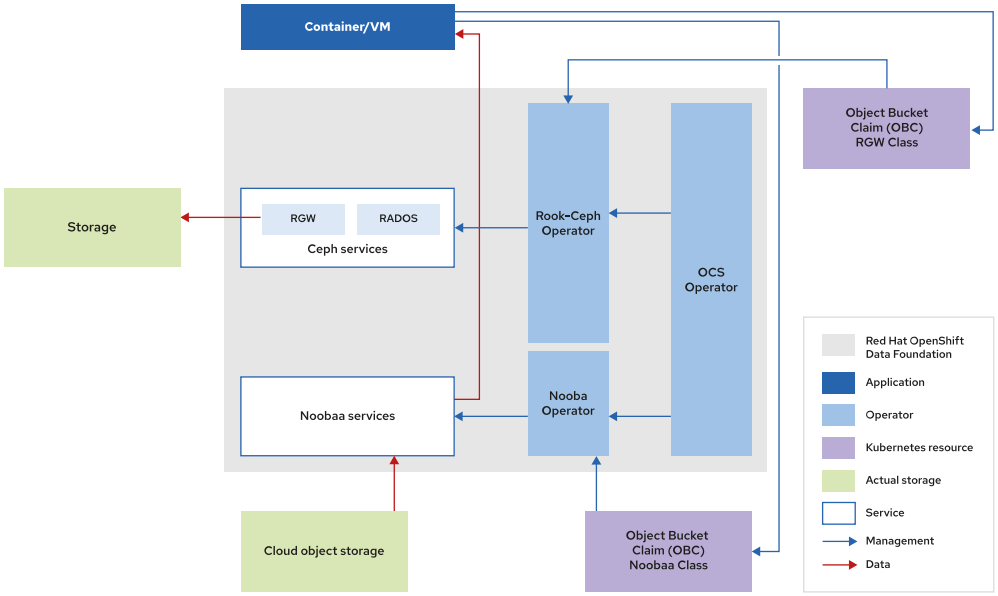


Figure 6.1: Red Hat OpenShift Data Foundation object storage providers

RADOS Object Gateway

Ceph RGW is an object storage interface built on top of the `librados` library to provide applications with a RESTful gateway to Ceph storage clusters. Ceph RGW provides an S3 interface for the object storage service that is backed by a Ceph storage cluster. The storage class for the Ceph RGW interface is `ocs-storagecluster-ceph-rgw`.

Multicloud Object Gateway

The NooBaa MCG gateway is a lightweight object storage service for OpenShift that can utilize native cloud storage on any AWS S3 compatible storage from a public or private cloud. The NooBaa MCG abstracts the specific parameters for the backing S3 storage and provides a unique endpoint to connect the applications and consume object storage. The storage class for the NooBa MCG gateway is `openshift-storage.noobaa.io`.

Deployment Modes for Ceph RGW and NooBaa MCG

You can configure the backing storage for Ceph RGW or NooBaa MCG in different ways depending on the type of infrastructure on which OpenShift Data Foundation is installed.

- You can use the Ceph RGW or NooBaa MCG object storage when deploying on bare metal infrastructure or in a hypervisor; both of them will interact with the storage provided by the Rook-Ceph operator.

- If deploying on a cloud provider, then you can configure NooBaa MCG to function as a gateway for one or more cloud object storage providers and to install Ceph RGW if needed.

- There is also a standalone deployment mode for NooBaa that uses local block or file storage. This deployment mode is only supported when using Quay and MTC.

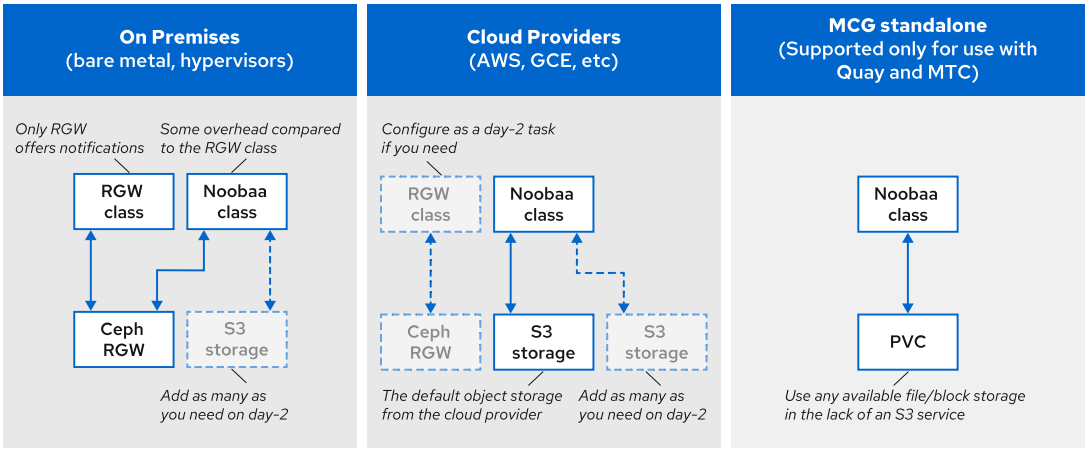


Figure 6.2: Red Hat OpenShift Data Foundation deployment modes and object storage providers

References

Red Hat, comparison between file, block, and object storage
https://www.redhat.com/en/topics/data-storage/file-block-object-storage

For more information, refer to the product documentation for the Multicloud Object Gateway and RADOS Object Gateway in the *Red Hat OpenShift Data Foundation* at https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/html-single/managing_hybrid_and_multicloud_resources/index

Guided Exercise

Introducing S3 Object Storage

In this exercise you will access files in an S3 bucket using a command line client.

Outcomes

You should be able to:

- Use a S3 client with a preconfigured object bucket.
- Create, move, retrieve, and delete objects from the bucket.
- Create and delete prefixes (folders) from the bucket.

Before You Begin

To perform this exercise, ensure that you have:

- A running OpenShift cluster with Red Hat OpenShift Data Foundation installed.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start object-define
```

Instructions

1. Inspect the environment.

   1.1. Log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

   1.2. Review the resources in the `object-define` project.

```
[student@workstation ~]$ watch oc get deployments,pods -n object-define
NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/s3-cli  1/1     1            1           31s

NAME                           READY   STATUS    RESTARTS   AGE
pod/s3-cli-7bff484c76-lsblk    1/1     Running   0          31s
```

   Press `Ctrl+C` to end the `watch` command when the pod is running and the ready status is `1/1`.

   1.3. Enter the preconfigured deployment to access the S3 bucket.

```
[student@workstation ~]$ oc exec -it deployment/s3-cli -n object-define -- \
  /bin/bash
(app-root) bash-4.4$
```

1.4. List all the available S3 buckets.

```
(app-root) bash-4.4$ aws s3 ls s3://
2021-06-16 23:39:30 object-bucket-8e722bfb-f687-4fa3-820a-bd1ffff08084
```

1.5. List the objects stored in the S3 bucket. The name of the bucket is stored in the BUCKET_NAME environment variable.

```
(app-root) bash-4.4$ printenv BUCKET_NAME
object-bucket-8e722bfb-f687-4fa3-820a-bd1ffff08084

(app-root) bash-4.4$ aws s3 ls s3://${BUCKET_NAME} --summarize

Total Objects: 0
   Total Size: 0
```

2. Upload and access objects in the S3 bucket.

2.1. Create a sample file and copy it to the S3 bucket.

```
(app-root) bash-4.4$ echo "Hello world" > sample-file.txt

(app-root) bash-4.4$ aws s3 cp sample-file.txt s3://${BUCKET_NAME}/
upload: ./sample-file.txt to s3://object-bucket-.../sample-file.txt
```

2.2. Retrieve the file from the S3 bucket and view its contents.

```
(app-root) bash-4.4$ aws s3 cp s3://${BUCKET_NAME}/sample-file.txt /tmp/
download: s3://object-bucket-.../sample-file.txt to ../../tmp/sample-file.txt

(app-root) bash-4.4$ cat /tmp/sample-file.txt
Hello world
```

2.3. Update the sample file and upload it again to the S3 bucket.

Note
When you copy files to the S3 bucket you are uploading the entire file.

```
(app-root) bash-4.4$ echo "updated content" > sample-file.txt

(app-root) bash-4.4$ aws s3 cp sample-file.txt s3://${BUCKET_NAME}/sample-file.txt
upload: ./sample-file.txt to s3://object-bucket-.../sample-file.txt
```

2.4. Duplicate an object in the S3 bucket.

```
(app-root) bash-4.4$ aws s3 cp s3://${BUCKET_NAME}/sample-file.txt
 s3://${BUCKET_NAME}/second-file.txt

(app-root) bash-4.4$ aws s3 cp s3://${BUCKET_NAME}/second-file.txt /tmp/
download: s3://object-bucket-.../second-file.txt to ../../tmp/second-file.txt

(app-root) bash-4.4$ cat /tmp/second-file.txt
updated content
```

2.5. List all the stored objects in the S3 bucket.

```
(app-root) bash-4.4$ aws s3 ls s3://${BUCKET_NAME} --summarize
2021-06-16 23:47:55         16 sample-file.txt
2021-06-16 23:48:04         16 second-file.txt

Total Objects: 2
   Total Size: 30
```

3. Create prefixes in the S3 bucket.

3.1. Create a prefix that allows objects to be sorted hierarchically.

```
(app-root) bash-4.4$ aws s3api put-object --bucket ${BUCKET_NAME} --key prefix
{
    "ETag": "\"d41d8cd98f00b204e9800998ecf8427e\""
}
```

3.2. Change the prefix name.

```
(app-root) bash-4.4$ aws s3 mv s3://${BUCKET_NAME}/prefix
 s3://${BUCKET_NAME}/my-s3-prefix
move: s3://object-bucket-.../prefix to s3://object-bucket-.../my-s3-prefix

(app-root) bash-4.4$ aws s3 ls s3://${BUCKET_NAME}
2021-06-16 23:47:55         16 sample-file.txt
2021-06-16 23:48:04         16 second-file.txt
2021-06-17 00:06:31          0 my-s3-prefix
```

3.3. Rename an object stored in the S3 bucket.

```
(app-root) bash-4.4$ aws s3 mv s3://${BUCKET_NAME}/second-file.txt
 s3://${BUCKET_NAME}/other-file.txt
move: s3://object-bucket-.../second-file.txt to s3://object-bucket-.../other-
file.txt
```

3.4. Move an object so that it is stored inside of a given prefix.

```
(app-root) bash-4.4$ aws s3 mv s3://${BUCKET_NAME}/sample-file.txt
 s3://${BUCKET_NAME}/my-s3-prefix/
move: s3://object-bucket-.../sample-file.txt to s3://object-bucket-.../my-s3-
prefix/sample-file.txt
```

3.5. List all the stored objects in the S3 bucket.

```
(app-root) bash-4.4$ aws s3 ls s3://${BUCKET_NAME} --recursive --summarize
2021-06-16 23:49:13         16 other-file.txt
2021-06-16 23:48:31          0 my-s3-prefix
2021-06-16 23:49:34         16 my-s3-prefix/sample-file.txt

Total Objects: 3
   Total Size: 30
```

4. Copy all the files in a local directory to the S3 bucket.

4.1. Create a another prefix inside the S3 bucket.

```
(app-root) bash-4.4$ aws s3api put-object --bucket ${BUCKET_NAME} --key icons
...output omitted...
```

4.2. Synchronize a local directory to the S3 bucket.

```
(app-root) bash-4.4$ aws s3 sync /usr/share/httpd/icons/
 s3://${BUCKET_NAME}/icons/
upload: ../../usr/share/httpd/icons/README to s3://object-bucket-.../icons/README
...output omitted...
```

4.3. List all the objects stored in the S3 bucket recursively.

```
(app-root) bash-4.4$ aws s3 ls s3://${BUCKET_NAME} --recursive --human-readable
 --summarize
2021-06-16 23:51:02    0 Bytes icons
2021-06-16 23:51:11    5.0 KiB icons/README
...output omitted...
2021-06-16 23:49:13   16 Bytes other-file.txt
2021-06-16 23:48:31    0 Bytes my-s3-prefix
2021-06-16 23:49:34   16 Bytes my-s3-prefix/sample-file.txt

Total Objects: 244
   Total Size: 424.3 KiB
```

5. Remove objects stored in the S3 bucket.

5.1. Delete a single object from the S3 bucket.

```
(app-root) bash-4.4$ aws s3 rm s3://${BUCKET_NAME}/other-file.txt
delete: s3://object-bucket-.../other-file.txt
```

5.2. Delete all objects under a given prefix in the S3 bucket.

```
(app-root) bash-4.4$ aws s3 rm s3://${BUCKET_NAME}/icons/small/ --recursive
delete: s3://object-bucket-.../icons/small/back.png
...output omitted...
```

5.3. Exit the preconfigured deployment.

```
(app-root) bash-4.4$ exit
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish object-define
```

This concludes the guided exercise.

Creating Object Bucket Claims and Accessing Object Storage

Objectives

After completing this section, you should be able to create object bucket claims and access the object buckets with S3 clients.

Object Bucket Claims

An object bucket claim (OBC) is similar to a persistent volume claim (PVC). You use a PVC to request a file system or block storage for your application and you use an OBC to request an S3 compatible bucket.

An OBC creates a new S3 bucket, a set of associated resources, and an application account with read/write permissions. The application account uses an access key and secret access key to authenticate with the S3 API when accessing the bucket.

Red Hat OpenShift Data Foundation provides object storage using two operators. You can create an OBC associated with the desired provider by specifying the appropriate storage class.

- Ceph RADOS Object Gateway
- Multicloud Object Gateway

Feature comparison between Ceph RGW and NooBaa MCG object storage

Type	Ceph RGW	NooBaa MCG
Storage Class	`ocs-storagecluster-ceph-rgw`	`openshift-storage.noobaa.io`
Protocol	http	https
Internal S3 endpoint	`rook-ceph-rgw-ocs-storagecluster-cephobjectstore.openshift-storage.svc`	`s3.openshift-storage.svc`
External S3 `route`	`route/ocs-storagecluster-cephobjectstore`	`route/s3`
Port	80	443
Region	us-east-1	(none)
Backing storage	Ceph cluster	• Ceph cluster • S3 storage

Note

- In previous versions of OpenShift Data Foundation, you had to manually expose the `rook-ceph-rgw-ocs-storagecluster-cephobjectstore` service to get a external S3 endpoint.
- You cannot change the storage class of an OBC after it has been created.

S3 Service Endpoints

The endpoint that the application uses to connect to the S3 API depends on the storage class used to create the OBC.

RADOS Object Gateway

The S3 service provided by the Ceph RGW gateway uses the `http` protocol on port 80. The internal endpoint is the name of the `rook-ceph-rgw-ocs-storagecluster-cephobjectstore` service in the `openshift-storage` namespace.

`rook-ceph-rgw-ocs-storagecluster-cephobjectstore.openshift-storage.svc`

OpenShift Data Foundation provides an external endpoint to connect to the S3 service from outside the OpenShift cluster.

```
[user@demo ~]$ oc get route/ocs-storagecluster-cephobjectstore \
  -n openshift-storage -o jsonpath='{.spec.host}{"\n"}'
ocs-storagecluster-cephobjectstore-openshift-storage.apps.ocp4.example.com
```

Note

In previous versions of OpenShift Data Foundation you had to manually expose the `rook-ceph-rgw-ocs-storagecluster-cephobjectstore` service to get an external S3 endpoint.

NooBaa MCG

The S3 service provided by NooBaa MCG uses the `https` protocol on port 443. The internal endpoint is the name of the `s3` service in the `openshift-storage` namespace.

`s3.openshift-storage.svc`

The external endpoint is provided by the S3 route in the `openshift-storage` namespace.

```
[user@demo ~]$ oc get route/s3 -n openshift-storage \
  -o jsonpath='{.spec.host}{"\n"}'
s3-openshift-storage.apps.ocp4.example.com
```

Note

NooBaa MCG has an independent management console that only works with Chromium browsers. The usage of this web console is beyond the scope of this course.

```
[user@demo ~]$ oc get route/noobaa-mgmt -n openshift-storage \
  -o jsonpath='{.spec.host}{"\n"}'
noobaa-mgmt-openshift-storage.apps.ocp4.example.com
```

Object Bucket Claim Syntax

The OBC resources belong to a specific namespace and have a set of parameters.

```
---
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: my-object-bucket-claim (1)
  namespace: default (2)
spec:
  storageClassName: openshift-storage.noobaa.io (3)
  generateBucketName: my-object-bucket-claim (4)
```

(1) Name of the OBC.

(2) Target namespace where the resource is created.

(3) Storage class name for the object bucket.

- To use NooBaa MCG, set the value to `openshift-storage.noobaa.io`.
- To use the Ceph RGW gateway, set the value to `ocs-storagecluster-ceph-rgw`.

(4) Base name of the object bucket to be generated. The S3 buckets do not belong to a namespace and the name must be unique.

Note

You cannot change the storage class of an OBC after it has been created.

You can create the OBC to request an S3 bucket from the OpenShift cluster.

```
[user@demo ~]$ oc apply -f my-object-bucket-claim.yaml
objectbucketclaim.objectbucket.io/my-object-bucket-claim created
```

The status of the new OBC is set to **bound**, similar to a PVC.

```
[user@demo ~]$ oc get objectbucketclaims
NAME                    STORAGE-CLASS                 PHASE   AGE
my-object-bucket-claim  openshift-storage.noobaa.io   Bound   60s
```

Resources Associated with an Object Bucket Claim

When you create an OBC resource, the API server creates the associated resources.

Object Bucket
Represents the S3 bucket that the controller created.

Configuration Map
Contains the S3 endpoint and the name of the S3 bucket.

Secret
Contains the access keys used to authenticate with the S3 API.

You can use the configuration map and secret to inject the settings in your application.

Object Bucket

After creating the OBC, the associated object bucket resource status is set to **bound** to indicate that the object bucket was created. The object bucket resources do not belong to a specific namespace.

```
[user@demo ~]$ oc get objectbuckets
NAME                                 STORAGE-CLASS                 ...   PHASE
obc-default-my-object-bucket-claim   openshift-storage.noobaa.io   ...   Bound
```

Configuration Map

The configuration map is created in the same namespace of the OBC and contains the S3 endpoint details and the name of the bucket.

The following ConfigMap correspond to an object bucket claim created with the NooBaa MCG storage class.

```
[user@demo ~]$ oc extract configmap/my-object-bucket-claim --to=-
# BUCKET_HOST
s3.openshift-storage.svc (1)
# BUCKET_PORT
443 (2)
# BUCKET_NAME
my-object-bucket-claim-27f4ce2f-13ca-44cc-b8a5-1a747989cfb3 (3)
# BUCKET_REGION
 (4)
# BUCKET_SUBREGION
```

(1) If you use the Ceph RGW storage class, then the host name used to access the S3 API is set to `rook-ceph-rgw-ocs-storagecluster-cephobjectstore.openshift-storage.svc`.

(2) If you use the Ceph RGW storage class, then the port is set to `80` and protocol is `https`.

(3) This is the name of the S3 bucket. Notice the unique identifier because all buckets are cluster wide resources.

(4) The default region is not set when using the NooBaa MCG storage class and is set to `us-east-1` when using the Ceph RGW storage class.

Secret

The secret is created in the same namespace of the OBC and contains the credentials to authenticate with the S3 API to access the bucket. Red Hat recommends that you protect both access keys as if they were a password because any applications that use them have access to the S3 bucket.

```
[user@demo ~]$ oc extract secret/my-object-bucket-claim --to=-
# AWS_ACCESS_KEY_ID
YEAsbMJnG3o1bGANZprt
# AWS_SECRET_ACCESS_KEY
xjaeCDhskn7lfrdA7WqzoUxpiRYuyjc9uDaWlMw3
```

Accessing Object Buckets with an S3 Client

Export the environment variables from the configuration map and secret associated with the OBC.

- `BUCKET_HOST`
- `BUCKET_PORT`
- `BUCKET_NAME`
- `AWS_ACCESS_KEY_ID`
- `AWS_SECRET_ACCESS_KEY`

Note

The following extra steps are needed when you use a certificate signed by a private CA.

Retrieve the CA certificate bundle used to validate the HTTPS connections to the S3 endpoint by using the following command.

```
[user@demo ~]$ oc extract configmap/serviceaccount-ca -n
 openshift-kube-controller-manager
```

Export the path to the certificate in the `AWS_CA_BUNDLE` environment variable.

```
[user@demo ~]$ export AWS_CA_BUNDLE="${PWD}/ca-bundle.crt"
```

List the available buckets by specifying the custom S3 endpoint.

```
[user@demo ~]$ aws s3 ls s3:// --endpoint-url "https://${BUCKET_HOST}"
2021-03-31 18:30:06 my-object-bucket-claim-27f4ce2f-13ca-44cc-b8a5-1a747989cfb3
```

Note

- The endpoint protocol is `http` when using the Ceph RGW gateway.
- You can disable the SSL certificate validation with the `--no-verify-ssl` option.

List the contents of the S3 bucket.

```
[user@demo ~]$ aws s3 ls s3://${BUCKET_NAME} \
  --endpoint-url "https://${BUCKET_HOST}" --summarize

Total Objects: 0
   Total Size: 0
```

Note

You can install the `awscli-plugin-endpoint` Python package to set the endpoint in the S3 client configuration file.

References

For more information, refer to the product documentation for the Multicloud Object Gateway and Ceph RADOS Object Gateway in *Red Hat OpenShift Data Foundation* at
https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/html-single/managing_hybrid_and_multicloud_resources/index

Guided Exercise

Creating Object Bucket Claims and Accessing Object Storage

In this exercise, you will create object bucket claims and access the object bucket using standard tools.

Outcomes

You should be able to:

- Create an OBC in the OpenShift Console.
- View the access parameters stored in the configuration map and secret.
- Configure the AWS command line tool to access the S3 storage in the object bucket.

Before You Begin

To perform this exercise, ensure that you have:

- A running OpenShift cluster.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start object-obc
```

Instructions

1. Access the OpenShift console.

   1.1. Log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

   1.2. Create a new project for this exercise.

```
[student@workstation ~]$ oc new-project obc-practice
Now using project "obc-practice" on server "https://api.ocp4.example.com:6443".
...output omitted...
```

   1.3. Show the URL for the OpenShift web console.

```
[student@workstation ~]$ oc whoami --show-console
https://console-openshift-console.apps.ocp4.example.com
```

1.4. Open the URL in a web browser and log in to the OpenShift web console. Click **htpasswd_provider** and enter the following credentials.

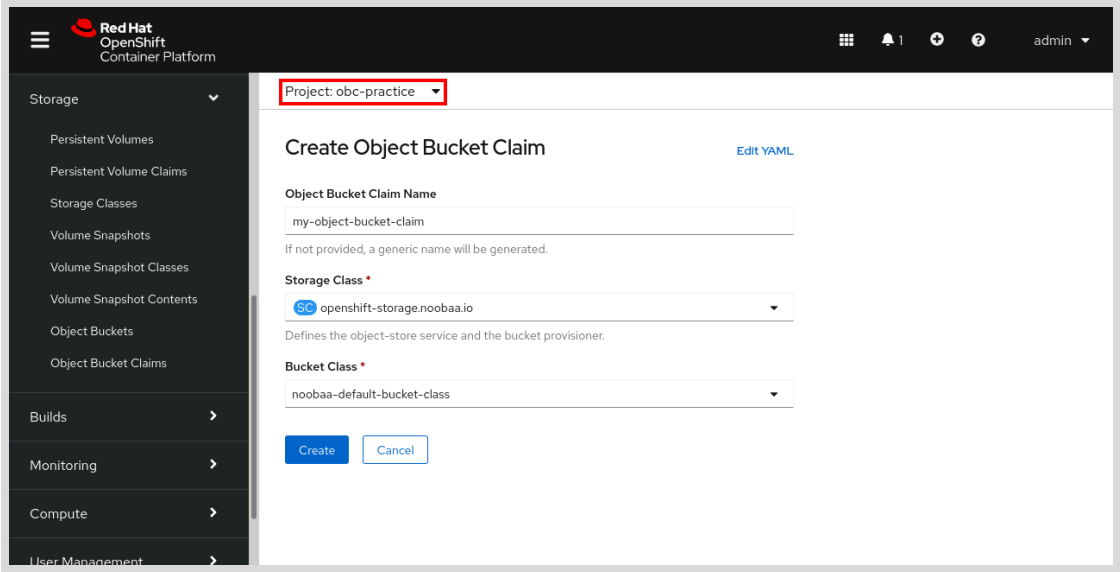
- Username: `admin`
- Password: `redhat`

1.5. Select **Storage** > **Object Bucket Claims** in the OpenShift web console. Click **Create Object Bucket Claim**.

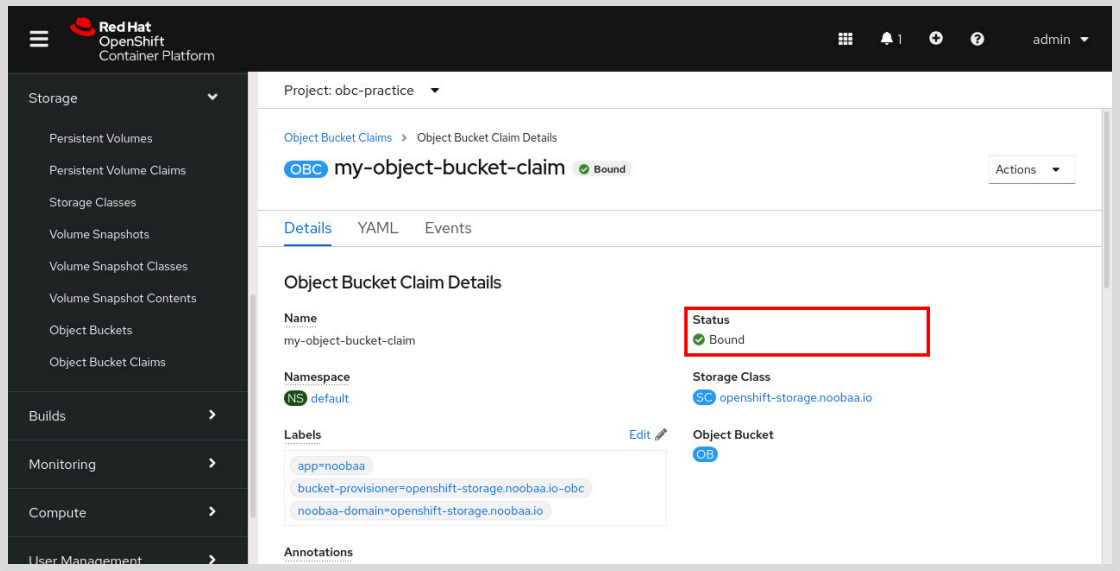
1.6. Select the `obc-practice` project.

1.7. Complete the following details and then click **Create**:

- Object Bucket Claim Name: `my-object-bucket-claim`
- Storage Class: `openshift-storage.noobaa.io`
- Bucket Class `noobaa-default-bucket-class`



1.8. Verify that the **status** field in the next screen is set to **Bound**.



2. The OBC resource creates a configuration map and a secret with all the parameters needed to connect to the bucket.

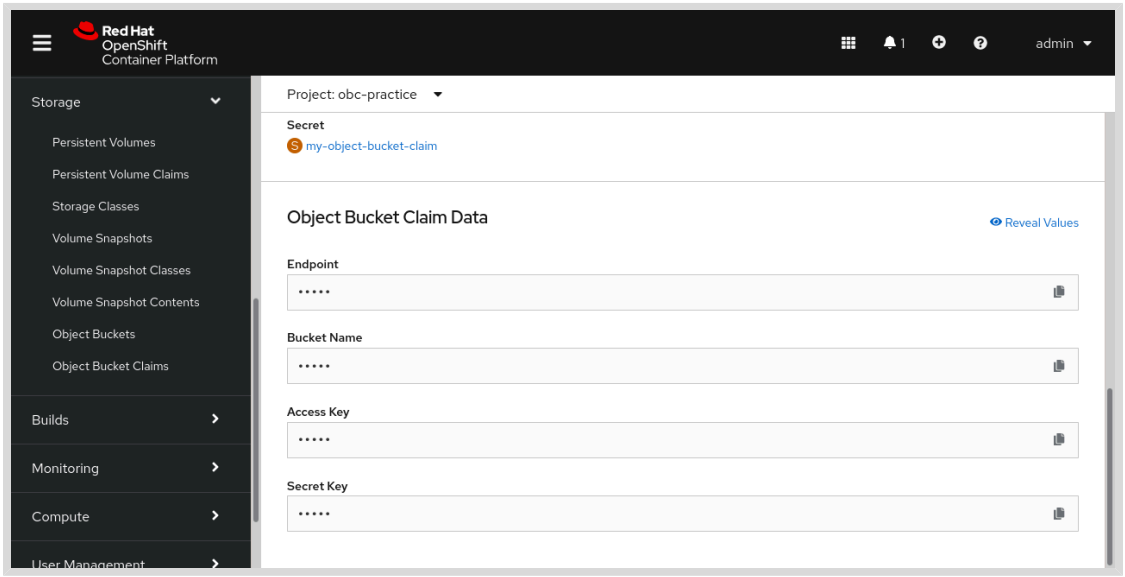
   The configuration map contains the following parameters:

   - `BUCKET_HOST`
   - `BUCKET_PORT`
   - `BUCKET_NAME`
   - `BUCKET_REGION`
   - `BUCKET_SUBREGION`

   The secret contains the keys required to access the bucket.

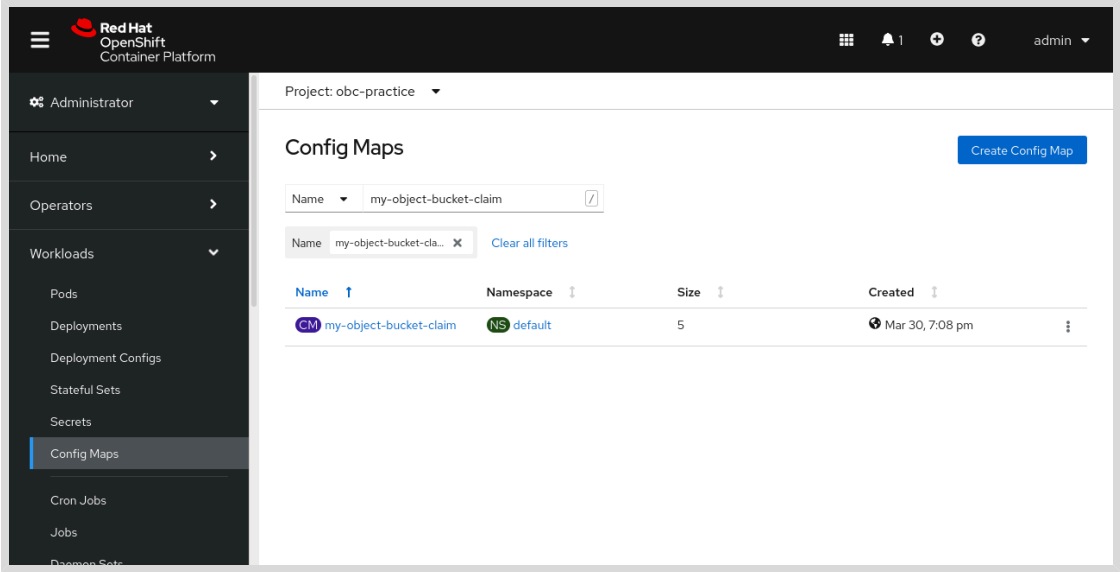
   - `AWS_ACCESS_KEY_ID`
   - `AWS_SECRET_ACCESS_KEY`

   2.1. You can view the parameters in the web console at the bottom of the OBC resource page.

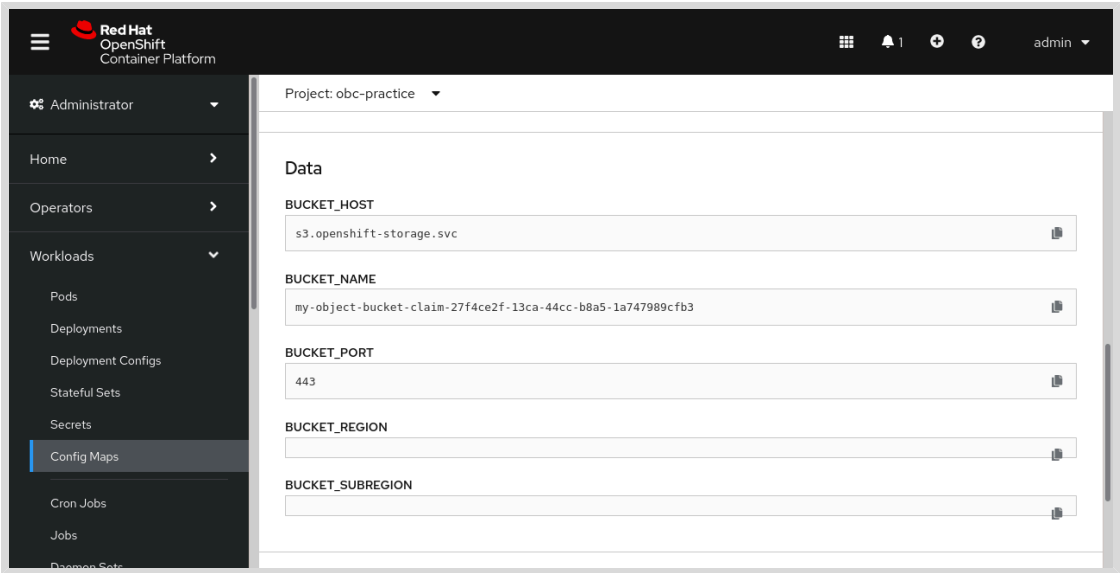


2.2. You can also view the configuration map and secret in the OpenShift web console.

Navigate to **Workloads** > **Config Maps** in the OpenShift web console. Type `my-object-bucket-claim` to filter the resources. Click on the `my-object-bucket-claim` configuration map resource to view it.

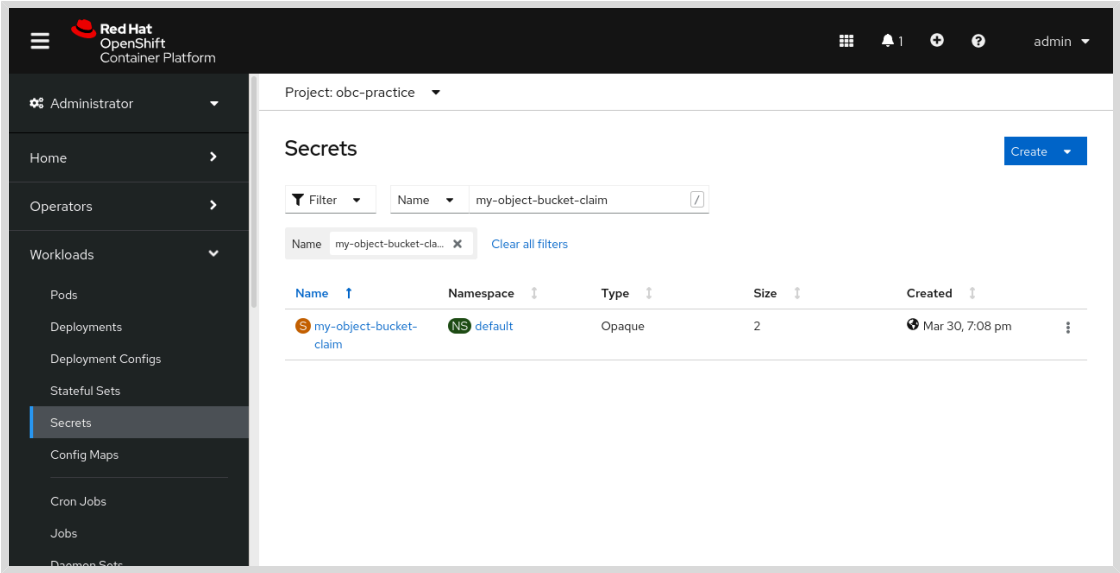


2.3. Scroll to the **Data** section of the configuration map resource to see the values.

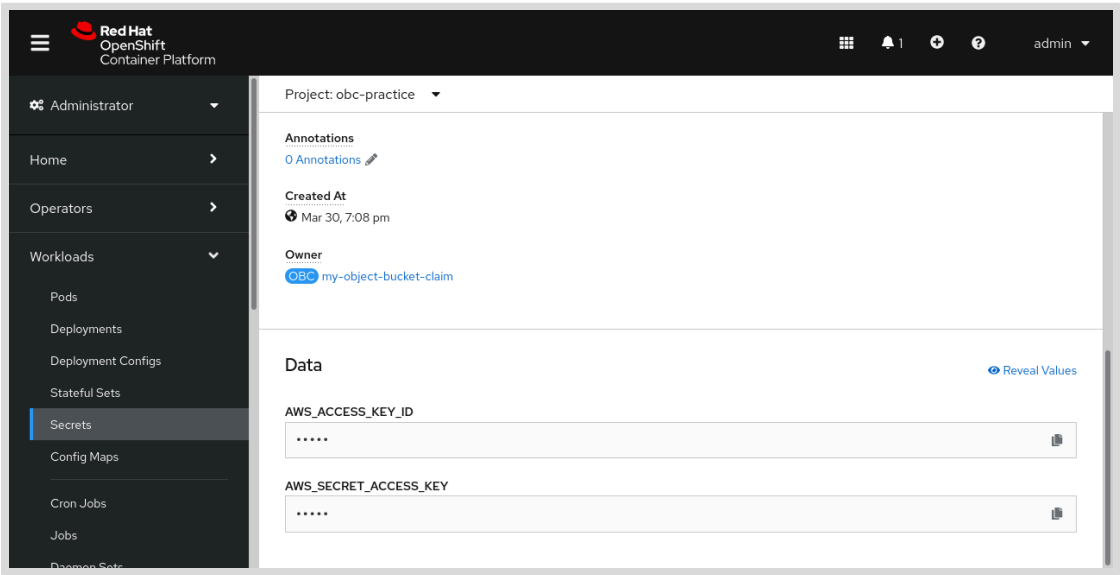


2.4. You can retrieve the access keys for the S3 bucket from the secret resource.

Navigate to **Workloads** > **Secrets** in the OpenShift web console. Type `my-object-bucket-claim` to filter the resources. Click on the `my-object-bucket-claim` secret resource to view it.



2.5. Scroll to the **Data** section of the secret resource and click **Reveal Values** to display the parameters.



3. Retrieve the route for the S3 endpoint and the bucket name.

3.1. Retrieve the endpoint for the S3 service.

```
[student@workstation ~]$ oc get route/s3 -n openshift-storage \
  -o jsonpath='{.spec.host}{"\n"}'
s3-openshift-storage.apps.ocp4.example.com
```

Note
You can also retrieve the S3 endpoint URL in the OpenShift web console in the **Networking** > **Routes** section.

3.2. Export the endpoint URL in an environment variable.

```
[student@workstation ~]$ export \
  BUCKET_HOST="s3-openshift-storage.apps.ocp4.example.com"
```

Note

You can also use the following command to export the BUCKET_HOST environment variable in a single step.

```
[student@workstation ~]$ export BUCKET_HOST="$(oc get route/s3 \
  -n openshift-storage -o jsonpath='{.spec.host}')"
...output omitted...
```

3.3. Retrieve the data from the configuration map.

```
[student@workstation ~]$ oc extract configmap/my-object-bucket-claim --to=-
# BUCKET_HOST
s3.openshift-storage.svc
# BUCKET_PORT
443
# BUCKET_NAME
my-object-bucket-claim-27f4ce2f-13ca-44cc-b8a5-1a747989cfb3
# BUCKET_REGION

# BUCKET_SUBREGION
```

3.4. Export the bucket name in an environment variable.

```
[student@workstation ~]$ export \
  BUCKET_NAME="my-object-bucket-claim-27f4ce2f-13ca-44cc-b8a5-1a747989cfb3"
```

Note

You can also use the following command to export the BUCKET_NAME environment variable in a single step.

```
[student@workstation ~]$ export BUCKET_NAME="$(oc extract \
  configmap/my-object-bucket-claim --keys=BUCKET_NAME --to=-)"
...output omitted...
```

4. Configure the AWS command line tool to access the S3 bucket.

4.1. Verify that the AWS CLI tool is installed.

```
[student@workstation ~]$ aws --version
aws-cli/1.22.76 Python/3.6.8 Linux/4.18.0-305.el8.x86_64 botocore/1.24.21
```

4.2. Export the path to the certificate in the AWS_CA_BUNDLE environment variable.

```
[student@workstation ~]$ export AWS_CA_BUNDLE=/etc/pki/tls/certs/ca-bundle.crt
```

4.3. Retrieve the *access key* and *secret access key* from the secret.

```
[student@workstation ~]$ oc extract secret/my-object-bucket-claim --to=-
# AWS_ACCESS_KEY_ID
YEAsbMJnG3o1bGANZprt
# AWS_SECRET_ACCESS_KEY
xjaeCDhskn7lfrdA7WqzoUxpiRYuyjc9uDaWlMw3
```

4.4. Run the following command to configure the tool. Enter the *access key* and *secret access key* that you retrieved in a previous step.

```
[student@workstation ~]$ aws configure
AWS Access Key ID [None]: YEAsbMJnG3o1bGANZprt
AWS Secret Access Key [None]: xjaeCDhskn7lfrdA7WqzoUxpiRYuyjc9uDaWlMw3
Default region name [None]:
Default output format [None]:
```

4.5. Check the `~/.aws/credentials` file to verify that both your *access key* and *secret access key* are present.

```
[student@workstation ~]$ cat ~/.aws/credentials
[default]
aws_access_key_id = YEAsbMJnG3o1bGANZprt
aws_secret_access_key = xjaeCDhskn7lfrdA7WqzoUxpiRYuyjc9uDaWlMw3
```

4.6. List the available buckets specifying the custom endpoint.

```
[student@workstation ~]$ aws s3 ls s3:// --endpoint-url \
  "https://${BUCKET_HOST}"
2021-03-31 18:30:06 my-object-bucket-claim-27f4ce2f-13ca-44cc-b8a5-1a747989cfb3
```

4.7. Install the `awscli-plugin-endpoint` package.

```
[student@workstation ~]$ pip install awscli-plugin-endpoint
Collecting awscli-plugin-endpoint
...output omitted...
```

4.8. Enable the plug-in in the configuration file.

```
[student@workstation ~]$ aws configure set plugins.endpoint awscli_plugin_endpoint
```

4.9. Set the custom endpoint in the configuration file.

```
[student@workstation ~]$ aws configure --profile default set s3.endpoint_url \
  https://${BUCKET_HOST}
```

4.10. Review the configuration file.

```
[student@workstation ~]$ cat ~/.aws/config
[default]
region = us-east-1
s3 =
    endpoint_url = https://s3-openshift-storage.apps.ocp4.example.com
[plugins]
endpoint = awscli_plugin_endpoint
```

4.11. List the available buckets without specifying the custom endpoint.

```
[student@workstation ~]$ aws s3 ls s3://
2021-03-31 18:51:12 my-object-bucket-claim-27f4ce2f-13ca-44cc-b8a5-1a747989cfb3
```

5. Access the S3 bucket with the AWS command-line tool.

5.1. List the bucket contents (the list will be empty).

```
[student@workstation ~]$ printenv BUCKET_NAME
my-object-bucket-claim-27f4ce2f-13ca-44cc-b8a5-1a747989cfb3

[student@workstation ~]$ aws s3 ls s3://${BUCKET_NAME} --summarize

Total Objects: 0
   Total Size: 0
```

5.2. Create a test file and upload it to the S3 bucket.

```
[student@workstation ~]$ echo "Hello world" > test-file.txt

[student@workstation ~]$ aws s3 cp test-file.txt s3://${BUCKET_NAME}/
upload: ./test-file.txt to s3://my-object-bucket-claim-27f4ce2f-13ca-44cc-
b8a5-1a747989cfb3/test-file.txt

[student@workstation ~]$ aws s3 ls s3://${BUCKET_NAME} --summarize
2021-03-31 19:10:27         12 test-file.txt

Total Objects: 1
   Total Size: 12
```

5.3. Retrieve the test file from the bucket.

```
[student@workstation ~]$ aws s3 cp s3://${BUCKET_NAME}/test-file.txt /tmp
download: s3://my-object-bucket-claim-27f4ce2f-13ca-44cc-b8a5-1a747989cfb3/test-
file.txt to ../../tmp/test-file.txt

[student@workstation ~]$ cat /tmp/test-file.txt
Hello world
```

5.4. Delete the test file from the bucket.

```
[student@workstation ~]$ aws s3 rm s3://${BUCKET_NAME}/test-file.txt
delete: s3://my-object-bucket-claim-27f4ce2f-13ca-44cc-b8a5-1a747989cfb3/test-
file.txt

[student@workstation ~]$ aws s3 ls s3://${BUCKET_NAME} --summarize

Total Objects: 0
   Total Size: 0
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish object-obc
```

This concludes the guided exercise.

Configuring Applications to Use Red Hat OpenShift Data Foundation Object Storage

Objectives

After completing this section, you should be able to configure an application to use S3 compatible object storage provided by Red Hat OpenShift Data Foundation.

S3 Bucket Settings and Access Keys

Applications use a client library to access buckets with the HTTP or HTTPS protocol. The client libraries authenticate with the S3 API by using a set of access keys.

When you create an object bucket claim, the API server creates a set of resources with the settings to access the S3 bucket.

Configuration Map
Contains the S3 endpoint and the name of the S3 bucket.

Secret
Contains the access keys used to authenticate with the S3 API.

You can pass the settings to a pod, deployment, job, or cron job that needs to access the S3 bucket.

Inject the S3 Settings to Environment Variables

You can inject all the settings from the configuration map and the secret into a container specification by adding the following section to the YAML file.

The `configMapRef` and `secretRef` keys indicate that all the keys in the configuration map and secret will be injected as environment variables into the container.

```
spec:
  containers:
  - name: image-tool-s3-obc
    image: quay.io/redhattraining/image-tool:latest
    env:
    - name: AWS_CA_BUNDLE (1)
      value: /run/secrets/kubernetes.io/serviceaccount/service-ca.crt
    envFrom:
    - configMapRef: (2)
        name: my-object-bucket-claim (3)
    - secretRef: (4)
        name: my-object-bucket-claim (5)
```

(1) Set the path to the CA used to validate the certificate of the S3 endpoint.

(2) Inject all keys from the configuration map as environment variables.

(3) Name of the configuration map associated with the object bucket claim.

4. Inject all keys from the secret as environment variables.

5. Name of the secret associated with the object bucket claim.

Note

The environment variables defined with `env` or `envFrom` override the environment variables defined on the container image, if any.

Inject the S3 Settings into Custom Environment Variables

The name of the environment variables needed to configure the S3 client library might be different across applications. You can use `valueFrom` to create environment variables that get their value from a specific key in a configuration map or secret.

- Use `configMapKeyRef` when the value is read from a key in a configuration map.
- Use `secretKeyRef` when the value is extracted from a key in a secret.

```
spec:
  containers:
  - name: image-tool-s3-obc
    image: quay.io/redhattraining/image-tool:latest
    env:
    - name: AWS_CA_BUNDLE (1)
      value: /run/secrets/kubernetes.io/serviceaccount/service-ca.crt
    - name: BUCKET_HOST (2)
      valueFrom:
        configMapKeyRef: (3)
          name: my-object-bucket-claim
          key: BUCKET_HOST
    - name: BUCKET_PORT
      valueFrom:
        configMapKeyRef:
          name: my-object-bucket-claim
          key: BUCKET_PORT
    - name: BUCKET_NAME
      valueFrom:
        configMapKeyRef:
          name: my-object-bucket-claim
          key: BUCKET_NAME
    - name: AWS_ACCESS_KEY_ID
      valueFrom:
        secretKeyRef: (4)
          name: my-object-bucket-claim
          key: AWS_ACCESS_KEY_ID
    - name: AWS_SECRET_ACCESS_KEY
      valueFrom:
        secretKeyRef:
          name: my-object-bucket-claim
          key: AWS_SECRET_ACCESS_KEY
```

1. Set the path to the CA used to validate the certificate of the S3 endpoint.

2. Define a custom environment variable.

3. Get the value for an environment variable from the configuration map associated with the object bucket claim.

4. Get the value for an environment variable from the secret associated with the object bucket claim.

Note

The environment variables defined with `env` or `envFrom` override the environment variables defined on the container image, if any.

References

For more information, refer to the product documentation for the Multicloud Object Gateway and RADOS Object Gateway in the *Red Hat OpenShift Data Foundation* at https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/html-single/managing_hybrid_and_multicloud_resources/index

Guided Exercise

Configuring Applications to Use Red Hat OpenShift Data Foundation Object Storage

In this exercise, you will configure an application to use S3 compatible object storage provided by OpenShift Data Foundation.

Outcomes

You should be able to:

- Deploy an application that uses an object bucket as backing storage.
- Interact with the S3 storage by using the example application.
- Access the files on the S3 bucket with the AWS S3 command line tool.

Before You Begin

To perform this exercise, ensure that you have:

- A running OpenShift cluster.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start object-configure
```

Instructions

1. Deploy the example application in the OpenShift cluster.

   1.1. Log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

   1.2. Create a new project.

```
[student@workstation ~]$ oc new-project image-tool
Now using project "image-tool" on server "https://api.ocp4.example.com:6443".
...output omitted...
```

   1.3. Change to the `~/DO370/labs/object-configure` directory.

```
[student@workstation ~]$ cd ~/DO370/labs/object-configure
```

1.4. Create the resources for the test application.

```
[student@workstation object-configure]$ oc apply -f serviceaccount.yaml -f
 deployment.yaml -f service.yaml -f route.yaml
serviceaccount/image-tool created
deployment.apps/image-tool-s3-obc created
service/image-tool-s3-obc created
route.route.openshift.io/image-tool-s3-obc created
```

1.5. Verify that the pods are running.

```
[student@workstation object-configure]$ oc get pods -l app=image-tool-s3-obc
NAME                                READY   STATUS    RESTARTS   AGE
image-tool-s3-obc-9855c8f6c-glb5n   1/1     Running   0          5s
```

1.6. Retrieve the hostname for the route.

```
[student@workstation object-configure]$ oc get route/image-tool-s3-obc \
  -o jsonpath='{.spec.host}{"\n"}'
image-tool-s3-obc-image-tool.apps.ocp4.example.com
```

1.7. Open the URL in the web browser.

- https://image-tool-s3-obc-image-tool.apps.ocp4.example.com/

1.8. View the logs for the deployment and review the message that says the application is using **ephemeral** storage.

This means that the application container does not have a persistent volume mounted in `/var/storage`.

```
[student@workstation object-configure]$ oc logs deployment/image-tool-s3-obc |
 head
[2021-04-06 22:21:09] INFO in app: Using EPHEMERAL as backing storage
[2021-04-06 22:21:09] INFO in app: Serving files from /var/storage
[2021-04-06 22:21:09 +0000] [1] [INFO] Starting gunicorn 20.0.4
[2021-04-06 22:21:09 +0000] [1] [INFO] Listening at: http://0.0.0.0:5000 (1)
[2021-04-06 22:21:09 +0000] [1] [INFO] Using worker: sync
[2021-04-06 22:21:09 +0000] [12] [INFO] Booting worker with pid: 12
```

1.9. Edit the `deployment.yaml` file and remove the comment from the following lines.

```
        env:
          - name: AWS_CA_BUNDLE
            value: /run/secrets/kubernetes.io/serviceaccount/service-ca.crt
        envFrom:
        - configMapRef:
            name: image-tool-obc
        - secretRef:
            name: image-tool-obc
```

Note

Only remove the # character and do not remove any white spaces that might affect the YAML indentation.

1.10. Create the OBC resource and apply the new version of the deployment. List the pods, and then wait until there is only one replica running.

```
[student@workstation object-configure]$ oc apply -f obc.yaml -f deployment.yaml
objectbucketclaim.objectbucket.io/image-tool-obc created
deployment.apps/image-tool-s3-obc configured

[student@workstation object-configure]$ oc get pods -l app=image-tool-s3-obc
NAME                                READY   STATUS        RESTARTS   AGE
image-tool-s3-obc-6847649bc-f6jcb   1/1     Running       0          7s
```

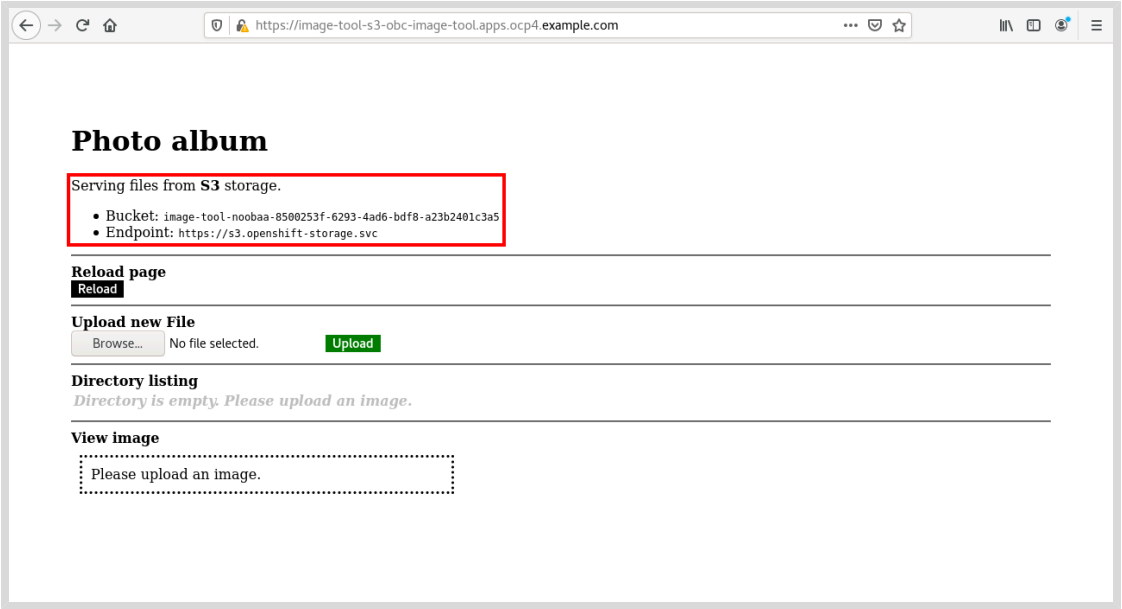
Note

You might need to run the command multiple times until the desired condition is reached.

2. Access the new version of the application.

2.1. Refresh the page in the web browser. The application displays a message specifying what type of storage it is using.

- https://image-tool-s3-obc-image-tool.apps.ocp4.example.com/



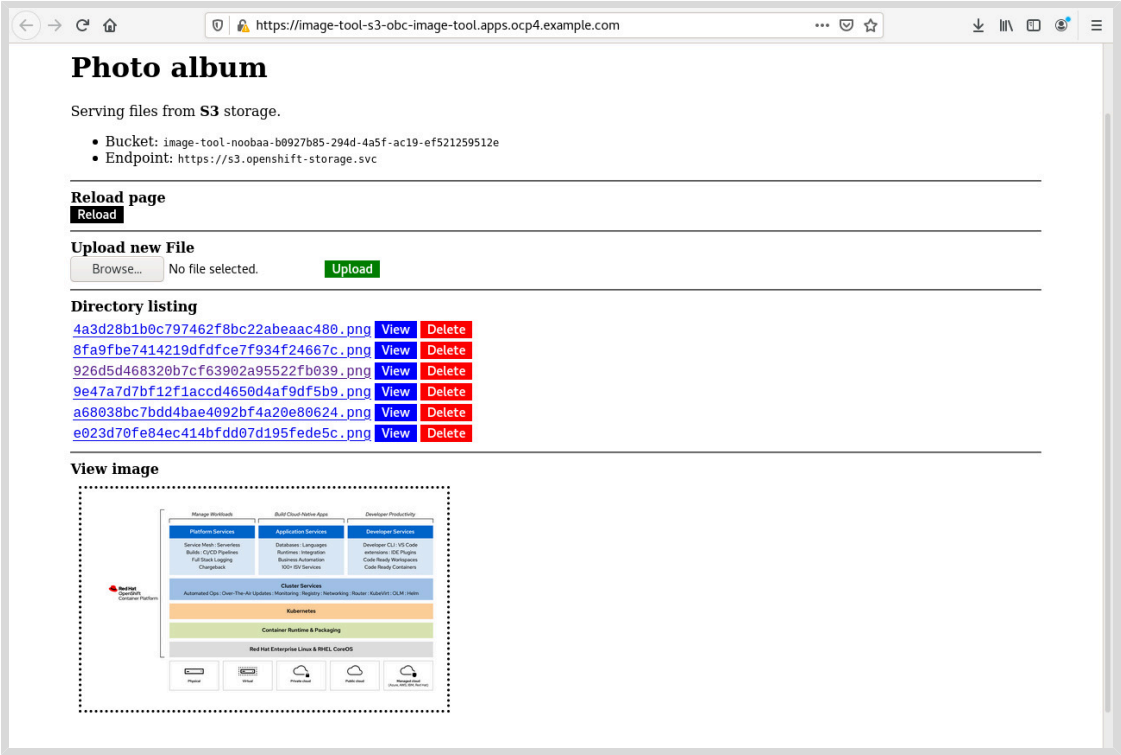
2.2. Review the logs again to see if the application is using S3 storage.

```
[student@workstation object-configure]$ oc logs deployment/image-tool-s3-obc |
 head
[2021-04-06 23:10:49] INFO in app: Using S3 as backing storage
[2021-04-06 23:10:49] INFO in app: Serving files from s3://image-tool-
noobaa-8500253f-6293-4ad6-bdf8-a23b2401c3a5
[2021-04-06 23:10:49 +0000] [1] [INFO] Starting gunicorn 20.0.4
[2021-04-06 23:10:49 +0000] [1] [INFO] Listening at: http://0.0.0.0:5000 (1)
[2021-04-06 23:10:49 +0000] [1] [INFO] Using worker: sync
[2021-04-06 23:10:49 +0000] [12] [INFO] Booting worker with pid: 12
[2021-04-06 23:11:05] INFO in app: S3: Listing bucket
...output omitted...
```

2.3. Click **Browse**, select an image from the `~/DO370/labs/object-configure/pictures` directory, and then click **Open**. Click **Upload** to send the file to the application.

Repeat this step for all the images in the `~/DO370/labs/object-configure/pictures` directory.

2.4. Click **View** to the right of the image name to view it in the lower pane.



2.5. Click **Delete** to the right of the **View** button to remove any of the uploaded images.

2.6. Switch to the terminal window and view the deployment logs.

```
[student@workstation object-configure]$ oc logs deployment/image-tool-s3-obc |
 grep S3
[2021-04-06 23:10:32] INFO in app: Using S3 as backing storage
[2021-04-06 23:11:11] INFO in app: S3: Listing bucket
[2021-04-06 23:11:11] INFO in app: S3: Put object: f309981d5285cf8efa2f7ab...png
[2021-04-06 23:11:11] INFO in app: S3: Write: f309981d5285cf8efa2f7ab...png
[2021-04-06 23:11:54] INFO in app: S3: Listing bucket
[2021-04-06 23:12:14] INFO in app: S3: Read: f309981d5285cf8efa2f7ab...png
[2021-04-06 23:12:21] INFO in app: S3: Delete: f309981d5285cf8efa2f7ab...png
...output omitted...
```

2.7. Delete the deployment. The application data is stored in a persistent volume (PV) and it will be available when new replica pods are executed.

```
[student@workstation object-configure]$ oc delete deployment/image-tool-s3-obc
deployment.apps "image-tool-s3-obc" deleted

[student@workstation object-configure]$ oc get pods -l app=image-tool-s3-obc
No resources found in image-tool-s3-obc namespace.
```

2.8. Create the deployment again, and then wait until all the new pods are running.

```
[student@workstation object-configure]$ oc create -f deployment.yaml
deployment.apps/image-tool-s3-obc created

[student@workstation object-configure]$ oc get pods -l app=image-tool-s3-obc
NAME                                READY   STATUS    RESTARTS   AGE
image-tool-s3-obc-6847649bc-cwbsc   1/1     Running   0          77s
```

2.9. Refresh the page in the web browser and review the message that says the application is using **S3** storage.

- https://image-tool-s3-obc-image-tool.apps.ocp4.example.com/

Verify that the images you uploaded previously are shown.

3. Access the files in the S3 bucket.

3.1. Enter the `image-tool-s3-obc` deployment.

```
[student@workstation object-configure]$ oc exec -it deployment/image-tool-s3-obc
 -- /bin/bash
...output omitted...
(app-root) bash-4.4$
```

3.2. List the environment variables inside the pod.

```
(app-root) sh-4.4$ env | egrep '^(AWS|BUCKET)_'
AWS_CA_BUNDLE=/run/secrets/kubernetes.io/serviceaccount/service-ca.crt
AWS_ACCESS_KEY_ID=XXXXXXXXXXXXXXXXXXX
AWS_SECRET_ACCESS_KEY=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
BUCKET_HOST=s3.openshift-storage.svc
BUCKET_PORT=443
BUCKET_NAME=image-tool-noobaa-8500253f-6293-4ad6-bdf8-a23b2401c3a5
BUCKET_REGION=
BUCKET_SUBREGION=
```

3.3. List the files in the S3 bucket.

```
(app-root) sh-4.4$ aws s3 ls s3://${BUCKET_NAME} --endpoint-url "https://
${BUCKET_HOST}"
2021-04-06 23:15:45      16773 82f755984d94a0a6a8104c5318bb21da.png
2021-04-06 23:15:33      19348 e8dd5c8cf7d70f6d19f66d1f8df6aac7.png
2021-04-06 23:15:28       5855 f309981d5285cf8efa2f7ab8e2149343.png
```

3.4. Review the contents of the `~/.aws/config` file. The configuration uses the `awscli-plugin-endpoint` Python package to allow specifying a custom endpoint in the AWS CLI configuration file.

```
(app-root) sh-4.4$ cat ~/.aws/config
[default]
region = us-east-1
s3 =
    endpoint_url = https://s3.openshift-storage.svc/

[plugins]
endpoint = awscli_plugin_endpoint
```

Note

The `endpoint_url` points to the `s3` service in the `openshift-storage` namespace; the DNS name is for internal use within the RHOCP cluster.

3.5. List the files in the S3 bucket without specifying the custom endpoint.

```
(app-root) sh-4.4$ aws s3 ls s3://${BUCKET_NAME}
2021-04-06 23:15:45      16773 82f755984d94a0a6a8104c5318bb21da.png
2021-04-06 23:15:33      19348 e8dd5c8cf7d70f6d19f66d1f8df6aac7.png
2021-04-06 23:15:28       5855 f309981d5285cf8efa2f7ab8e2149343.png
```

3.6. Exit the deployment pod.

```
(app-root) bash-4.4$ exit
```

3.7. Change to the `/home/student` directory.

```
[student@workstation object-configure]$ cd
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish object-configure
```

This concludes the guided exercise.

Monitoring Red Hat OpenShift Data Foundation Object Buckets

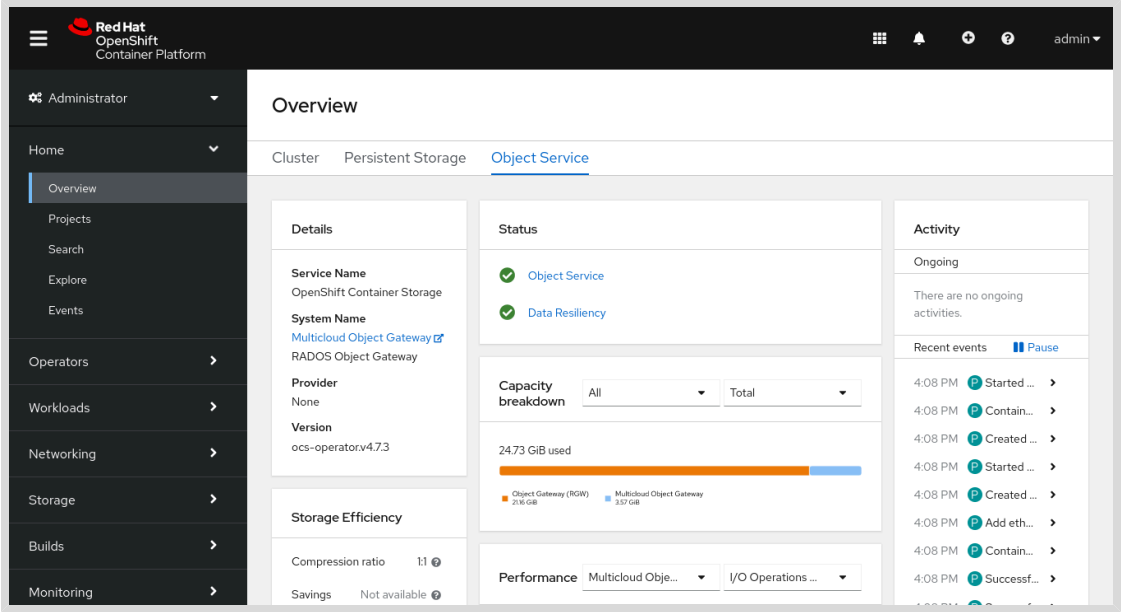
Objectives

After completing this section, you should be able to identify the storage metrics associated with the object buckets.

Monitoring Overview

The object service provided by OpenShift Data Foundation has a storage overview dashboard that includes the following items.

- Details panel for the NooBaa MCG and Ceph RGW S3 storage providers
- Overall status of the object service and the data resiliency
- Storage efficiency that shows the compression ratio and the storage savings
- Capacity breakdown of the storage used by the NooBaa MCG and Ceph RGW S3 buckets
- Number of object buckets and OBCs
- Recent events in the activity section
- Performance charts for the NooBaa MCG and Ceph RGW storage providers



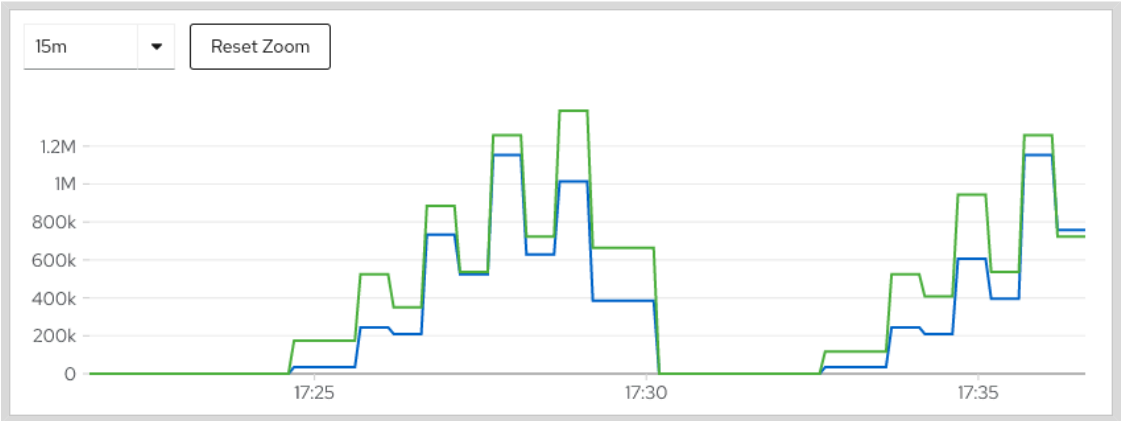
Accessing Metrics on the OpenShift Web Console

The metrics for the NooBaa MCG and Ceph RGW storage providers is available in the Monitoring section of the OpenShift web console.

You can set the refresh rate of the graph on the upper right of the page. You can also set the zoom to a given number of minutes on the upper left of the graph.

You can add the metrics in the expression field. Press **Add Query** to add another metric query. Press **Run Queries** when you are finished adding queries.

These trends represent the input and output bandwidth rate for NooBaa MCG over the previous two minutes.



Metrics for NooBaa MCG Gateway

The following table lists some of the available metrics for the NooBaa MCG gateway.

Prometheus metrics for the NooBaa MCG gateway

Metric Identifier	Description
NooBaa_total_usage	Total disk usage of NooBaa MCG object buckets
NooBaa_providers_logical_size	Total disk usage of the raw data
NooBaa_providers_physical_size	Total disk usage of the compressed data
NooBaa_providers_bandwidth_read_size	Total bytes transferred on read requests
NooBaa_providers_bandwidth_write_size	Total bytes transferred on write requests
NooBaa_num_objects_buckets_claims	Total number of objects in all buckets
NooBaa_providers_ops_read_num	Total number of read operations
NooBaa_providers_ops_write_num	Total number of write operations

Metrics for the Ceph RGW Gateway

The following table lists some of the available metrics for the Ceph RGW gateway.

Prometheus metrics for Ceph RGW gateway

Metric Identifier	Description
ceph_rgw_get	Number of objects retrieved from all buckets
ceph_rgw_put	Number of objects uploaded to all buckets
ceph_rgw_req	Number of requests to the RADOS Object Gateway endpoint
ceph_rgw_get_b	Total bytes on transferred read requests
ceph_rgw_put_b	Total bytes on transferred write requests
ceph_rgw_qlen	Request queue length
ceph_rgw_qactive	Number of active requests
ceph_rgw_failed_req	Total number of failed requests

References

For more information, refer to the product documentation for *Red Hat OpenShift Data Foundation* monitoring at
https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.7/html-single/monitoring_openshift_container_storage/index

Guided Exercise

Monitoring Red Hat OpenShift Data Foundation Object Buckets

In this exercise, you will inspect storage metrics for the object buckets in the OpenShift web console.

Outcomes

You should be able to:

- Identify the metrics used to measure input and output bandwidth rate for the NooBaa object buckets.
- Deploy an application that generates read and write activity on the object bucket.
- Observe the bandwidth rate changes in the OpenShift web console.

Before You Begin

To perform this exercise, ensure that you have:

- A running OpenShift cluster with OpenShift Data Foundation installed.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start bucket-monitor
```

Instructions

1. Access the OpenShift console.

   1.1. Log in to your OpenShift cluster as the `developer` user.

```
[student@workstation ~]$ oc login -u developer -p developer \
  https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

   1.2. Change to the `~/DO370/labs/bucket-monitor` directory.

```
[student@workstation ~]$ cd ~/DO370/labs/bucket-monitor
```

   1.3. Create a new project for this exercise.

```
[student@workstation bucket-monitor]$ oc new-project object-monitor
Now using project "object-monitor" on server "https://api.ocp4.example.com:6443".
...output omitted...
```

1.4. Create an OBC.

```
[student@workstation bucket-monitor]$ oc apply -f obc.yaml
objectbucketclaim.objectbucket.io/object-bucket created
```

1.5. Create a deployment that generates read and write operations to an object bucket.

```
[student@workstation bucket-monitor]$ oc apply -f s3-app.yaml
deployment.apps/awscli created
configmap/settings created
configmap/entrypoint created
```

1.6. Confirm that the pod is running.

```
[student@workstation bucket-monitor]$ oc get pods
NAME                      READY   STATUS    RESTARTS   AGE
awscli-67fdd4459c-m9dwj   1/1     Running   0          30s
```

Note

You might need to run the command multiple times until the desired condition is reached.

1.7. Get the deployment logs to confirm that there is read and write activity on the object bucket.

```
[student@workstation bucket-monitor]$ oc logs deployment/awscli | tail
...output omitted...
Tue Jun  1 17:25:12 UTC 2021
==> Listing bucket <==
2021-06-01 17:24:52    1.0 MiB object-001
2021-06-01 17:25:01    2.0 MiB object-002

Total Objects: 2
   Total Size: 3.0 MiB
==> Copying 'object-003' to S3 bucket <==
3+0 records in
3+0 records out
3145728 bytes (3.1 MB) copied, 5.54899 s, 567 kB/s
==> Reading 'object-001' from S3 bucket <==
d41d8cd98f00b204e9800998ecf8427e object-001
```

Note

Your output might be different but should have a similar structure.

2. Access the Metrics dashboard in the OpenShift web console.

2.1. Show the URL for the web console.

```
[student@workstation bucket-monitor]$ oc whoami --show-console
https://console-openshift-console.apps.ocp4.example.com
```

2.2. Open the URL in the web browser, select the `htpasswd` login provider, and then log in using the `admin` user and `redhat` password.

2.3. Navigate to **Monitoring** > **Metrics** in the OpenShift web console. Set the refresh rate to `15 seconds` on the upper right of the page, and also set the zoom to `15m` on the upper left of the graph.

2.4. Add the following metrics in the expression field. Enter the first metric in the expression field and press **Add Query** to add the second metric. Press **Run Queries** when done.

- `rate(NooBaa_providers_bandwidth_read_size[2m])`
- `rate(NooBaa_providers_bandwidth_write_size[2m])`

2.5. Observe the growing trends in the metrics over time. These trends represent the input and output bandwidth rate over the previous two minutes.

Note

It might take up to five minutes for the increased bandwidth rate to show in the graph.

3. Stop the read and write operations to the S3 bucket and observe the metrics behavior.

3.1. Delete the application deployment.

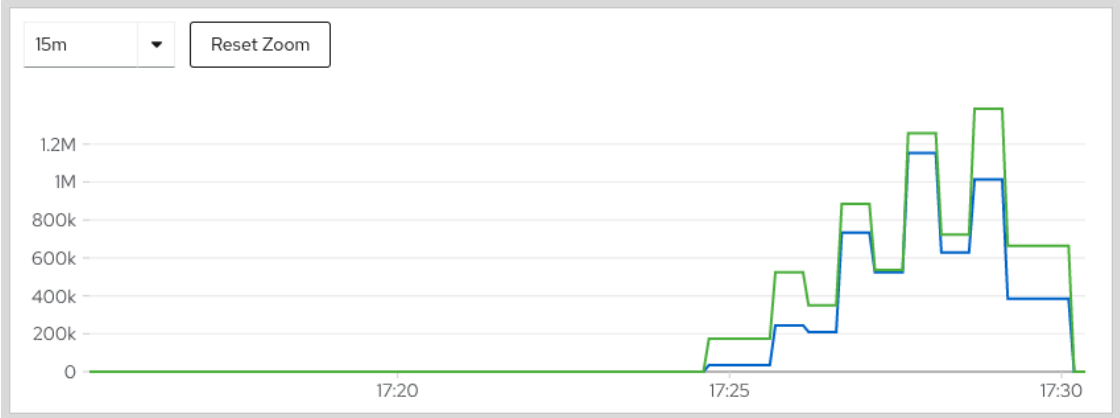
```
[student@workstation bucket-monitor]$ oc delete deployment/awscli
deployment.apps "awscli" deleted
```

3.2. Wait until the deployment and the pods are deleted.

```
[student@workstation bucket-monitor]$ watch oc get deployments,pods
```

Press `Ctrl+C` to end the `watch` command when the command displays that no resources were found in the namespace.

3.3. Wait for the bandwidth rate to go down in the monitoring graph.



Note

It might take up to five minutes for the decreased bandwidth rate to show in the graph.

4. Deploy the example application again and observe the metrics behavior.

4.1. Create the deployment again to generate read and write operations to the S3 bucket.

```
[student@workstation bucket-monitor]$ oc apply -f s3-app.yaml
deployment.apps/awscli created
configmap/settings configured
configmap/entrypoint configured
```

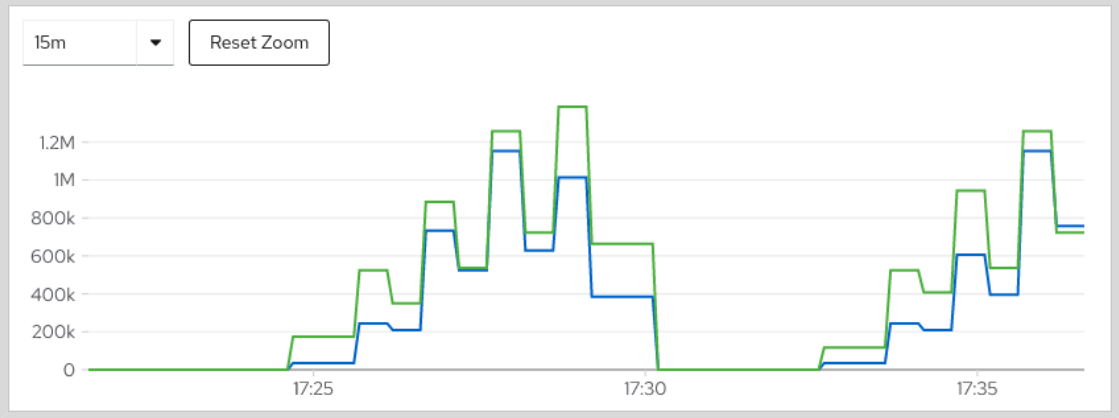
4.2. Review the logs for the deployment to check that there is read and write activity to the S3 bucket. Press `Ctrl+C` to exit the command after the bucket listing shows fifteen or more objects.

```
[student@workstation bucket-monitor]$ oc logs -f deployment/awscli
...output omitted...
Tue Jun  1 17:35:15 UTC 2021
==> Listing bucket <==
2021-06-01 17:32:59    1.0 MiB object-001
```

```
2021-06-01 17:33:07    2.0 MiB object-002
...output omitted...
2021-06-01 17:34:54   14.0 MiB object-014
2021-06-01 17:35:09   15.0 MiB object-015

Total Objects: 15
   Total Size: 120.0 MiB
==> Copying 'object-016' to S3 bucket <==
16+0 records in
16+0 records out
16777216 bytes (17 MB) copied, 5.6506 s, 3.0 MB/s
==> Reading 'object-014' from S3 bucket <==
105afb3604006ce6aec0cd27ff194bbc object-014
--
^C
```

4.3. Wait for the bandwidth rate to go up in the monitoring graph.



Note
It might take up to five minutes for the increased bandwidth rate to show in the graph.

4.4. Change to the `/home/student` directory.

```
[student@workstation bucket-monitor]$ cd
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish bucket-monitor
```

This concludes the guided exercise.

Lab

Configuring Applications to Use Red Hat OpenShift Data Foundation Object Storage

In this lab, you will:

- Create an object bucket claim.
- Deploy an application that uses the S3 bucket to save its data.
- Generate read and write traffic on the S3 bucket.
- Monitor the metrics for the object bucket.

Outcomes

You should be able to:

- Expose the settings for accessing the S3 bucket as environment variables in the deployment.
- Observe the trends in the monitoring graph.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start object-review
```

Instructions

1. Change to the `bucket-review` project.
2. Create an OBC named `object-review` that uses the `ocs-storagecluster-ceph-rgw` storage class. You can use the `~/DO370/labs/object-review/obc-review.yaml` file as a template.
3. Deploy an application that generates traffic to the S3 bucket associated with the `object-review` OBC. You can use the `~/DO370/labs/object-review/s3-app.yaml` file as a template.
4. Deploy an application that uses the S3 bucket associated with the `object-review` OBC to store its data. You can use the `~/DO370/labs/object-review/photo-album.yaml` file as a template.

   Open the URL https://photo-album-bucket-review.apps.ocp4.example.com/ in the web browser, and then upload the images contained in the `~/DO370/labs/object-review/pictures` folder to the application.
5. Navigate to **Monitoring** > **Metrics** in the OpenShift web console, and then inspect the rate of change over a two-minute period for the following object bucket metrics.

- Number of objects retrieved from all buckets
- Number of objects uploaded to all buckets
- Total bytes on transferred read requests
- Total bytes on transferred write requests

6. Delete the `s3-app` deployment and wait until the monitoring graph shows a downward rate.
7. Change to the `/home/student` directory.

```
[student@workstation object-review]$ cd
```

Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade object-review
```

Finish

As the `student` user on the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish object-review
```

This concludes the lab.

Solution

Configuring Applications to Use Red Hat OpenShift Data Foundation Object Storage

In this lab, you will:

- Create an object bucket claim.
- Deploy an application that uses the S3 bucket to save its data.
- Generate read and write traffic on the S3 bucket.
- Monitor the metrics for the object bucket.

Outcomes

You should be able to:

- Expose the settings for accessing the S3 bucket as environment variables in the deployment.
- Observe the trends in the monitoring graph.

Before You Begin

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

```
[student@workstation ~]$ lab start object-review
```

Instructions

1. Change to the `bucket-review` project.

   1.1. Log in to your OpenShift cluster as the `admin` user.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

   1.2. Change to the `bucket-review` project.

```
[student@workstation ~]$ oc project bucket-review
Now using project "bucket-review" on server "https://api.ocp4.example.com:6443".
```

2. Create an OBC named `object-review` that uses the `ocs-storagecluster-ceph-rgw` storage class. You can use the `~/DO370/labs/object-review/obc-review.yaml` file as a template.

2.1. Change to the `~/DO370/labs/object-review` directory.

```
[student@workstation ~]$ cd ~/DO370/labs/object-review
```

2.2. Edit the `obc-review.yaml` file and modify the placeholder text to match the following content.

```
---
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: object-review
  namespace: bucket-review
  labels:
    app: photo-album
spec:
  generateBucketName: object-review
  storageClassName: ocs-storagecluster-ceph-rgw
```

2.3. Create the new OBC.

```
[student@workstation object-review]$ oc apply -f obc-review.yaml
...output omitted...
```

3. Deploy an application that generates traffic to the S3 bucket associated with the `object-review` OBC. You can use the `~/DO370/labs/object-review/s3-app.yaml` file as a template.

3.1. Edit the `s3-app.yaml` file and modify the placeholder text to match the following content in both the `init` container and the `awscli` container.

```
      envFrom:
      - configMapRef:
          name: object-review
      - secretRef:
          name: object-review
```

3.2. Create the new deployment.

```
[student@workstation object-review]$ oc apply -f s3-app.yaml
...output omitted...
```

4. Deploy an application that uses the S3 bucket associated with the `object-review` OBC to store its data. You can use the `~/DO370/labs/object-review/photo-album.yaml` file as a template.

Open the URL https://photo-album-bucket-review.apps.ocp4.example.com/ in the web browser, and then upload the images contained in the `~/DO370/labs/object-review/pictures` folder to the application.

4.1. Edit the `s3-app.yaml` file and modify the placeholder text to match the following content.

```
        envFrom:
        - configMapRef:
            name: object-review
        - secretRef:
            name: object-review
```

4.2. Create the new deployment.

```
[student@workstation object-review]$ oc apply -f photo-album.yaml
...output omitted...
```

4.3. Get the URL for the web application.

```
[student@workstation object-review]$ oc get route/photo-album -o
 jsonpath='{.spec.host}{"\n"}'
photo-album-bucket-review.apps.ocp4.example.com
```

4.4. Open the URL https://photo-album-bucket-review.apps.ocp4.example.com/ in the web browser, and then upload the images contained in the `~/DO370/labs/object-review/pictures` folder to the application.

5. Navigate to **Monitoring** > **Metrics** in the OpenShift web console, and then inspect the rate of change over a two-minute period for the following object bucket metrics.

- Number of objects retrieved from all buckets
- Number of objects uploaded to all buckets
- Total bytes on transferred read requests
- Total bytes on transferred write requests

5.1. Get the URL for the OpenShift web console and open it in the web browser.

```
[student@workstation object-review]$ oc whoami --show-console
https://console-openshift-console.apps.ocp4.example.com
```

5.2. Set the refresh rate to `15 seconds` on the upper right of the page, and also set the zoom to `15m` on the upper left of the graph.

5.3. Navigate to **Monitoring** > **Metrics** in the OpenShift web console and enter the following metrics as queries.

- `rate(ceph_rgw_get[2m])`
- `rate(ceph_rgw_put[2m])`
- `rate(ceph_rgw_get_b[2m])`
- `rate(ceph_rgw_put_b[2m])`

6. Delete the `s3-app` deployment and wait until the monitoring graph shows a downward rate.

6.1. Delete the deployment.

```
[student@workstation object-review]$ oc delete deployment/s3-app
...output omitted...
```

7. Change to the `/home/student` directory.

```
[student@workstation object-review]$ cd
```

Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade object-review
```

Finish

As the `student` user on the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish object-review
```

This concludes the lab.

Summary

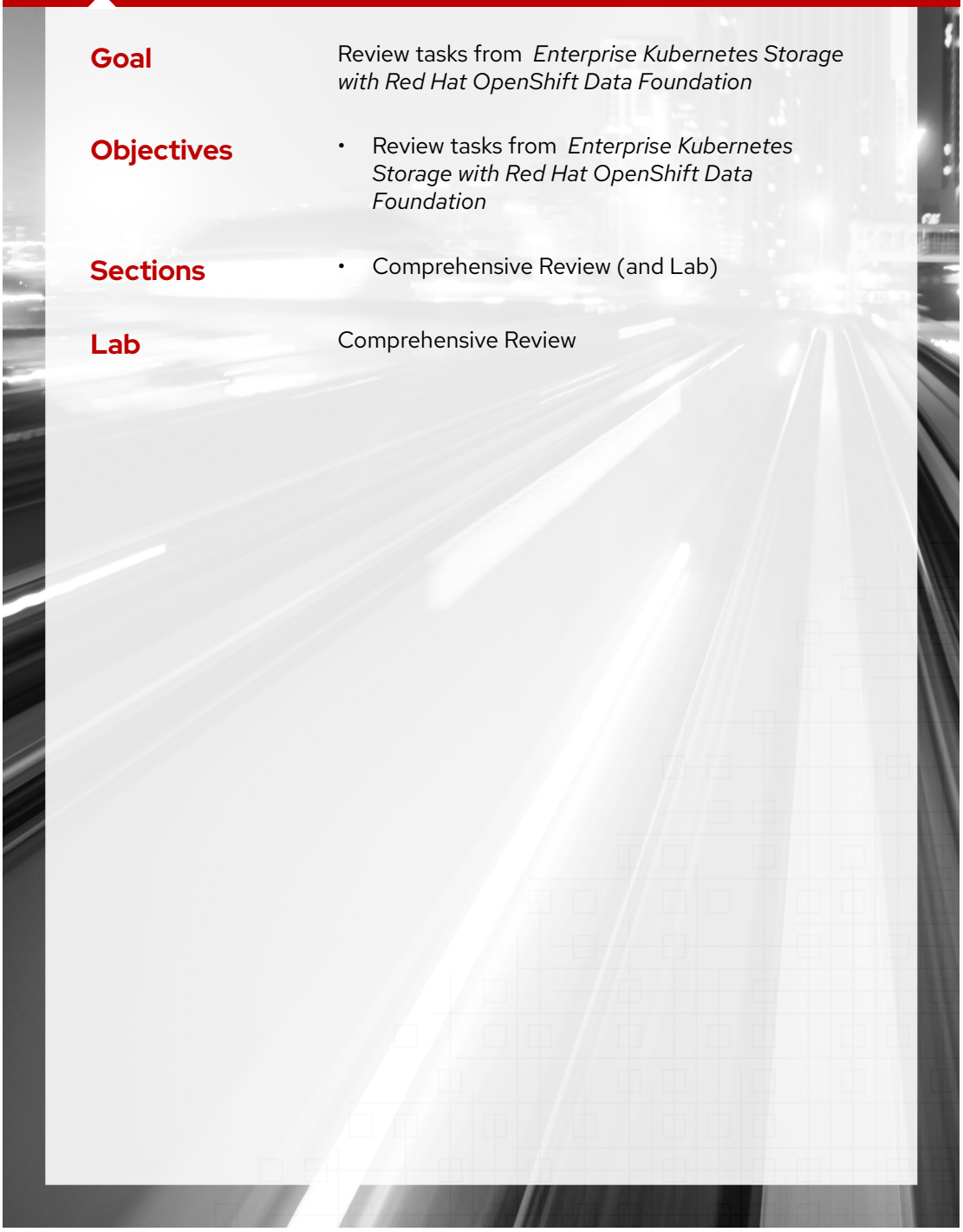
In this chapter, you learned about:

- Object storage, such as objects, buckets, prefixes, and the S3 API.
- Access keys needed to authenticate with the S3 API.
- The types of object storage provided by Red Hat OpenShift Data Foundation:
  - RADOS Object Gateway
  - Multicloud Object Gateway
- S3 service endpoints for the object storage classes.
- Object bucket claims and their associated resources, including object buckets, configuration maps, and secrets.
- Injecting S3 settings as environment variables using a YAML container specification.
- Monitoring object bucket metrics in the OpenShift web console.

Chapter 7

Comprehensive Review

Goal	Review tasks from *Enterprise Kubernetes Storage with Red Hat OpenShift Data Foundation*
Objectives	• Review tasks from *Enterprise Kubernetes Storage with Red Hat OpenShift Data Foundation*
Sections	• Comprehensive Review (and Lab)
Lab	Comprehensive Review



Comprehensive Review

Objectives

After completing this section, you should have reviewed and refreshed the knowledge and skills learned in *Enterprise Kubernetes Storage with Red Hat OpenShift Data Foundation*.

Reviewing Enterprise Kubernetes Storage with Red Hat OpenShift Data Foundation

Before beginning the comprehensive review for this course, you should be comfortable with the topics covered in each chapter.

You can refer to earlier sections in the textbook for extra study.

Chapter 1, Describing the Architecture and Deploying Red Hat OpenShift Data Foundation Using Internal Mode

Install an OpenShift Data Foundation cluster on an OpenShift cluster using the internal mode.

- Describe the architecture of Red Hat OpenShift Data Foundation.
- Deploy Red Hat OpenShift Data Foundation on-premises using the OpenShift web console
- Deploy Red Hat OpenShift Data Foundation on premises using the CLI.

Chapter 2, Configuring OpenShift Cluster Services to Use Red Hat OpenShift Data Foundation

Configure OpenShift Monitoring, Registry, and Logging to use storage provided by OpenShift Data Foundation.

- Identify and describe Red Hat OpenShift Data Foundation storage classes
- Configure the internal registry to use storage provided by Red Hat OpenShift Data Foundation
- Configure the OpenShift monitoring subsystem to use block storage from Red Hat OpenShift Data Foundation

Chapter 3, Configuring Application Workloads to Use Red Hat OpenShift Data Foundation File and Block Storage

Select and configure OpenShift Data Foundation storage classes to meet application requirements.

- Identify the components and gather information from a Ceph storage implementation for Red Hat OpenShift Data Foundation.
- Configure applications to use file storage provided by Red Hat OpenShift Data Foundation.
- Configure applications to use block storage provided by Red Hat OpenShift Data Foundation.
- Configure a new storage class with custom settings.

Chapter 4, Managing Red Hat OpenShift Data Foundation Block and File Storage Capacity

Monitor and expand overall OpenShift Data Foundation cluster capacity.

- Verify storage health metrics using cluster monitoring.
- Configuring and verify quotas and permissions for the Red Hat OpenShift Data Foundation cluster storage.
- Detect when a storage volume is close to full and expand the volume size without disrupting an application.
- Add additional disks to a Red Hat OpenShift Data Foundation cluster.

Chapter 5, Performing Backup and Restore of Kubernetes Block and File Volumes

Backup and restore application data using Kubernetes CSI APIs.

- Compare traditional backup and containerized applications backup approaches.
- Performing backup and restore of applications resources, images, and data volumes.
- Create volume snapshots for use in restoration and/or duplication of storage volumes.

Chapter 6, Configuring Applications to Use OpenShift Data Foundation Object Storage

Configure applications to use object storage from OpenShift Data Foundation.

- Identify the use cases for object storage and benefits provided by Red Hat OpenShift Data Foundation.
- Create object bucket claims and access the object buckets with S3 clients.
- Configure an application to use S3 compatible object storage provided by Red Hat OpenShift Data Foundation.
- Identify the storage metrics associated with the object buckets.

Lab

Comprehensive Review

In this review, you will review and refresh the knowledge and skills learned in Enterprise Kubernetes Storage with Red Hat OpenShift Data Foundation.

Outcomes

You should be able to:

- Update the internal OpenShift registry to use persistent storage.
- Create custom object classes.
- Deploy applications that use OpenShift Data Foundation block storage.
- Deploy applications that use OpenShift Data Foundation file storage.
- Deploy applications that use OpenShift Data Foundation object storage.
- Expand persistent volume claims for applications.
- Back up persistent volume claims.

Before You Begin

If you did not reset your `workstation` and `server` machines at the end of the previous chapter, save any work from earlier exercises and reset them now.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the environment for this lab is configured.

```
[student@workstation ~]$ lab start comprehensive-review
```

Instructions

In this review, you create persistent volume claims (PVCs) and bucket claims, use them in applications, and perform backups by using snapshots and clones.

- Log in to the cluster as the `admin` user with password `redhat` and create a project named `comprehensive-review`.
- Create a custom storage class named `ocs-storagecluster-ceph-rbd-xfs` that provides a block device that uses the `xfs` file system format. Use the `openshift-storage.rbd.csi.ceph.com` provisioner. Use the `~/DO370/labs/comprehensive-review/xfs-storageclass.yaml` file as starting point.
- Create a PostgreSQL database deployment named `pg-compreview` by using the `postgresql-persistent-sc` template and applying the following settings: `STORAGECLASS_NAME=ocs-storagecluster-ceph-rbd-xfs`, `VOLUME_CAPACITY=150Mi`, `POSTGRESQL_USER=student`, `POSTGRESQL_PASSWORD=redhat`, `POSTGRESQL_DATABASE=compreview`, `DATABASE_SERVICE_NAME=compreview`.

- Create an object bucket claim (OBC) named `image-object-bucket` that is based on the `ocs-storagecluster-ceph-rgw` storage class. Use the `~/DO370/labs/comprehensive-review/object-bucket-claim.yaml` file as starting point.
- Create an application that uses object storage based on the `~/DO370/labs/comprehensive-review/object-storage-app.yaml` file. Use the `image-object-bucket` OBC as storage for this deployment.
- Use the `image-tool-app.yaml` reference file to create an application named `compreview-file` that mounts a file storage PVC named `compreview-file-cephfs` under the `/var/storage` directory. Create the PVC with the `ocs-storagecluster-cephfs` object class.
- Backup the `compreview` PVC by creating a snapshot named `pg-compreview-snapshot`. Change the `volumesnapshotclass/ocs-storagecluster-rbdplugin-snapclass` deletion policy of the volume snapshot classes to preserve the snapshots. Use the `~/DO370/solutions/comprehensive-review/volumesnapshot.yaml` file as starting point.

Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade comprehensive-review
```

Finish

As the `student` user on the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish comprehensive-review
```

This concludes the lab.

Solution

Comprehensive Review

In this review, you will review and refresh the knowledge and skills learned in Enterprise Kubernetes Storage with Red Hat OpenShift Data Foundation.

Outcomes

You should be able to:

- Update the internal OpenShift registry to use persistent storage.
- Create custom object classes.
- Deploy applications that use OpenShift Data Foundation block storage.
- Deploy applications that use OpenShift Data Foundation file storage.
- Deploy applications that use OpenShift Data Foundation object storage.
- Expand persistent volume claims for applications.
- Back up persistent volume claims.

Before You Begin

If you did not reset your `workstation` and `server` machines at the end of the previous chapter, save any work from earlier exercises and reset them now.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the environment for this lab is configured.

```
[student@workstation ~]$ lab start comprehensive-review
```

Instructions

In this review, you create persistent volume claims (PVCs) and bucket claims, use them in applications, and perform backups by using snapshots and clones.

- Log in to the cluster as the `admin` user with password `redhat` and create a project named `comprehensive-review`.
- Create a custom storage class named `ocs-storagecluster-ceph-rbd-xfs` that provides a block device that uses the `xfs` file system format. Use the `openshift-storage.rbd.csi.ceph.com` provisioner. Use the `~/DO370/labs/comprehensive-review/xfs-storageclass.yaml` file as starting point.
- Create a PostgreSQL database deployment named `pg-compreview` by using the `postgresql-persistent-sc` template and applying the following settings: `STORAGECLASS_NAME=ocs-storagecluster-ceph-rbd-xfs`, `VOLUME_CAPACITY=150Mi`, `POSTGRESQL_USER=student`, `POSTGRESQL_PASSWORD=redhat`, `POSTGRESQL_DATABASE=compreview`, `DATABASE_SERVICE_NAME=compreview`.

- Create an object bucket claim (OBC) named `image-object-bucket` that is based on the `ocs-storagecluster-ceph-rgw` storage class. Use the `~/DO370/labs/comprehensive-review/object-bucket-claim.yaml` file as starting point.
- Create an application that uses object storage based on the `~/DO370/labs/comprehensive-review/object-storage-app.yaml` file. Use the `image-object-bucket` OBC as storage for this deployment.
- Use the `image-tool-app.yaml` reference file to create an application named `compreview-file` that mounts a file storage PVC named `compreview-file-cephfs` under the `/var/storage` directory. Create the PVC with the `ocs-storagecluster-cephfs` object class.
- Backup the `compreview` PVC by creating a snapshot named `pg-compreview-snapshot`. Change the `volumesnapshotclass/ocs-storagecluster-rbdplugin-snapclass` deletion policy of the volume snapshot classes to preserve the snapshots. Use the `~/DO370/solutions/comprehensive-review/volumesnapshot.yaml` file as starting point.

1. Log in to the cluster as the `admin` user with password `redhat` and create a project named `comprehensive-review`.

1.1. Log in to the OpenShift cluster as the `admin` user with the `redhat` password.

```
[student@workstation ~]$ oc login -u admin -p redhat \
  https://api.ocp4.example.com:6443
...output omitted...
```

1.2. Create a project named `comprehensive-review`.

```
[student@workstation ~]$ oc new-project comprehensive-review
...output omitted...
```

2. Create a custom storage class named `ocs-storagecluster-ceph-rbd-xfs` that provides a block device that uses the `xfs` file system format. Use the `openshift-storage.rbd.csi.ceph.com` provisioner. Use the `~/DO370/labs/comprehensive-review/xfs-storageclass.yaml` file as starting point.

2.1. Edit the `~/DO370/labs/comprehensive-review/xfs-storageclass.yaml` file to match the following lines.

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ocs-storagecluster-ceph-rbd-xfs
  annotations:
    description: Provides RWO Filesystem volumes, and RWO and RWX Block volumes
 with XFS filesystem
parameters:
  clusterID: openshift-storage
  imageFeatures: layering
```

```
  imageFormat: "2"
  pool: ocs-storagecluster-cephblockpool
  csi.storage.k8s.io/fstype: xfs
  csi.storage.k8s.io/controller-expand-secret-name: rook-csi-rbd-provisioner
  csi.storage.k8s.io/controller-expand-secret-namespace: openshift-storage
  csi.storage.k8s.io/node-stage-secret-name: rook-csi-rbd-node
  csi.storage.k8s.io/node-stage-secret-namespace: openshift-storage
  csi.storage.k8s.io/provisioner-secret-name: rook-csi-rbd-provisioner
  csi.storage.k8s.io/provisioner-secret-namespace: openshift-storage
allowVolumeExpansion: true
provisioner: openshift-storage.rbd.csi.ceph.com
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

The `~/DO370/solutions/comprehensive-review/xfs-storageclass.yaml` file contains the correct configuration and can be used for comparison.

2.2. Create the storage class by using the `oc apply` command with the `~/DO370/labs/comprehensive-review/xfs-storageclass.yaml` file.

```
[student@workstation ~]$ oc apply -f \
  ~/DO370/labs/comprehensive-review/xfs-storageclass.yaml
storageclass.storage.k8s.io/ocs-storagecluster-ceph-rbd-xfs created
```

3. Create a PostgreSQL database deployment named `pg-compreview` by using the `postgresql-persistent-sc` template.

3.1. Use the `postgresql-persistent-sc` template to create the `pg-compreview` deployment with the following settings:

- `STORAGECLASS_NAME=ocs-storagecluster-ceph-rbd-xfs`
- `VOLUME_CAPACITY=150Mi`
- `POSTGRESQL_USER=student`
- `POSTGRESQL_PASSWORD=redhat`
- `POSTGRESQL_DATABASE=compreview`
- `DATABASE_SERVICE_NAME=compreview`

```
[student@workstation ~]$ oc new-app --name=pg-compreview \
  --template=postgresql-persistent-sc \
  -p STORAGECLASS_NAME=ocs-storagecluster-ceph-rbd-xfs \
  -p VOLUME_CAPACITY=150Mi \
  -p POSTGRESQL_USER=student \
  -p POSTGRESQL_PASSWORD=redhat \
  -p POSTGRESQL_DATABASE=compreview \
  -p DATABASE_SERVICE_NAME=compreview
...output omitted...
--> Creating resources ...
    secret "compreview" created
    service "compreview" created
    persistentvolumeclaim "compreview" created
    deploymentconfig.apps.openshift.io "compreview" created
--> Success

...output omitted...
```

4. Create an OBC named `image-object-bucket` that is based on the `ocs-storagecluster-ceph-rgw` storage class. Use the `~/DO370/labs/comprehensive-review/object-bucket-claim.yaml` file as starting point.

4.1. Edit the `~/DO370/labs/comprehensive-review/object-bucket-claim.yaml` file to match the following lines:

```
---
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: image-object-bucket
  namespace: comprehensive-review
  labels:
    app: comprehensive-review
spec:
  generateBucketName: object-bucket
  storageClassName: ocs-storagecluster-ceph-rgw
```

The `~/DO370/solutions/comprehensive-review/object-bucket-claim.yaml` file contains the correct configuration and can be used for comparison.

4.2. Use the `oc apply` command to create the `image-object-bucket` OBC.

```
[student@workstation ~]$ oc apply -f \
  ~/DO370/labs/comprehensive-review/object-bucket-claim.yaml
objectbucketclaim.objectbucket.io/image-object-bucket created
```

5. Create an application that uses object storage based on the `~/DO370/labs/comprehensive-review/object-storage-app.yaml` file. Use the `image-object-bucket` object bucket claim as storage for this deployment.

5.1. Use your favorite editor to edit the `~/DO370/labs/comprehensive-review/object-storage-app.yaml` file and remove the comments from the following lines:

```
...output omitted...
        envFrom:
        - configMapRef:
            name: image-object-bucket
        - secretRef:
            name: image-object-bucket
...output omitted...
```

The `~/DO370/solutions/comprehensive-review/object-storage-app.yaml` file contains the correct configuration and can be used for comparison.

5.2. Create the application deployment by using the `oc apply` command and the `~/DO370/labs/comprehensive-review/object-storage-app.yaml` file.

```
[student@workstation ~]$ oc apply -f \
  ~/DO370/labs/comprehensive-review/object-storage-app.yaml
deployment.apps/image-tool-compreview-object created
service/image-tool-compreview-object created
route.route.openshift.io/image-tool-compreview-object created
```

6. Use the `image-tool-app.yaml` file as a reference to create an application named `compreview-file` that mounts a file storage PVC named `compreview-file-cephfs` under the `/var/storage` directory. Create the PVC by using the `ocs-storagecluster-cephfs` object class.

   6.1. Edit the `~/DO370/labs/comprehensive-review/cephfs-pvc.yaml` file to match the following lines:

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: compreview-file-cephfs
spec:
  accessModes:
  - ReadWriteMany
  storageClassName: ocs-storagecluster-cephfs
  resources:
    requests:
      storage: 1Gi
```

   The `~/DO370/solutions/comprehensive-review/object-storage-app.yaml` file contains the correct configuration and can be used for comparison.

   6.2. Use the `oc apply` command to create the PVC from the file modified in the previous step.

```
[student@workstation ~]$ oc apply -f \
  ~/DO370/labs/comprehensive-review/cephfs-pvc.yaml
persistentvolumeclaim/compreview-file-cephfs created
```

   6.3. Edit the `~/DO370/labs/comprehensive-review/image-tool-app.yaml` file to match the following lines:

```
...output omitted...
        volumeMounts:
        - name: compreview-file-cephfs-vol
          mountPath: "/var/storage"
      volumes:
      - name: compreview-file-cephfs-vol
        persistentVolumeClaim:
          claimName: compreview-file-cephfs
...output omitted...
```

   6.4. Use the `oc apply` command and the `~/DO370/labs/comprehensive-review/image-tool-app.yaml` file to create the `compreview-file` application.

```
[student@workstation ~]$ oc apply -f \
  ~/DO370/labs/comprehensive-review/image-tool-app.yaml
deployment.apps/compreview-file created
```

7. Back up the `compreview` PVC by creating a snapshot named `pg-compreview-snapshot`. Change the `volumesnapshotclass/ocs-storagecluster-rbdplugin-snapclass`

deletion policy of the volume snapshot classes to preserve the snapshots. Use the `~/DO370/solutions/comprehensive-review/volumesnapshot.yaml` file as starting point.

7.1. Use the `oc patch` command to change the deletion policy of the `volumesnapshotclass/ocs-storagecluster-rbdplugin-snapclass` volume snapshot class to preserve the snapshots.

```
[student@workstation ~]$ oc patch \
  volumesnapshotclass/ocs-storagecluster-rbdplugin-snapclass \
  --type merge -p '{"deletionPolicy":"Retain"}'
volumesnapshotclass..../ocs-storagecluster-rbdplugin-snapclass  patched
```

7.2. Edit the `~/DO370/labs/comprehensive-review/volumesnapshot.yaml` file to match the following lines:

```
---
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: pg-compreview-snapshot
spec:
  volumeSnapshotClassName: ocs-storagecluster-rbdplugin-snapclass
  source:
    persistentVolumeClaimName: compreview
```

The `~/DO370/solutions/comprehensive-review/volumesnapshot.yaml` file contains the correct configuration and can be used for comparison.

7.3. Create the `pg-compreview-snapshot` by using the `oc apply` command and the `~/DO370/labs/comprehensive-review/volumesnapshot.yaml` file.

```
[student@workstation ~]$ oc apply -f \
  ~/DO370/labs/comprehensive-review/volumesnapshot.yaml
volumesnapshot.snapshot.storage.k8s.io/pg-compreview-snapshot created
```

Evaluation

As the `student` user on the `workstation` machine, use the `lab` command to grade your work. Correct any reported failures and rerun the command until successful.

```
[student@workstation ~]$ lab grade comprehensive-review
```

Finish

As the `student` user on the `workstation` machine, use the `lab` command to complete this exercise. This is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish comprehensive-review
```

This concludes the lab.