



Red Hat Training and Certification

Student Workbook (ROLE)

Red Hat Ceph Storage 5.0 CL260

Cloud Storage with Red Hat Ceph Storage

Edition 1



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 community members



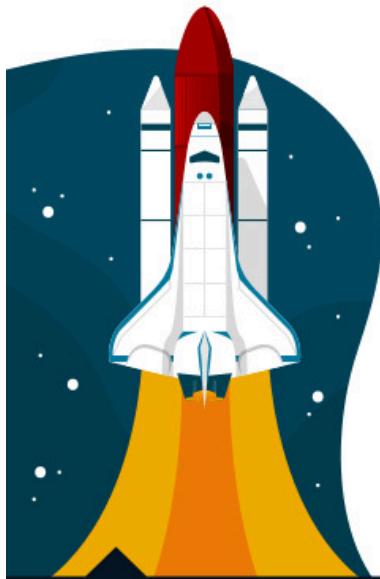
Engage in thousands of active conversations and posts



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



Copyright © 2020 Red Hat, Inc. Red Hat, Red Hat Enterprise Linux, the Red Hat logo, and Ansible are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Cloud Storage with Red Hat Ceph Storage



Red Hat Ceph Storage 5.0 CL260
Cloud Storage with Red Hat Ceph Storage
Edition 1 20211117
Publication date 20211117

Authors: Patrick Gomez, Ed Parenti, Trey Feagle, Mauricio Santacruz,
Bernardo Gargallo
Course Architect: Philip Sweany
DevOps Engineer: Artur Glogowski
Editor: Julian Cable, Sam Ffrench, Nicole Muller

Copyright © 2021 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are
Copyright © 2021 Red Hat, Inc.

No part of this publication may be stored in a retrieval system, transmitted or reproduced in any way, including, but not limited to, photocopy, photograph, magnetic, electronic or other record, without the prior written permission of Red Hat, Inc.

This instructional program, including all material provided herein, is supplied without any guarantees from Red Hat, Inc. Red Hat, Inc. assumes no liability for damages or legal action arising from the use or misuse of contents or details contained herein.

If you believe Red Hat training materials are being used, copied, or otherwise improperly distributed, please send email to training@redhat.com or phone toll-free (USA) +1 (866) 626-2994 or +1 (919) 754-3700.

Red Hat, Red Hat Enterprise Linux, the Red Hat logo, JBoss, OpenShift, Fedora, Hibernate, Ansible, CloudForms, RHCA, RHCE, RHCSA, Ceph, and Gluster are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a registered trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation or the OpenStack community.

All other trademarks are the property of their respective owners.

Contributors: Roberto Velazquez, David Sacco, Sajith Eyamkuzhy Sugathan

Document Conventions	xi
	xi
Introduction	xiii
Cloud Storage with Red Hat Ceph Storage	xiii
Orientation to the Classroom Environment	xiv
Performing Lab Exercises	xviii
1. Introducing Red Hat Ceph Storage Architecture	1
Describing Storage Personas	2
Quiz: Describing Storage Personas	5
Describing Red Hat Ceph Storage Architecture	7
Guided Exercise: Describing Red Hat Ceph Storage Architecture	15
Describing Red Hat Ceph Storage Management Interfaces	20
Guided Exercise: Describing Red Hat Ceph Storage Management Interfaces	25
Summary	31
2. Deploying Red Hat Ceph Storage	33
Deploying Red Hat Ceph Storage	34
Guided Exercise: Deploying Red Hat Ceph Storage	39
Expanding Red Hat Ceph Storage Cluster Capacity	45
Guided Exercise: Expanding Red Hat Ceph Storage Cluster Capacity	49
Lab: Deploying Red Hat Ceph Storage	54
Summary	61
3. Configuring a Red Hat Ceph Storage Cluster	63
Managing Cluster Configuration Settings	64
Guided Exercise: Managing Cluster Configuration Settings	69
Configuring Cluster Monitors	76
Guided Exercise: Configuring Cluster Monitors	80
Configuring Cluster Networking	84
Guided Exercise: Configuring Cluster Networking	88
Lab: Configuring a Red Hat Ceph Storage Cluster	91
Summary	95
4. Creating Object Storage Cluster Components	97
Creating BlueStore OSDs Using Logical Volumes	98
Guided Exercise: Creating BlueStore OSDs Using Logical Volumes	106
Creating and Configuring Pools	114
Guided Exercise: Creating and Configuring Pools	123
Managing Ceph Authentication	131
Guided Exercise: Managing Ceph Authentication	138
Lab: Creating Object Storage Cluster Components	141
Summary	147
5. Creating and Customizing Storage Maps	149
Managing and Customizing the CRUSH Map	150
Guided Exercise: Managing and Customizing the CRUSH Map	161
Managing the OSD Map	170
Guided Exercise: Managing the OSD Map	174
Lab: Creating and Customizing Storage Maps	178
Summary	184
6. Providing Block Storage Using RADOS Block Devices	185
Managing RADOS Block Devices	186
Guided Exercise: Managing RADOS Block Devices	193
Managing RADOS Block Device Snapshots	200
Guided Exercise: Managing RADOS Block Device Snapshots	206

Importing and Exporting RBD Images	211
Guided Exercise: Importing and Exporting RBD Images	214
Lab: Providing Block Storage Using RADOS Block Devices	222
Summary	229
7. Expanding Block Storage Operations	231
Configuring RBD Mirrors	232
Guided Exercise: Configuring RBD Mirrors	238
Providing iSCSI Block Storage	245
Quiz: Providing iSCSI Block Storage	251
Lab: Expanding Block Storage Operations	253
Summary	261
8. Providing Object Storage Using a RADOS Gateway	263
Deploying an Object Storage Gateway	264
Guided Exercise: Deploying an Object Storage Gateway	271
Configuring a Multisite Object Storage Deployment	275
Guided Exercise: Configuring a Multisite Object Storage Deployment	282
Lab: Providing Object Storage Using a RADOS Gateway	290
Summary	294
9. Accessing Object Storage Using a REST API	295
Providing Object Storage Using the Amazon S3 API	296
Guided Exercise: Providing Object Storage Using the Amazon S3 API	303
Providing Object Storage Using the Swift API	306
Guided Exercise: Providing Object Storage Using the Swift API	310
Lab: Accessing Object Storage Using a REST API	314
Summary	320
10. Providing File Storage with CephFS	321
Deploying Shared File Storage	322
Guided Exercise: Deploying Shared File Storage	332
Managing Shared File Storage	339
Guided Exercise: Managing Shared File Storage	345
Lab: Providing File Storage with CephFS	350
Summary	356
11. Managing a Red Hat Ceph Storage Cluster	357
Performing Cluster Administration and Monitoring	358
Guided Exercise: Performing Cluster Administration and Monitoring	373
Performing Cluster Maintenance Operations	380
Guided Exercise: Performing Cluster Maintenance Operations	384
Lab: Managing a Red Hat Ceph Storage Cluster	391
Summary	399
12. Tuning and Troubleshooting Red Hat Ceph Storage	401
Optimizing Red Hat Ceph Storage Performance	402
Guided Exercise: Optimizing Red Hat Ceph Storage Performance	411
Tuning Object Storage Cluster Performance	420
Guided Exercise: Tuning Object Storage Cluster Performance	425
Troubleshooting Clusters and Clients	428
Guided Exercise: Troubleshooting Clusters and Clients	439
Lab: Tuning and Troubleshooting Red Hat Ceph Storage	444
Summary	454
13. Managing Cloud Platforms with Red Hat Ceph Storage	455
Introducing OpenStack Storage Architecture	456
Quiz: Introducing OpenStack Storage Architecture	464

Implementing Storage in OpenStack Components	466
Quiz: Implementing Storage in OpenStack Components	470
Introducing OpenShift Storage Architecture	472
Quiz: Introducing OpenShift Storage Architecture	478
Implementing Storage in OpenShift Components	480
Quiz: Implementing Storage in OpenShift Components	490
Summary	492
14. Comprehensive Review	493
Comprehensive Review	494
Lab: Deploying Red Hat Ceph Storage	497
Lab: Configuring Red Hat Ceph Storage	506
Lab: Deploying CephFS	515
Lab: Deploying and Configuring Block Storage with RBD	523
Lab: Deploying and Configuring RADOS Gateway	535

Document Conventions

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

Admonitions

Red Hat Training courses use the following admonitions:



References

These describe where to find external documentation relevant to a subject.



Note

These 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

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



Warning

These 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 covered in Red Hat Training courses. Red Hat appreciates your patience during this process.

Introduction

Cloud Storage with Red Hat Ceph Storage

Cloud Storage with Red Hat Ceph Storage (CL260) is designed for storage administrators or cloud operators who intend to deploy Red Hat Ceph Storage to their production data center environment or OpenStack installation. Learn how to deploy, manage, and scale out a Ceph storage cluster.

Course Objectives

- Explain the architecture of a Ceph cluster.
- Deploy a Red Hat Ceph Storage cluster using Cephadm.
- Manage operations on a Red Hat Ceph Storage cluster.
- Provide servers with storage from the Ceph cluster using block, object, and file-based access methods.
- Integrate Red Hat Ceph Storage as back-end storage for Red Hat OpenStack Platform.
- Prepare for the Red Hat Certified Specialist in Ceph Storage Architecture and Administration exam.

Audience

- This course is intended for storage administrators, cloud operators, and cloud developers who want to learn how to deploy and manage Red Hat Ceph Storage for use by servers in an enterprise data center or within a Red Hat OpenStack Platform environment.

Prerequisites

- Red Hat Certified System Administrator (RHCSA) in Red Hat Enterprise Linux certification or equivalent Linux system administration skills.
- Some experience with storage administration is recommended but not required.

Orientation to the Classroom Environment

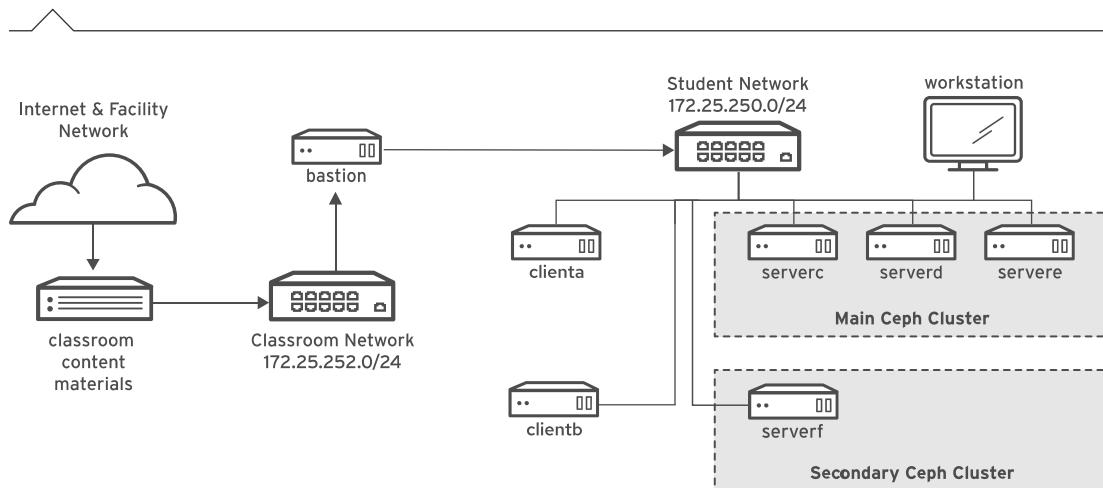


Figure 0.1: Classroom environment

In this classroom environment, your primary system for hands-on activities is **workstation**. The **workstation** virtual machine (VM) is the only system with a graphical desktop, which is required for using a browser to access web-based tools. You should always log in directly to **workstation** first. From **workstation**, use **ssh** for command-line access to all other VMs. Use a web browser from **workstation** to access the Red Hat Ceph Storage web-based dashboard and other graphical tools.

As seen in Figure 0.1, all VMs share an external network, 172.25.250.0/24, with a gateway of 172.25.250.254 (**bastion**). External network DNS and container registry services are provided by **utility**.

Additional student VMs used for hands-on exercises include **clienta**, **clientb**, **serverc**, **serverd**, **servere**, **serverf**, and **serverg**. All ten of these systems are in the `lab.example.com` DNS domain.

All student computer systems have a standard user account, **student**, which has the password **student**. The root password on all student systems is **redhat**.

Classroom Machines

Machine name	IP addresses	Role
<code>workstation.lab.example.com</code>	172.25.250.9	Graphical student workstation
<code>clienta.lab.example.com</code>	172.25.250.10	Client "A" as cluster client and admin node
<code>clientb.lab.example.com</code>	172.25.250.11	Client "B" cluster client
<code>serverc.lab.example.com</code>	172.25.250.12	Server "C" cluster storage node
<code>serverd.lab.example.com</code>	172.25.250.13	Server "D" cluster storage node

Machine name	IP addresses	Role
servere.lab.example.com	172.25.250.14	Server "E" cluster storage node
serverf.lab.example.com	172.25.250.15	Server "F" second cluster storage node
serverg.lab.example.com	172.25.250.16	Server "G" spare cluster storage node
utility.lab.example.com	172.25.250.17	Utility services such as DNS and container registry
classroom.example.com	172.25.254.254	The classroom materials and content server
bastion.lab.example.com	172.25.250.254	Router to link VMs to central servers

The environment uses the `classroom` server as a NAT router to the outside network, and as a file server using the URLs `content.example.com` and `materials.example.com`, serving course content for certain exercises. Information on how to use these servers is provided in the instructions for those activities.

Managing your storage cluster

Your classroom environment comes with two Ceph storage clusters installed. The primary Ceph cluster is composed of `serverc`, `serverd`, and `servere`. The secondary Ceph cluster is composed of `serverf`. The `clienta.lab.example.com` server is installed with the `cephadm` container and it has the required configuration to manage the primary Ceph cluster. The `serverf.lab.example.com` server is installed with the `cephadm` container and it has the required configuration to manage the secondary Ceph cluster.

You can reset your classroom environment to set all of your classroom nodes back to their beginning state when the classroom was first created. Resetting allows you to clean your virtual machines, and start exercises over again. It is also a simple method for clearing a classroom issue which is blocking your progress and is not easily solved. Some chapters, such as chapters 02, 12, and 14, might ask you to reset your classroom to ensure you work on a clean environment. It is highly recommended to follow this instruction.

When your lab environment is first provisioned, and each time you restart the lab environment, the monitor (MON) services might fail to properly initialize and can cause a cluster warning message. On your Red Hat Online Learning cloud platform, this behavior is caused by the random order in which complex network interfaces and services are started. This timing issue does not occur in production Ceph storage cluster environments. To resolve the cluster warning, use the `ceph orch restart mon` command to restart the monitor services, which should then result in a `HEALTH_OK` cluster state.

Controlling Your Systems

You are assigned remote computers in a Red Hat Online Learning classroom. Self-paced courses are accessed through a web application hosted at `rol.redhat.com` [<http://rol.redhat.com>]. If your course is an instructor-led virtual training, you will be provided with your course location URL. 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 web page interface controls. The state of each classroom virtual machine is displayed on the **Lab Environment** tab.

The screenshot shows a web-based interface for managing a lab environment. At the top, there are tabs for 'Table of Contents', 'Course', 'Lab Environment', and icons for star and help. Below the tabs, a section titled '► Lab Controls' contains instructions: 'Click CREATE to build all of the virtual machines needed for the classroom lab environment. This may take several minutes to complete. Once created the environment can then be stopped and restarted to pause your experience.' and 'If you DELETE your lab, you will remove all of the virtual machines in your classroom and lose all of your progress.' Below this are three buttons: 'DELETE' (red), 'STOP' (blue), and a blue button with an info icon. A table lists five virtual machines with their current state, actions, and console access:

Virtual Machine	State	Action	Open Console
bastion	active	ACTION -	OPEN CONSOLE
classroom	active	ACTION -	OPEN CONSOLE
servera	building	ACTION -	OPEN CONSOLE
serverb	building	ACTION -	OPEN CONSOLE
workstation	active	ACTION -	OPEN CONSOLE

Figure 0.2: An example course Lab Environment management page

Machine States

Virtual Machine State	Description
building	The virtual machine is being created.
active	The virtual machine is running and available. If just started, it may still be starting services.
stopped	The virtual machine is completely shut down. Upon starting, the virtual machine boots into the same state as it was before it was shut down. The disk state is preserved.

Classroom Actions

Button or Action	Description
CREATE	Create the ROLE classroom. Creates and starts all of the virtual machines needed for this classroom.
CREATING	The ROLE classroom virtual machines are being created. Creates and starts all of the virtual machines needed for this classroom. Creation can take several minutes to complete.
DELETE	Delete the ROLE classroom. Destroys all virtual machines in the classroom. All work saved on that system's disks is lost.
START	Start all virtual machines in the classroom.

Button or Action	Description
STARTING	All virtual machines in the classroom are starting.
STOP	Stop all virtual machines in the classroom.

Machine Actions

Button or Action	Description
OPEN CONSOLE	Connect to the system console of the virtual machine in a new browser tab. You can log in directly to the virtual machine and run commands, when required. Normally, log in to the workstation virtual machine only, and from there, 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 disk contents.
ACTION → Power Off	Forcefully shut down the virtual machine, while still preserving disk contents. 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. All work saved on that system's disks 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** on every virtual machine in the list.

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



Warning

The **DELETE** operation cannot be undone. All work you have completed in the classroom environment will be lost.

The Auto-stop and Auto-destroy Timers

The Red Hat Online Learning enrollment entitles you to a set allotment of computer time. To help conserve your allotted time, the ROLE classroom uses timers, which shut down or delete the classroom when the appropriate timer expires.

To adjust the timers, locate the two + buttons at the bottom of the course management page. Click the auto-stop + button to add another hour to the auto-stop timer. Click the auto-destroy + button to add another day to the auto-destroy timer. There is a maximum for auto-stop at 11 hours, and a maximum auto-destroy at 14 days. Be careful to keep the timers set while you are working, so as to not have your environment unexpectedly shut down. Be careful not to set the timers unnecessarily high, which could waste your subscription time allotment.

Performing Lab Exercises

Run the `lab` command from `workstation` 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, it 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 to 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 as follows:

```
[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

A script's start logic verifies the required resources to begin an exercise. It might include configuring settings, creating resources, checking prerequisite services, and verifying necessary outcomes from previous exercises. With exercise start logic, you can perform any exercise at any time, even if you did not perform prerequisite exercises.

grade

End-of-chapter labs help to verify what you 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 rerun the grade action.

finish

A script's finish logic deletes exercise resources that are no longer necessary. With cleanup logic, you can repeatedly perform an exercise, and it helps course performance by ensuring that unneeded objects release their resources.

To list the available exercises, use tab completion in the `lab` command with an action:

```
[student@workstation ~]$ lab start Tab Tab
    api-review           cluster-admin      comprehensive-review1
    api-s3               cluster-maint     comprehensive-review2
    api-swift            cluster-review    comprehensive-review3
    block-devices        component-auth   comprehensive-review4
    block-import          component-osd    comprehensive-review5
    block-review          component-pool   configure-monitor
    block-snapshot        component-review  configure-network
...output omitted...
```

Troubleshooting Lab Scripts

The `lab` command displays a list of action steps or grading criteria while it runs, with a PASS or FAIL status for each. If the status is FAIL, the script will display additional information captured from the step's command output. The output displayed, and how useful it might be for troubleshooting, will vary depending on what the step was doing.

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

Grading lab.

· Checking lab systems ..... SUCCESS
· Verify that cluster is running on the hosts ..... SUCCESS
· Verify that cluster health is ok ..... SUCCESS
· Verify that number of OSDs is 11 ..... FAIL
  - '11 osds' not found in command output at 'clienta'
· Verify OSD public_network ..... FAIL
  - '172.25.250.0/24' not found in command output at 'clienta'
· Verify OSD cluster_network ..... SUCCESS

Overall lab grade: FAIL
```

All exercise scripts are stored on the `workstation` system in the folder `/home/student/.venv/labs/lib/python3.6/site-packages/SKU/`, where the SKU is the course code. When you run the `lab` command with a valid action and exercise, it creates an exercise log file in `/tmp/log/labs`, and captures command output and error messages into the file.

```
[student@workstation ~]$ lab start cluster-review
...output omitted...
[student@workstation ~]$ ls -l /tmp/log/labs/
-rw-rw-r-- 1 student student 7520 Nov 02 05:34 cluster-review
```

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 and components by using REST 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.

Updating the Lab Files

Performing maintenance on the `lab` command and script files should only be done on the advice of your instructor or Red Hat support personnel. Incorrect use of the `lab` command or versioning can make your course environment unable to run `lab` commands or for you to perform exercises. List all available `lab` actions by using the `--help` option.

```
[student@workstation ~]$ lab upgrade SKU
...output omitted...
[student@workstation ~]$ lab --help
...output omitted...
```


Chapter 1

Introducing Red Hat Ceph Storage Architecture

Goal

Describe Red Hat Ceph Storage architecture, including data organization, distribution, and client access methods.

Objectives

- Describe the personas in the cloud storage ecosystem that characterize the use cases and tasks taught in this course.
- Describe the Red Hat Ceph Storage architecture, introduce the Object Storage Cluster, and describe the choices in data access methods.
- Describe and compare the use cases for the various management interfaces provided for Red Hat Ceph Storage.

Sections

- Describing Storage Personas (and Quiz)
- Describing Red Hat Ceph Storage Architecture (and Guided Exercise)
- Describing Red Hat Ceph Storage Management Interfaces (and Guided Exercise)

Describing Storage Personas

Objectives

After completing this section, you should be able to describe the personas in the cloud storage ecosystem that characterize the use cases and tasks taught in this course.

Introducing Cloud Storage Personas

Personas are user definitions that are created to represent user types in cloud storage environments. Personas help you to understand user scenarios and goals by researching trends and use cases with Red Hat Ceph Storage organizations. Red Hat uses personas to focus training on relevant user tasks and behavior, not on features and tools.

Information from real Red Hat Ceph Storage users builds the various personas. Personas might describe multiple job titles, and your organization's job titles might map to multiple personas. The personas that are presented here embody the most common roles of Red Hat Ceph Storage users. Roles might change depending on your organization's size and user ecosystem. This course uses the *storage administrator* persona to define Red Hat Ceph Storage operations and use cases.

Cloud Storage Personas in This Course

This course focuses primarily on the roles and responsibilities of the storage administrator persona and introduces the storage operator persona. The storage administrator also supports and communicates with other cloud personas to implement cloud storage.

The Storage Administrator as the Primary Persona

The primary persona for this course is the storage administrator. A Ceph storage administrator performs the following tasks:

- Installs, configures, and maintains a Ceph storage cluster.
- Educates infrastructure architects about Ceph capabilities and features.
- Informs users about Ceph data presentation and methods, as choices for their data applications.
- Provides resilience and recovery, such as replication, backup, and disaster recovery methods.
- Automates and integrates through Infrastructure as Code.
- Provides access for data analytics and advanced mass data mining.

This course includes material to cover Ceph storage cluster deployment and configuration, and incorporates automation techniques whenever appropriate.

The Storage Operator as the Secondary Persona

A storage operator assists in the daily operations of the storage cluster, and is commonly less experienced than a storage administrator. Storage operators primarily use the Ceph Dashboard GUI to view and respond to cluster alerts and statistics. They also perform routine storage administration tasks that are defined as Dashboard workflows, such as replacing a failed storage device.

Other Storage-related Personas

Other personas that use Ceph directly include *application developers*, *project managers*, and *service administrators* with data processing, data warehouse, big data, and similar application needs. The storage administrator frequently communicates with these personas. Defining these relationships helps to illustrate the context for the storage administrator's duties, and clarifies the task limits or boundaries of the storage administrator's scope.

Cloud Operator

A cloud operator administers cloud resources at their organization, such as OpenStack or OpenShift infrastructures. The storage administrator works closely with a cloud operator to maintain the Ceph cluster that is configured to provide storage for those platforms.

Automation Engineer

Automation engineers frequently use Ceph directly. An automation engineer is responsible for creating playbooks for commonly repeated tasks. All user interface and end-user discussions in this course are examples of actions that might be automated. Storage administrators would be familiar with these same actions because they are typically the foremost Ceph subject matter experts.

Application Developer (DevOps Developer)

An application developer can be an original coder, maintainer, or other cloud user who is responsible for the correct deployment and behavior of an application. A storage administrator coordinates with the application developer to ensure that storage resources are available, sets quotas, and secures the application storage.

Service Administrator

Service administrators manage end-user services (as distinct from operating system services). Service administrators have a similar role to project managers, but for an existing production service offering.

Deployment Engineer (DevOps Engineer)

In larger environments, dedicated personnel perform, manage, and tune application deployments, working with the storage administrator and the application developer.

Application Architect

A storage administrator relies on the application architect as a subject matter expert who can correlate between Ceph infrastructure layout and resource availability, scaling, and latency. This architecture expertise helps the storage administrator to design complex application deployments effectively. To support the cloud users and their applications, a storage administrator must comprehend those same aspects of resource availability, scaling, and latency.

Infrastructure Architect

A storage administrator must master the storage cluster's architectural layout to manage resource location, capacity, and latency. The infrastructure architect for the Ceph cluster deployment and maintenance is a primary source of information for the storage administrator. The infrastructure architect might be a cloud service provider employee or a vendor solutions architect or consultant.

Data Center Operator

Personas at the lower Ceph storage infrastructure layer support data provisioning. Data center operators are typically employed by the public cloud service provider or the organization's internal IT group in a private data center cloud. The storage administrator opens service tickets with the relevant public cloud service provider or internal IT group.

Roles Vary by Organization

Because staffing, skills, security, and sizing differ between organizations, personas and cloud roles are often implemented differently. Although personas sometimes match individuals, users commonly assume more than one persona depending on their workplace responsibilities.

- At telecommunications service providers (telcos) and cloud service providers, the prevalent personas are the cloud and storage operators, infrastructure architects, and cloud service developers. Their customers request support with the provider's service ticketing system. Commonly, customers only consume storage services and do not maintain them.
- At organizations that require a secure, private, dedicated infrastructure, such as in banking and finance, all roles are internally staffed. The storage administrator, cloud operator, and infrastructure architect personas act as service providers and support all other personas.
- At universities and smaller implementations, technical support personnel can potentially assume all roles. A single individual might handle the storage administrator, infrastructure architect, and cloud operator personas.



References

The Storage Administrator Role Is Evolving: Meet the Cloud Administrator

<https://cloud.netapp.com/blog/meet-the-cloud-administrator>

Data Storage Security: What Storage Professionals Need to Know

<https://www.enterprisestorageforum.com/storage-management/data-storage-security-guide.html>

Storage Survey Highlights Challenges of Restricted Hiring Trends

<https://www.enterprisestorageforum.com/storage-management/storage-survey-highlights-challenges-of-restricted-hiring-trends.html>

Survey Reveals Tech Trends Reshaping Data Storage

<https://www.enterprisestorageforum.com/storage-management/survey-reveals-tech-trends-reshaping-data-storage.html>

► Quiz

Describing Storage Personas

Match the items below to their counterparts in the table.

Application architect

Automation engineer

Cloud operator

Infrastructure architect

Service administrator

Storage administrator

Storage operator

Responsibility	Persona
Designs distributed cloud deployment capacities and configurations.	
Manages end-user services.	
Configures cloud services at the infrastructure level.	
Installs, configures, and maintains Ceph storage clusters.	
Designs cloud applications based on the comprehension of modern cloud protocols and configurations.	
Implements solutions designed as effortless application deployment and scaling.	
Manages cluster operations by using the Ceph Dashboard GUI.	

► Solution

Describing Storage Personas

Match the items below to their counterparts in the table.

Responsibility	Persona
Designs distributed cloud deployment capacities and configurations.	Infrastructure architect
Manages end-user services.	Service administrator
Configures cloud services at the infrastructure level.	Cloud operator
Installs, configures, and maintains Ceph storage clusters.	Storage administrator
Designs cloud applications based on the comprehension of modern cloud protocols and configurations.	Application architect
Implements solutions designed as effortless application deployment and scaling.	Automation engineer
Manages cluster operations by using the Ceph Dashboard GUI.	Storage operator

Describing Red Hat Ceph Storage Architecture

Objectives

After completing this section, you should be able to describe the Red Hat Ceph Storage architecture, introduce the Object Storage Cluster, and describe the choices in data access methods.

Introducing the Ceph Cluster Architecture

Red Hat Ceph Storage is a distributed data object store. It is an enterprise-ready, software-defined storage solution that scales to thousands of clients who access exabytes of data and beyond. Ceph is designed to provide excellent performance, reliability, and scalability.

Ceph has a modular and distributed architecture that contains the following elements:

- An object storage back end that is known as RADOS (Reliable Autonomic Distributed Object Store)
- Various access methods to interact with RADOS

RADOS is a self-healing and self-managing software-based object store.

Ceph Storage Back-end Components

The Red Hat Ceph Storage cluster has the following daemons:

- *Monitors (MONs)* maintain maps of the cluster state. They help the other daemons to coordinate with each other.
- *Object Storage Devices (OSDs)* store data and handle data replication, recovery, and rebalancing.
- *Managers (MGRs)* track runtime metrics and expose cluster information through a browser-based dashboard and REST API.
- *Metadata Servers (MDSes)* store metadata that CephFS uses (but not object storage or block storage) so that clients can run POSIX commands efficiently.

These daemons can scale to meet the requirements of a deployed storage cluster.

Ceph Monitors

Ceph Monitors (MONs) are the daemons that maintain the cluster map. The cluster map is a collection of five maps that contain information about the state of the cluster and its configuration. Ceph must handle each cluster event, update the appropriate map, and replicate the updated map to each MON daemon.

To apply updates, the MONs must establish a consensus on the state of the cluster. A majority of the configured monitors must be available and agree on the map update. Configure your Ceph clusters with an odd number of monitors to ensure that the monitors can establish a quorum when they vote on the state of the cluster. More than half of the configured monitors must be functional for the Ceph storage cluster to be operational and accessible.

Ceph Object Storage Devices

Ceph Object Storage Devices (OSDs) are the building blocks of a Ceph storage cluster. OSDs connect a storage device (such as a hard disk or other block device) to the Ceph storage cluster. An individual storage server can run multiple OSD daemons and provide multiple OSDs to the cluster. Red Hat Ceph Storage 5 supports a feature called BlueStore to store data within RADOS. BlueStore uses the local storage devices in raw mode and is designed for high performance.

One design goal for OSD operation is to bring computing power as close as possible to the physical data so that the cluster can perform at peak efficiency. Both Ceph clients and OSD daemons use the Controlled Replication Under Scalable Hashing (CRUSH) algorithm to efficiently compute information about object location, instead of depending on a central lookup table.

CRUSH Map

CRUSH assigns every object to a Placement Group (PG), which is a single hash bucket. PGs are an abstraction layer between the objects (application layer) and the OSDs (physical layer). CRUSH uses a pseudo-random placement algorithm to distribute the objects across the PGs and uses rules to determine the mapping of the PGs to the OSDs. In the event of failure, Ceph remaps the PGs to different physical devices (OSDs) and synchronizes their content to match the configured data protection rules.

Primary and Secondary OSDs

One OSD is the *primary OSD* for the object's placement group, and Ceph clients always contact the primary OSD in the acting set when it reads or writes data. Other OSDs are *secondary OSDs* and play an important role in ensuring the resilience of data in the event of failures in the cluster.

Primary OSD functions:

- Serve all I/O requests
- Replicate and protect the data
- Check data coherence
- Rebalance the data
- Recover the data

Secondary OSD functions:

- Act always under control of the primary OSD
- Can become the primary OSD



Warning

A host that runs OSDs must not mount Ceph RBD images or CephFS file systems by using the kernel-based client. Mounted resources can become unresponsive due to memory deadlocks or blocked I/O that is pending on stale sessions.

Ceph Managers

Ceph Managers (MGRs) provide for the collection of cluster statistics.

If no MGRs are available in a cluster, client I/O operations are not negatively affected, but attempts to query cluster statistics fail. To avoid this scenario, Red Hat recommends that you deploy at least two Ceph MGRs for each cluster, each running in a separate failure domain.

The MGR daemon centralizes access to all data that is collected from the cluster and provides a simple web dashboard to storage administrators. The MGR daemon can also export status information to an external monitoring server, such as Zabbix.

Metadata Server

The Ceph Metadata Server (MDS) manages Ceph File System (CephFS) metadata. It provides POSIX-compliant, shared file-system metadata management, including ownership, time stamps, and mode. The MDS uses RADOS instead of local storage to store its metadata. It has no access to file contents.

The MDS enables CephFS to interact with the Ceph Object Store, mapping an inode to an object and the location where Ceph stored the data within a tree. Clients who access a CephFS file system first send a request to an MDS, which provides the needed information to get file content from the correct OSDs.

Cluster Map

Ceph clients and OSDs require knowledge of the cluster topology. Five maps represent the cluster topology, which is collectively referred to as the *cluster map*. The Ceph Monitor daemon maintains the cluster map. A cluster of Ceph MONs ensures high availability if a monitor daemon fails.

- The *Monitor Map* contains the cluster's `fsid`; the position, name, address, and port of each monitor; and map time stamps. The `fsid` is a unique, auto-generated identifier (UUID) that identifies the Ceph cluster. View the Monitor Map with the `ceph mon dump` command.
- The *OSD Map* contains the cluster's `fsid`, a list of pools, replica sizes, placement group numbers, a list of OSDs and their status, and map time stamps. View the OSD Map with the `ceph osd dump` command.
- The *Placement Group (PG) Map* contains the PG version; the full ratios; details on each placement group such as the PG ID, the Up Set, the Acting Set, the state of the PG, data usage statistics for each pool; and map time stamps. View the PG Map statistics with the `ceph pg dump` command.
- The *CRUSH Map* contains a list of storage devices, the failure domain hierarchy (such as device, host, rack, row, room), and rules for traversing the hierarchy when storing data. View the CRUSH map with the `ceph osd crush dump` command.
- The *Metadata Server (MDS) Map* contains the pool for storing metadata, a list of metadata servers, metadata servers status, and map time stamps. View the MDS Map with the `ceph fs dump` command.

Ceph Access Methods

Ceph provides the following methods for accessing a Ceph cluster:

- Ceph Native API (`librados`)
- Ceph Block Device (RBD, `librbd`), also known as a RADOS Block Device (RBD) image
- Ceph Object Gateway
- Ceph File System (CephFS, `libcephfs`)

The following diagram depicts the four data access methods of a Ceph cluster, the libraries that support the access methods, and the underlying Ceph components that manage and store the data.

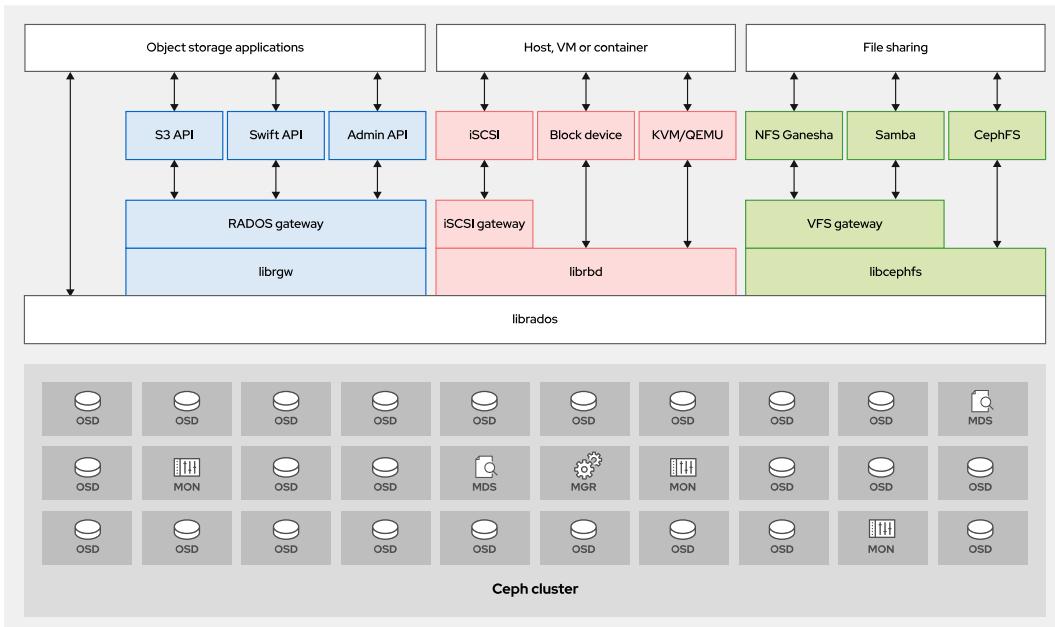


Figure 1.1: Ceph components

Ceph Native API (`librados`)

The foundational library that implements the other Ceph interfaces, such as Ceph Block Device and Ceph Object Gateway, is `librados`. The `librados` library is a native C library that allows applications to work directly with RADOS to access objects stored by the Ceph cluster. Similar libraries are available for C++, Java, Python, Ruby, Erlang, and PHP.

To maximize performance, write your applications to work directly with `librados`. This method gives the best results to improve storage performance in a Ceph environment. For easier Ceph storage access, instead use the higher-level access methods that are provided, such as the RADOS Block Devices, Ceph Object Gateway (RADOSGW), and CephFS.

RADOS Block Device

A Ceph Block Device (RADOS Block Device or RBD) provides block storage within a Ceph cluster through RBD images. Ceph scatters the individual objects that compose RBD images across different OSDs in the cluster. Because the objects that make up the RBD are on different OSDs, access to the block device is automatically parallelized.

RBD provides the following features:

- Storage for virtual disks in the Ceph cluster
- Mount support in the Linux kernel
- Boot support in QEMU, KVM, and OpenStack Cinder

Ceph Object Gateway (RADOS Gateway)

Ceph Object Gateway (RADOS Gateway, RADOSGW, or RGW) is an object storage interface that is built with `librados`. It uses this library to communicate with the Ceph cluster and writes to OSD processes directly. It provides applications with a gateway with a RESTful API, and supports two interfaces: Amazon S3 and OpenStack Swift.

**Note**

You need to update the endpoint only to port existing applications that use the Amazon S3 API or the OpenStack Swift API.

The Ceph Object Gateway offers scalability support by not limiting the number of deployed gateways and by providing support for standard HTTP load balancers. The Ceph Object Gateway includes the following use cases:

- Image storage (for example, SmugMug, Tumblr)
- Backup services
- File storage and sharing (for example, Dropbox)

Ceph File System (CephFS)

Ceph File System (CephFS) is a parallel file system that provides a scalable, single-hierarchy shared disk. Red Hat Ceph Storage provides production environment support for CephFS, including support for snapshots.

The Ceph Metadata Server (MDS) manages the metadata that is associated with files stored in CephFS, including file access, change, and modification time stamps.

Ceph Client Components

Cloud-aware applications need a simple object storage interface with asynchronous communication capability. The Red Hat Ceph Storage Cluster provides such an interface. Clients have direct, parallel access to objects and access throughout the cluster, including:

- Pool Operations
- Snapshots
- Read/Write Objects
 - Create or Remove
 - Entire Object or Byte Range
 - Append or Truncate
- Create/Set/Get/Remove XATTRs
- Create/Set/Get/Remove Key/Value Pairs
- Compound operations and dual-ack semantics

The `object map` tracks the presence of backing RADOS objects when a client writes to an RBD image. When a write occurs, it is translated to an offset within a backing RADOS object. When the `object map` feature is enabled, the presence of RADOS objects is tracked to signify that the objects exist. The `object map` is kept in-memory on the librbd client to avoid querying the OSDs for objects that do not exist.

The `object map` is beneficial for certain operations, such as:

- Resize
- Export
- Copy
- Flatten
- Delete
- Read

Storage devices have throughput limitations, which impact performance and scalability. Storage systems often support **striping**, which is storing sequential pieces of information across multiple

storage devices, to increase throughput and performance. Ceph clients can use data striping to increase performance when writing data to the cluster.

Data Distribution and Organization in Ceph

This section describes the mechanisms that Ceph uses to distribute and organize data across the various storage devices in a cluster.

Partitioning Storage with Pools

Ceph OSDs protect and constantly check the integrity of the data that is stored in the cluster. *Pools* are logical partitions of the Ceph storage cluster that are used to store objects under a common name tag. Ceph assigns each pool a specific number of hash buckets, called *Placement Groups (PGs)*, to group objects for storage.

Each pool has the following adjustable properties:

- Immutable ID
- Name
- Number of PGs to distribute the objects across the OSDs
- CRUSH rule to determine the mapping of the PGs for this pool
- Protection type (replicated or erasure coding)
- Parameters that are associated with the protection type
- Various flags to influence the cluster behavior

Configure the number of placement groups that are assigned to each pool independently to fit the type of data and the required access for the pool.

The CRUSH algorithm determines the OSDs that host the data for a pool. Each pool has a single CRUSH rule that is assigned as its placement strategy. The CRUSH rule determines which OSDs store the data for all the pools that are assigned that rule.

Placement Groups

A *Placement Group (PG)* aggregates a series of objects into a hash bucket, or group. Ceph maps each PG to a set of OSDs. An object belongs to a single PG, and all objects that belong to the same PG return the same hash result.

The CRUSH algorithm maps an object to its PG based on the hashing of the object's name. The placement strategy is also called the CRUSH placement rule. The placement rule identifies the failure domain to choose within the CRUSH topology to receive each replica or erasure code chunk.

When a client writes an object to a pool, it uses the pool's CRUSH placement rule to determine the object's placement group. The client then uses its copy of the cluster map, the placement group, and the CRUSH placement rule to calculate which OSDs to write a copy of the object to (or its erasure-coded chunks).

The layer of indirection that the placement group provides is important when new OSDs become available to the Ceph cluster. When OSDs are added to or removed from a cluster, placement groups are automatically rebalanced between operational OSDs.

Mapping an Object to Its Associated OSDs

A Ceph client gets the latest copy of the cluster map from a MON. The cluster map provides information to the client about all the MONs, OSDs, and MDSs in the cluster. It does not provide

the client with the location of the objects; the client must use CRUSH to compute the locations of objects that it needs to access.

To calculate the Placement Group ID (PG ID) for an object, the Ceph client needs the object ID and the name of the object's storage pool. The client calculates the PG ID, which is the hash of the object ID modulo the number of PGs. It then looks up the numeric ID for the pool, based on the pool's name, and prepends the pool ID to the PG ID.

The CRUSH algorithm is then used to determine which OSDs are responsible for a placement group (the *Acting Set*). The OSDs in the Acting Set that are up are in the *Up Set*. The first OSD in the Up Set is the current primary OSD for the object's placement group, and all other OSDs in the Up Set are secondary OSDs.

The Ceph client can then directly work with the primary OSD to access the object.

Data Protection

Like Ceph clients, OSD daemons use the CRUSH algorithm, but the OSD daemon uses it to compute where to store the object replicas and for rebalancing storage. In a typical write scenario, a Ceph client uses the CRUSH algorithm to compute where to store the original object. The client maps the object to a pool and placement group and then uses the CRUSH map to identify the primary OSD for the mapped placement group. When creating pools, set them as either replicated or erasure coded pools. Red Hat Ceph Storage 5 supports erasure coded pools for Ceph RBD and CephFS.

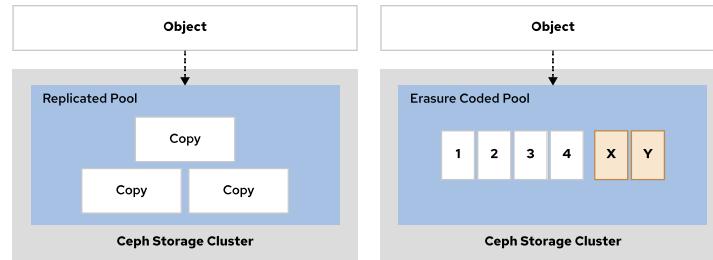


Figure 1.2: Ceph pool data protection methods

For resilience, configure pools with the number of OSDs that can fail without losing data. For a replicated pool, which is the default pool type, the number determines the number of copies of an object to create and distribute across different devices. Replicated pools provide better performance than erasure coded pools in almost all cases at the cost of a lower usable-to-raw storage ratio.

Erasure coding provides a more cost-efficient way to store data but with lower performance. For an erasure coded pool, the configuration values determine the number of coding chunks and parity blocks to create.

A primary advantage of erasure coding is its ability to offer extreme resilience and durability. You can configure the number of coding chunks (parities) to use.

The following figure illustrates how data objects are stored in a Ceph cluster. Ceph maps one or more objects in a pool to a single PG, represented by the colored boxes. Each of the PGs in this figure is replicated and stored on separate OSDs within the Ceph cluster.

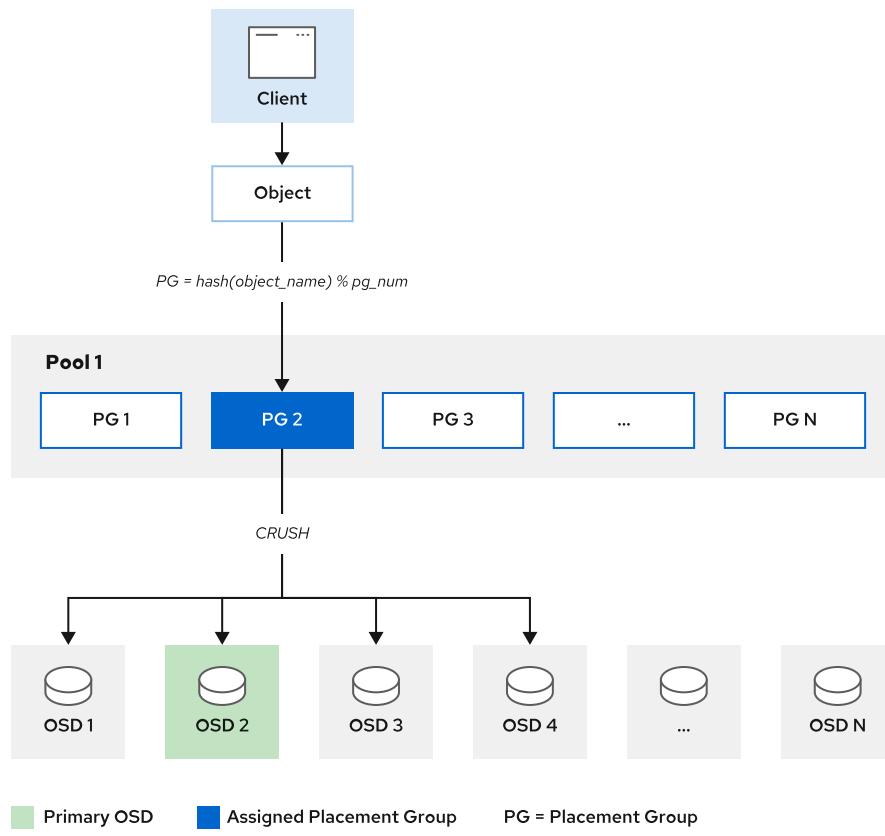


Figure 1.3: Objects in a Ceph pool stored in placement groups



References

For more information, refer to the *Red Hat Ceph Storage 5 Architecture Guide* at https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/architecture_guide

For more information, refer to the *Red Hat Ceph Storage 5 Hardware Guide* at https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/hardware_guide

For more information, refer to the *Red Hat Ceph Storage 5 Storage Strategies Guide* at https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/storage_strategies_guide

► Guided Exercise

Describing Red Hat Ceph Storage Architecture

In this exercise, you navigate and query the Red Hat Ceph Storage cluster.

Outcomes

You should be able to navigate and work with services within the Ceph cluster.

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 intro-arch
```

This command confirms that the Ceph cluster in the classroom is operating.

Instructions

- 1. Log in to the admin node, `clienta`, and view Ceph services.

- 1.1. Log in to `clienta` as the `admin` user and switch to the `root` user.

```
[student@workstation ~]$ ssh admin@clienta
...output omitted...
[admin@clienta ~]$ sudo -i
[root@clienta ~]#
```

- 1.2. Use the `ceph orch ls` command within the `cephadm` shell to view the running services.

```
[root@clienta ~]# cephadm shell -- ceph orch ls
Inferring fsid 2ae6d05a-229a-11ec-925e-52540000fa0c
Inferring config /var/lib/ceph/2ae6d05a-229a-11ec-925e-52540000fa0c/mon.clienta/
config
Using recent ceph image...
NAME          RUNNING  REFRESHED  AGE  PLACEMENT
alertmanager   1/1     8m ago    2d   count:1
crash          4/4     8m ago    2d   *
grafana        1/1     8m ago    2d   count:1
mgr            4/4     8m ago    2d
clienta.lab.example.com;serverc.lab.example.com;serverd.lab.example.com;servere.lab.example.com
mon            4/4     8m ago    2d
clienta.lab.example.com;serverc.lab.example.com;serverd.lab.example.com;servere.lab.example.com
node-exporter   4/4     8m ago    2d   *
```

```
osd.default_drive_group    9/12 8m ago   2d  server*
prometheus                 1/1  8m ago   2d  count:1
rgw.realm.zone             2/2  8m ago   2d
serverc.lab.example.com;serverd.lab.example.com
```

- 1.3. Use the `cephadm shell` command to launch the shell, and then use the `ceph orch ps` command to view the status of all cluster daemons.

```
[root@clienta ~]# cephadm shell
...output omitted...
[ceph: root@clienta /]# ceph orch ps
NAME                      HOST          STATUS
REFRESHED    AGE    PORTS      VERSION      IMAGE ID      CONTAINER ID
alertmanager.serverc        serverc.lab.example.com  running (43m)  2m ago
    53m *:9093 *:9094  0.20.0        4c997545e699  95767d5df632
crash.serverc               serverc.lab.example.com  running (43m)  2m ago
    53m -           16.2.0-117.el8cp  2142b60d7974  0f19ee9f42fa
crash.serverc               serverc.lab.example.com  running (52m)  7m ago
    52m -           16.2.0-117.el8cp  2142b60d7974  036bafc0145c
crash.serverd               serverd.lab.example.com  running (52m)  7m ago
    52m -           16.2.0-117.el8cp  2142b60d7974  2e112369ca35
crash.servere               servere.lab.example.com  running (51m)  2m ago
    51m -           16.2.0-117.el8cp  2142b60d7974  3a2b9161c49e
grafana.serverc              serverc.lab.example.com  running (43m)  2m ago
    53m *:3000       6.7.4          09cf77100f6a  ff674835c5fc
mgr.serverc.lab.example.com.ccjsrd serverc.lab.example.com  running (43m)  2m ago
    54m *:9283       16.2.0-117.el8cp  2142b60d7974  449c4ba94638
mgr.serverc.lvsxza            serverc.lab.example.com  running (51m)  7m ago
    51m *:8443 *:9283  16.2.0-117.el8cp  2142b60d7974  855376edd5f8
mon.serverc.lab.example.com            serverc.lab.example.com  running (43m)  2m ago
    55m -           16.2.0-117.el8cp  2142b60d7974  3e1763669c29
mon.serverc                  serverc.lab.example.com  running (51m)  7m ago
    51m -           16.2.0-117.el8cp  2142b60d7974  d56f57a637a8
...output omitted...
```

▶ 2. View the cluster health.

- 2.1. Use the `ceph health` command to view the health of your Ceph cluster.

```
[ceph: root@clienta /]# ceph health
HEALTH_OK
```



Note

If the reported cluster status is not `HEALTH_OK`, the `ceph health detail` command shows further information about the cause of the health alert.

- 2.2. Use the `ceph status` command to view the full cluster status.

```
[ceph: root@clienta /]# ceph status
cluster:
  id: 2ae6d05a-229a-11ec-925e-52540000fa0c
  health: HEALTH_OK
```

```

services:
  mon: 4 daemons, quorum serverc.lab.example.com,clienta,serverd,servere (age
    10m)
    mgr: serverc.lab.example.com.aiqepd(active, since 19m), standbys:
      clienta.nncugs, serverd.klrkci, servere.kjwyko
    osd: 9 osds: 9 up (since 19m), 9 in (since 2d)
    rgw: 2 daemons active (2 hosts, 1 zones)

data:
  pools:   5 pools, 105 pgs
  objects: 221 objects, 4.9 KiB
  usage:   156 MiB used, 90 GiB / 90 GiB avail
  pgs:     105 active+clean

```

► 3. Explore the Ceph cluster by viewing the cluster components.

3.1. Use the `ceph mon dump` command to view the cluster MON map.

```
[ceph: root@clienta /]# ceph mon dump
epoch 4
fsid 2ae6d05a-229a-11ec-925e-52540000fa0c
last_changed 2021-10-01T09:33:53.880442+0000
created 2021-10-01T09:30:30.146231+0000
min_mon_release 16 (pacific)
election_strategy: 1
0: [v2:172.25.250.12:3300/0,v1:172.25.250.12:6789/0] mon.serverc.lab.example.com
1: [v2:172.25.250.10:3300/0,v1:172.25.250.10:6789/0] mon.clienta
2: [v2:172.25.250.13:3300/0,v1:172.25.250.13:6789/0] mon.serverd
3: [v2:172.25.250.14:3300/0,v1:172.25.250.14:6789/0] mon.servere
dumped monmap epoch 4
```

3.2. Use the `ceph mgr stat` command to view the cluster MGR status.

```
[ceph: root@clienta /]# ceph mgr stat
{
  "epoch": 32,
  "available": true,
  "active_name": "serverc.lab.example.com.aiqepd",
  "num_standby": 3
}
```

3.3. Use the `ceph osd pool ls` command to view the cluster pools.

```
[ceph: root@clienta /]# ceph osd pool ls
device_health_metrics
.rgw.root
default.rgw.log
default.rgw.control
default.rgw.meta
```

3.4. Use the `ceph pg stat` command to view Placement Group (PG) status.

```
[ceph: root@clienta /]# ceph pg stat
105 pgs: 105 active+clean; 4.9 KiB data, 162 MiB used, 90 GiB / 90 GiB avail
```

3.5. Use the `ceph osd status` command to view the status of all OSDs.

<code>[ceph: root@clienta /]# ceph osd status</code>												
ID	HOST	USED	AVAIL	WR OPS	WR DATA	RD OPS	RD DATA	STATE				
0	serverc.lab.example.com	12.5M	9.98G	0	0	0	0	exists,up				
1	serverc.lab.example.com	13.8M	9.98G	0	0	0	0	exists,up				
2	serverc.lab.example.com	19.0M	9.97G	0	0	0	0	exists,up				
3	serverd.lab.example.com	16.8M	9.97G	0	0	0	0	exists,up				
4	servere.lab.example.com	23.8M	9.97G	0	0	0	0	exists,up				
5	serverd.lab.example.com	24.0M	9.97G	0	0	0	0	exists,up				
6	servere.lab.example.com	12.1M	9.98G	0	0	1	0	exists,up				
7	serverd.lab.example.com	15.6M	9.98G	0	0	0	0	exists,up				
8	servere.lab.example.com	23.7M	9.97G	0	0	0	0	exists,up				

3.6. Use the `ceph osd crush tree` command to view the cluster CRUSH hierarchy.

<code>[ceph: root@clienta /]# ceph osd crush tree</code>			
ID	CLASS	WEIGHT	TYPE NAME
-1		0.08817	root default
-3		0.02939	host serverc
0	hdd	0.00980	osd.0
1	hdd	0.00980	osd.1
2	hdd	0.00980	osd.2
-5		0.02939	host serverd
3	hdd	0.00980	osd.3
5	hdd	0.00980	osd.5
7	hdd	0.00980	osd.7
-7		0.02939	host servere
4	hdd	0.00980	osd.4
6	hdd	0.00980	osd.6
8	hdd	0.00980	osd.8

3.7. Return to `workstation` as the `student` user.

```
[ceph: root@clienta /]# exit
[root@clienta ~]# exit
[admin@clienta ~]$ exit
[student@workstation ~]$
```

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 intro-arch
```

This concludes the guided exercise.

Describing Red Hat Ceph Storage Management Interfaces

Objectives

After completing this section, you should be able to describe and compare the use cases for the various management interfaces provided for Red Hat Ceph Storage.

Introducing Ceph Interfaces

Previous Ceph versions used the Ansible Playbooks from the `ceph-ansible` package to deploy and manage the cluster. Red Hat Ceph Storage 5 introduces `cephadm` as the tool to manage the whole lifecycle of the cluster (deployment, management, and monitoring), replacing the previous functions that `ceph-ansible` provided.

`Cephadm` is implemented as a module in the Manager daemon (MGR), which is the first daemon that starts when deploying a new cluster. The Ceph cluster core integrates all the management tasks, and `Cephadm` is ready to use when the cluster starts.

`Cephadm` is provided by the `cephadm` package. You should install this package in the first cluster node, which acts as the bootstrap node. As Ceph 5 is deployed in the containerized version, the only package requirements to have a Ceph cluster up and running are `cephadm`, `podman`, `python3`, and `chrony`. This containerized version reduces the complexity and package dependencies to deploy a Ceph cluster.

The following diagram illustrates how `Cephadm` interacts with the other services.

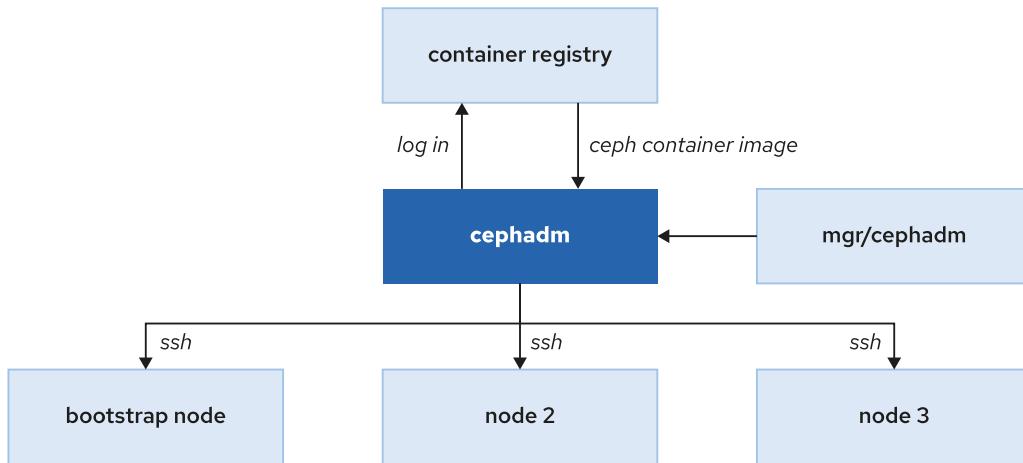


Figure 1.4: Cephadm interaction with other services

`Cephadm` can log in to the container registry to pull a Ceph image and deploy services on the nodes that use that image. This Ceph container image is necessary when bootstrapping the cluster, because the deployed Ceph containers are based on that image.

To interact with the Ceph cluster nodes, Cephadm uses SSH connections. By using these SSH connections, Cephadm can add new hosts to the cluster, add storage, or monitor these hosts.

Exploring Ceph Management Interfaces

Ceph is deployed in a containerized version, and no extra software is necessary in bootstrap node. You can bootstrap the cluster from the command-line interface in the bootstrap node of the cluster. Bootstrapping the cluster sets up a minimal cluster configuration with only one host (the bootstrap node) and two daemons (the monitor and manager daemons). You interact with the cluster for maintenance and scaling operations, such as adding more cluster hosts or storage. Red Hat Ceph Storage 5 provides two interfaces: the Ceph CLI and the Dashboard GUI. Both interfaces are deployed by default when you bootstrap the cluster.

The Ceph Orchestrator

You can use the *Ceph orchestrator* to add hosts and daemons to the cluster easily. Use the orchestrator to provision Ceph daemons and services and to expand or contract the cluster. To use the Ceph orchestrator via the command-line interface (CLI), use the `ceph orch` command. You can also use the Red Hat Ceph Storage Dashboard interface to run orchestrator tasks. The `cephadm` script interacts with the Ceph Manager orchestration module.

The following diagram illustrates the Ceph Orchestrator.

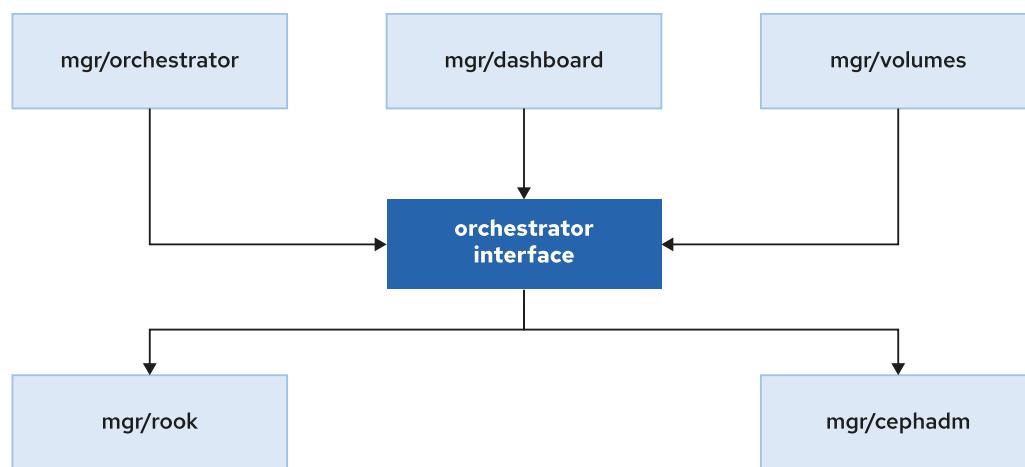


Figure 1.5: The Ceph Orchestrator

The Ceph Command-line Interface

Cephadm can launch a containerized version of the shell with all the required Ceph packages installed. The command to run this containerized shell is `cephadm shell`. You should run this command only in the bootstrap node, because only this node can access the admin key ring in `/etc/ceph` when bootstrapping the cluster.

```
[root@node ~]# cephadm shell
Inferring fsid 6016ad68-0be8-11ec-9a05-52540000fa0c
Inferring config /var/lib/ceph/6016ad68-0be8-11ec-9a05-52540000fa0c/
mon.node.example.com/config
Using recent ceph image registry.redhat.io/rhceph/rhceph-5-
rhel8@sha256:efe1...b7dd
[ceph: root@node /]#
```

You can also use `cephadm shell -- command` to directly run commands through the containerized shell.

```
[root@node ~]# cephadm shell -- ceph status
Inferring fsid 6016ad68-0be8-11ec-9a05-52540000fa0c
Inferring config /var/lib/ceph/6016ad68-0be8-11ec-9a05-52540000fa0c/
mon.node.example.com/config
Using recent ceph image registry.redhat.io/rhceph/rhceph-5-
rhel8@sha256:efe1...b7dd
cluster:
  id: 6016ad68-0be8-11ec-9a05-52540000fa0c
  health: HEALTH_OK

  services:
    mon: 2 daemons, quorum node.example.com,othernode (age 29m)
    mgr: serveve.zyngtp(active, since 30m), standbys: node.example.com.jbfojo
    osd: 15 osds: 15 up (since 29m), 15 in (since 33m)
    rgw: 2 daemons active (2 hosts, 1 zones)

  data:
    pools: 5 pools, 105 pgs
    objects: 189 objects, 4.9 KiB
    usage: 231 MiB used, 150 GiB / 150 GiB avail
    pgs: 105 active+clean
```

You can use the Ceph CLI for tasks to deploy, manage, and monitor a cluster. This course uses the Ceph CLI for many cluster management operations.

The Ceph Dashboard Interface

The Red Hat Ceph Storage 5 Dashboard GUI is enhanced to support many cluster tasks with this interface. The Ceph Dashboard GUI is a web-based application to monitor and manage the cluster. It provides cluster information in a more visual way than the Ceph CLI. Like the Ceph CLI, Ceph implements the Dashboard GUI web server as a module of the `ceph-mgr` daemon. By default, Ceph deploys the Dashboard GUI in the bootstrap node when the cluster is created, and uses TCP port 8443.

The Ceph Dashboard GUI provides these features:

Multi-user and role management

You can create different user accounts with multiple permissions and roles.

Single Sign-On

The Dashboard GUI permits authentication via an external identity provider.

Auditing

You can configure the Dashboard to log all the REST API requests.

Security

The Dashboard uses SSL/TLS by default to secure all HTTP connections.

The Ceph Dashboard GUI also implements different features for managing and monitoring the cluster. The next lists, although not exhaustive, summarize important management and monitoring features:

- Management features
 - Review the cluster hierarchy by using the CRUSH map
 - Enable, edit, and disable manager modules
 - Create, remove, and manage OSDs
 - Manage iSCSI
 - Manage pools
- Monitoring features
 - Check the overall cluster health
 - View the hosts in the cluster and its services
 - Review the logs
 - Review the cluster alerts
 - Check the cluster capacity

The following image shows the status screen from the Dashboard GUI. You can quickly review some important cluster parameters, such as the cluster status, the number of hosts in the cluster, or the number of OSDs.

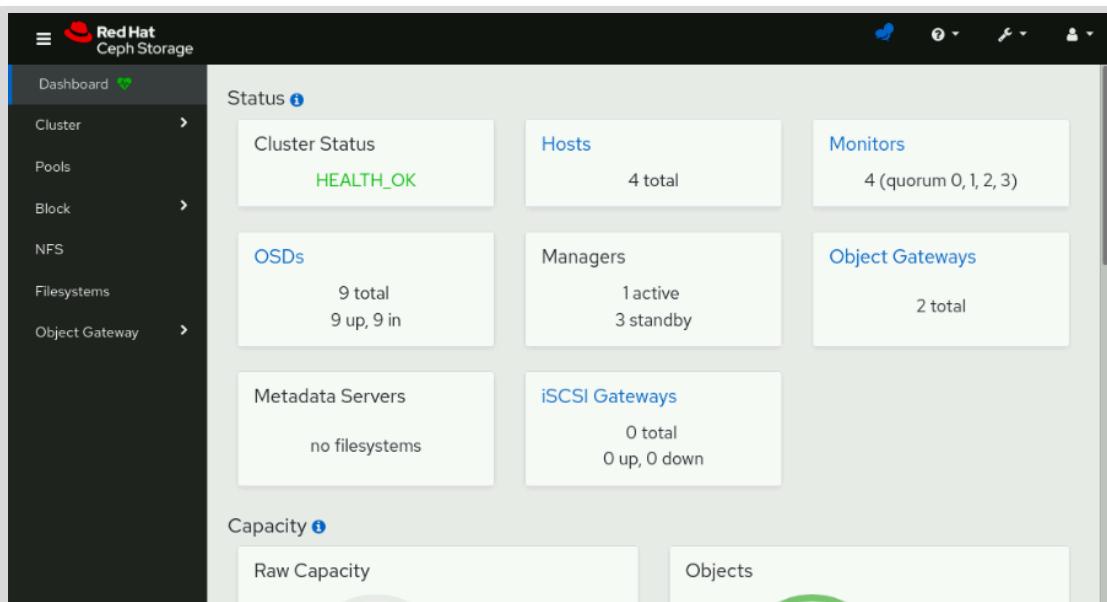


Figure 1.6: Ceph Dashboard GUI status screen



References

For more information, refer to the *Red Hat Ceph Storage 5 Administration Guide* at
https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/administration_guide

For more information, refer to the *Red Hat Ceph Storage 5 Dashboard Guide* at
https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/dashboard_guide/index

► Guided Exercise

Describing Red Hat Ceph Storage Management Interfaces

In this exercise, you navigate the Dashboard GUI primary screens and activities.

Outcomes

You should be able to navigate the Dashboard GUI primary screens.

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 intro-interface
```

This command confirms that the Ceph cluster in the classroom is operating.

Instructions

- ▶ 1. Use Firefox to navigate to the Dashboard web GUI URL at <https://serverc.lab.example.com:8443>. If prompted, accept the self-signed certificates that are used in this classroom.
On the Dashboard login screen, enter your credentials.
 - User name: `admin`
 - Password: `redhat`
- ▶ 2. The Dashboard main screen appears, with three sections: **Status**, **Capacity**, and **Performance**.
 - 2.1. The **Status** section shows an overview of the whole cluster status. You can see the cluster health status, which can be `HEALTH_OK`, `HEALTH_WARN`, or `HEALTH_ERR`. Check that your cluster is in the `HEALTH_OK` state. This section displays the number of cluster hosts, the number of monitors and the cluster quorum that uses those monitors, the number and status of OSDs, and other options.

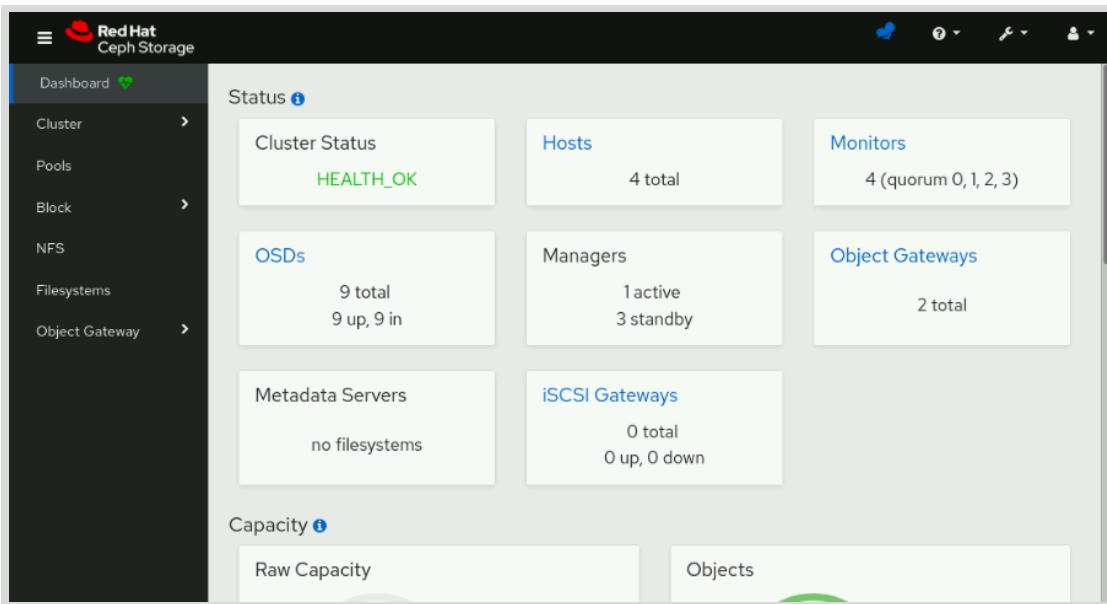


Figure 1.7: Dashboard interface status screen

- 2.2. The Dashboard **Capacity** section displays overall Ceph cluster capacity, the number of objects, the placement groups, and the pools. Check that the capacity of your cluster is approximately 90 GiB.

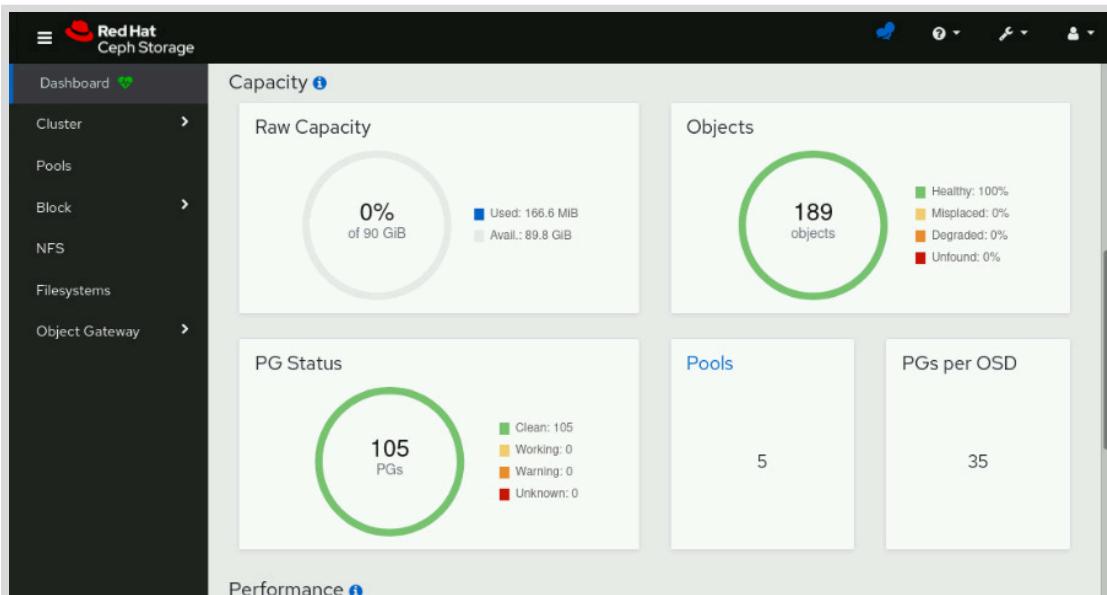


Figure 1.8: Dashboard interface capacity screen

- 2.3. The **Performance** section displays throughput information, and read and write disk speed. Because the cluster just started, the throughput and speed should be 0.

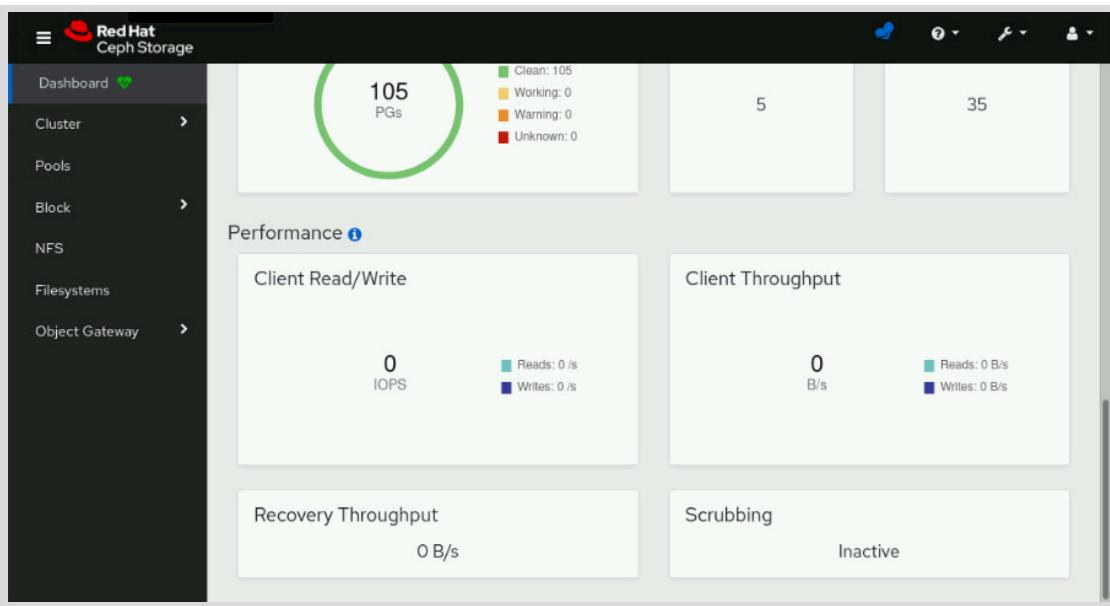


Figure 1.9: Dashboard interface performance screen

► 3. Navigate to the **Cluster** menu.

- 3.1. The **Hosts** section displays the host members of the Ceph cluster, the Ceph services that are running on each host, and the cephadm version that is running. In your cluster, check that the three hosts `serverc`, `serverd`, and `servere` are running the same cephadm version. In this menu, you can add hosts to the cluster, and edit or delete the existing hosts.

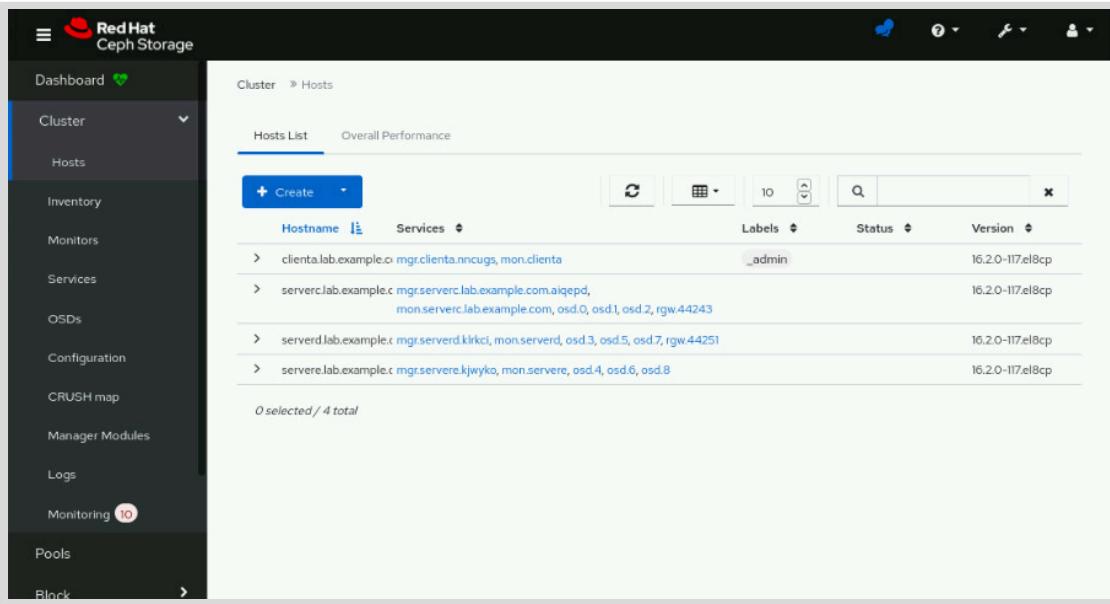


Figure 1.10: Dashboard interface hosts screen

- 3.2. The **Inventory** section displays the physical disks that the Ceph cluster detects. You can view physical disk attributes, such as their host, device path, and size. Verify that the total number of physical disks on your cluster is 20. In this menu, if you select one physical disk and press **Identify**, then that disk's LED starts flashing to make it easy to physically locate disks in your cluster.

The screenshot shows the Red Hat Ceph Storage Dashboard. The left sidebar has a dark theme with white text and icons. The 'Inventory' section is currently selected. The main area is titled 'Physical Disks'. It contains a table with the following data:

Hostname	Device path	Type	Available	Vendor	Model	Size	OSDs
clienta.lab.example	/dev/vdb	HDD	✓	Oxlaef4		10 GiB	
clienta.lab.example	/dev/vdc	HDD	✓	Oxlaef4		10 GiB	
clienta.lab.example	/dev/vdd	HDD	✓	Oxlaef4		10 GiB	
clienta.lab.example	/dev/vde	HDD	✓	Oxlaef4		10 GiB	
clienta.lab.example	/dev/vdf	HDD	✓	Oxlaef4		10 GiB	
serverc.lab.example	/dev/vde	HDD	✓	Oxlaef4		10 GiB	
serverc.lab.example	/dev/vdf	HDD	✓	Oxlaef4		10 GiB	
serverc.lab.example	/dev/vdb	HDD		Oxlaef4		10 GiB	osd.0
serverc.lab.example	/dev/vdc	HDD		Oxlaef4		10 GiB	osd.1
serverc.lab.example	/dev/vdd	HDD		Oxlaef4		10 GiB	osd.2

At the bottom of the table, it says '0 selected / 20 total'. There are navigation buttons for the table at the bottom right.

Figure 1.11: Dashboard interface inventory screen

- 3.3. Navigate to the OSDs section to view information about the cluster OSDs. This section displays the number of OSDs, which host they reside on, the number and usage of placement groups, and the disk read and write speeds. Verify that `serverc` contains three OSDs. You can create, edit, and delete OSDs from this menu.

The screenshot shows the Red Hat Ceph Storage Dashboard. The left sidebar has a dark theme with white text and icons. The 'OSDs' section is currently selected. The main area is titled 'OSDs'. It contains a table with the following data:

ID	Host	Status	Device	PGs	Size	Flags	Usage	Read bytes	Write bytes	Read ops	Write ops
0	serverc	in up	hdd	34	10 GiB		0%	0.4 /s	0 /s		
1	serverc	in up	hdd	42	10 GiB		0%	0.4 /s	0 /s		
2	serverc	in up	hdd	29	10 GiB		0%	0 /s	0 /s		
3	serverd	in up	hdd	39	10 GiB		0%	0.4 /s	0 /s		
4	servere	in up	hdd	34	10 GiB		0%	0.8 /s	0 /s		
5	serverd	in up	hdd	31	10 GiB		0%	0.4 /s	0 /s		
6	servere	in up	hdd	39	10 GiB		0%	1.2 /s	0 /s		

Figure 1.12: Dashboard interface OSDs screen

- 3.4. Navigate to the CRUSH Map section, which displays your cluster's CRUSH map. This map provides information about your cluster's physical hierarchy. Verify that the three host buckets `serverc`, `serverd`, and `servere` are defined within the default bucket of type root.

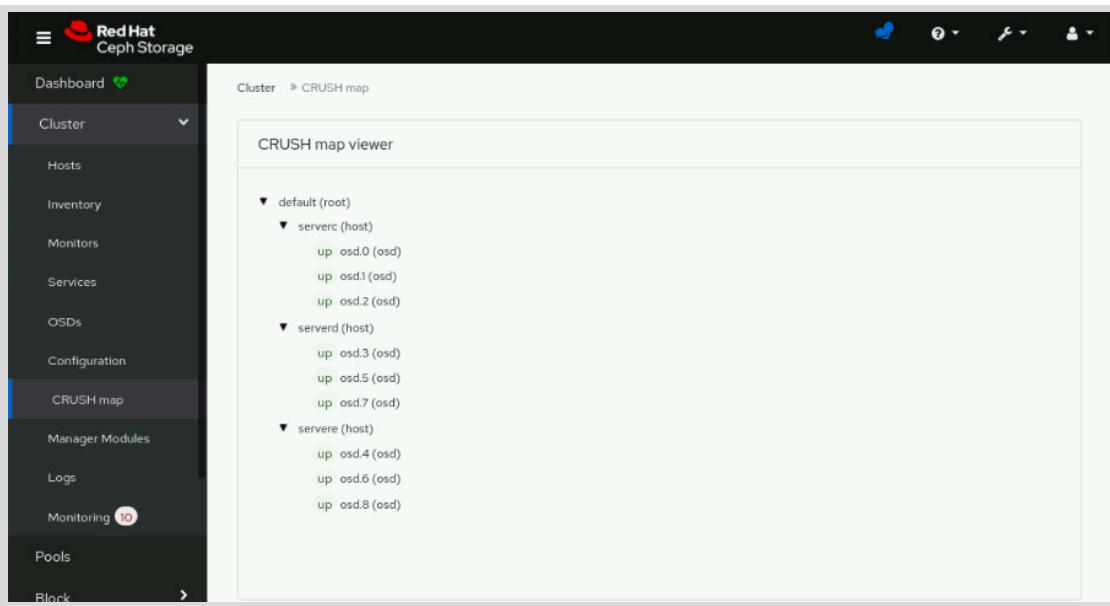


Figure 1.13: Dashboard interface CRUSH screen

- 3.5. The **Logs** section displays the Ceph logs. View the **Cluster Logs** and the **Audit Logs**. You can filter logging messages by priority, keyword, and date. View the **Info** log messages by filtering by **Priority**.

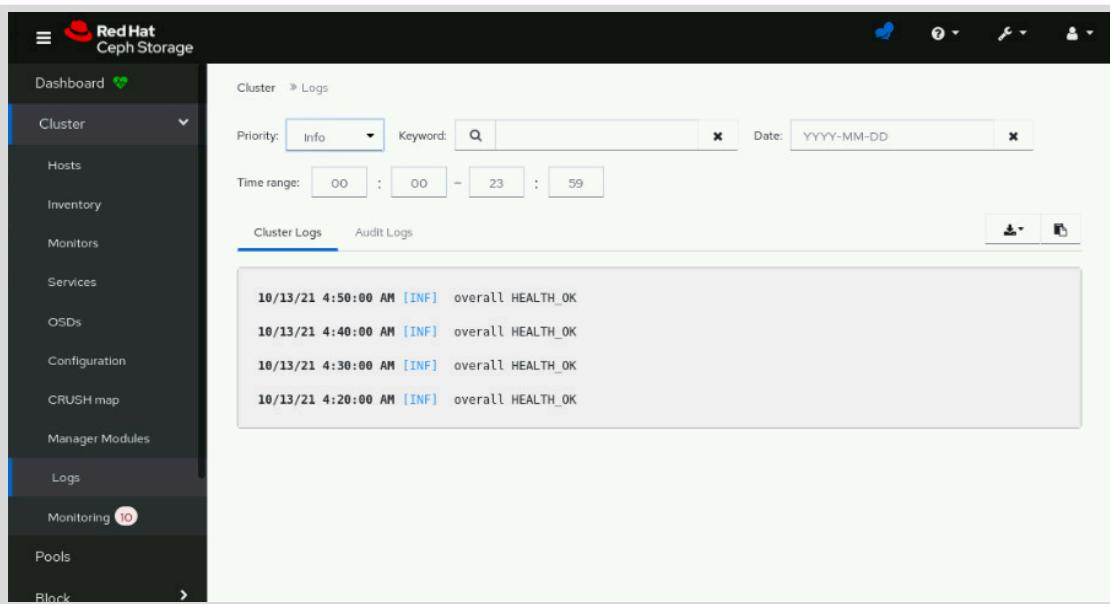


Figure 1.14: Dashboard interface logs screen

- ▶ 4. Navigate to the **Pools** menu.

The screenshot shows the Red Hat Ceph Storage Dashboard. The left sidebar has a dark theme with white text and icons. It includes links for Dashboard, Cluster, Pools (which is selected and highlighted in blue), Block, NFS, Filesystems, and Object Gateway. The main content area is titled 'Pools' and has tabs for 'Pools List' (selected) and 'Overall Performance'. At the top of the list table are buttons for '+ Create' and '- Delete'. The table columns are: Name (sorted by name), Data Protection (replica: x3), Applications (rgw), PG Status (active+clean), Usage (0%), Read bytes (0 /s), Write bytes (0 /s), Read ops (0 /s), and Write ops (0 /s). Below the table, it says '0 selected / 5 total'. There are also various icons at the top right of the dashboard.

Figure 1.15: Dashboard interface pools screen

The **Pools** menu displays existing pool information, including the pool name and type of data protection, and the application. You can also create, edit, or delete the pools from this menu. Verify that your cluster contains a pool called `default.rgw.log`.

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 intro-interface
```

This concludes the guided exercise.

Summary

In this chapter, you learned:

- The following services provide the foundation for a Ceph storage cluster:
 - Monitors (MONs) maintain cluster maps.
 - Object Storage Devices (OSDs) store and manage objects.
 - Managers (MGRs) track and expose cluster runtime metrics.
 - Metadata Servers (MDSes) store metadata that CephFS uses to efficiently run POSIX commands for clients.
- RADOS (Reliable Autonomic Distributed Object Store) is the back end for storage in the Ceph cluster, a self-healing and self-managing object store.
- RADOS provides four access methods to storage: the `librados` native API, the object-based RADOS Gateway, the RADOS Block Device (RBD), and the distributed file-based CephFS file system.
- A Placement Group (PG) aggregates a set of objects into a hash bucket. The CRUSH algorithm maps the hash buckets to a set of OSDs for storage.
- Pools are logical partitions of the Ceph storage that are used to store object data. Each pool is a name tag for grouping objects. A pool groups objects for storage by using placement groups.
- Red Hat Ceph Storage provides two interfaces, a command line and a Dashboard GUI, for managing clusters. Both interfaces use the same `cephadm` module to perform operations and to interact with cluster services.

Chapter 2

Deploying Red Hat Ceph Storage

Goal

Deploy a new Red Hat Ceph Storage cluster and expand the cluster capacity.

Objectives

- Prepare for and perform a Red Hat Ceph Storage cluster deployment using cephadm command-line tools.
- Expand capacity to meet application storage requirements by adding OSDs to an existing cluster.

Sections

- Deploying Red Hat Ceph Storage (and Guided Exercise)
- Expanding Red Hat Ceph Storage Cluster Capacity (and Guided Exercise)

Lab

Deploying Red Hat Ceph Storage

Deploying Red Hat Ceph Storage

Objectives

After completing this section, you should be able to prepare for and perform a Red Hat Ceph Storage cluster deployment using cephadm command-line tools.

Preparing for Cluster Deployment

Use the `cephadm` utility to deploy a new Red Hat Ceph Storage 5 cluster.

The `cephadm` utility consists of two main components:

- The `cephadm shell`.
- The `cephadm orchestrator`.

The `cephadm shell` command runs a bash shell within a Ceph-supplied management container. Use the `cephadm shell` to perform cluster deployment tasks initially and cluster management tasks after the cluster is installed and running.

Launch the `cephadm shell` to run multiple commands interactively, or to run a single command. To run it interactively, use the `cephadm shell` command to open the shell, then run Ceph commands.

The `cephadm orchestrator` provides a command-line interface to the orchestrator `ceph-mgr` modules, which interface with external orchestration services. The purpose of an orchestrator is to coordinate configuration changes that must be performed cooperatively across multiple nodes and services in a storage cluster.

```
[root@node ~]# cephadm shell  
[ceph: root@node /]#
```

To run a single command, use the `cephadm shell` command followed by two dashes and the Ceph command.

```
[root@node ~]# cephadm shell -- CEPH_COMMAND
```

Planning for Cluster Service Colocation

All of the cluster services now run as containers. Containerized Ceph services can run on the same node; this is called **colocation**. Colocation of Ceph services allows for better resource utilization while maintaining secure isolation between the services.

The following daemons can be collocated with OSD daemons: RADOSGW, MDS, RBD-mirror, MON, MGR, Grafana, and NFS Ganesha.

Secure Communication Between Hosts

The `cephadm` command uses SSH to communicate with storage cluster nodes. The cluster SSH key is created during the cluster bootstrap process. Copy the cluster public key to each host that will be a cluster member.

Use the following command to copy the cluster key to a cluster node:

```
[root@node ~]# cephadm shell
[ceph: root@node /]# ceph cephadm get-pub-key > ~/ceph.pub
[ceph: root@node /]# ssh-copy-id -f -i ~/ceph.pub root@node.example.com
```

Deploying a New Cluster

The steps to deploy a new cluster are:

- Install the `cephadm-ansible` package on the host you have chosen as the bootstrap node, which is the first node in the cluster.
- Run the `cephadm preflight` playbook. This playbook verifies that the host has the required prerequisites.
- Use `cephadm` to bootstrap the cluster. The bootstrap process accomplishes the following tasks:
 - Installs and starts a Ceph Monitor and a Ceph Manager daemon on the bootstrap node.
 - Creates the `/etc/ceph` directory.
 - Writes a copy of the cluster public SSH key to `/etc/ceph/ceph.pub` and adds the key to the `/root/.ssh/authorized_keys` file.
 - Writes a minimal configuration file needed to communicate with the new cluster to the `/etc/ceph/ceph.conf` file.
 - Writes a copy of the `client.admin` administrative secret key to the `/etc/ceph/ceph.client.admin.keyring` file.
 - Deploys a basic monitoring stack with `prometheus` and `grafana` services, as well as other tools such as `node-exporter` and `alert-manager`.

Installing Prerequisites

Install `cephadm-ansible` on the bootstrap node:

```
[root@node ~]# yum install cephadm-ansible
```

Run the `cephadm-preflight.yaml` playbook. This playbook configures the Ceph repository and prepares the storage cluster for bootstrapping. It also installs prerequisite packages, such as `podman`, `lvm2`, `chrony`, and `cephadm`.

The preflight playbook uses the `cephadm-ansible` inventory file to identify the admin and client nodes in the storage cluster.

The default location for the inventory file is `/usr/share/cephadm-ansible/hosts`. The following example shows the structure of a typical inventory file:

```
[admin]
node00

[clients]
client01
client02
client03
```

To run the pre-flight playbook:

```
[root@node ~]# ansible-playbook -i INVENTORY-FILE cephadm-preflight.yml \
--extra-vars "ceph_origin=rhcs"
```

Bootstrapping the Cluster

The cephadm bootstrapping process creates a small storage cluster on a single node, consisting of one Ceph Monitor and one Ceph Manager, plus any required dependencies.

Expand the storage cluster by using the `ceph orchestrator` command or the Dashboard GUI to add cluster nodes and services.



Note

Before bootstrapping, you must create a username and password for the `registry.redhat.io` container registry. Visit <https://access.redhat.com/RegistryAuthentication> for instructions.

Use the `cephadm bootstrap` command to bootstrap a new cluster:

```
[root@node ~]# cephadm bootstrap --mon-ip=MON_IP \
--registry-url=registry.redhat.io \
--registry-username=REGISTRY_USERNAME --registry-password=REGISTRY_PASSWORD \
--initial-dashboard-password=DASHBOARD_PASSWORD --dashboard-password-noupdate \
--allow-fqdn-hostname
```

The script displays this output when finished:

Ceph Dashboard is now available at:

```
URL: https://bootstrapnode.example.com:8443/
User: admin
Password: adminpassword
```

You can access the Ceph CLI with:

```
sudo /usr/sbin/cephadm shell --fsid 266ee7a8-2a05-11eb-b846-5254002d4916 -c /etc/ceph/ceph.conf -k /etc/ceph/ceph.client.admin.keyring
```

Please consider enabling telemetry to help improve Ceph:

```
ceph telemetry on
```

For more information see:

<https://docs.ceph.com/docs/master/mgr/telemetry/>

Bootstrap complete.

Using a Service Specification file

Use the `cephadm bootstrap` command with the `--apply-spec` option and a service specification file to both bootstrap a storage cluster and configure additional hosts and daemons. The configuration file is a YAML file that contains the service type, placement, and designated nodes for services to deploy.

The following is an example of a services configuration file:

```
service_type: host
addr: node-00
hostname: node-00
---
service_type: host
addr: node-01
hostname: node-01
---
service_type: host
addr: node-02
hostname: node-02
---
service_type: mon
placement:
  hosts:
    - node-00
    - node-01
    - node-02
---
service_type: mgr
placement:
  hosts:
    - node-00
    - node-01
    - node-02
---
service_type: rgw
service_id: realm.zone
placement:
  hosts:
    - node-01
    - node-02
---
service_type: osd
placement:
  host_pattern: "*"
  data_devices:
    all: true
```

Here is an example command to bootstrap a cluster using a services configuration file:

```
[root@node ~]# cephadm bootstrap --apply-spec CONFIGURATION_FILE_NAME \
--mon-ip MONITOR-IP-ADDRESS
```

Labeling Cluster Nodes

The Ceph orchestrator supports assigning labels to hosts. Labels can be used to group cluster hosts so that you can deploy Ceph services to multiple hosts at the same time. A host can have multiple labels.

Labeling simplifies cluster management tasks by helping to identify the daemons running on each host. For example, you can use the `ceph orch host ls` command to list You can use the Ceph orchestrator or a YAML service specification file to deploy or remove daemons on specifically labeled hosts.

Except for the `_admin` label, labels are free-form and have no specific meaning. You can use labels such as `mon`, `monitor`, `mycluster_monitor`, or other text strings to label and group cluster nodes. For example, assign the `mon` label to nodes that you deploy MON daemons to. Assign the `mgr` label for nodes that you deploy MGR daemons to, and assign `rgw` for RADOS gateways.

For example, the following command applies the `_admin` label to a host to designate it as the admin node.

```
[ceph: root@node /]# ceph orch host label add ADMIN_NODE _admin
```

Deploy cluster daemons to specific hosts by using labels.

```
[ceph: root@node /]# ceph orch apply prometheus --placement="label:prometheus"
```

Setting up the Admin Node

To configure the admin node, perform the following steps:

- Assign the `admin` label to the node, as shown previously.
- Copy the admin key to the admin node.
- Copy the `ceph.conf` file to the admin node.

```
[root@node ~]# scp /etc/ceph/ceph.client.admin.keyring ADMIN_NODE:/etc/ceph/
[root@node ~]# scp /etc/ceph/ceph.conf ADMIN_NODE:/etc/ceph/
```



References

For more information, refer to the *Red Hat Ceph Storage 5 Installation Guide* at https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/installation_guide/index

► Guided Exercise

Deploying Red Hat Ceph Storage

In this exercise, you will install a Red Hat Ceph Storage cluster.

Outcomes

You should be able to install a containerized Ceph cluster by using a service specification file.

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 deploy-deploy
```

This command confirms that the local container registry for the classroom is running and deletes the prebuilt Ceph cluster so it can be redeployed with the steps in this exercise.



Important

This lab start script immediately deletes the prebuilt Ceph cluster and takes a few minutes to complete. Wait for the command to finish before continuing.

Instructions

- 1. Log in to `serverc` as the `admin` user and switch to the `root` user.

```
[student@workstation ~]$ ssh admin@serverc
[admin@serverc ~]$ sudo -i
[root@serverc ~]#
```

- 2. Install the `cephadm-ansible` package, create the inventory file, and run the `cephadm-preflight.yml` playbook to prepare cluster hosts.

- 2.1. Install the `cephadm-ansible` package in `serverc`.

```
[root@serverc ~]# yum install cephadm-ansible
...output omitted...
Complete!
```

- 2.2. Create the `hosts` inventory in the `/usr/share/cephadm-ansible` directory.

```
[root@serverc ~]# cd /usr/share/cephadm-ansible
```

```
[root@serverc cephadm-ansible]# cat hosts
clienta.lab.example.com
serverc.lab.example.com
serverd.lab.example.com
servere.lab.example.com
```

2.3. Run the `cephadm-preflight.yml` playbook.

```
[root@serverc cephadm-ansible]# ansible-playbook -i hosts \
cephadm-preflight.yml --extra-vars "ceph_origin="
...output omitted...
```

The `ceph_origin` variable is set to empty, which causes some playbook tasks to be skipped because, in this classroom, the Ceph packages are installed from a local classroom repository. For your production environment, set `ceph_origin` to `rhcs` to enable the Red Hat Storage Tools repository for your supported deployment.

- 3. Review the `initial-config-primary-cluster.yaml` file in the `/root/ceph/` directory.

```
---
service_type: host ①
addr: 172.25.250.10
hostname: clienta.lab.example.com
---

service_type: host
addr: 172.25.250.12
hostname: serverc.lab.example.com
---

service_type: host
addr: 172.25.250.13
hostname: serverd.lab.example.com
---

service_type: host
addr: 172.25.250.14
hostname: servere.lab.example.com
---

service_type: mon ②
placement:
  hosts:
    - clienta.lab.example.com
    - serverc.lab.example.com
    - serverd.lab.example.com
    - servere.lab.example.com
---

service_type: rgw ③
service_id: realm.zone
placement:
  hosts:
    - serverc.lab.example.com
    - serverd.lab.example.com
---

service_type: mgr ④
```

```
placement:
  hosts:
    - clienta.lab.example.com
    - serverc.lab.example.com
    - serverd.lab.example.com
    - servere.lab.example.com
  ...
  service_type: osd 5
  service_id: default_drive_group
  placement: 6
    host_pattern: 'server*'
  data_devices:
    paths:
      - /dev/vdb
      - /dev/vdc
      - /dev/vdd
```

- 1** The `service_type: host` defines the nodes to add after the `cephadm bootstrap` completes. Host `clienta` will be configured as an admin node.
 - 2** The Ceph Orchestrator deploys one monitor daemon by default. In the file the `service_type: mon` deploys a Ceph monitor daemon in the listed hosts.
 - 3** The `service_type: mgr` deploys a Ceph Object Gateway daemon in the listed hosts.
 - 4** The `service_type: mgr` deploys a Ceph Manager daemon in the listed hosts.
 - 5** The `service_type: osd` deploys a ceph-osd daemon in the listed hosts backed by the `/dev/vdb` device.
 - 6** Defines where and how to deploy the daemons.
- **4.** As the `root` user on the `serverc` node, run the `cephadm bootstrap` command to create the Ceph cluster. Use the service specification file located at `initial-config-primary-cluster.yaml`

```
[root@serverc ~]# cd /root/ceph
```

```
[root@serverc ceph]# cephadm bootstrap --mon-ip=172.25.250.12 \
--apply-spec=initial-config-primary-c luster.yaml \
--initial-dashboard-password=redhat \
--dashboard-password-noupdate \
--allow-fqdn-hostname \
--registry-url=registry.lab.example.com \
--registry-username=registry \
--registry-password=redhat
...output omitted...
Ceph Dashboard is now available at:

  URL: https://serverc.lab.example.com:8443/
  User: admin
  Password: redhat
```

Chapter 2 | Deploying Red Hat Ceph Storage

```
Applying initial-config-primary-cluster.yaml to cluster
Adding ssh key to clienta.lab.example.com
Adding ssh key to serverd.lab.example.com
Adding ssh key to servere.lab.example.com
Added host 'clienta.lab.example.com' with addr '172.25.250.10'
Added host 'serverc.lab.example.com' with addr '172.25.250.12'
Added host 'serverd.lab.example.com' with addr '172.25.250.13'
Added host 'servere.lab.example.com' with addr '172.25.250.14'
Scheduled mon update...
Scheduled rgw.realm.zone update...
Scheduled mgr update...
Scheduled osd.default_drive_group update...
```

You can access the Ceph CLI with:

```
sudo /usr/sbin/cephadm shell --fsid 8896efec-21ea-11ec-b6fe-52540000fa0c -c /etc/ceph/ceph.conf -k /etc/ceph/ceph.client.admin.keyring
```

Please consider enabling telemetry to help improve Ceph:

```
ceph telemetry on
```

For more information see:

```
https://docs.ceph.com/docs/pacific/mgr/telemetry/
```

Bootstrap complete.

▶ **5.** Verify the status of the Ceph storage cluster.

5.1. Run the `cephadm shell`.

```
[root@serverc ~]# cephadm shell
...output omitted...
[ceph: root@serverc /]#
```

5.2. Verify that the cluster status is `HEALTH_OK`.

```
[ceph: root@serverc /]# ceph status
cluster:
  id: 8896efec-21ea-11ec-b6fe-52540000fa0c
  health: HEALTH_OK

  services:
    mon: 4 daemons, quorum serverc.lab.example.com,serverd,servere,clienta (age
10s)
    mgr: serverc.lab.example.com.bypxer(active, since 119s), standbys:
serverd.lflgzj, clienta.hloibd, servere.jhegip
    osd: 9 osds: 9 up (since 55s), 9 in (since 75s)

  data:
    pools: 1 pools, 1 pgs
```

```
objects: 0 objects, 0 B
usage: 47 MiB used, 90 GiB / 90 GiB avail
pgs: 1 active+clean
```

Your cluster might be in the HEALTH_WARN state for a few minutes until all services and OSDs are ready.

- 6. Label `clienta` as the admin node. Verify that you can execute `cephadm` commands from `clienta`.

- 6.1. Apply the `_admin` label to `clienta` to label it as the admin node.

```
[ceph: root@serverc /]# ceph orch host label add clienta.lab.example.com _admin
Added label _admin to host clienta.lab.example.com
```

- 6.2. Manually copy the `ceph.conf` and `ceph.client.admin.keyring` files from `serverc` to `clienta`. These files are located in `/etc/ceph`.

```
[ceph: root@serverc /]# exit
exit
[root@serverc ceph]# cd /etc/ceph
[root@serverc ceph]# scp {ceph.client.admin.keyring,ceph.conf} \
root@clienta:/etc/ceph/
Warning: Permanently added 'clienta' (ECDSA) to the list of known hosts.
ceph.client.admin.keyring          100%   63    105.6KB/s  00:00
ceph.conf                          100%  177    528.3KB/s  00:00
```

- 6.3. Return to `workstation` as the student user, then log into `clienta` as the `admin` user and start the `cephadm` shell. Verify that you can execute `cephadm` commands from `clienta`.

```
[root@serverc ceph]# exit
[admin@serverc ~]$ exit
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
Inferring fsid 8896efec-21ea-11ec-b6fe-52540000fa0c
Inferring config /var/lib/ceph/8896efec-21ea-11ec-b6fe-52540000fa0c/mon.clienta/
config
Using recent ceph image registry.redhat.io/rhceph/rhceph-5-
rhel8@sha256:6306...47ff
[ceph: root@clienta /]# ceph health
HEALTH_OK
```

- 6.4. Return to `workstation` as the student user.

```
[ceph: root@clienta /]# exit
[root@clienta ~]# exit
[admin@clienta ~]$ exit
[student@workstation ~]$
```

Finish

On the **workstation** machine, use the `lab` command to complete this exercise. This command does not disable or modify the Ceph cluster you just deployed. Your new cluster will be used in the next exercise in this chapter.

```
[student@workstation ~]$ lab finish deploy-deploy
```

This concludes the guided exercise.

Expanding Red Hat Ceph Storage Cluster Capacity

Objectives

After completing this section, you should be able to expand capacity to meet application storage requirements by adding OSDs to an existing cluster.

Expanding Your Ceph Cluster Capacity

You can expand the storage capacity of your Red Hat Ceph Storage cluster without disrupting active storage activity. There are two ways to expand the storage in your cluster:

- Add additional OSD nodes to the cluster, referred to as *scaling out*.
- Add additional storage space to the existing OSD nodes, referred to as *scaling up*.

Use the `cephadm shell -- ceph health` command to verify that the cluster is in the `HEALTH_OK` state before starting to deploy additional OSDs.

Configuring Additional OSD Servers

As a storage administrator, you can add more hosts to a Ceph storage cluster to maintain cluster health and provide sufficient load capacity. Add one or more OSDs to expand the storage cluster capacity when the current storage space is becoming full.

- As the root user, add the Ceph storage cluster public SSH key to the root user's `authorized_keys` file on the new host.

```
[root@adm ~]# ssh-copy-id -f -i /etc/ceph/ceph.pub root@new-osd-1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/etc/ceph/ceph.pub"

Number of key(s) added: 1
...output omitted...
```

- As the root user, add new nodes to the inventory file located in `/usr/share/cephadm-ansible/hosts/`. Run the preflight playbook with the `--limit` option to restrict the playbook's tasks to run only on the nodes specified. The Ansible Playbook verifies that the nodes to be added meet the prerequisite package requirements.

```
[root@adm ~]# ansible-playbook -i /usr/share/cephadm-ansible/hosts/ \
/usr/share/cephadm-ansible/cephadm-preflight.yml \
--limit new-osd-1
```

- Choose one of the methods to add new hosts to the Ceph storage cluster:
 - As the root user, in the Cephadm shell, use the `ceph orch host add` command to add a new host to the storage cluster. In this example, the command also assigns host labels.

```
[ceph: root@adm /]# ceph orch host add new-osd-1 --labels=mon,osd,mgr
Added host 'new-osd-1' with addr '192.168.122.102'
```

Chapter 2 | Deploying Red Hat Ceph Storage

- To add multiple hosts, create a YAML file with host descriptions. Create the YAML file within the admin container where you will then run `ceph orch ..`.

```
service_type: host
addr:
hostname: new-osd-1
labels:
- mon
- osd
- mgr
---
service_type: host
addr:
hostname: new-osd-2
labels:
- mon
- osd
```

After creating the YAML file, run the `ceph orch apply` command to add the OSDs:

```
[ceph: root@admin ~]# ceph orch apply -i host.yaml
Added host 'new-osd-1' with addr '192.168.122.102'
Added host 'new-osd-1' with addr '192.168.122.103'
```

Listing Hosts

Use `ceph orch host ls` from the cephadm shell to list the cluster nodes. The STATUS column is blank when the host is online and operating normally.

```
[ceph: root@admin /]# ceph orch host ls
HOST          ADDR          LABELS      STATUS
existing-osd-1 192.168.122.101  mon
new-osd-1      192.168.122.102  mon osd mgr
new-osd-2      192.168.122.103  mon osd
```

Configuring Additional OSD Storage for OSD Servers

As a storage administrator, you can expand the Ceph cluster capacity by adding new storage devices to your existing OSD nodes, then using the cephadm orchestrator to configure them as OSDs. Ceph requires that the following conditions are met to consider a storage device:

- The device must not have any partitions.
- The device must not have any LVM state.
- The device must not be mounted.
- The device must not contain a file system.
- The device must not contain a Ceph BlueStore OSD.
- The device must be larger than 5 GB.

Run the `ceph orch device ls` command from the cephadm shell to list the available devices. The `--wide` option provides more device detail.

```
[ceph: root@adm /]# ceph orch device ls
Hostname Path Type Serial Size Health Ident Fault Available
osd-1   /dev/vdb hdd 8a8d3399-4da0-b 10.7G Unknown N/A N/A Yes
osd-1   /dev/vdc hdd 8b06b0af-4350-b 10.7G Unknown N/A N/A Yes
osd-1   /dev/vdd hdd e15146bc-4970-a 10.7G Unknown N/A N/A Yes
osd-2   /dev/vdb hdd 82dc7aff-45bb-9 10.7G Unknown N/A N/A Yes
osd-2   /dev/vdc hdd e7f82a83-44f2-b 10.7G Unknown N/A N/A Yes
osd-2   /dev/vdd hdd fc290db7-4636-a 10.7G Unknown N/A N/A Yes
osd-3   /dev/vdb hdd cb17228d-45d3-b 10.7G Unknown N/A N/A Yes
osd-3   /dev/vdc hdd d11bb434-4275-a 10.7G Unknown N/A N/A Yes
osd-3   /dev/vdd hdd 68e406a5-4954-9 10.7G Unknown N/A N/A Yes
```

As the root user, run the `ceph orch daemon add osd` command to create an OSD using a specific device on a specific host.

```
[ceph: root@admin /]# ceph orch daemon add osd osd-1:/dev/vdb
Created osd(s) 0 on host 'osd-1'
```

Alternately, run the `ceph orch apply osd --all-available-devices` command to deploy OSDs on all available and unused devices.

```
[ceph: root@adm /]# ceph orch apply osd --all-available-devices
Scheduled osd.all-available-devices update...
```

You can create OSDs by using only specific devices on specific hosts by including selective disk properties. The following example creates two OSDs in the group `default_drive_group` backed by `/dev/vdc` and `/dev/vdd` on each host.

```
[ceph: root@adm /]# cat /var/lib/ceph/osd/osd_spec.yml
service_type: osd
service_id: default_drive_group
placement:
  hosts:
    - osd-1
    - osd-2
data_devices:
  paths:
    - /dev/vdc
    - /dev/vdd
```

Run the `ceph orch apply` command to implement the configuration in the YAML file.

```
[ceph: root@adm /]# ceph orch apply -i /var/lib/ceph/osd/osd_spec.yml
Scheduled osd.default_drive_group update...
```



References

For more information, refer to the *Adding hosts* chapter in the *Red Hat Ceph Storage Installation Guide* at

https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/installation_guide/index#adding-hosts_install

For more information, refer to the *Management of OSDs using the Ceph Orchestrator* chapter in the *Red Hat Ceph Storage Operations Guide* at

https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/operations_guide/index#management-of-osds-using-the-ceph-orchestrator

► Guided Exercise

Expanding Red Hat Ceph Storage Cluster Capacity

In this exercise, you will expand the cluster by adding new OSDs to each of the nodes in the cluster, and by adding an OSD node to the Ceph cluster.

Outcomes

You should be able to expand your cluster by adding new OSDs.

Before You Begin

Start this exercise only after having successfully completed the previous *Guided Exercise: Deploying Red Hat Ceph Storage*:

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

```
[student@workstation ~]$ lab start deploy-expand
```

This command confirms that the Ceph cluster is reachable and provides an example service specification file.

Instructions

In this exercise, expand the amount of storage in your Ceph storage cluster.

► 1. List the inventory of storage devices on the cluster hosts.

1.1. Log in to `serverc` as the `admin` user and use `sudo` to run the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]# sudo cephadm shell
[ceph: root@clienta /]#
```

1.2. Display the storage device inventory on the Ceph cluster.

```
[ceph: root@clienta /]# ceph orch device ls
Hostname          Path    Type  Serial           Size  Health
Ident  Fault Available
clienta.lab.example.com  /dev/vdb  hdd   f97f3019-2827-4b84-8  10.7G Unknown N/A
      N/A   Yes
clienta.lab.example.com  /dev/vdc  hdd   fdaeb489-de2c-4165-9  10.7G Unknown N/A
      N/A   Yes
clienta.lab.example.com  /dev/vdd  hdd   0159ff1a-d45a-4ada-9  10.7G Unknown N/A
      N/A   Yes
clienta.lab.example.com  /dev/vde  hdd   69f43b17-9af3-45f1-b  10.7G Unknown N/A
      N/A   Yes
```

clienta.lab.example.com	/dev/vdf	hdd	bfc36a25-1680-49eb-8	10.7G	Unknown	N/A
N/A	Yes					
serverc.lab.example.com	/dev/vdc	hdd	8b06b0af-ff15-4350-b	10.7G	Unknown	N/A
N/A	Yes					
serverc.lab.example.com	/dev/vdd	hdd	e15146bc-22dd-4970-a	10.7G	Unknown	N/A
N/A	Yes					
serverc.lab.example.com	/dev/vde	hdd	f24fe7d7-b400-44b8-b	10.7G	Unknown	N/A
N/A	Yes					
serverc.lab.example.com	/dev/vdf	hdd	e5747c44-0afa-4918-8	10.7G	Unknown	N/A
N/A	Yes					
serverc.lab.example.com	/dev/vdb	hdd	8a8d3399-52d9-4da0-b	10.7G	Unknown	N/A
N/A	No					
serverd.lab.example.com	/dev/vdc	hdd	e7f82a83-56f6-44f2-b	10.7G	Unknown	N/A
N/A	Yes					
serverd.lab.example.com	/dev/vdd	hdd	fc290db7-fa22-4636-a	10.7G	Unknown	N/A
N/A	Yes					
serverd.lab.example.com	/dev/vde	hdd	565c1d73-48c4-4448-a	10.7G	Unknown	N/A
N/A	Yes					
serverd.lab.example.com	/dev/vdf	hdd	90bf4d1f-83e5-4901-b	10.7G	Unknown	N/A
N/A	Yes					
serverd.lab.example.com	/dev/vdb	hdd	82dc7aff-3c2a-45bb-9	10.7G	Unknown	N/A
N/A	No					
servere.lab.example.com	/dev/vdc	hdd	d11bb434-5829-4275-a	10.7G	Unknown	N/A
N/A	Yes					
servere.lab.example.com	/dev/vdd	hdd	68e406a5-9f0f-4954-9	10.7G	Unknown	N/A
N/A	Yes					
servere.lab.example.com	/dev/vde	hdd	2670c8f2-acde-4948-8	10.7G	Unknown	N/A
N/A	Yes					
servere.lab.example.com	/dev/vdf	hdd	5628d1f0-bdbf-4b05-8	10.7G	Unknown	N/A
N/A	Yes					
servere.lab.example.com	/dev/vdb	hdd	cb17228d-c039-45d3-b	10.7G	Unknown	N/A
N/A	No					

- 2. Deploy two OSDs by using /dev/vdc and /dev/vdd on serverc.lab.example.com, serverd.lab.example.com, and servere.lab.example.com.
- 2.1. Create the osd_spec.yml file in the /var/lib/ceph/osd/ directory with the correct configuration. For your convenience, you can copy and paste the content from the /root/expand-osd/osd_spec.yml file on clienta.

```
[ceph: root@clienta /]# exit
exit
[admin@clienta ~]# sudo cephadm shell --mount /root/expand-osd/osd_spec.yml
[ceph: root@clienta /]# cd /mnt
[ceph: root@clienta mnt]# cp osd_spec.yml /var/lib/ceph/osd/
[ceph: root@clienta mnt]# cat /var/lib/ceph/osd/osd_spec.yml
service_type: osd
service_id: default_drive_group
placement:
  hosts:
    - serverc.lab.example.com
    - serverd.lab.example.com
    - servere.lab.example.com
data_devices:
```

```
paths:
  - /dev/vdb
  - /dev/vdc
  - /dev/vdd
```

- 2.2. Deploy the `osd_spec.yml` file, then run the `ceph orch apply` command to implement the configuration.

```
[ceph: root@clienta mnt]# ceph orch apply -i /var/lib/ceph/osd/osd_spec.yml
Scheduled osd.default_drive_group update...
```

- 3. Add OSDs to `servere` by using devices `/dev/vde`, and `/dev/vdf`.

- 3.1. Display the `servere` storage device inventory on the Ceph cluster.

```
[ceph: root@clienta mnt]# ceph orch device ls --hostname=servere.lab.example.com
Hostname          Path      Type  Serial           Size   Health
Ident  Fault Available
servere.lab.example.com  /dev/vdb  hdd   4f0e87b2-dc76-457a-a  10.7G Unknown N/A
  N/A   No
servere.lab.example.com  /dev/vdc  hdd   b8483b8a-13a9-4992-9  10.7G Unknown N/A
  N/A   No
servere.lab.example.com  /dev/vdd  hdd   ff073d1f-31d8-477e-9  10.7G Unknown N/A
  N/A   No
servere.lab.example.com  /dev/vde  hdd   705f4daa-f63f-450e-8  10.7G Unknown N/A
  N/A   Yes
servere.lab.example.com  /dev/vdf  hdd   63adfb7c-9d9c-4575-8  10.7G Unknown N/A
  N/A   Yes
```

- 3.2. Create the OSDs by using `/dev/vde` and `/dev/vdf` on `servere`.

```
[ceph: root@clienta mnt]# ceph orch daemon add osd \
servere.lab.example.com:/dev/vde
Created osd(s) 9 on host 'servere.lab.example.com'
```

```
[ceph: root@clienta mnt]# ceph orch daemon add osd \
servere.lab.example.com:/dev/vdf
Created osd(s) 10 on host 'servere.lab.example.com'
```

- 4. Verify that the cluster is in a healthy state and that the OSDs were successfully added.

- 4.1. Verify that the cluster status is `HEALTH_OK`.

```
[ceph: root@clienta mnt]# ceph status
cluster:
  id: 29179c6e-10ac-11ec-b149-52540000fa0c
  health: HEALTH_OK

  services:
    mon: 4 daemons, quorum serverc.lab.example.com,clienta,servere,serverd (age
         12m)
```

Chapter 2 | Deploying Red Hat Ceph Storage

```

mgr: serverc.lab.example.com.dwsvgt(active, since 12m), standbys:
serverd.kdkmia, servere.rdbtge, clienta.etpong
osd: 11 osds: 11 up (since 11m), 11 in (since 29m)

...output omitted...

```

- 4.2. Use the `ceph osd tree` command to display the CRUSH tree. Verify that the new OSDs' location in the infrastructure is correct.

```

[ceph: root@clienta mnt]# ceph osd tree
ID CLASS WEIGHT TYPE NAME           STATUS REWEIGHT PRI-AFF
-1          0.10776 root default
-3          0.02939 host serverc
  2  hdd  0.00980     osd.2       up   1.00000  1.00000
  5  hdd  0.00980     osd.5       up   1.00000  1.00000
  8  hdd  0.00980     osd.8       up   1.00000  1.00000
-7          0.02939 host serverd
  1  hdd  0.00980     osd.1       up   1.00000  1.00000
  4  hdd  0.00980     osd.4       up   1.00000  1.00000
  7  hdd  0.00980     osd.7       up   1.00000  1.00000
-5          0.04898 host servere
  0  hdd  0.00980     osd.0       up   1.00000  1.00000
  3  hdd  0.00980     osd.3       up   1.00000  1.00000
  6  hdd  0.00980     osd.6       up   1.00000  1.00000
  9  hdd  0.00980     osd.9       up   1.00000  1.00000
 10  hdd 0.00980     osd.10      up   1.00000  1.00000

```

- 4.3. Use the `ceph osd df` command to verify the data usage and the number of placement groups for each OSD.

```

[ceph: root@clienta mnt]# ceph osd df
ID CLASS WEIGHT  REWEIGHT SIZE    RAW USE  DATA   OMAP  META  AVAIL
%USE  VAR    PGS  STATUS
  2  hdd  0.00980 1.00000 10 GiB  16 MiB 2.1 MiB 0 B  14 MiB 10 GiB
 0.15 1.05  38     up
  5  hdd  0.00980 1.00000 10 GiB  15 MiB 2.1 MiB 0 B  13 MiB 10 GiB
 0.15 1.03  28     up
  8  hdd  0.00980 1.00000 10 GiB  20 MiB 2.1 MiB 0 B  18 MiB 10 GiB
 0.20 1.37  39     up
  1  hdd  0.00980 1.00000 10 GiB  16 MiB 2.1 MiB 0 B  14 MiB 10 GiB
 0.16 1.08  38     up
  4  hdd  0.00980 1.00000 10 GiB  16 MiB 2.1 MiB 0 B  14 MiB 10 GiB
 0.15 1.06  34     up
  7  hdd  0.00980 1.00000 10 GiB  15 MiB 2.1 MiB 0 B  13 MiB 10 GiB
 0.15 1.04  33     up
  0  hdd  0.00980 1.00000 10 GiB  16 MiB 2.1 MiB 0 B  14 MiB 10 GiB
 0.15 1.05  22     up
  3  hdd  0.00980 1.00000 10 GiB  16 MiB 2.1 MiB 0 B  14 MiB 10 GiB
 0.15 1.05  25     up
  6  hdd  0.00980 1.00000 10 GiB  16 MiB 2.1 MiB 0 B  14 MiB 10 GiB
 0.15 1.05  23     up
  9  hdd  0.00980 1.00000 10 GiB  16 MiB 2.1 MiB 0 B  14 MiB 10 GiB
 0.15 1.05  24     up

```

```
10    hdd  0.00980   1.00000   10  GiB   16  MiB  2.1  MiB   0  B   14  MiB   10  GiB
0.15   1.05   26      up
                                TOTAL   110  GiB   330  MiB   68  MiB  2.8  KiB  262  MiB   110  GiB
0.29
MIN/MAX VAR: 0.65/1.89  STDDEV: 0.16
```

4.4. Return to `workstation` as the `student` user.

```
[ceph: root@clienta mnt]# exit
[admin@admin ~]$ exit
[student@workstation ~]$
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This command will not delete your Ceph cluster or modify your cluster configuration, allowing you to browse your expanded configuration before continuing with the next chapter.

```
[student@workstation ~]$ lab finish deploy-expand
```

This concludes the guided exercise.

► Lab

Deploying Red Hat Ceph Storage

In this lab, you will deploy a Red Hat Ceph Storage cluster.

Outcomes

You should be able to deploy a new Ceph cluster.

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 deploy-review
```

This command confirms that the local container registry for the classroom is running and deletes the prebuilt Ceph cluster so it can be redeployed with the steps in this exercise.



Important

This lab start script immediately deletes the prebuilt Ceph cluster and takes a few minutes to complete. Wait for the command to finish before continuing.

Instructions

Deploy a new cluster with `serverc`, `serverd`, and `servere` as MON, MGR, and OSD nodes. Use `serverc` as the deployment bootstrap node. Add OSDs to the cluster after the cluster deploys.

1. Use `serverc` as the bootstrap node. Log in to `serverc` as the `admin` user and switch to the `root` user. Run the `cephadm-preflight.yml` playbook to prepare the cluster hosts.
2. Create a services specification file called `initial-cluster-config.yaml`.
Using the following template, add hosts `serverd.lab.example.com` and `servere.lab.example.com`, with their IP addresses, as `service_type: host`. Add `serverc` and `serverd` to the `mon` and `mgr` sections.

```
---
service_type: host
addr: 172.25.250.12
hostname: serverc.lab.example.com
---
service_type: mon
placement:
  hosts:
    - serverc.lab.example.com
---
service_type: mgr
placement:
```

```
hosts:
  - serverc.lab.example.com
---
service_type: osd
service_id: default_drive_group
placement:
  host_pattern: 'server*'
data_devices:
  paths:
    - /dev/vdb
    - /dev/vdc
    - /dev/vdd
```

3. Create the Ceph cluster by using the `initial-cluster-config.yaml` service specification file. Verify that the cluster was successfully deployed.
4. Expand the cluster by adding OSDs to `serverc`, `serverd`, and `servere`. Use the following service specification file.

```
service_type: osd
service_id: default_drive_group
placement:
  hosts:
    - serverc.lab.example.com
    - serverd.lab.example.com
    - servere.lab.example.com
data_devices:
  paths:
    - /dev/vde
    - /dev/vdf
```

5. Return to `workstation` as the `student` user.

Evaluation

Grade your work by running the `lab grade deploy-review` command from your `workstation` machine. Correct any reported failures and rerun the script until successful.

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

Finish

Reset your classroom environment. This restores your Ceph environment to the original, prebuilt Ceph cluster that is expected by other course chapters.

This concludes the lab.

► Solution

Deploying Red Hat Ceph Storage

In this lab, you will deploy a Red Hat Ceph Storage cluster.

Outcomes

You should be able to deploy a new Ceph cluster.

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 deploy-review
```

This command confirms that the local container registry for the classroom is running and deletes the prebuilt Ceph cluster so it can be redeployed with the steps in this exercise.



Important

This lab start script immediately deletes the prebuilt Ceph cluster and takes a few minutes to complete. Wait for the command to finish before continuing.

Instructions

Deploy a new cluster with `serverc`, `serverd`, and `servere` as MON, MGR, and OSD nodes. Use `serverc` as the deployment bootstrap node. Add OSDs to the cluster after the cluster deploys.

1. Use `serverc` as the bootstrap node. Log in to `serverc` as the `admin` user and switch to the `root` user. Run the `cephadm-preflight.yml` playbook to prepare the cluster hosts.
 - 1.1. Log in to `serverc` as the `admin` user and switch to the `root` user.

```
[student@workstation ~]$ ssh admin@serverc
[admin@serverc ~]$ sudo -i
[root@serverc ~]#
```

- 1.2. Run the `cephadm-preflight.yml` playbook to prepare the cluster hosts.

```
[root@serverc ~]# cd /usr/share/cephadm-ansible
[root@serverc cephadm-ansible]# ansible-playbook -i /tmp/hosts \
cephadm-preflight.yml --extra-vars "ceph_origin="
...output omitted...
```

2. Create a services specification file called `initial-cluster-config.yaml`.

Using the following template, add hosts `serverd.lab.example.com` and `servere.lab.example.com`, with their IP addresses, as `service_type: host`. Add `serverc` and `serverd` to the `mon` and `mgr` sections.

```
---
  service_type: host
  addr: 172.25.250.12
  hostname: serverc.lab.example.com
---
  service_type: mon
  placement:
    hosts:
      - serverc.lab.example.com
---
  service_type: mgr
  placement:
    hosts:
      - serverc.lab.example.com
---
  service_type: osd
  service_id: default_drive_group
  placement:
    host_pattern: 'server*'
  data_devices:
    paths:
      - /dev/vdb
      - /dev/vdc
      - /dev/vdd
```

2.1. Create a services specification file called `/tmp/initial-cluster-config.yaml`.

```
[root@serverc cephadm-ansible]# cat /tmp/initial-cluster-config.yaml
---
  service_type: host
  addr: 172.25.250.12
  hostname: serverc.lab.example.com
---
  service_type: host
  addr: 172.25.250.13
  hostname: serverd.lab.example.com
---
  service_type: host
  addr: 172.25.250.14
  hostname: servere.lab.example.com
---
  service_type: mon
  placement:
    hosts:
      - serverc.lab.example.com
      - serverd.lab.example.com
      - servere.lab.example.com
---
  service_type: mgr
```

```

placement:
  hosts:
    - serverc.lab.example.com
    - serverd.lab.example.com
    - servere.lab.example.com
  ...
service_type: osd
service_id: default_drive_group
placement:
  host_pattern: 'server*'
data_devices:
  paths:
    - /dev/vdb
    - /dev/vdc
    - /dev/vdd

```

3. Create the Ceph cluster by using the `initial-cluster-config.yaml` service specification file. Verify that the cluster was successfully deployed.
 - 3.1. Run the `cephadm bootstrap` command to create the Ceph cluster. Use the `initial-cluster-config.yaml` service specification file that you just created.

```
[root@serverc cephadm-ansible]# cephadm bootstrap --mon-ip=172.25.250.12 \
--apply-spec=/tmp/initial-cluster-config.yaml \
--initial-dashboard-password=redhat \
--dashboard-password-noupdate \
--allow-fqdn-hostname \
--registry-url=registry.redhat.io \
--registry-username=registry \
--registry-password=redhat
...output omitted...
Ceph Dashboard is now available at:
```

```
    URL: https://serverc.lab.example.com:8443/
    User: admin
    Password: redhat
```

```
Applying /tmp/initial-cluster-config.yaml to cluster
Adding ssh key to serverd.lab.example.com
Adding ssh key to servere.lab.example.com
Added host 'serverc.lab.example.com' with addr '172.25.250.12'
Added host 'serverd.lab.example.com' with addr '172.25.250.13'
Added host 'servere.lab.example.com' with addr '172.25.250.14'
Scheduled mon update...
Scheduled mgr update...
Scheduled osd.default_drive_group update...
```

You can access the Ceph CLI with:

```
sudo /sbin/cephadm shell --fsid 0bbab748-30ee-11ec-abc4-52540000fa0c -c /etc/
ceph/ceph.conf -k /etc/ceph/ceph.client.admin.keyring
...output omitted...
Bootstrap complete.
```

- 3.2. Using the `cephadm` shell, verify that the cluster was successfully deployed. Wait for the cluster to finish deploying and reach the `HEALTH_OK` status.

```
[root@serverc cephadm-ansible]# cephadm shell
...output omitted...
[ceph: root@serverc /]# ceph status
cluster:
  id:      0bbab748-30ee-11ec-abc4-52540000fa0c
  health:  HEALTH_OK

  services:
    mon: 3 daemons, quorum serverc.lab.example.com,servere,serverd (age 2m)
    mgr: serverc.lab.example.com.blxerd(active, since 3m), standbys:
          serverd.nibyts, servere.rkpsii
    osd: 9 osds: 9 up (since 2m), 9 in (since 2m)

  data:
    pools:   1 pools, 1 pgs
    objects: 0 objects, 0 B
    usage:   46 MiB used, 90 GiB / 90 GiB avail
    pgs:     1 active+clean
```

4. Expand the cluster by adding OSDs to `serverc`, `serverd`, and `servere`. Use the following service specification file.

```
service_type: osd
service_id: default_drive_group
placement:
  hosts:
    - serverc.lab.example.com
    - serverd.lab.example.com
    - servere.lab.example.com
data_devices:
  paths:
    - /dev/vde
    - /dev/vdf
```

- 4.1. Create a service specification file called `osd-spec.yaml`.

```
[ceph: root@serverc /]# cat /tmp/osd-spec.yaml
service_type: osd
service_id: default_drive_group
placement:
  hosts:
    - serverc.lab.example.com
    - serverd.lab.example.com
    - servere.lab.example.com
data_devices:
  paths:
    - /dev/vde
    - /dev/vdf
```

- 4.2. Use the `ceph orch apply` command to add the OSDs to the cluster OSD nodes.

```
[ceph: root@serverc /]# ceph orch apply -i /tmp/osd-spec.yaml
Scheduled osd.default_drive_group update...
```

- 4.3. Verify that the OSDs were added. Wait for the new OSDs to display as **up** and **in**.

```
[ceph: root@serverc /]# ceph status
cluster:
  id:      0bbab748-30ee-11ec-abc4-52540000fa0c
  health:  HEALTH_OK

  services:
    mon: 3 daemons, quorum serverc.lab.example.com,servere,serverd (age 5m)
    mgr: serverc.lab.example.com.blxerd(active, since 6m), standbys:
          serverd.nibyts, servere.rkpsii
    osd: 15 osds: 15 up (since 10s), 15 in (since 27s)

  data:
    pools:   1 pools, 1 pgs
    objects: 0 objects, 0 B
    usage:   83 MiB used, 150 GiB / 150 GiB avail
    pgs:     1 active+clean
```

5. Return to workstation as the student user.

- 5.1. Return to workstation as the student user.

```
[ceph: root@serverc /]# exit
[root@serverc cephadm-ansible]# exit
[student@serverc ~]$ exit
[student@workstation ~]$
```

Evaluation

Grade your work by running the `lab grade deploy-review` command from your workstation machine. Correct any reported failures and rerun the script until successful.

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

Finish

Reset your classroom environment. This restores your Ceph environment to the original, prebuilt Ceph cluster that is expected by other course chapters.

This concludes the lab.

Summary

In this chapter, you learned:

- The two main components of the `cephadm` utility:
 - The `cephadm shell` runs a bash shell within a specialized management container. Use the `cephadm` shell to perform cluster deployment tasks and cluster management tasks after the cluster is installed.
 - The `cephadm orchestrator` provides a command-line interface to the orchestrator `ceph-mgr` modules. The orchestrator coordinates configuration changes that must be performed cooperatively across multiple nodes and services in a storage cluster.
- As of version 5.0, all Red Hat Ceph Storage cluster services are containerized.
- Preparing for a new cluster deployment requires planning cluster service placement and distributing SSH keys to nodes.
- Use `cephadm` to bootstrap a new cluster:
 - Installs and starts the MON and MGR daemons on the bootstrap node.
 - Writes a copy of the cluster public SSH key and adds the key to authorized keys file.
 - Writes a minimal configuration file to communicate with the new cluster.
 - Writes a copy of the administrative secret key to the key ring file.
 - Deploys a basic monitoring stack.
- Use the `cephadm-preflight.yml` playbook to verify cluster host prerequisites.
- Assign labels to the cluster hosts to identify the daemons running on each host. The `_admin` label is reserved for administrative nodes.
- Expand cluster capacity by adding OSD nodes to the cluster or additional storage space to existing OSD nodes.

Chapter 3

Configuring a Red Hat Ceph Storage Cluster

Goal

Manage the Red Hat Ceph Storage configuration, including the primary settings, the use of monitors, and the cluster network layout.

Objectives

- Identify and configure the primary settings for the overall Red Hat Ceph Storage cluster.
- Describe the purpose of cluster monitors and the quorum procedures, query the monitor map, manage the configuration database, and describe Cephx.
- Describe the purpose for each of the cluster networks, and view and modify the network configuration.

Sections

- Managing Cluster Configuration Settings (and Guided Exercise)
- Configuring Cluster Monitors (and Guided Exercise)
- Configuring Cluster Networking (and Guided Exercise)

Lab

Configuring a Red Hat Ceph Storage Cluster

Managing Cluster Configuration Settings

Objectives

After completing this section, you should be able to identify and configure the primary settings for the overall Red Hat Ceph Storage cluster.

Ceph Cluster Configuration Overview

All Red Hat Ceph Storage cluster configurations contain these required definitions:

- Cluster network configuration
- Cluster monitor (MON) configuration and bootstrap options
- Cluster authentication configuration
- Daemon configuration options

Ceph configuration settings use unique names that consist of lowercase character words connected with underscores.



Note

Configuration settings might contain dash or space characters when using some configuration methods. However, using underscores in configuration naming is a consistent, recommended practice.

Every Ceph daemon, process, and library accesses its configuration from one of these sources:

- The compiled-in default value
- The centralized configuration database
- A configuration file that is stored on the local host
- Environment variables
- Command-line arguments
- Runtime overrides



Important

Later settings override those found in earlier sources when multiple setting sources are present. The configuration file configures the daemons when they start. Configuration file settings override those stored in the central database.

The monitor (MON) nodes manage a centralized configuration database. On startup, Ceph daemons parse configuration options that are provided via command-line options, environment variables, and the local cluster configuration file. The daemons then contact the MON cluster to retrieve configuration settings that are stored in the centralized configuration database.

Red Hat Ceph Storage 5 deprecates the `ceph.conf` cluster configuration file, making the centralized configuration database the preferred way to store configuration settings.

Modifying the Cluster Configuration File

Each Ceph node stores a local cluster configuration file. The default location of the cluster configuration file is `/etc/ceph/ceph.conf`. The `cephadm` tool creates an initial Ceph configuration file with a minimal set of options.

The configuration file uses an INI file format, containing several sections that include configuration for Ceph daemons and clients. Each section has a name that is defined with the `[name]` header, and one or more parameters that are defined as a key-value pair.

```
[name]
parameter1 = value1
parameter2 = value2
```

Use a hash sign (#) or semicolon (;) to disable settings or to add comments.

Bootstrap a cluster with custom settings by using a cluster configuration file. Use the `cephadm bootstrap` command with the `--config` option to pass the configuration file.

```
[root@node ~]# cephadm bootstrap --config ceph-config.yaml
```

Configuration Sections

Ceph organizes configuration settings into groups, whether stored in the configuration file or in the configuration database, using sections called for the daemons or clients to which they apply.

- The `[global]` section stores general configuration that is common to all daemons or any process that reads the configuration, including clients. You can override `[global]` parameters by creating called sections for individual daemons or clients.
- The `[mon]` section stores configuration for the Monitors (MON).
- The `[osd]` section stores configuration for the OSD daemons.
- The `[mgr]` section stores configuration for the Managers (MGR).
- The `[mds]` section stores configuration for the Metadata Servers (MDS).
- The `[client]` section stores configuration that applies to all the Ceph clients.



Note

A well-commented `sample.ceph.conf` example file can be found in `/var/lib/containers/storage/overlays/{ID}/merged/usr/share/doc/ceph/` on any cluster node, the area where podman manages containerized service data.

Instance Settings

Group the settings that apply to a specific daemon instance in their own section, with a name of the form `[daemon-type.instance-ID]`.

```
[mon]
# Settings for all mon daemons

[mon.serverc]
# Settings that apply to the specific MON daemon running on serverc
```

The same naming applies to the [osd], [mgr], [mds], and [client] sections. For OSD daemons, the instance ID is always numeric, for example [osd.0]. For clients, the instance ID is the active user name, such as [client.operator3].

Meta Variables

Meta variables are Ceph-defined variables. Use them to simplify the configuration.

\$cluster

The name of the Red Hat Ceph Storage 5 cluster. The default cluster name is ceph.

\$type

The daemon type, such as the value mon for a monitor. OSDs use osd, MDSes use mds, MGRs use mgr, and client applications use client.

\$id

The daemon instance ID. This variable has the value serverc for the Monitor on serverc. \$id is 1 for osd.1, and is the user name for a client application.

\$name

The daemon name and instance ID. This variable is a shortcut for \$type.\$id.

\$host

The host name on which the daemon is running.

Using the Centralized Configuration Database

The MON cluster manages and stores the centralized configuration database on the MON nodes. You can either change a setting temporarily, until the daemons restart, or configure a setting permanently and store it in the database. You can change most configuration settings while the cluster is running.

Use `ceph config` commands to query the database and view configuration information.

- `ceph config ls`, to list all possible configuration settings.
- `ceph config help setting`, for help with a particular configuration setting.
- `ceph config dump`, to show the cluster configuration database settings.
- `ceph config show $type.$id`, to show the database settings for a specific daemon. Use `show-with-defaults` to include default settings.
- `ceph config get $type.$id`, to get a specific configuration setting.
- `ceph config set $type.$id`, to set a specific configuration setting.

Use the `assimilate-conf` subcommand to apply configuration from a file to a running cluster. This process recognizes and applies the changed settings from the configuration file to the centralized database. This command is useful to import custom settings from a previous storage cluster to a new one. Invalid or unrecognized options display on standard output, and require manual handling. Redirect screen output to a file using the `-o output-file`.

```
[ceph: root@node /]# ceph config assimilate-conf -i ceph.conf
```

Cluster Bootstrap Options

Some options provide the information needed to start the cluster. MON nodes read the monmap to find other MONs and establish a quorum. MON nodes read the `ceph.conf` file to identify how to communicate with other MONs.

The `mon_host` option lists cluster monitors. This option is essential and cannot be stored in the configuration database. To avoid using a cluster configuration file, Ceph clusters support using DNS service records to provide the `mon_host` list.

The local cluster configuration file can contain other options to fit your requirements.

- `mon_host_override`, the initial list of monitors for the cluster to contact to start communicating.
- `mon_dns_serv_name`, the name of the DNS SRV record to check to identify the cluster monitors via DNS.
- `mon_data`, `osd_data`, `mds_data`, `mgr_data`, define the daemon's local data storage directory.
- `keyring`, `keyfile`, and `key`, the authentication credentials to authenticate with the monitor.

Using Service Configuration Files

Service configuration files are YAML files that bootstrap a storage cluster and additional Ceph services. The `cephadm` tool orchestrates the service deployment, sizing, and placement by balancing the running daemons in the cluster. Various parameters can deploy services such as OSDs or MONs in a more defined manner.

An example service configuration file follows.

```
service_type: mon
placement:
  host_pattern: "mon*"
  count: 3
---
service_type: osd
service_id: default_drive_group
placement:
  host_pattern: "osd*"
data_devices:
  all: true
```

- `service_type` defines the type of service, such as `mon`, `mds`, `mgr`, or `rgw`.
- `placement` defines the location and quantity of the services to deploy. You can define the hosts, host pattern, or label to select the target servers.
- `data_devices` is specific to OSD services, supporting filters such as size, model, or paths.

Use the `cephadm bootstrap --apply-spec` command to apply the service configurations from the specified file.

```
[root@node ~]# cephadm bootstrap --apply-spec service-config.yaml
```

Overriding Configuration Settings at Runtime

You can change most cluster configuration settings while running. You can temporarily change a configuration setting while the daemon is running.

The `ceph tell $type.$id config` command temporarily overrides configuration settings, and requires that both the MONs and the daemon being configured are running.

Run this command from any cluster host configured to run `ceph` commands. Settings that are changed with this command revert to their original settings when the daemon restarts.

- `ceph tell $type.$id config get` gets a specific runtime setting for a daemon.
- `ceph tell $type.$id config set` sets a specific runtime setting for a daemon. When the daemon restarts, these temporary settings revert to their original values.

The `ceph tell $type.$id config` command can also accept wildcards to get or set the value on all daemons of the same type. For example, `ceph tell osd.* config get debug_ms` displays the value of that setting for all OSD daemons in the cluster.

You can use the `ceph daemon $type.$id config` command to temporarily override a configuration setting. Run this command on the cluster node where the setting is required.

The `ceph daemon` does not need to connect through the MONs. The `ceph daemon` command will still function even when the MONs are not running, which can be useful for troubleshooting.

- `ceph daemon $type.$id config get` gets a specific runtime setting for a daemon.
- `ceph daemon $type.$id config set` sets a specific runtime setting for a daemon. Temporary settings revert to their original values when their daemon restarts.



References

For more information, refer to the *Red Hat Ceph Storage 5 Configuration Guide* at https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/configuration_guide/index#the-basics-of-ceph-configuration

► Guided Exercise

Managing Cluster Configuration Settings

In this exercise, you query and modify Red Hat Ceph Storage configuration settings.

Outcomes

You should be able to view stored configuration settings, and view and set the value of a specific stored setting and a specific runtime setting.

Before You Begin



Important

Do you need to reset your environment before performing this exercise?

If you performed the practice cluster deployments in the *Deploying Red Hat Ceph Storage* chapter exercises, but have not reset your environment back to the default classroom cluster since that chapter, then you must reset your environment before executing the `lab start` command.

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

```
[student@workstation ~]$ lab start configure-settings
```

This command confirms that the required hosts for this exercise are accessible.

Instructions

- 1. Log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]#
```

- 2. View the cluster configuration database.

```
[ceph: root@clienta /]# ceph config dump
WHO                                MASK  LEVEL      OPTION
          VALUE
global                            RO
                                advanced  cluster_network
                                *
                                advanced
auth_allow_insecure_global_id_reclaim  false
mon                               advanced  public_network
                                *
                                advanced
mon                               *
                                advanced
                                *
```

mgr		advanced	mgr/cephadm/
container_init	True	*	
mgr		advanced	mgr/cephadm/
migration_current	2	*	
mgr		advanced	mgr/cephadm/
registry_password	redhat	*	
mgr	registry.redhat.io	advanced	mgr/cephadm/registry_url
mgr		advanced	mgr/cephadm/
registry_username	registry	*	
mgr		advanced	mgr/dashboard/
ALERTMANAGER_API_HOST	http://172.25.250.12:9093	*	
mgr		advanced	mgr/dashboard/
GRAFANA_API_SSL_VERIFY	false	*	
mgr		advanced	mgr/dashboard/
GRAFANA_API_URL	https://172.25.250.12:3000	*	
mgr		advanced	mgr/dashboard/
PROMETHEUS_API_HOST	http://172.25.250.12:9095	*	
mgr		advanced	mgr/dashboard/
ssl_server_port	8443	*	
mgr		advanced	mgr/orchestrator/
orchestrator	cephadm		
client.rgw.realm.zone.serverc.xhyhfk		basic	rgw_frontends
beast port=80		*	
client.rgw.realm.zone.serverd.fonetu		basic	rgw_frontends
beast port=80		*	

- ▶ 3. View the runtime configuration settings for the osd.1 daemon.

```
[ceph: root@clienta /]# ceph config show osd.1
NAME          VALUE
SOURCE        OVERRIDES  IGNORES
cluster_network 172.25.249.0/24
    mon
container_image   registry.redhat.io/rhceph/rhceph-5-rhel8@sha256:6306...47ff
    mon
daemonize      false
    override
keyring        $osd_data/keyring
    default
leveldb_log
    default
log_stderr_prefix debug
    default
log_to_file    false
    default
log_to_stderr  true
    default
mon_host       [v2:172.25.250.12:3300/0,v1:172.25.250.12:6789/0]
    file
```

```
no_config_file      false
                   override
rbd_default_features 61
                   default
setgroup           ceph
                   cmdline
setuser            ceph
                   cmdline
```

- ▶ 4. View the runtime and cluster configuration database value of the `debug_ms` setting for the `osd.1` daemon. Edit the setting value to 10. Verify that the runtime and cluster configuration database value is applied. Restart the `osd.1` daemon. Verify that the values for the `debug_ms` setting persist after the restart.

- 4.1. View the `debug_ms` runtime configuration setting for `osd.1`.

```
[ceph: root@clienta /]# ceph config show osd.1 debug_ms
0/0
```

- 4.2. View the `debug_ms` setting value that is stored in the cluster configuration database for the `osd.1` daemon.

```
[ceph: root@clienta /]# ceph config get osd.1 debug_ms
0/0
```

- 4.3. Modify the `debug_ms` setting value for `osd.1`.

```
[ceph: root@clienta /]# ceph config set osd.1 debug_ms 10
```

- 4.4. Verify that the runtime and cluster configuration database value is applied.

```
[ceph: root@clienta /]# ceph config show osd.1 debug_ms
10/10
[ceph: root@clienta /]# ceph config get osd.1 debug_ms
10/10
```

- 4.5. Restart the `osd.1` daemon.

```
[ceph: root@clienta /]# ceph orch daemon restart osd.1
Scheduled to restart osd.1 on host 'serverd.lab.example.com'
```

- 4.6. Run the `cephadm` shell and verify that the values persist after the restart.

```
[ceph: root@clienta /]# ceph config show osd.1 debug_ms
10/10
[ceph: root@clienta /]# ceph config get osd.1 debug_ms
10/10
```

- ▶ 5. Use `ceph tell` to temporarily override the `debug_ms` setting value for the `osd.1` daemon to 5. Restart the `osd.1` daemon in `serverc` and verify that the value is reverted.

- 5.1. View the runtime value of the `debug_ms` setting for `osd.1`.

```
[ceph: root@clienta /]# ceph tell osd.1 config get debug_ms
{
    "debug_ms": "10/10"
}
```

- 5.2. Temporarily change the runtime value of the `debug_ms` setting for `osd.1` to 5. This setting reverts when you restart the `osd.1` daemon.

```
[ceph: root@clienta /]# ceph tell osd.1 config set debug_ms 5
{
    "success": ""
}
```

- 5.3. Verify the runtime value of the `debug_ms` setting for `osd.1`.

```
[ceph: root@clienta /]# ceph tell osd.1 config get debug_ms
{
    "debug_ms": "5/5"
}
```

- 5.4. Restart the `osd.1` daemon.

```
[ceph: root@clienta /]# ceph orch daemon restart osd.1
Scheduled to restart osd.1 on host 'serverd.lab.example.com'
```

- 5.5. Verify that the runtime value is reverted.

```
[ceph: root@clienta /]# ceph tell osd.1 config get debug_ms
{
    "debug_ms": "10/10"
}
```

▶ 6. Return to workstation as the student user.

```
[ceph: root@clienta /]# exit
[admin@clienta ~]$ exit
[student@workstation ~]$
```

▶ 7. Use a web browser to access the Ceph Dashboard GUI and edit the `mon_allow_pool_delete` advanced configuration setting to `true` in the `global` section.

- 7.1. Open a web browser and navigate to `https://serverc:8443`. If necessary, accept the certificate warning. The URL might redirect to `serverd` or `servere`, depending on the active MGR. Log in with user name `admin` and password `redhat`.
- 7.2. In the Ceph Dashboard web UI, click **Cluster** → **Configuration** to display the **Configuration Settings** page.

The screenshot shows the Red Hat Ceph Storage Dashboard. On the left, a sidebar menu is open under the 'Cluster' section, listing options like Hosts, Inventory, Monitors, Services, OSDs, Configuration, CRUSH map, Manager Modules, Logs, Monitoring, and Pools. The 'Configuration' option is currently selected. The main area is divided into two sections: 'Status' and 'Capacity'. The 'Status' section contains several cards: 'Cluster Status' (HEALTH_OK), 'Hosts' (4 total), 'Monitors' (4 (quorum 0, 1, 2, 3)), 'OSDs' (15 total, 15 up, 15 in), 'Managers' (1 active, 3 standby), 'Object Gateways' (2 total), 'Metadata Servers' (no filesystems), and 'iSCSI Gateways' (0 total, 0 up, 0 down). The 'Capacity' section has two tabs: 'Raw Capacity' and 'Objects'.

Figure 3.1: Configuration settings in the Ceph Dashboard

- 7.3. Select the advanced option from the **Level** menu to view advanced configuration settings. Type `mon_allow_pool_delete` in the search bar to find the setting.

The screenshot shows the Red Hat Ceph Storage Dashboard with the 'Configuration' page selected. The search bar at the top contains the query `mon_allow_pool_delete`. The results table shows one entry: `mon_allow_pool_delete` with a current value of `false` and a default value of `false`. The 'Level' dropdown is set to 'advanced'. The table has columns for Name, Description, Current value, Default, and Edit. A note at the bottom of the table says `0 selected / 1 found / 1768 total`.

Figure 3.2: Configuration setting search in the Ceph Dashboard

- 7.4. Click `mon_allow_pool_delete` and then click **Edit**.

The screenshot shows the Red Hat Ceph Storage Dashboard. The left sidebar is titled 'Cluster' and includes options like Hosts, Inventory, Monitors, Services, OSDs, Configuration, CRUSH map, Manager Modules, Logs, Monitoring (9), Pools, and Block. The 'Configuration' option is currently selected. The main pane shows a table with columns: Name, Description, Current value, Default, and Edit. A search bar at the top right shows 'mon_allow_pool_del...'. The table has one row: 'mon_allow_pool_delete' with a current value of 'false' and a default value of 'false'. The status bar at the bottom indicates '1 selected / 1 found / 1768 total'.

Figure 3.3: Editing configuration settings in the Ceph Dashboard

75. Edit the `mon_allow_pool_delete` setting, set the `global` value to `true`, and then click **Save**.

This screenshot shows the configuration dialog for the 'mon_allow_pool_delete' setting. The left sidebar is identical to Figure 3.3. The main area has a 'Default' field set to 'false' and a 'Services' field set to 'mon'. Below these, under 'Values', there is a table with rows for 'global', 'mon', 'mgr', 'osd', 'mds', and 'client'. The 'global' row has its value set to 'true' and is highlighted with a green border. The other rows have their values set to '-- Default --'. At the bottom are 'Cancel' and 'Update' buttons, with 'Update' being highlighted.

Figure 3.4: Modifying a configuration setting in the Ceph Dashboard

The dashboard displays a message to confirm the new setting.

Name	Description	Current value	Default	Edit
client_cache_size	soft maximum number of directory entries in client cache	16384	-	
cluster_addr	cluster-facing address to bind to	-	-	
container_image	container image (used by cephadm orchestrator)	global:registry.redhat.io/rhceph/rhceph-5-rhel8@sha256:6306de945a6c940439ab584aba9b622f2aa6222947d3d4cde75a4b82649a47ff	docker.io/ceph/daemon-base:latest-pacific-devel	
device_failure_prio	Method used to predict device failures	none	-	
err_to_graylog	send critical error log lines to remote graylog server	false	-	
err_to_stderr	send critical error log lines to stderr	false	-	
err_to_syslog	send critical error log lines to syslog	false	-	

Figure 3.5: Modifying a configuration setting in the Ceph Dashboard

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 configure-settings
```

This concludes the guided exercise.

Configuring Cluster Monitors

Objectives

After completing this section, you should be able to describe the purpose of cluster monitors and the quorum procedures, query the monitor map, manage the configuration database, and describe Cephx.

Configuring Ceph Monitors

Ceph monitors (MONs) store and maintain the cluster map that clients use to find the MON and OSD nodes. Ceph clients must connect to a MON to retrieve the cluster map before they can read or write any data to OSDs. For this reason, the proper configuration of cluster MONs is critical.

MONs form a quorum and elect a *leader* by using a variation of the Paxos algorithm, to achieve consensus among a distributed set of computers.

MONs each have one of the following roles:

- *Leader*: the first MON to obtain the most recent version of the cluster map.
- *Provider*: a MON that has the most recent version of the cluster map, but is not the leader.
- *Requester*: a MON that does not have the most recent version of the cluster map and must synchronize with a provider before it can rejoin the quorum.

Synchronization always occurs when a new MON joins the cluster. Each MON periodically checks whether a neighboring monitor has a more recent version of the cluster map. If a MON does not have the most recent version of the cluster map, then it must synchronize and obtain it.

A majority of the MONs in a cluster must be running to establish a quorum. For example, if five MONs are deployed, then three must be running to establish a quorum. Deploy at least three MON nodes in your production Ceph cluster to ensure high availability. You can add or remove MONs in a running cluster.

The cluster configuration file defines the MON host IP addresses and ports for the cluster to operate. The `mon_host` setting can contain IP addresses or DNS names. The `cephadm` tool does not update the cluster configuration file. Define a strategy to keep the cluster configuration files synchronized across cluster nodes, such as with `rsync`.

```
[global]
mon_host = [v2:172.25.250.12:3300,v1:172.25.250.12:6789],
[v2:172.25.250.13:3300,v1:172.25.250.13:6789],
[v2:172.25.250.14:3300,v1:172.25.250.14:6789]
```



Important

Changing MON node IP addresses is not recommended after the cluster is deployed and running.

Viewing the Monitor Quorum

Verify the MON quorum status by using the `ceph status` or `ceph mon stat` commands.

```
[ceph: root@node /]# ceph mon stat
e4: 4 mons at {nodea=[v2:172.25.250.10:3300/0,v1:172.25.250.10:6789/0],
nodeb=[v2:172.25.250.12:3300/0,v1:172.25.250.12:6789/0],
nodec=[v2:172.25.250.13:3300/0,v1:172.25.250.13:6789/0]}, election epoch 66,
leader 0 nodea, quorum 0,1,2 nodea,nodeb,nodec
```

Alternately, use the `ceph quorum_status` command. Add the `-f json-pretty` option to create a more readable output.

```
[ceph: root@node /]# ceph quorum_status -f json-pretty
{
    "election_epoch": 5,
    "quorum": [
        0,
        1,
        2
    ],
    "quorum_names": [
        "nodea",
        "nodeb",
        "nodec"
    ],
    "quorum_leader_name": "nodea",
    "quorum_age": 1172,
    "features": {
        ...output omitted...
    }
}
```

You can also view the status of MONs in the Dashboard. In the Dashboard, click **Cluster** → **Monitors** to view the status of the Monitor nodes and quorum.

Analyzing the Monitor Map

The Ceph cluster map contains the MON map, OSD map, PG map, MDS map, and CRUSH map.

The MON map contains the cluster *fsid* (File System ID), and the name, IP address, and network port to communicate with each MON node. The *fsid* is a unique, auto-generated identifier (UUID) that identifies the Ceph cluster.

The MON map also keeps map version information, such as the epoch and time of the last change. MON nodes maintain the map by synchronizing changes and agreeing on the current version.

Use the `ceph mon dump` command to view the current MON map.

```
[ceph: root@node /]# ceph mon dump
epoch 4
fsid 11839bde-156b-11ec-bb71-52540000fa0c
last_changed 2021-09-14T14:54:23.611787+0000
created 2021-09-14T14:50:37.372360+0000
min_mon_release 16 (pacific)
election_strategy: 1
```

```
0: [v2:172.25.250.12:3300/0,v1:172.25.250.12:6789/0] mon.serverc
1: [v2:172.25.250.13:3300/0,v1:172.25.250.13:6789/0] mon.serverd
2: [v2:172.25.250.14:3300/0,v1:172.25.250.14:6789/0] mon.servere
dumped monmap epoch 4
```

Managing the Centralized Configuration Database

The MON nodes store and maintain the centralized configuration database. The default location of the database on each MON node is `/var/lib/ceph/$fsid/mon.$host/store.db`. It is not recommended to change the location of the database.

The database might grow large over time. Run the `ceph tell mon.$id compact` command to compact the database to improve performance. Alternately, set the `mon_compact_on_start` configuration to `true` to compact the database on each daemon start:

```
[ceph: root@node /]# ceph config set mon mon_compact_on_start true
```

Define threshold settings that trigger a change in health status based on the database size.

Description	Setting	Default
Change the cluster health status to <code>HEALTH_WARN</code> when the configuration database exceeds this size.	<code>mon_data_size_warn</code>	15 (GB)
Change the cluster health status to <code>HEALTH_WARN</code> when the file system that holds the configuration database has a remaining capacity that is less than or equal to this percentage.	<code>mon_data_avail_warn</code>	30 (%)
Change the cluster health status to <code>HEALTH_ERR</code> when the file system that holds the configuration database has a remaining capacity that is less than or equal to this percentage.	<code>mon_data_avail_crit</code>	5 (%)

Cluster Authentication

Ceph uses the Cephx protocol by default for cryptographic authentication between Ceph components, using shared secret keys for authentication. Deploying the cluster with `cephadm` enables Cephx by default. You can disable Cephx if needed, but it is not recommended because it weakens cluster security. To enable or disable the Cephx protocol, use the `ceph config set` command to manage multiple settings.

```
[ceph: root@node /]# ceph config get mon auth_service_required  
cephx  
[ceph: root@node /]# ceph config get mon auth_cluster_required  
cephx  
[ceph: root@node /]# ceph config get mon auth_client_required  
cephx
```

The /etc/ceph directory and daemon data directories contain the Cephx key-ring files. For MONs, the data directory is /var/lib/ceph/\$fsid/mon.\$host/.



Note

Key-ring files store the secret key as plain text. Secure them with appropriate Linux file permissions.

Use the `ceph auth` command to create, view, and manage cluster keys. Use the `ceph-authtool` command to create key-ring files.

The following command creates a key-ring file for the MON nodes.

```
[ceph: root@node /]# ceph-authtool --create-keyring /tmp/ceph.mon.keyring \  
--gen-key -n mon. --cap mon 'allow *'  
creating /tmp/ceph.mon.keyring
```

The `cephadm` tool creates a `client.admin` user in the /etc/ceph directory, which allows you to run administrative commands and to create other Ceph client user accounts.



References

For more information, refer to the *Monitor Configuration Reference* chapter in the *Red Hat Ceph Storage 5 Configuration Guide* at
https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/configuration_guide/index

► Guided Exercise

Configuring Cluster Monitors

In this exercise, you view the cluster settings for the monitor (MON) nodes.

Outcomes

You should be able to view the cluster quorum settings, quorum status, and MON map. You should be able to analyze cluster authentication settings and to compact the monitor configuration database.

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 configure-monitor
```

This command confirms that the required hosts for this exercise are accessible.

Instructions

- 1. Log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]#
```

- 2. Use the `ceph status` command to view the cluster quorum status.

```
[admin@clienta ~]$ ceph status
      id: 472b24e2-1821-11ec-87d7-52540000fa0c
      health: HEALTH_OK

      services:
        mon: 4 daemons, quorum serverc.lab.example.com,serverd,servere,clienta (age 1h)
        ...output omitted...
```

- 3. View the MON map.

```
[ceph: root@clienta /]# ceph mon dump
epoch 4
fsid 472b24e2-1821-11ec-87d7-52540000fa0c
last_changed 2021-09-20T01:41:44.138014+0000
created 2021-09-18T01:39:57.614592+0000
min_mon_release 16 (pacific)
election_strategy: 1
```

```
0: [v2:172.25.250.12:3300/0,v1:172.25.250.12:6789/0] mon.serverc.lab.example.com
1: [v2:172.25.250.13:3300/0,v1:172.25.250.13:6789/0] mon.serverd
2: [v2:172.25.250.14:3300/0,v1:172.25.250.14:6789/0] mon.servere
3: [v2:172.25.250.10:3300/0,v1:172.25.250.10:6789/0] mon.clienta
dumped monmap epoch 4
```

- ▶ 4. View the value of the `mon_host` setting for the `serverc` MON.

```
[ceph: root@clienta /]# ceph config show mon.serverc.lab.example.com mon_host
[v2:172.25.250.10:3300/0,v1:172.25.250.10:6789/0]
[v2:172.25.250.12:3300/0,v1:172.25.250.12:6789/0]
[v2:172.25.250.13:3300/0,v1:172.25.250.13:6789/0]
[v2:172.25.250.14:3300/0,v1:172.25.250.14:6789/0]
```

- ▶ 5. View the MON IP, port information, and quorum status in the MON stats.

```
[ceph: root@clienta /]# ceph mon stat
e4: 4 mons at {clienta=[v2:172.25.250.10:3300/0,v1:172.25.250.10:6789/0],
serverc.lab.example.com=[v2:172.25.250.12:3300/0,v1:172.25.250.12:6789/0],
serverd=[v2:172.25.250.13:3300/0,v1:172.25.250.13:6789/0],
servere=[v2:172.25.250.14:3300/0,v1:172.25.250.14:6789/0]},
election epoch 66, leader 0 serverc.lab.example.com, quorum 0,1,2,3
serverc.lab.example.com,serverd,servere,clienta
```

- ▶ 6. Use the `ceph auth ls` command to view the cluster authentication settings.

```
[ceph: root@clienta /]# ceph auth ls
osd.0
key: AQDiQ0Vh905yKBAAI0l2a+53fcCbNVBeJ AeJvQ==
caps: [mgr] allow profile osd
caps: [mon] allow profile osd
caps: [osd] allow *
...output omitted...
client.admin
key: AQBsq0Vhe3UqMhAA+44H1rvTDjgBPvdDGXBK+A==
caps: [mds] allow *
caps: [mgr] allow *
caps: [mon] allow *
caps: [osd] allow *
...output omitted...
mgr.serverc.lab.example.com.xgbgpo
key: AQBtQ0VhFCRnDhAAbnmlnzHbcmq2akAjZuigg==
caps: [mds] allow *
caps: [mon] profile mgr
caps: [osd] allow *
...output omitted...
```

- 7. Export the `client.admin` key ring.

**Note**

The admin key ring is stored by default in the `/etc/ceph/ceph.client.admin.keyring` file.

```
[ceph: root@clienta /]# ceph auth get client.admin -o /tmp/adminkey
exported keyring for client.admin
[ceph: root@clienta /]# cat /tmp/adminkey
[client.admin]
key = AQBsQ0Vhe3UqMhAA+44H1rvTDjgBPvdDGXBK+A==
caps mds = "allow *"
caps mgr = "allow *"
caps mon = "allow *"
caps osd = "allow *"
```

- 8. Verify the space that the MON database uses on `serverc`. Set the option to compact the MON database on start. Use `ceph orch` to restart the MON daemons and wait for the cluster to reach a healthy state. Verify again the space of the MON database on `serverc`.

**Note**

Your cluster is expected to have a different size than in the examples.

- 8.1. Verify the space that the MON database uses on `serverc`. The name of the `fsid` folder inside `/var/lib/ceph` can be different in your environment.

```
[ceph: root@clienta /]# exit
exit
[admin@clienta ~]$ ssh serverc sudo du -sch \
/var/lib/ceph/472b...a0c/mon.serverc.lab.example.com/store.db/
admin@serverc's password: redhat
74M /var/lib/ceph/472b...a0c/mon.serverc.lab.example.com/store.db/
74M total
```

- 8.2. Set the option to compact the MON database on start. Use `ceph orch` to restart the MON daemons, then wait for the cluster to reach a healthy state. This process can take many seconds.

```
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]# ceph config set mon mon_compact_on_start true
[ceph: root@clienta /]# ceph orch restart mon
Scheduled to restart mon.clienta on host 'clienta.lab.example.com'
Scheduled to restart mon.serverc.lab.example.com on host 'serverc.lab.example.com'
Scheduled to restart mon.serverd on host 'serverd.lab.example.com'
Scheduled to restart mon.servere on host 'servere.lab.example.com'
[ceph: root@clienta /]# ceph health
HEALTH_OK
```

- 8.3. Exit the `cephadm` shell, then verify the current space of the MON database on `serverc`.

```
[ceph: root@clienta /]# exit  
exit  
[admin@clienta ~]$ ssh serverc sudo du -sch \  
/var/lib/ceph/472b...a0c/mon.serverc.lab.example.com/store.db/  
admin@serverc's password: redhat  
71M /var/lib/ceph/472b...a0c/mon.serverc.lab.example.com/store.db/  
71M total
```



Note

The MON database compact process can take a few minutes depending on the size of the database. If the file size is bigger than before, then wait a few seconds until the file is compacted.

- 9. Return to workstation as the student user.

```
[admin@clienta ~]$ exit  
[student@workstation ~]$
```

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 configure-monitor
```

This concludes the guided exercise.

Configuring Cluster Networking

Objectives

After completing this section, you should be able to describe the purpose for each of the cluster networks, and view and modify the network configuration.

Configuring the Public and Cluster Networks

The `public` network is the default network for all Ceph cluster communication. The `cephadm` tool assumes that the network of the first MON daemon IP address is the `public` network. New MON daemons are deployed in the `public` network unless you explicitly define a different network.

Ceph clients make requests directly to OSDs over the cluster's `public` network. OSD replication and recovery traffic uses the `public` network unless you configure a separate `cluster` network for this purpose.

Configuring a separate `cluster` network might improve cluster performance by decreasing the `public` network traffic load and separating client traffic from back-end OSD operations traffic.

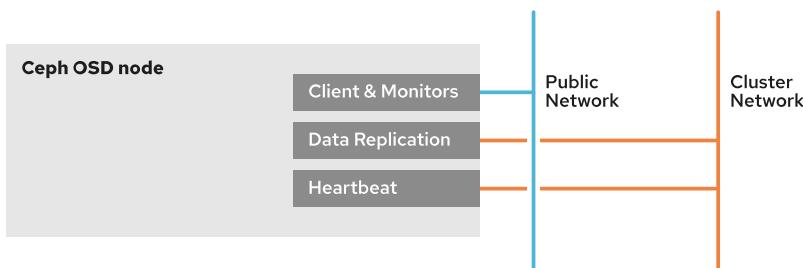


Figure 3.6: OSD network communication

Configure the nodes for a separate `cluster` network by performing the following steps.

- Configure an additional network interface on each cluster node.
- Configure the appropriate `cluster` network IP addresses on the new network interface on each node.
- Use the `--cluster-network` option of the `cephadm bootstrap` command to create the `cluster` network at the cluster bootstrap.

You can use a cluster configuration file to set `public` and `cluster` networks. You can configure more than one subnet for each network, separated by commas. Use CIDR notation for the subnets (for example, `172.25.250.0/24`).

```

[global]
public_network = 172.25.250.0/24,172.25.251.0/24
cluster_network = 172.25.249.0/24
  
```

**Important**

If you configure multiple subnets for a network, those subnets must be able to route to each other.

The `public` and `cluster` networks can be changed with the `ceph config set` command or with the `ceph config assimilate-conf` command.

Configuring Individual Daemons

MON daemons bind to specific IP addresses, but MGR, OSD, and MDS daemons bind to any available IP address by default. In Red Hat Ceph Storage 5, `cephadm` deploys daemons in arbitrary hosts by using the `public` network for most services. To handle where `cephadm` deploys new daemons, you can define a specific subnet for a service to use. Only hosts that have an IP address in the same subnet are considered for deployment of that service.

To set the `172.25.252.0/24` subnet to MON daemons:

```
[ceph: root@node /]# ceph config set mon public_network 172.25.252.0/24
```

The example command is the equivalent of the following `[mon]` section in a cluster configuration file.

```
[mon]
public_network = 172.25.252.0/24
```

Use the `ceph orch daemon add` command to manually deploy daemons to a specific subnet or IP address.

```
[ceph: root@node /]# ceph orch daemon add mon cluster-host02:172.25.251.0/24
[ceph: root@node /]# ceph orch daemon rm mon.cluster-host01
```

Using runtime `ceph orch daemon` commands for configuration changes is not recommended. Instead, use service specification files as the recommended method for managing Ceph clusters.

Running IPv6

The default value of the `ms_bind_ipv4` setting is `true` for the cluster and the value of the `ms_bind_ipv6` setting is `false`. To bind Ceph daemons to IPv6 addresses, set `ms_bind_ipv6` to `true` and set `ms_bind_ipv4` to `false` in a cluster configuration file.

```
[global]
public_network = <IPv6 public-network/netmask>
cluster_network = <IPv6 cluster-network/netmask>
```

Enabling Jumbo Frames

Configuring the network MTU to support jumbo frames is a recommended practice on storage networks and might improve performance. Configure an MTU value of 9000 on the `cluster` network interface to support jumbo frames.

**Important**

All nodes and networking devices in a communication path must have the same MTU value. For bonded network interfaces, set the MTU value on the bonded interface and the underlying interfaces will inherit the same MTU value.

Configuring Network Security

Configuring a separate cluster network might also increase cluster security and availability, by reducing the attack surface over the public network, thwarting some types of Denial of Service (DoS) attacks against the cluster, and preventing traffic disruption between OSDs. When traffic between OSDs gets disrupted, clients are prevented from reading and writing data.

Separating back-end OSD traffic onto its own network might help to prevent data breaches over the public network. To secure the back-end cluster network, ensure that traffic is not routed between the cluster and public networks.

Configuring Firewall Rules

Ceph OSD and MDS daemons bind to TCP ports in the 6800 to 7300 range by default. To configure a different range, change the `ms_bind_port_min` and `ms_bind_port_max` settings.

The following table lists the default ports for Red Hat Ceph Storage 5.

Default Ports by Red Hat Ceph Storage Service

Service name	Ports	Description
Monitor (MON)	6789/TCP (msgr), 3300/TCP (msgr2)	Communication within the Ceph cluster
OSD	6800-7300/TCP	Each OSD uses three ports in this range: one for communicating with clients and MONs over the public network; one for sending data to other OSDs over a cluster network, or over the public network if the former does not exist; and another for exchanging heartbeat packets over a cluster network or over the public network if the former does not exist.
Metadata Server (MDS)	6800-7300/TCP	Communication with the Ceph Metadata Server
Dashboard/Manager (MGR)	8443/TCP	Communication with the Ceph Manager Dashboard over SSL
Manager RESTful Module	8003/TCP	Communication with the Ceph Manager RESTful module over SSL
Manager Prometheus Module	9283/TCP	Communication with the Ceph Manager Prometheus plug-in
Prometheus Alertmanager	9093/TCP	Communication with the Prometheus Alertmanager service

Service name	Ports	Description
Prometheus Node Exporter	9100/TCP	Communication with the Prometheus Node Exporter daemon
Grafana server	3000/TCP	Communication with the Grafana service
Ceph Object Gateway (RGW)	80/TCP	Communication with Ceph RADOSGW. If the <code>client.rgw</code> configuration section is empty, <code>cephadm</code> uses the default port 80.
Ceph iSCSI Gateway	9287/TCP	Communication with Ceph iSCSI Gateway

MONs always operate on the public network. To secure MON nodes for firewall rules, configure rules with the public interface and public network IP address. You can do it by manually adding the port to the firewall rules.

```
[root@node ~]# firewall-cmd --zone=public --add-port=6789/tcp
[root@node ~]# firewall-cmd --zone=public --add-port=6789/tcp --permanent
```

You can also secure MON nodes by adding the ceph-mon service to the firewall rules.

```
[root@node ~]# firewall-cmd --zone=public --add-service=ceph-mon
[root@node ~]# firewall-cmd --zone=public --add-service=ceph-mon --permanent
```

To configure a cluster network, OSDs need rules for both the public and cluster networks. Clients connect to OSDs by using the public network and OSDs communicate with each other over the cluster network.

To secure OSD nodes for firewall rules, configure rules with the appropriate network interface and IP address.

```
[root@node ~]# firewall-cmd --zone=<public-or-cluster> --add-port=6800-7300/tcp
[root@node ~]# firewall-cmd --zone=<public-or-cluster> \
--add-port=6800-7300/tcp --permanent
```

You can also secure OSD nodes by adding the ceph service to the firewall rules.

```
[root@node ~]# firewall-cmd --zone=<public-or-cluster> --add-service=ceph
[root@node ~]# firewall-cmd --zone=<public-or-cluster> \
--add-service=ceph --permanent
```



References

For more information, refer to the *Network Configuration Reference* chapter in the *Red Hat Ceph Storage 5 Configuration Guide* at
https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/configuration_guide/index#ceph-network-configuration

► Guided Exercise

Configuring Cluster Networking

In this exercise, you view and modify the network configuration.

Outcomes

You should be able to configure public and cluster network settings and secure the cluster with firewall rules.

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 configure-network
```

This command confirms that the required hosts for this exercise are accessible.

Instructions

- 1. Log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]#
```

- 2. Use the `ceph health` command to view the health of the cluster.

```
[ceph: root@clienta /]# ceph health
HEALTH_OK
```

- 3. View the configured `public_network` setting for the OSD and MON services.

```
[ceph: root@clienta /]# ceph config get osd public_network
[ceph: root@clienta /]# ceph config get mon public_network
[ceph: root@clienta /]#
```

- ▶ 4. Exit the cephadm shell. Create the osd-cluster-network.conf file and add a public_network setting with the IPv4 network address value of 172.25.250.0/24 in the [osd] section.

```
[ceph: root@clienta /]# exit  
exit  
[admin@clienta ~]$ cat osd-cluster-network.conf  
[osd]  
cluster network = 172.25.249.0/24
```

- ▶ 5. Use the cephadm shell with the --mount option to mount the osd-cluster-network.conf file in the default location (/mnt). Use the ceph config assimilate-conf command with the public-network.conf file to apply the configuration. Verify that cluster-network is defined for the service.
- 5.1. Use the cephadm shell with the --mount option to mount the osd-cluster-network.conf file and verify the integrity of the file.

```
[admin@clienta ~]$ sudo cephadm shell --mount osd-cluster-network.conf  
[ceph: root@clienta /]# cat /mnt/osd-cluster-network.conf  
[osd]  
public network = 172.25.250.0/24
```

- 5.2. Use the ceph config assimilate-conf command with the osd-cluster-network.conf file to apply the configuration. Verify that cluster_network is defined for the service.

```
[ceph: root@clienta /]# ceph config assimilate-conf \  
-i /mnt/osd-cluster-network.conf  
[ceph: root@clienta /]# ceph config get osd cluster_network  
172.25.249.0/24
```

- ▶ 6. Use the ceph config command to set the public_network setting to 172.25.250.0/24 for the MON services. Verify that the service has the new setting. Exit the cephadm shell.

```
[ceph: root@clienta /]# ceph config set mon public_network 172.25.250.0/24  
[ceph: root@clienta /]# ceph config get mon public_network  
172.25.250.0/24  
[ceph: root@clienta /]# exit  
exit  
[admin@clienta ~]$
```



Note

You must restart the cluster for this setting to take effect. Omit that step for this exercise, to save time.

- ▶ 7. Log in to serverc as the admin user and switch to the root user. Configure a firewall rule to secure the MON service on serverc.

```
[admin@clienta ~]$ ssh admin@serverc
admin@serverc's password: redhat
[admin@serverc ~]$ sudo -i
[root@serverc ~]# firewall-cmd --zone=public --add-service=ceph-mon
success
[root@serverc ~]# firewall-cmd --zone=public --add-service=ceph-mon --permanent
success
[root@serverc ~]#
```

- 8. Configure a firewall rule to secure the OSD services on serverc.

```
[root@serverc ~]# firewall-cmd --zone=public --add-service=ceph
success
[root@serverc ~]# firewall-cmd --zone=public --add-service=ceph --permanent
success
[root@serverc ~]#
```

- 9. Increase the MTU for the cluster network interface to support jumbo frames.

```
[root@serverc ~]# nmcli conn modify 'Wired connection 2' 802-3-ethernet.mtu 9000
[root@serverc ~]# nmcli conn down 'Wired connection 2'
Connection 'Wired connection 2' successfully deactivated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/10)
[root@serverc ~]# nmcli conn up 'Wired connection 2'
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/11)
[root@serverc ~]# ip link show eth1
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8942 qdisc fq_codel state UP mode
    DEFAULT group default qlen 1000
        link/ether 52:54:00:01:fa:0c brd ff:ff:ff:ff:ff:ff
```

- 10. Return to workstation as the student user.

```
[root@serverc ~]# exit
[admin@serverc ~]$ exit
[admin@clienta ~]$ exit
[student@workstation ~]$
```

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 configure-network
```

This concludes the guided exercise.

► Lab

Configuring a Red Hat Ceph Storage Cluster

In this lab, you query and modify Red Hat Ceph Storage configuration settings.

Outcomes

You should be able to configure cluster settings.

Before You Begin

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

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

This command confirms that the required hosts for this exercise are accessible.

Instructions

Configure Ceph cluster settings using both the command line and Ceph Dashboard GUI. View MON settings and configure firewall rules for MON and RGW nodes.

1. Configure your Red Hat Ceph Storage cluster settings. Set `mon_data_avail_warn` to 15 and `mon_max_pg_per_osd` to 400. These changes must persist across cluster restarts.
2. Configure the `mon_data_avail_crit` setting to 10 by using the Ceph Dashboard GUI.
3. Display the MON map and view the cluster quorum status.
4. Configure firewall rules for the MON and RGW nodes on `serverd`.
5. Return to `workstation` as the student user.

Evaluation

Grade your work by running the `lab grade configure-review` command from your `workstation` machine. Correct any reported failures and rerun the script until successful.

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

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 configure-review
```

This concludes the lab.

► Solution

Configuring a Red Hat Ceph Storage Cluster

In this lab, you query and modify Red Hat Ceph Storage configuration settings.

Outcomes

You should be able to configure cluster settings.

Before You Begin

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

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

This command confirms that the required hosts for this exercise are accessible.

Instructions

Configure Ceph cluster settings using both the command line and Ceph Dashboard GUI. View MON settings and configure firewall rules for MON and RGW nodes.

- Configure your Red Hat Ceph Storage cluster settings. Set `mon_data_avail_warn` to 15 and `mon_max_pg_per_osd` to 400. These changes must persist across cluster restarts.
 - Log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell.
Configure `mon_data_avail_warn` to 15 and `mon_max_pg_per_osd` to 400.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]# ceph config set mon mon_data_avail_warn 15
[ceph: root@clienta /]# ceph config set mon mon_max_pg_per_osd 400
```

- Verify the new values for each setting.

```
[ceph: root@clienta /]# ceph config get mon.serverc mon_data_avail_warn
15
[ceph: root@clienta /]# ceph config get mon.serverc mon_max_pg_per_osd
400
```

- Configure the `mon_data_avail_crit` setting to 10 by using the Ceph Dashboard GUI.
 - Open a web browser and go to `https://serverc:8443`. If necessary, accept the certificate warning. If the URL redirects to the active MGR node, you might need to accept the certificate warning again.
 - Log in as the `admin` user, with `redhat` as the password.

- 2.3. In the Ceph Dashboard web UI, click **Cluster** → **Configuration** to display the **Configuration Settings** page.
 - 2.4. Select the advanced option from the **Level** menu to view advanced configuration settings. Type `mon_data_avail_crit` in the search bar.
 - 2.5. Click `mon_data_avail_crit` and then click **Edit**.
 - 2.6. Set the `global` value to `10` and then click **Update**.
 - 2.7. Verify that a message indicates that the configuration option is updated.
3. Display the MON map and view the cluster quorum status.
 - 3.1. Display the MON map.

```
[ceph: root@clienta /]# ceph mon dump
epoch 4
fsid 472b24e2-1821-11ec-87d7-52540000fa0c
last_changed 2021-09-20T01:41:44.138014+0000
created 2021-09-18T01:39:57.614592+0000
min_mon_release 16 (pacific)
election_strategy: 1
0: [v2:172.25.250.12:3300/0,v1:172.25.250.12:6789/0] mon.serverc.lab.example.com
1: [v2:172.25.250.13:3300/0,v1:172.25.250.13:6789/0] mon.serverd
2: [v2:172.25.250.14:3300/0,v1:172.25.250.14:6789/0] mon.servere
3: [v2:172.25.250.10:3300/0,v1:172.25.250.10:6789/0] mon.clienta
dumped monmap epoch 4
```

- 3.2. Display the cluster quorum status.

```
[ceph: root@clienta /]# ceph mon stat
e4: 4 mons at {clienta=[v2:172.25.250.10:3300/0,v1:172.25.250.10:6789/0],
serverc.lab.example.com=[v2:172.25.250.12:3300/0,v1:172.25.250.12:6789/0],
serverd=[v2:172.25.250.13:3300/0,v1:172.25.250.13:6789/0],
servere=[v2:172.25.250.14:3300/0,v1:172.25.250.14:6789/0]},
election epoch 66, leader 0 serverc.lab.example.com, quorum 0,1,2,3
serverc.lab.example.com,serverd,servere,clienta
```

4. Configure firewall rules for the MON and RGW nodes on `serverd`.

- 4.1. Exit the `cephadm` shell. Log in to `serverd` as the `admin` user and switch to the `root` user.
Configure a firewall rule for the MON node on `serverd`.

```
[ceph: root@clienta /]# exit
exit
[admin@servera ~]$ ssh serverd
admin@serverd's password: redhat
[admin@serverd ~]$ sudo -i
[root@serverd ~]# firewall-cmd --zone=public --add-service=ceph-mon
success
[root@serverd ~]# firewall-cmd --zone=public --add-service=ceph-mon --permanent
success
```

4.2. Configure a firewall rule for the RGW node on serverd.

```
[root@serverd ~]# firewall-cmd --zone=public --add-port=7480/tcp  
success  
[root@serverd ~]# firewall-cmd --zone=public --add-port=7480/tcp --permanent  
success
```

5. Return to workstation as the student user.

5.1. Return to workstation as the student user.

```
[root@serverd ~]# exit  
[admin@serverd ~]$ exit  
[root@clienta ~]$ exit  
[admin@clienta ~]$ exit  
[student@workstation ~]$
```

Evaluation

Grade your work by running the `lab grade configure-review` command from your workstation machine. Correct any reported failures and rerun the script until successful.

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

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 configure-review
```

This concludes the lab.

Summary

In this chapter, you learned:

- Most cluster configuration settings are stored in the cluster configuration database on the MON nodes. The database is automatically synchronized across MONs.
- Certain configuration settings, such as cluster boot settings, can be stored in the cluster configuration file. The default file name is `ceph.conf`. This file must be synchronized manually between all cluster nodes.
- Most configuration settings can be modified when the cluster is running. You can change a setting temporarily or make it persistent across daemon restarts.
- The MON map holds the MON cluster quorum information that can be viewed with `ceph` commands or with the dashboard. You can configure MON settings to ensure high cluster availability.
- Cephx provides cluster authentication via shared secret keys. The `client.admin` key ring is required for administering the cluster.
- Cluster nodes operate across the `public` network. You can configure an additional `cluster` network to separate OSD replication, heartbeat, backfill, and recovery traffic. Cluster performance and security might be increased by configuring a `cluster` network.
- You can use firewall rules to secure communication to cluster nodes.

Chapter 4

Creating Object Storage Cluster Components

Goal

Create and manage the components that comprise the object storage cluster, including OSDs, pools, and the cluster authorization method.

Objectives

- Describe OSD configuration scenarios and create BlueStore OSDs using ceph-volume.
- Describe and compare replicated and erasure coded pools, and create and configure each pool type.
- Describe Cephx and configure user authentication and authorization for Ceph clients.

Sections

- Creating BlueStore OSDs Using Logical Volumes (and Guided Exercise)
- Creating and Configuring Pools (and Guided Exercise)
- Managing Ceph Authentication (and Guided Exercise)

Lab

Creating Object Storage Cluster Components

Creating BlueStore OSDs Using Logical Volumes

Objectives

After completing this section, you should be able to describe OSD configuration scenarios and create BlueStore OSDs using cephadm.

Introducing BlueStore

BlueStore replaced FileStore as the default storage back end for OSDs. FileStore stores objects as files in a file system (Red Hat recommends XFS) on top of a block device. BlueStore stores objects directly on raw block devices and eliminates the file-system layer, which improves read and write operation speeds.

FileStore is deprecated. Continued use of FileStore in RHCS 5 requires a Red Hat support exception. Newly created OSDs, whether by cluster growth or disk replacement, use BlueStore by default.

BlueStore Architecture

Objects that are stored in a Ceph cluster have a cluster-wide unique identifier, binary object data, and object metadata. BlueStore stores the object metadata in the *block database*. The block database stores metadata as key-value pairs in a *RocksDB* database, which is a high-performing key-value store.

The block database resides on a small BlueFS partition on the storage device. BlueFS is a minimal file system that is designed to hold the RocksDB files.

BlueStore writes data to block storage devices by utilizing the *write-ahead log (WAL)*. The write-ahead log performs a journaling function and logs all transactions.

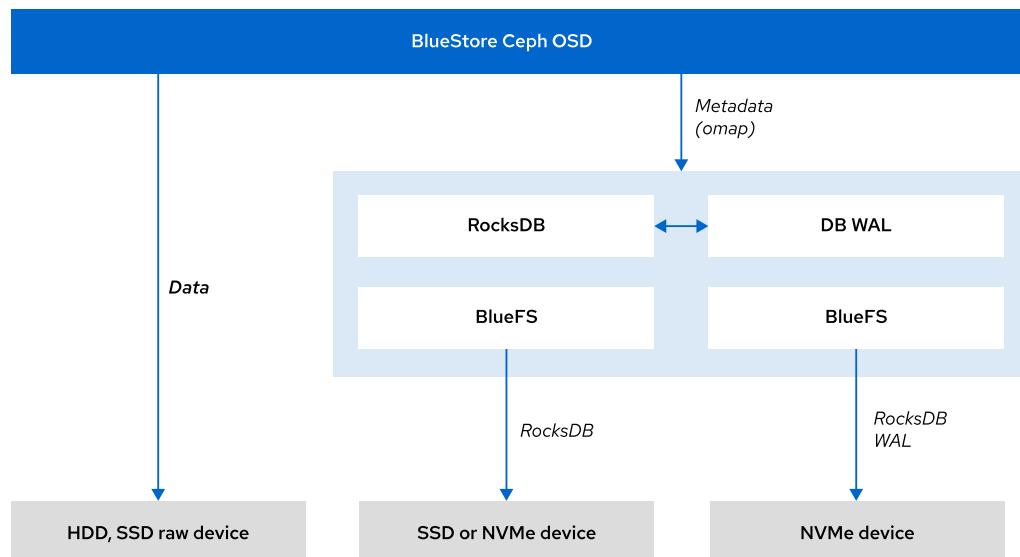


Figure 4.1: BlueStore OSD layout

BlueStore Performance

FileStore writes to a journal and then writes from the journal to the block device. BlueStore avoids this double-write performance penalty by writing data directly to the block device and logging transactions to the write-ahead log simultaneously with a separate data stream. BlueStore write operations are approximately twice as fast as FileStore with similar workloads.

When using a mix of different cluster storage devices, customize BlueStore OSDs to improve performance. When you create a BlueStore OSD, the default is to place the data, block database, and write-ahead log all on the same block device. Many of the performance advantages come from the block database and the write-ahead log, so placing those components on separate, faster devices might improve performance.



Note

You might improve performance by moving BlueStore devices if the new device is faster than the primary storage device. For example, if object data is on HDD devices, then improve performance by placing the block database on SSD devices and the write-ahead log on NVMe devices.

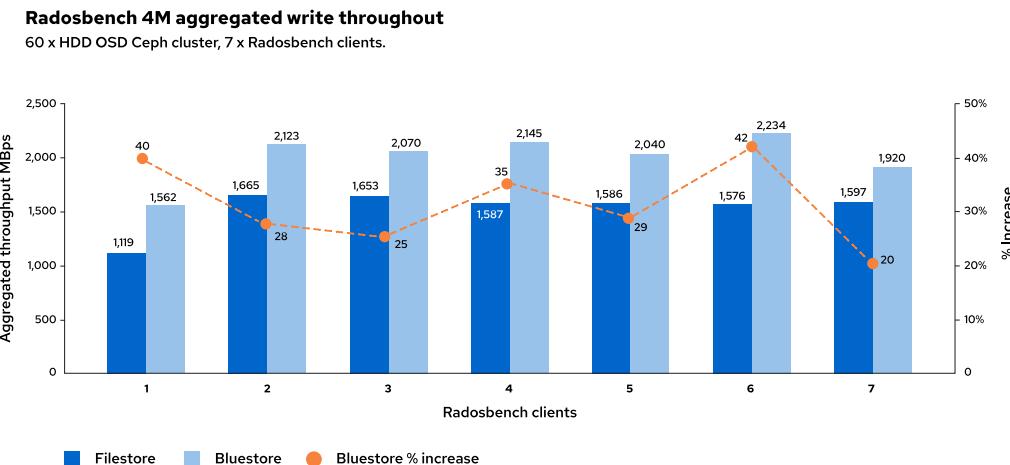
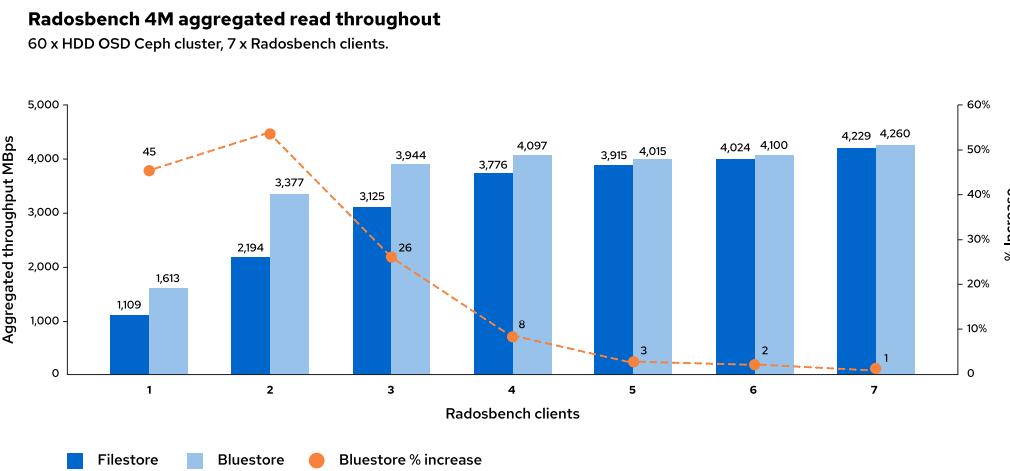
Use service specification files to define the location of the BlueStore data, block database, and write-ahead log devices. The following example specifies the BlueStore devices for an OSD service.

```
service_type: osd
service_id: osd_example
placement:
  host_pattern: '*'
data_devices:
  paths:
    - /dev/vda
db_devices:
  paths:
    - /dev/nvme0
wal_devices:
  paths:
    - /dev/nvme1
```

The BlueStore storage back end provides the following features:

- Allows use of separate devices for the data, block database, and write-ahead log (WAL).
- Supports use of virtually any combination of HDD, SSD, and NVMe devices.
- Operates over raw devices or partitions, eliminating double writes to storage devices, with increased metadata efficiency.
- Writes all data and metadata with checksums. All read operations are verified with their corresponding checksums before returning to the client.

The following graphs show the performance of BlueStore versus the deprecated FileStore architecture.

**Figure 4.2: FileStore versus BlueStore write throughput****Figure 4.3: FileStore versus BlueStore read throughput**

BlueStore runs in user space, manages its own cache and database, and can have a lower memory footprint than FileStore. BlueStore uses RocksDB to store key-value metadata. BlueStore is self-tuning by default, but you can manually tune BlueStore parameters if required.

The BlueStore partition writes data in chunks of the size of the `bluestore_min_alloc_size` parameter. The default value is 4 KiB. If the data to write is less than the size of the chunk, BlueStore fills the remaining space of the chunk with zeroes. It is a recommended practice to set the parameter to the size of the smallest typical write on the raw partition.

It is recommended to re-create FileStore OSDs as BlueStore to take advantage of the performance improvements and to maintain Red Hat support.

Introducing BlueStore Database Sharding

BlueStore can limit the size of large *omap* objects stored in RocksDB and distribute them into multiple column families. This process is known as *sharding*. When using sharding, the cluster groups keys that have similar access and modification frequency to improve performance and to save disk space. Sharding can alleviate the impacts of RocksDB compaction. RocksDB must reach a certain level of used space before compacting the database, which can affect OSD

performance. With sharding, these operations are independent from the used space level, allowing a more precise compaction and minimizing the effect on OSD performance.



Note

Red Hat recommends that the configured space for RocksDB is at least 4% of the data device size.

In Red Hat Ceph Storage 5, sharding is enabled by default. Sharding is not enabled in OSDs from clusters that are migrated from earlier versions. OSDs from clusters migrated from previous versions will not have sharding enabled.

Use `ceph config get` to verify whether sharding is enabled for an OSD and to view the current definition.

```
[ceph: root@node /]# ceph config get osd.1 bluestore_rocksdb_cf
true
[ceph: root@node /]# ceph config get osd.1 bluestore_rocksdb_cfs
m(3) p(3,0-12) 0(3,0-13)=block_cache={type=binned_lru} L P
```

The default values result in good performance in most Ceph use cases. The optimal sharding definition for your production cluster depends on several factors. Red Hat recommends use of default values unless you are faced with significant performance issues. In a production-upgraded cluster, you might want to weigh the performance benefits against the maintenance effort to enable sharding for RocksDB in a large environment.

You can use the BlueStore administrative tool, `ceph-bluestore-tool`, to reshards the RocksDB database without reprovisioning OSDs. To reshards an OSD, stop the daemon and pass the new sharding definition with the `--sharding` option. The `--path` option refers to the OSD data location, which defaults to `/var/lib/ceph/$fsid/osd.$ID/`.

```
[ceph: root@node /]# ceph-bluestore-tool --path <data path> \
--sharding="m(3) p(3,0-12) 0(3,0-13)= block_cache={type=binned_lru} L P" \
reshard
```

Provisioning BlueStore OSDs

As a storage administrator, you can use the Ceph Orchestrator service to add or remove OSDs in a cluster. To add an OSD, the device must meet the following conditions:

- The device must not have partitions.
- The device must not be mounted.
- The device must have at least 5 GB of space.
- The device must not contain a Ceph BlueStore OSD.

Use the `ceph orch device ls` command to list devices across the hosts in the cluster.

Hostname	Path	Type	Serial	Size	Health	Ident	Fault
Available							
nodea	/dev/vda	hdd	00000000-0000-0000-a	20G	Unknown	N/A	N/A Yes
nodea	/dev/vdb	hdd	00000000-0000-0000-b	20G	Unknown	N/A	N/A Yes
nodeb	/dev/vda	hdd	00000000-0000-0001-a	20G	Unknown	N/A	N/A Yes
nodeb	/dev/vdb	hdd	00000000-0000-0001-b	20G	Unknown	N/A	N/A Yes

Nodes with the **Yes** label in the **Available** column are candidates for OSD provisioning. To view only in-use storage devices, use the `ceph device ls` command.



Note

Some devices might not be eligible for OSD provisioning. Use the `--wide` option to view the details of why the cluster rejects the device.

To prepare a device for provisioning, Use the `ceph orch device zap` command. This command removes all partitions and purges the data in the device so it can be used for provisioning. Use the `--force` option to ensure the removal of any partition that a previous OSD might have created.

```
[ceph: root@node /]# ceph orch device zap node /dev/vda --force
```

Reviewing BlueStore Provisioning Methods

In RHCS 5, `cephadm` is the recommended tool to provision and manage OSDs. It uses the `ceph-volume` utility in the background for OSD operations. The `cephadm` tool might not see manual operations that use `ceph-volume`. It is recommended to limit manual `ceph-volume` OSD use cases to troubleshooting.

There are multiple ways to provision OSDs with `cephadm`. Consider the appropriate method according to the wanted cluster behavior.

Orchestrator-Managed Provisioning

The Orchestrator service can discover available devices among cluster hosts, add the devices, and create the OSD daemons. The Orchestrator handles the placement for the new OSDs that are balanced between the hosts, as well as handling BlueStore device selection.

Use the `ceph orch apply osd --all-available-devices` command to provision all available, unused devices.

```
[ceph: root@node /]# ceph orch apply osd --all-available-devices
```

This command creates an OSD service called `osd.all-available-devices` and enables the Orchestrator service to manage all OSD provisioning. The Orchestrator automatically creates OSDs from both new disk devices in the cluster and from existing devices that are prepared with the `ceph orch device zap` command.

To disable the Orchestrator from automatically provisioning OSDs, set the `unmanaged` flag to `true`.

```
[ceph: root@node /]# ceph orch apply osd --all-available-devices --unmanaged=true
Scheduled osd.all-available-devices update...
```

**Note**

You can also update the unmanaged flag with a service specification file.

Specific Target Provisioning

You can create OSD daemons by using a specific device and host. To create a single OSD daemon with a specific host and storage device, use the `ceph orch daemon add` command.

```
[ceph: root@node /]# ceph orch daemon add osd node:/dev/vdb
Created osd(s) 12 on host 'node'
```

To stop an OSD daemon, use the `ceph orch daemon stop` command with the OSD ID.

```
[ceph: root@node /]# ceph orch daemon stop osd.12
```

To remove an OSD daemon, use the `ceph orch daemon rm` command with the OSD ID.

```
[ceph: root@node /]# ceph orch daemon rm osd.12
Removed osd.12 from host 'node'
```

To release an OSD ID, use the `ceph osd rm` command.

```
[ceph: root@node /]# ceph osd rm 12
removed osd.12
```

Service Specification Provisioning

Use service specification files to describe the cluster layout for OSD services. You can customize the service provisioning with filters. With filters, you can configure the OSD service without knowing the specific hardware architecture. This method is useful when automating cluster bootstrap and maintenance windows.

The following is an example service specification YAML file that defines two OSD services, each using different filters for placement and BlueStore device location.

```
service_type: osd
service_id: osd_size_and_model
placement:
  host_pattern: '*'
data_devices:
  size: '100G:'
db_devices:
  model: My-Disk
wal_devices:
  size: '10G:20G'
unmanaged: true
```

```

---
service_type: osd
service_id: osd_host_and_path
placement:
  host_pattern: 'node[6-10]'
data_devices:
  paths:
    - /dev/sdb
db_devices:
  paths:
    - /dev/sdc
wal_devices:
  paths:
    - /dev/sdd
encrypted: true

```

The `osd_size_and_model` service specifies that any host can be used for placement and the service will be managed by the storage administrator. The data device must have a device with 100 GB or more, and the write-ahead log must have a device of 10 - 20 GB. The database device must be of the My-Disk model.

The `osd_host_and_path` service specifies that the target host must be provisioned on nodes between node6 and node10 and the service will be managed by the orchestrator service. The device paths for data, database, and write-ahead log must be `/dev/sdb`, `/dev/sdc`, and `/dev/sdd`. The devices in this service will be encrypted.

Run the `ceph orch apply` command to apply the service specification.

```
[ceph: root@node /]# ceph orch apply -i service_spec.yaml
```

Other OSD Utilities

The `ceph-volume` command is a modular tool to deploy logical volumes as OSDs. It uses a plug-in type framework. The `ceph-volume` utility supports the `lvm` plug-in and raw physical disks. It can also manage devices that are provisioned with the legacy `ceph-disk` utility.

Use the `ceph-volume lvm` command to manually create and delete BlueStore OSDs. The following command creates a new BlueStore OSD on block storage device `/dev/vdc`:

```
[ceph: root@node /]# ceph-volume lvm create --bluestore --data /dev/vdc
```

An alternative to the `create` subcommand is to use the `ceph-volume lvm prepare` and `ceph-volume lvm activate` subcommands. With this method, OSDs are gradually introduced into the cluster. You can control when the new OSDs are in the `up` or `in` state, so you can ensure that large amounts of data are not unexpectedly rebalanced across OSDs.

The `prepare` subcommand configures logical volumes for the OSD to use. You can specify a logical volume or a device name. If you specify a device name, then a logical volume is automatically created.

```
[ceph: root@node /]# ceph-volume lvm prepare --bluestore --data /dev/vdc
```

The `activate` subcommand enables a `systemd` unit for the OSD so that it starts at boot time. You need the OSD `fsid` (UUID) from the output of the `ceph-volume lvm list` command to use the `activate` subcommand. Providing the unique identifier ensures that the correct OSD is activated, because OSD IDs can be reused.

```
[ceph: root@node /]# ceph-volume lvm activate <osd-fsid>
```

When the OSD is created, use the `systemctl start ceph-osd@$id` command to start the OSD so it has the `up` and `in` state in the cluster.

The `batch` subcommand creates multiple OSDs at the same time.

```
[ceph: root@node /]# ceph-volume lvm batch \  
--bluestore /dev/vdc /dev/vdd /dev/nvme0n1
```

The `inventory` subcommand provides information about all physical storage devices on a node.

```
[ceph: root@node /]# ceph-volume inventory
```



References

For more information, refer to the *Red Hat Ceph Storage 5 Architecture Guide* at
https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/architecture_guide/index

For more information, refer to the *BlueStore* chapter in the *Red Hat Ceph Storage 5 Administration Guide* at
https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/administration_guide/osd-bluestore

For more information, refer to the *Advanced service specifications and filters for deploying OSDs* chapter in the *Red Hat Ceph Storage 5 Operation Guide* at
https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/operations_guide/index#advanced-service-specifications-and-filters-for-deploying-osds_ops

► Guided Exercise

Creating BlueStore OSDs Using Logical Volumes

In this exercise, you will create new BlueStore OSDs.

Outcomes

You should be able to create BlueStore OSDs and place the data, write-ahead log (WAL), and metadata (DB) onto separate storage devices.

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 component-osd
```

This command confirms that the hosts required for this exercise are accessible.

Instructions

- 1. Log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]#
```

- 2. Verify the health of the cluster. View the current cluster size. View the current OSD tree.

- 2.1. Verify the health of the cluster.

```
[ceph: root@clienta /]# ceph health
HEALTH_OK
```

- 2.2. View the current cluster storage size.

```
[ceph: root@clienta /]# ceph df
--- RAW STORAGE ---
CLASS      SIZE     AVAIL      USED    RAW USED   %RAW USED
hdd       90 GiB   90 GiB    180 MiB   180 MiB      0.20
TOTAL     90 GiB   90 GiB    180 MiB   180 MiB      0.20
...output omitted...
```

- 2.3. View the current OSD tree.

```
[ceph: root@clienta /]# ceph osd tree
ID CLASS WEIGHT TYPE NAME STATUS REWEIGHT PRI-AFF
-1          0.08817  root default
-3          0.02939  host serverc
 0  hdd  0.00980    osd.0   up   1.00000  1.00000
 1  hdd  0.00980    osd.1   up   1.00000  1.00000
 2  hdd  0.00980    osd.2   up   1.00000  1.00000
-5          0.02939  host serverd
 3  hdd  0.00980    osd.3   up   1.00000  1.00000
 4  hdd  0.00980    osd.4   up   1.00000  1.00000
 5  hdd  0.00980    osd.5   up   1.00000  1.00000
-7          0.02939  host servere
 6  hdd  0.00980    osd.6   up   1.00000  1.00000
 7  hdd  0.00980    osd.7   up   1.00000  1.00000
 8  hdd  0.00980    osd.8   up   1.00000  1.00000
```

- 3. List all active disk devices in the cluster. Use grep or awk to filter available devices from the `ceph orch device ls` command.

3.1. List all in-use disk devices in the cluster.

```
[ceph: root@clienta /]# ceph device ls
DEVICE                      HOST:DEV                         DAEMONS
WEAR  LIFE EXPECTANCY
11fa06d1-e33e-45d1-a  serverc.lab.example.com:vda  mon.serverc.lab.example.com
44b28a3a-a008-4f9f-8  servere.lab.example.com:vda  mon.servere
69a83d4c-b41a-45e8-a  serverd.lab.example.com:vdc  osd.4
6d0064d3-0c4e-4458-8  serverc.lab.example.com:vdd  osd.2
74995d99-2cae-4683-b  clienta.lab.example.com:vda  mon.clienta
7f14ca40-eaad-4bf6-a   servere.lab.example.com:vdb  osd.6
90b06797-6a94-4540-a   servere.lab.example.com:vdc  osd.7
97ed8eda-4ca4-4437-b   serverd.lab.example.com:vdb  osd.3
a916efa8-d749-408e-9   serverd.lab.example.com:vda  mon.serverd
b1b94eac-1975-41a2-9   serverd.lab.example.com:vdd  osd.5
ca916831-5492-4014-a   serverc.lab.example.com:vdb  osd.0
caee7c62-d12a-45c4-b   serverc.lab.example.com:vdc  osd.1
f91f6439-0eb9-4df0-a   servere.lab.example.com:vdd  osd.8
```

3.2. Use grep or awk to filter available devices from the `ceph orch device ls` command.

```
[ceph: root@clienta /]# ceph orch device ls | awk '/server/ | grep Yes
serverc.lab.example.com /dev/vde hdd c63...82a-b 10.7G Unknown N/A N/A Yes
serverc.lab.example.com /dev/vdf hdd 6f2...b8e-9 10.7G Unknown N/A N/A Yes
serverd.lab.example.com /dev/vde hdd f84...6f0-8 10.7G Unknown N/A N/A Yes
serverd.lab.example.com /dev/vdf hdd 297...63c-9 10.7G Unknown N/A N/A Yes
servere.lab.example.com /dev/vde hdd 2aa...c03-b 10.7G Unknown N/A N/A Yes
servere.lab.example.com /dev/vdf hdd 41c...794-b 10.7G Unknown N/A N/A Yes
```

- 4. Create two OSD daemons by using the disk devices `/dev/vde` and `/dev/vdf` on `serverc.lab.example.com`. Record the ID assigned to each OSD. Verify that the daemons are running correctly. View the cluster storage size and the new OSD tree.

Chapter 4 | Creating Object Storage Cluster Components

- 4.1. Create an OSD daemon by using device /dev/vde on serverc.lab.example.com.

```
[ceph: root@clienta /]# ceph orch daemon add osd serverc.lab.example.com:/dev/vde  
Created osd(s) 9 on host 'serverc.lab.example.com'
```

- 4.2. Create an OSD daemon by using device /dev/vdf on serverc.lab.example.com.

```
[ceph: root@clienta /]# ceph orch daemon add osd serverc.lab.example.com:/dev/vdf  
Created osd(s) 10 on host 'serverc.lab.example.com'
```

- 4.3. Verify that the daemons are running.

```
[ceph: root@clienta /]# ceph orch ps | grep -ie osd.9 -ie osd.10  
osd.9          serverc.lab.example.com  running (6m)  6m ago   6m  ...  
osd.10         serverc.lab.example.com  running (6m)  6m ago   6m  ...
```

- 4.4. View the cluster storage size.

```
[ceph: root@clienta /]# ceph df  
--- RAW STORAGE ---  
CLASS      SIZE    AVAIL     USED  RAW USED %RAW USED  
hdd       110 GiB  110 GiB  297 MiB 297 MiB      0.26  
TOTAL     110 GiB  110 GiB  297 MiB 297 MiB      0.26  
...output omitted...
```

- 4.5. View the OSD tree.

```
[ceph: root@clienta /]# ceph osd tree  
ID CLASS WEIGHT  TYPE NAME            STATUS REWEIGHT PRI-AFF  
-1          0.10776  root default  
-3          0.04898  host serverc  
 0  hdd  0.00980    osd.0        up  1.00000  1.00000  
 1  hdd  0.00980    osd.1        up  1.00000  1.00000  
 2  hdd  0.00980    osd.2        up  1.00000  1.00000  
 9  hdd  0.00980    osd.9        up  1.00000  1.00000  
10  hdd  0.00980    osd.10       up  1.00000  1.00000  
-5          0.02939  host serverd  
 3  hdd  0.00980    osd.3        up  1.00000  1.00000  
 4  hdd  0.00980    osd.4        up  1.00000  1.00000  
 5  hdd  0.00980    osd.5        up  1.00000  1.00000  
-7          0.02939  host servere  
 6  hdd  0.00980    osd.6        up  1.00000  1.00000  
 7  hdd  0.00980    osd.7        up  1.00000  1.00000  
 8  hdd  0.00980    osd.8        up  1.00000  1.00000
```

- ▶ 5. Enable the orchestrator service to create OSD daemons automatically from the available cluster devices. Verify the creation of the all-available-devices OSD service, and the existence of new OSD daemons.

- 5.1. Enable the orchestrator service to create OSD daemons automatically from the available cluster devices.

```
[ceph: root@clienta /]# ceph orch apply osd --all-available-devices
Scheduled osd.all-available-devices update...
```

- 5.2. Verify the creation of the all-available-devices OSD service.

```
[ceph: root@clienta /]# ceph orch ls
NAME          RUNNING  REFRESHED  AGE  PLACEMENT
alertmanager      1/1   11s ago   13h  count:1
crash           4/4   5m ago    13h  *
grafana          1/1   11s ago   13h  count:1
mgr              4/4   5m ago    13h  ...
mon              4/4   5m ago    13h  ...
node-exporter     1/4   5m ago    13h  *
osd.all-available-devices  0/4   -        10s  *
osd.default_drive_group  9/12  5m ago   13h  server*
osd.unmanaged      2/2   11s ago   -    <unmanaged>
prometheus        1/1   11s ago   13h  count:1
rgw.realm.zone     2/2   5m ago    8m   ...
```

- 5.3. Verify the existence of new OSDs.

```
[ceph: root@clienta /]# ceph osd tree
ID  CLASS  WEIGHT  TYPE NAME          STATUS  REWEIGHT  PRI-AFF
-1          0.19592  root default
-9          0.04898  host clienta
12  hdd  0.00980  osd.12       up    1.00000  1.00000
16  hdd  0.00980  osd.16       up    1.00000  1.00000
17  hdd  0.00980  osd.17       up    1.00000  1.00000
18  hdd  0.00980  osd.18       up    1.00000  1.00000
19  hdd  0.00980  osd.19       up    1.00000  1.00000
-3          0.04898  host serverc
 0  hdd  0.00980  osd.0        up    1.00000  1.00000
 1  hdd  0.00980  osd.1        up    1.00000  1.00000
 2  hdd  0.00980  osd.2        up    1.00000  1.00000
 9  hdd  0.00980  osd.9        up    1.00000  1.00000
10  hdd  0.00980  osd.10       up    1.00000  1.00000
-5          0.04898  host serverd
 3  hdd  0.00980  osd.3        up    1.00000  1.00000
 4  hdd  0.00980  osd.4        up    1.00000  1.00000
 5  hdd  0.00980  osd.5        up    1.00000  1.00000
11  hdd  0.00980  osd.11       up    1.00000  1.00000
14  hdd  0.00980  osd.14       up    1.00000  1.00000
-7          0.04898  host servere
 6  hdd  0.00980  osd.6        up    1.00000  1.00000
 7  hdd  0.00980  osd.7        up    1.00000  1.00000
 8  hdd  0.00980  osd.8        up    1.00000  1.00000
13  hdd  0.00980  osd.13       up    1.00000  1.00000
15  hdd  0.00980  osd.15       up    1.00000  1.00000
```

- 6. Stop the OSD daemon associated with the /dev/vde device on `servere` and remove it. Verify the removal process ends correctly. Zap the /dev/vde device on `servere`. Verify that the Orchestrator service then re-adds the OSD daemon correctly.

**Note**

It is expected that the OSD ID might be different in your lab environment. Review the output of the `ceph device ls | grep 'servere.lab.example.com:vde'` and use the ID to perform the next steps.

- 6.1. Stop the OSD daemon associated with the /dev/vde device on `servere.lab.example.com` and remove it.

```
[ceph: root@clienta /]# ceph device ls | grep 'servere.lab.example.com:vde'
2aa9ab0c-9d12-4c03-b servere.lab.example.com:vde osd.ID
[ceph: root@clienta /]# ceph orch daemon stop osd.ID
Scheduled to stop osd.ID on host 'servere.lab.example.com'
[ceph: root@clienta /]# ceph orch daemon rm osd.ID --force
Scheduled OSD(s) for removal
[ceph: root@clienta /]# ceph osd rm ID
removed osd.ID
[ceph: root@clienta /]# ceph orch osd rm status
OSD_ID HOST STATE PG_COUNT REPLACE FORCE
DRAIN_STARTED_AT
ID servere.lab.example.com done, waiting for purge 0 False False
None
```

- 6.2. Verify that the removal process ends correctly.

```
[ceph: root@clienta /]# ceph orch osd rm status
No OSD remove/replace operations reported
```

- 6.3. Zap the /dev/vde device on `servere`. Verify that the Orchestrator service re-adds the OSD daemon correctly.

```
[ceph: root@clienta /]# ceph orch device zap --force \
servere.lab.example.com /dev/vde
[ceph: root@clienta /]# ceph orch device ls | awk '/servere/
servere.lab.example.com /dev/vdb hdd 50a...56b-8 10.7G Unknown N/A N/A No
servere.lab.example.com /dev/vdc hdd eb7...5a6-9 10.7G Unknown N/A N/A No
servere.lab.example.com /dev/vdd hdd 407...56f-a 10.7G Unknown N/A N/A No
servere.lab.example.com /dev/vde hdd 2aa...c03-b 10.7G Unknown N/A N/A Yes
servere.lab.example.com /dev/vdf hdd 41c...794-b 10.7G Unknown N/A N/A No
[ceph: root@clienta /]# ceph orch device ls | awk '/servere/
servere.lab.example.com /dev/vdb hdd 50a...56b-8 10.7G Unknown N/A N/A No
servere.lab.example.com /dev/vdc hdd eb7...5a6-9 10.7G Unknown N/A N/A No
servere.lab.example.com /dev/vdd hdd 407...56f-a 10.7G Unknown N/A N/A No
servere.lab.example.com /dev/vde hdd 2aa...c03-b 10.7G Unknown N/A N/A No
servere.lab.example.com /dev/vdf hdd 41c...794-b 10.7G Unknown N/A N/A No
[ceph: root@clienta /]# ceph device ls | grep 'servere.lab.example.com:vde'
osd.ID
[ceph: root@clienta /]# ceph orch ps | grep osd.ID
osd.ID servere.lab.example.com running (76s) 66s ago 76s ...
```

- 7. View the OSD services in YAML format. Copy the definition corresponding to the all-available-devices service. Create the all-available-devices.yaml file and add the copied service definition.

```
[ceph: root@clienta /]# ceph orch ls --service-type osd --format yaml  
...output omitted...  
service_type: osd  
service_id: all-available-devices  
service_name: osd.all-available-devices  
...output omitted...  
service_type: osd  
service_id: default_drive_group  
service_name: osd.default_drive_group  
...output omitted...
```

The resulting all-available-devices.yaml file should look like this.

```
[ceph: root@clienta /]# cat all-available-devices.yaml  
service_type: osd  
service_id: all-available-devices  
service_name: osd.all-available-devices  
placement:  
  host_pattern: '*'  
spec:  
  data_devices:  
    all: true  
  filter_logic: AND  
  objectstore: bluestore
```

- 8. Modify the all-available-device.yaml file to add the unmanaged: true flag, then apply the service change to the orchestrator. Verify that the service now has the unmanaged flag from the orchestrator service list.

8.1. Modify the all-available-device.yaml file to add the unmanaged: true flag.

```
[ceph: root@clienta /]# cat all-available-devices.yaml  
service_type: osd  
service_id: all-available-devices  
service_name: osd.all-available-devices  
placement:  
  host_pattern: '*'  
spec:  
  data_devices:  
    all: true  
  filter_logic: AND  
  objectstore: bluestore  
unmanaged: true
```

8.2. Apply the service change to the service.

```
[ceph: root@clienta /]# ceph orch apply -i all-available-devices.yaml  
Scheduled osd.all-available-devices update...
```

- 8.3. Verify that the `osd.all-available-devices` service now has the `unmanaged` flag.

```
[ceph: root@clienta /]# ceph orch ls
NAME          RUNNING  REFRESHED  AGE  PLACEMENT
alertmanager   1/1     2m ago    8d   count:1
crash         4/4     9m ago    8d   *
grafana        1/1     2m ago    8d   count:1
mgr            4/4     9m ago    8d   ...
mon            4/4     9m ago    8d   ...
node-exporter  4/4     9m ago    8d   *
osd.all-available-devices  8/12    7m ago   10s <unmanaged>
osd.default_drive_group  9/12    9m ago   13h server*
osd.unmanaged    2/3     9m ago   -    <unmanaged>
prometheus      1/1     2m ago    8d   count:1
rgw.realm.zone  2/2     2m ago    2d   ...
```

- ▶ 9. Stop the OSD daemon associated with the `/dev/vdf` device on `serverd` and remove it. Verify that the removal process ends correctly. Zap the `/dev/vdf` device on `serverd`. Verify that the Orchestrator service does not create a new OSD daemon from the cleaned device.



Note

It is expected that the OSD ID might be different in your lab environment. Use `ceph device ls | grep 'serverd.lab.example.com:vdf'` to obtain your OSD ID.

- 9.1. Stop the OSD daemon associated with the `/dev/vdf` device on `serverd` and remove it.

```
[ceph: root@clienta /]# ceph device ls | grep 'serverd.lab.example.com:vdf'
297fa02b-7e18-463c-9 serverd.lab.example.com:vdf osd.ID
[ceph: root@clienta /]# ceph orch daemon stop osd.ID
stop osd.ID.

[ceph: root@clienta /]# ceph orch daemon rm osd.ID --force
Scheduled OSD(s) for removal
[ceph: root@clienta /]# ceph orch osd rm status
OSD_ID HOST STATE PG_COUNT REPLACE FORCE
DRAIN_STARTED_AT
ID serverd.lab.example.com done, waiting for purge 0 False False
None
```

- 9.2. Verify that the removal was successful, then remove the OSD ID.

```
[ceph: root@clienta /]# ceph orch osd rm status
No OSD remove/replace operations reported
[ceph: root@clienta /]# ceph osd rm ID
```

- 9.3. Verify that the orchestrator service does not create a new OSD daemon from the cleaned device.

```
[ceph: root@clienta /]# ceph orch device zap --force \
serverd.lab.example.com /dev/vdf
[ceph: root@clienta /]# ceph orch device ls | awk '/serverd/
serverd.lab.example.com /dev/vdf hdd 297...63c-9 10.7G Unknown N/A N/A Yes
serverd.lab.example.com /dev/vdb hdd 87a...c28-a 10.7G Unknown N/A N/A No
serverd.lab.example.com /dev/vdc hdd 850...7cf-8 10.7G Unknown N/A N/A No
serverd.lab.example.com /dev/vdd hdd 6bf...ecb-a 10.7G Unknown N/A N/A No
serverd.lab.example.com /dev/vde hdd f84...6f0-8 10.7G Unknown N/A N/A No
[ceph: root@clienta /]# sleep 60
[ceph: root@clienta /]# ceph orch device ls | awk '/serverd/
serverd.lab.example.com /dev/vdf hdd 297...63c-9 10.7G Unknown N/A N/A
Yes
serverd.lab.example.com /dev/vdb hdd 87a...c28-a 10.7G Unknown N/A N/A No
serverd.lab.example.com /dev/vdc hdd 850...7cf-8 10.7G Unknown N/A N/A No
serverd.lab.example.com /dev/vdd hdd 6bf...ecb-a 10.7G Unknown N/A N/A No
serverd.lab.example.com /dev/vde hdd f84...6f0-8 10.7G Unknown N/A N/A No
```

- 10. Return to workstation as the student user.

```
[ceph: root@clienta /]# exit
[admin@clienta ~]$ exit
[student@workstation ~]$
```

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 component-osd
```

This concludes the guided exercise.

Creating and Configuring Pools

Objectives

After completing this section, you should be able to describe and compare replicated and erasure coded pools, and create and configure each pool type.

Understanding Pools

Pools are logical partitions for storing objects. Ceph clients write objects to pools.

Ceph clients require the cluster name (ceph by default) and a monitor address to connect to the cluster. Ceph clients usually obtain this information from a Ceph configuration file, or by being specified as command-line parameters.

A Ceph client uses the list of pools retrieved with the cluster map, to determine where to store new objects.

The Ceph client creates an input/output context to a specific pool and the Ceph cluster uses the CRUSH algorithm to map these pools to placement groups, which are then mapped to specific OSDs.

Pools provide a layer of resilience for the cluster because pools define the number of OSDs that can fail without losing data.

Pool Types

The available pool types are **replicated** and **erasure coded**. You decide which pool type to use based on your production use case and the type of workload.

The default pool type is **replicated**, which functions by copying each object to multiple OSDs. This pool type requires more storage because it creates multiple copies of objects, however, read operation availability is increased through redundancy.

Erasure coded pools require less storage and network bandwidth but use more CPU processing time because of parity calculations.

Erasure coded pools are recommended for infrequently accessed data that does not require low latency. Replicated pools are recommended for frequently accessed data that requires fast read performance. The recovery time for each pool type can vary widely and is based on the cluster deployment, failure, and sizing characteristics.

A pool's type cannot be changed after creating the pool.

Pool Attributes

You must specify certain attributes when you create a pool:

- The `pool name`, which must be unique in the cluster.
- The `pool type`, which determines the protection mechanism the pool uses to ensure data durability. The **replicated** type distributes multiple copies of each object across the cluster. The **erasure coded** type splits each object into chunks, and distributes them along with

additional erasure coded chunks to protect objects using an automatic error correction mechanism.

- The number of `placement groups` (PGs) in the pool, which store their objects in a set of OSDs determined by the CRUSH algorithm.
- Optionally, a `CRUSH rule` set that Ceph uses to identify which placement groups to use to store objects for the pool.



Note

Change the `osd_pool_default_pg_num` and `osd_pool_default_pgp_num` configuration settings to set the default number of PGs for a pool.

Creating Replicated Pools

Ceph protects data within replicated pools by creating multiple copies of each object, called replicas. Ceph uses the CRUSH failure domain to determine the primary OSDs of the acting set to store the data. Then, the primary OSD finds the current replica size for the pool and calculates the secondary OSDs to write the objects to. After the primary OSD receives the acknowledgment of the writes and finishes writing the data, the primary OSD acknowledges a successful write operation to the Ceph client. This protects the data in the object if one or more of the OSDs fail.

Use the following command to create a replicated pool.

```
[ceph: root@node /]# ceph osd pool create pool-name pg-num ppg-num  
replicated crush-rule-name
```

Where:

- `pool_name` is the name of the new pool.
- `pg_num` is the total configured number of placement groups (PGs) for this pool.
- `ppg_num` is the effective number of placement groups for this pool. Set this equal to `pg_num`.
- `replicated` specifies that this is a replicated pool, and is the default if not included in the command.
- `crush-rule-name` is the name of the CRUSH rule set you want to use for this pool. The `osd_pool_default_crush_replicated_ruleset` configuration parameter sets the default value.

The number of placement groups in a pool can be adjusted after it is initially configured. If `pg_num` and `ppg_num` are set to the same number, then any future adjustments to `pg_num` automatically adjusts the value of `ppg_num`. The adjustment to `ppg_num` triggers the movement of PGs across OSDs, if needed, to implement the change. Define a new number of PGs in a pool by using the following command.

```
[ceph: root@node /]# ceph osd pool set my_pool pg_num 32  
set pool 6 pg_num to 32
```

When you create a pool with the `ceph osd pool create` command, you do not specify the number of replicas (size). The `osd_pool_default_size` configuration parameter defines the number of replicas, and defaults to a value of 3.

```
[ceph: root@node /]# ceph config get mon osd_pool_default_size  
3
```

Change the size of a pool with the `ceph osd pool set pool-name size number-of-replicas` command. Alternatively, update the default setting of the `osd_pool_default_size` configuration setting.

The `osd_pool_default_min_size` parameter sets the number of copies of an object that must be available to accept I/O requests. The default value is 2.

Configuring Erasure Coded Pools

An erasure coded pool uses erasure coding instead of replication to protect object data.

Objects stored in an erasure coded pool are divided into a number of data chunks which are stored in separate OSDs. The number of coding chunks are calculated based on the data chunks and are stored in different OSDs. The coding chunks are used to reconstruct the object's data if an OSD fails. The primary OSD receives the write operation, then encodes the payload into K+M chunks and sends them to the secondary OSDs in erasure coded pools.

Erasure coded pools use this method to protect their objects and, unlike replicated pools, do not rely on storing multiple copies of each object.

To summarize how erasure coded pools work:

- Each object's data is divided into k data chunks.
- m coding chunks are calculated.
- The coding chunk size is the same as the data chunk size.
- The object is stored on a total of k + m OSDs.

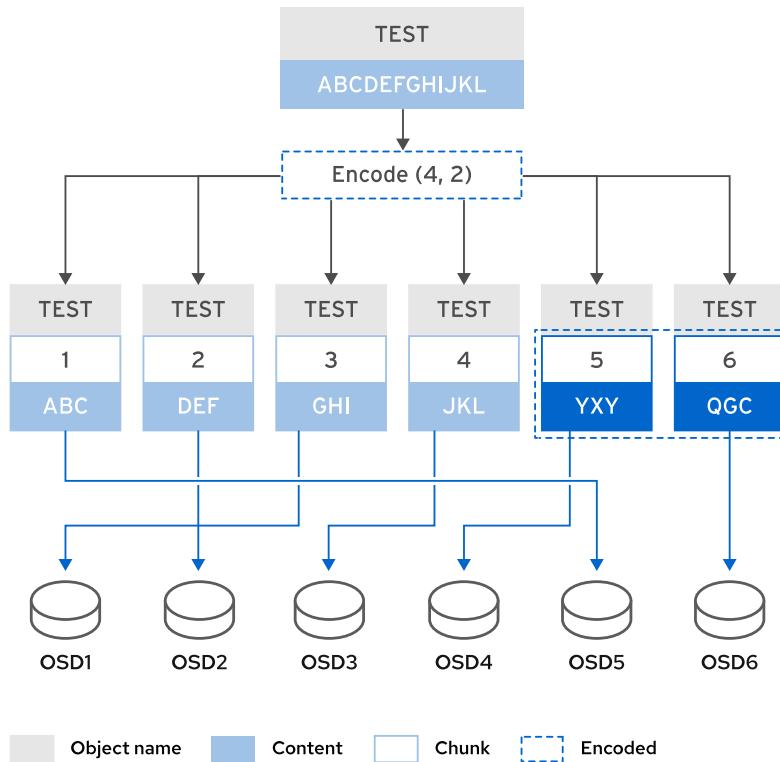


Figure 4.4: Erasure coded pools

Erasure coding uses storage capacity more efficiently than replication. Replicated pools maintain n copies of an object, whereas erasure coding maintains only $k + m$ chunks. For example, replicated pools with 3 copies use 3 times the storage space. Erasure coded pools with $k=4$ and $m=2$ use only 1.5 times the storage space.



Note

Red Hat supports the following $k+m$ values which result in the corresponding usable-to-raw ratio:

- **4+2** (1:1.5 ratio)
- **8+3** (1:1.375 ratio)
- **8+4** (1:1.5 ratio)

The formula for calculating the erasure code overhead is $n\text{OSD} * k / (k+m) * \text{OSD Size}$. For example, if you have 64 OSDs of 4 TB each (256 TB total), with $k=8$ and $m=4$, then the formula is $64 * 8 / (8+4) * 4 = 170.67$. Then divide the raw storage capacity by the overhead to get the ratio. 256 TB/170.67 TB equals a ratio of 1.5.

Erasure coded pools require less storage than replicated pools to obtain a similar level of data protection, which can reduce the cost and size of the storage cluster. However, calculating coding chunks adds CPU processing and memory overhead for erasure coded pools, reducing overall performance.

Use the following command to create an erasure coded pool.

```
[ceph: root@node /]# ceph osd pool create pool-name pg-num pgp-num \
erasure erasure-code-profile crush-rule-name
```

Where:

- **pool-name** is the name of the new pool.
- **pg-num** is the total number of placement groups (PGs) for this pool.
- **pgp-num** is the effective number of placement groups for this pool. Normally, this should be equal to the total number of placement groups.
- **erasure** specifies that this is an erasure coded pool.
- **erasure-code-profile** is the name of the profile to use. You can create new profiles with the `ceph osd erasure-code-profile set` command. A profile defines the **k** and **m** values and the erasure code plug-in to use. By default, Ceph uses the **default** profile.
- **crush-rule-name** is the name of the CRUSH rule set to use for this pool. If not set, Ceph uses the one defined in the erasure code profile.

You can configure placement group autoscaling on a pool. Autoscaling allows the cluster to calculate the number of placement groups and to choose appropriate `pg_num` values automatically. Autoscaling is enabled by default in Red Hat Ceph Storage 5.

Every pool in the cluster has a `pg_autoscale_mode` option with a value of `on`, `off`, or `warn`.

- **on**: Enables automated adjustments of the PG count for the pool.
- **off**: Disables PG autoscaling for the pool.
- **warn**: Raises a health alert and changes the cluster health status to `HEALTH_WARN` when the PG count needs adjustment.

This example enables the `pg_autoscaler` module on the Ceph MGR nodes and sets the autoscaling mode to `on` for a pool:

```
[ceph: root@node /]# ceph mgr module enable pg_autoscaler
module 'pg_autoscaler' is already enabled (always-on)
[ceph: root@node /]# ceph osd pool set pool-name pg_autoscale_mode on
set pool 7 pg_autoscale_mode to on
```

Erasure coded pools cannot use the *Object Map* feature. An object map is an index of objects that tracks where the blocks of an rbd object are allocated. Having an object map for a pool improves the performance of resize, export, flatten, and other operations.

Erasure Code Profiles

An erasure code profile configures the number of data chunks and coding chunks that your erasure-coded pool uses to store objects, and which erasure coding plug-in and algorithm to use.

Create profiles to define different sets of erasure coding parameters. Ceph automatically creates the `default` profile during installation. This profile is configured to divide objects into two data chunks and one coding chunk.

Use the following command to create a new profile.

```
[ceph: root@node /]# ceph osd erasure-code-profile set profile-name arguments
```

The following arguments are available:

k

The number of data chunks that are split across OSDs. The default value is 2.

m

The number of OSDs that can fail before the data becomes unavailable. The default value is 1.

directory

This optional parameter is the location of the plug-in library. The default value is `/usr/lib64/ceph/erasure-code`.

plugin

This optional parameter defines the erasure coding algorithm to use.

crush-failure-domain

This optional parameter defines the CRUSH failure domain, which controls chunk placement. By default, it is set to `host`, which ensures that an object's chunks are placed on OSDs on different hosts. If set to `osd`, then an object's chunks can be placed on OSDs on the same host. Setting the failure domain to `osd` is less resilient because all OSDs on a host will fail if the host fails. Failure domains can be defined and used to ensure chunks are placed on OSDs on hosts in different data center racks or other customization.

crush-device-class

This optional parameter selects only OSDs backed by devices of this class for the pool. Typical classes might include `hdd`, `ssd`, or `nvme`.

crush-root

This optional parameter sets the root node of the CRUSH rule set.

key=value

Plug-ins might have key-value parameters unique to that plug-in.

technique

Each plug-in provides a different set of techniques that implement different algorithms.



Important

You cannot modify or change the erasure code profile of an existing pool.

Use the `ceph osd erasure-code-profile ls` command to list existing profiles.

Use the `ceph osd erasure-code-profile get` command to view the details of an existing profile.

Use the `ceph osd erasure-code-profile rm` command to remove an existing profile.

Managing and Operating Pools

You can view and modify existing pools and change pool configuration settings.

- Rename a pool by using the `ceph osd pool rename` command. This does not affect the data stored in the pool. If you rename a pool and you have per-pool capabilities for an authenticated user, you must update the user's capabilities with the new pool name.
- Delete a pool by using the `ceph osd pool delete` command.

**Warning**

Deleting a pool removes all data in the pool and is not reversible. You must set `mon_allow_pool_delete` to `true` to enable pool deletion.

- Prevent pool deletion for a specific pool by using the `ceph osd pool set pool_name nodelete true` command. Set `nodelete` back to `false` to allow deletion of the pool.
- View and modify pool configuration settings by using the `ceph osd pool set` and `ceph osd pool get` commands.
- List pools and pool configuration settings by using the `ceph osd ls pools` and `ceph osd pool ls detail` commands.
- List pools usage and performance statistics by using the `ceph df` and `ceph osd pool stats` commands.
- Enable Ceph applications for a pool by using the `ceph osd pool application enable` command. Application types are `cephfs` for Ceph File System, `rbd` for Ceph Block Device, and `rgw` for RADOS Gateway.
- Set pool quotas to limit the maximum number of bytes or the maximum number of objects that can be stored in the pool by using the `ceph osd pool set-quota` command.

**Important**

When a pool reaches the configured quota, operations are blocked. You can remove a quota by setting its value to 0.

Configure these example setting values to enable protection against pool reconfiguration:

`osd_pool_default_flag_nodelete`

Sets the default value of the `nodelete` flag on pools. Set the value to `true` to prevent pool deletion.

`osd_pool_default_flag_nopgchange`

Sets the default value of the `nopgchange` flag on pools. Set the value to `true` to prevent changes to `pg_num`, and `pgp_num`.

`osd_pool_default_flag_nosizechange`

Sets the default value of the `nosizechange` flag on pools. Set the value to `true` to prevent pool size changes.

Pool Namespaces

A *namespace* is a logical group of objects in a pool. Access to a pool can be limited so that a user can only store or retrieve objects in a particular namespace. One advantage of namespaces is to restrict user access to part of a pool.

Namespaces are useful for restricting storage access by an application. They allow you to logically partition a pool and restrict applications to specific namespaces inside the pool.

You could dedicate an entire pool to each application, but having more pools means more PGs per OSD, and PGs are computationally expensive. This might degrade OSD performance as load increases. With namespaces, you can keep the number of pools the same and not dedicate an entire pool to each application.

**Important**

Namespaces are currently only supported for applications that directly use librados. RBD and Ceph Object Gateway clients do not currently support this feature.

To store an object inside a namespace, the client application must provide the pool and the namespace names. By default, each pool contains a namespace with an empty name, known as the default namespace.

Use the rados command to store and retrieve objects from a pool. Use the -N *name* and --namespace=*name* options to specify the pool and namespace to use.

The following example stores the /etc/services file as the `srv` object in the `mytestpool` pool, under the `system` namespace.

```
[ceph: root@node /]# rados -p mytestpool -N system put srv /etc/services
[ceph: root@node /]# rados -p mytestpool -N system ls
srv
```

List all the objects in all namespaces in a pool by using the --all option. To obtain JSON formatted output, add the --format=json-pretty option.

The following example lists the objects in the `mytestpool` pool. The `mytest` object has an empty namespace. The other objects belong to the `system` or the `flowers` namespaces.

```
[ceph: root@node /]# rados -p mytestpool --all ls
system  srv
flowers  anemone
flowers  iris
system  magic
flowers  rose
      mytest
system  networks
[ceph: root@node /]# rados -p mytestpool --all ls --format=json-pretty
[
  {
    "name": "srv",
    "namespace": "system"
  },
  {
    "name": "anemone",
    "namespace": "flowers"
  },
  {
    "name": "iris",
    "namespace": "flowers"
  },
  {
    "name": "magic",
    "namespace": "system"
  },
  {
    "name": "networks",
    "namespace": "system"
  }
]
```

```
"name": "rose",
"namespace": "flowers"
},
{
  "name": "mytest",
  "namespace": ""
},
{
  "name": "networks",
  "namespace": "system"
}
]
```



References

For more information, refer to the *Pools* and *Erasure Code Pools* chapters in the *Red Hat Ceph Storage 5 Storage Strategies Guide* at
https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/storage_strategies_guide

► Guided Exercise

Creating and Configuring Pools

In this exercise, you will create and configure replicated and erasure coded storage pools.

Outcomes

You should be able to create, delete, and rename pools as well as view and configure pool settings.

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 component-pool
```

This command confirms that the hosts required for this exercise are accessible.

Instructions

- 1. Log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]#
```

- 2. Create a replicated pool called `replpool1` with 64 Placement Groups (PGs).

```
[ceph: root@clienta /]# ceph osd pool create replpool1 64 64
pool 'replpool1' created
```

- 3. Verify that PG autoscaling is enabled for the `replpool1` pool and that it is the default for new pools.

```
[ceph: root@clienta /]# ceph osd pool get replpool1 pg_autoscale_mode
pg_autoscale_mode: on
[ceph: root@clienta /]# ceph config get mon osd_pool_default_pg_autoscale_mode
on
```

- 4. List the pools, verify the existence of the `replpool` pool, and view the autoscale status for the pools.

- 4.1. List the pools and verify the existence of the `replpool` pool.

```
[ceph: root@clienta /]# ceph osd lspools
1 device_health_metrics
2 .rgw.root
3 default.rgw.log
4 default.rgw.control
5 default.rgw.meta
6 replpool1
```

4.2. View the autoscale status.

```
[ceph: root@clienta /]# ceph osd pool autoscale-status
POOL          SIZE ... AUTOSCALE
device_health_metrics    0      on
.rgw.root            2466   on
default.rgw.control    0      on
default.rgw.meta       393   on
default.rgw.log         3520   on
replpool1             0      on
```

- ▶ 5. Set the number of replicas for the `replpool1` pool to 4. Set the minimum number of replicas required for I/O to 2, allowing up to two OSDs to fail without losing data. Set the application type for the pool to `rbd`. Use the `ceph osd pool ls detail` command to verify the pool configuration settings. Use the `ceph osd pool get` command to get the value of a specific setting.

- 5.1. Set the number of replicas for the `replpool1` pool to 4. Set the minimum number of replicas required for I/O to two.

```
[ceph: root@clienta /]# ceph osd pool set replpool1 size 4
set pool 6 size to 4
[ceph: root@clienta /]# ceph osd pool set replpool1 min_size 2
set pool 6 min_size to 2
```

- 5.2. Set the application type for the pool to `rbd`.

```
[ceph: root@clienta /]# ceph osd pool application enable replpool1 rbd
enabled application 'rbd' on pool 'replpool1'
```

- 5.3. Use the `ceph osd pool ls detail` command to verify the pool configuration settings.

```
[ceph: root@clienta /]# ceph osd pool ls detail
...output omitted...
pool 6 'replpool1' replicated size 4 min_size 2 crush_rule 0 object_hash rjenkins
pg_num 64 pgp_num 64 autoscale_mode on last_change 366 flags hashpspool
stripe_width 0 application rbd
[ceph: root@clienta /]# ceph osd pool get replpool1 size
size: 4
```

**Note**

The pool uses CRUSH rule 0. Configuring CRUSH rules and pool CRUSH rules is covered in a later chapter.

- 6. Rename the `replpool1` pool to `newpool`. Delete the `newpool` pool.

- 6.1. Rename the `replpool1` pool to `newpool`.

```
[ceph: root@clienta /]# ceph osd pool rename replpool1 newpool
pool 'replpool1' renamed to 'newpool'
```

- 6.2. Delete the `newpool` pool.

```
[ceph: root@clienta /]# ceph osd pool delete newpool
Error EPERM: WARNING: this will *PERMANENTLY DESTROY* all data stored in pool
newpool. If you are *ABSOLUTELY CERTAIN* that is what you want, pass the pool
name *twice*, followed by --yes-i-really-really-mean-it.

[ceph: root@clienta /]# ceph osd pool delete newpool newpool \
--yes-i-really-really-mean-it
Error EPERM: pool deletion is disabled; you must first set the
mon_allow_pool_delete config option to true before you can destroy a pool

[ceph: root@clienta /]# ceph tell mon.* config set mon_allow_pool_delete true
mon.serverc.lab.example.com: {
    "success": ""
}
mon.serverd: {
    "success": ""
}
mon.servere: {
    "success": ""
}
mon.clienta: {
    "success": ""
}

[ceph: root@clienta /]# ceph osd pool delete newpool newpool \
--yes-i-really-really-mean-it
pool 'newpool' removed
```

**Important**

When you rename a pool, you must update any associated user authentication settings with the new pool name. User authentication and capabilities are covered in a later chapter.

- 7. List the existing erasure coded profiles and view the details of the `default` profile. Create an erasure code profile called `ecprofile-k4-m2` with `k=4` and `m=2` values. These values allow the simultaneous loss of two OSDs without losing any data and meets the minimum requirement for Red Hat support.

- 7.1. View the configured erasure coded profiles.

```
[ceph: root@clienta /]# ceph osd erasure-code-profile ls
default
```

- 7.2. View the details of the default profile.

```
[ceph: root@clienta /]# ceph osd erasure-code-profile get default
k=2
m=1
plugin=jerasure
technique=reed_sol_van
```

- 7.3. Create an erasure code profile called ecprofile-k4-m2 with k=4 and m=2 values.

```
[ceph: root@clienta /]# ceph osd erasure-code-profile set ecprofile-k4-m2 k=4 m=2
```

- 8. Create an erasure coded pool called ecpool1 using the ecprofile-k4-m2 profile with 64 placement groups and an rgw application type. View the details of the ecpool1 pool. Configure the ecpool1 pool to allow partial overwrites so that RBD and CephFS can use it. Delete the ecpool1.

- 8.1. Create an erasure coded pool called ecpool1 by using the ecprofile-k4-m2 profile with 64 placement groups and set the application type to rgw.

```
[ceph: root@clienta /]# ceph osd pool create ecpool1 64 64 erasure ecprofile-k4-m2
pool 'ecpool1' created
[ceph: root@clienta /]# ceph osd pool application enable ecpool1 rgw
enabled application 'rgw' on pool 'ecpool1'
```

- 8.2. View the details of the ecpool1 pool. Your pool ID is expected to be different.

```
[ceph: root@clienta /]# ceph osd pool ls detail
...output omitted...
pool 7 'ecpool1' erasure profile ecprofile-k4-m2 size 6 min_size 5 crush_rule 2
object_hash rjenkins pg_num 64 ppg_num 64 autoscale_mode on last_change 373 flags
hashpspool,creating stripe_width 16384 application rgw
```

- 8.3. Configure the ecpool1 pool to allow partial overwrites so that RBD and CephFS can use it.

```
[ceph: root@clienta /]# ceph osd pool set ecpool1 allow_ec_overwrites true
set pool 7 allow_ec_overwrites to true
```

- 8.4. Delete the ecpool1 pool.

```
[ceph: root@clienta /]# ceph osd pool delete ecpool1 ecpool1 \
--yes-i-really-really-mean-it
pool 'ecpool1' removed
```

- 9. Return to workstation as the student user.

```
[ceph: root@clienta /]# exit
[admin@clienta ~]$ exit
[student@workstation ~]$
```

- 10. Create a replicated pool using the Ceph Dashboard.

- 10.1. Open a web browser and navigate to <https://serverc:8443>.
- 10.2. Log in as admin by using redhat as the password. You should see the Dashboard page.

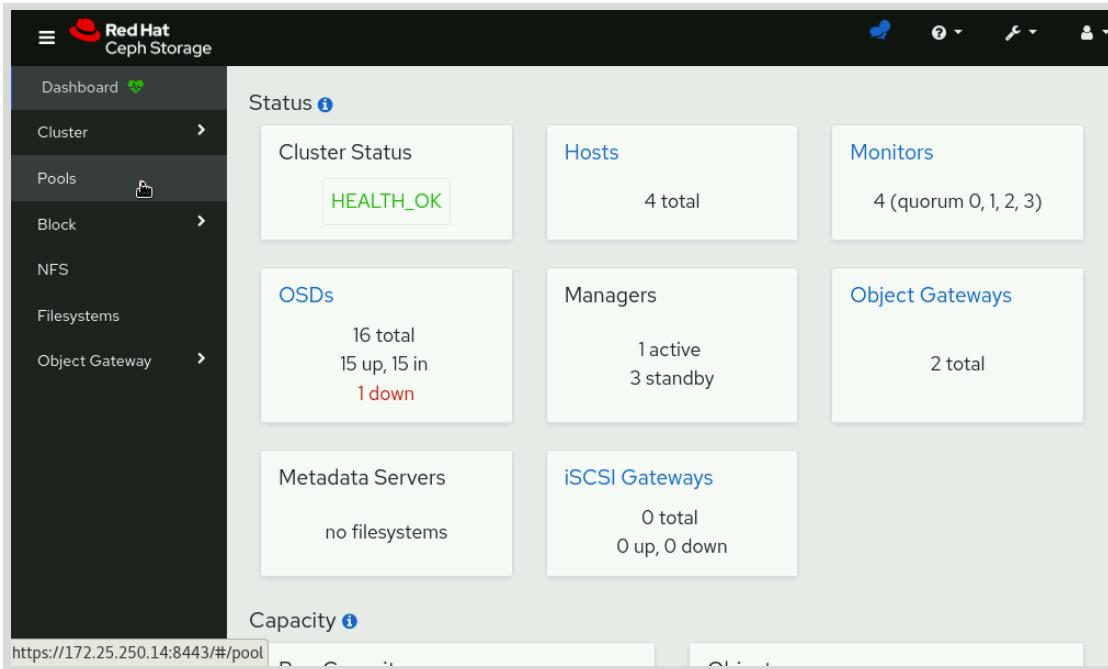


Figure 4.5: The Ceph Storage Dashboard

- 10.3. Click Pools to display the Pools page Click Create.

The screenshot shows the 'Pools' page in the Red Hat Ceph Storage interface. On the left, a sidebar lists 'Dashboard', 'Cluster', 'Pools', 'Block', 'NFS', 'Filesystems', and 'Object Gateway'. The 'Pools' section is selected. The main area displays a table of existing pools:

Name	Data Protection	Applications	PG Status	Usage	Read bytes	Write bytes	Read ops	Write ops
.rgw.root	replica: 3	rgw	active+cle 2 stale+acth	30 0%	0/s	0/s	0/s	0/s
default.rgw.control	replica: 3	rgw	active+cle 2 stale+acth	30 0%	0/s	0/s	0/s	0/s
default.rgw.log	replica: 3	rgw	active+cle 2 stale+acth	30 0%	0/s	0/s	0/s	0/s
default.rgw.meta	replica: 7	rgw	active+cle 1	0% 7	0/s	0/s	0/s	0/s

A blue 'Create' button with a plus sign is located at the top of the pool list.

Figure 4.6: The Pools page

- 10.4. Enter **replpool1** in the **Name** field, **replicated** in the **Pool type** field, **on** in the **PG Autoscale** field, and **3** in the **Replicated size**. Leave other values as default. Click **CreatePool**.

The screenshot shows the 'Create Pool' dialog box. The 'Name' field is set to 'replpool1'. The 'Pool type' dropdown is set to 'replicated'. The 'PG Autoscale' dropdown is set to 'on'. The 'Replicated size' dropdown is set to '3'. The 'Applications' section shows 'No applications added'. The 'CRUSH' section shows 'Crush ruleset: replicated_rule'. The 'Create' button is visible at the bottom right of the dialog.

Figure 4.7: Creating a replicated pool in the Ceph Dashboard

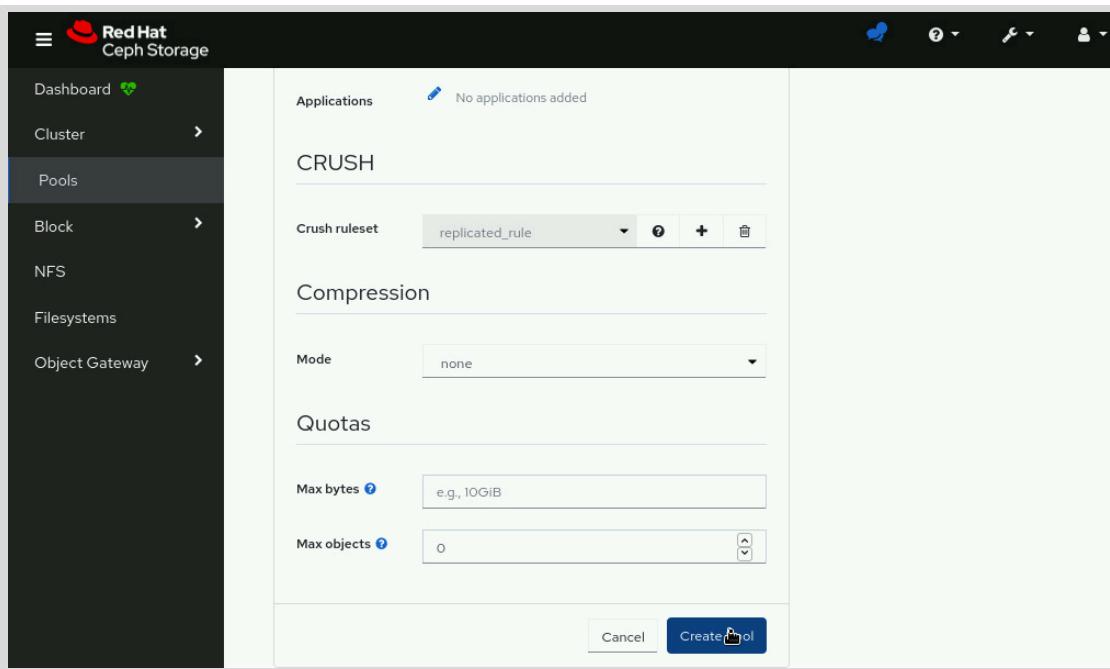


Figure 4.8: Creating a replicated pool in the Ceph Dashboard

- 11. Create an erasure coded pool using the Ceph Dashboard.

- 11.1. Click Pools to display the Pools page. Click Create.

Name	Data Protection	Applications	PG Status	Usage	Read bytes	Write bytes	Read ops	Write ops
.rgw.root	replica: x3	rgw	30 active+cle 2 stale+active	0%	0 /s	0 /s	0 /s	0 /s
default.rgw.control	replica: x3	rgw	30 active+cle 2 stale+active	0%	0 /s	0 /s	0 /s	0 /s
default.rgw.log	replica: x3	rgw	30 active+cle 2 stale+active	0%	0 /s	0 /s	0 /s	0 /s
default.rgw.meta	replica: x3	rgw	7 active+cle 1 stale+active	0%	0 /s	0 /s	0 /s	0 /s

Figure 4.9: The Pools page

- 11.2. Enter **ecpool1** in the Name field, **erasure** in the Pool type field, **off** in the PG Autoscale field, and **64** in the Placement groups field. Check the EC Overwrites box of the Flags section, and select the **ecprofile-k4-m2** profile from the Erasure code profile field. Click **CreatePool**.

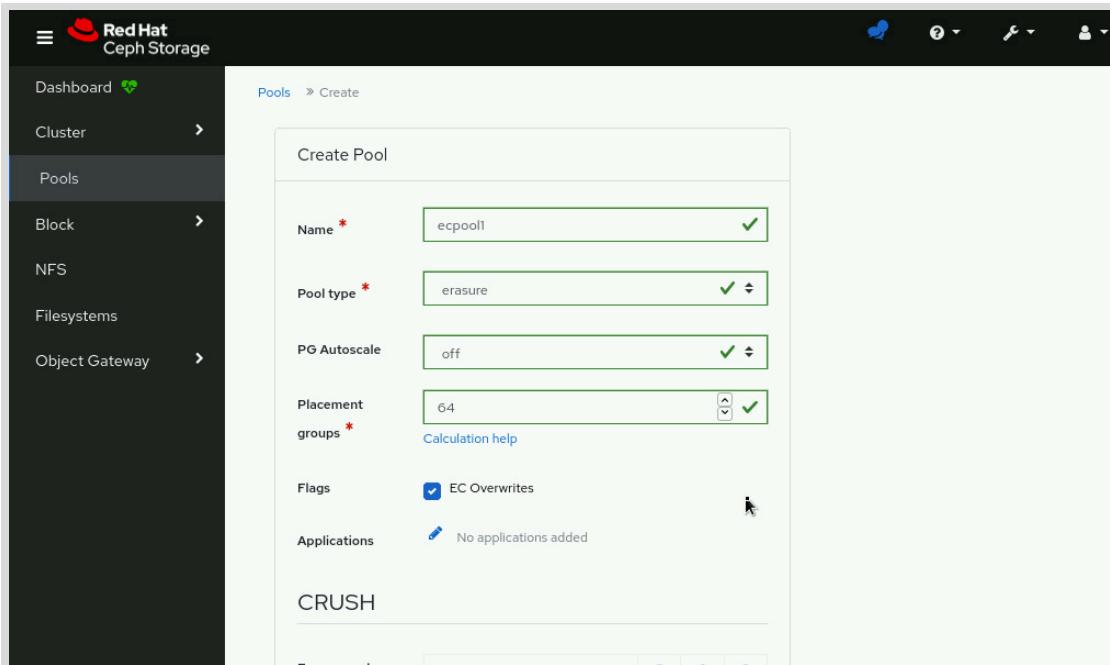


Figure 4.10: Creating an erasure coded pool in the Ceph Dashboard

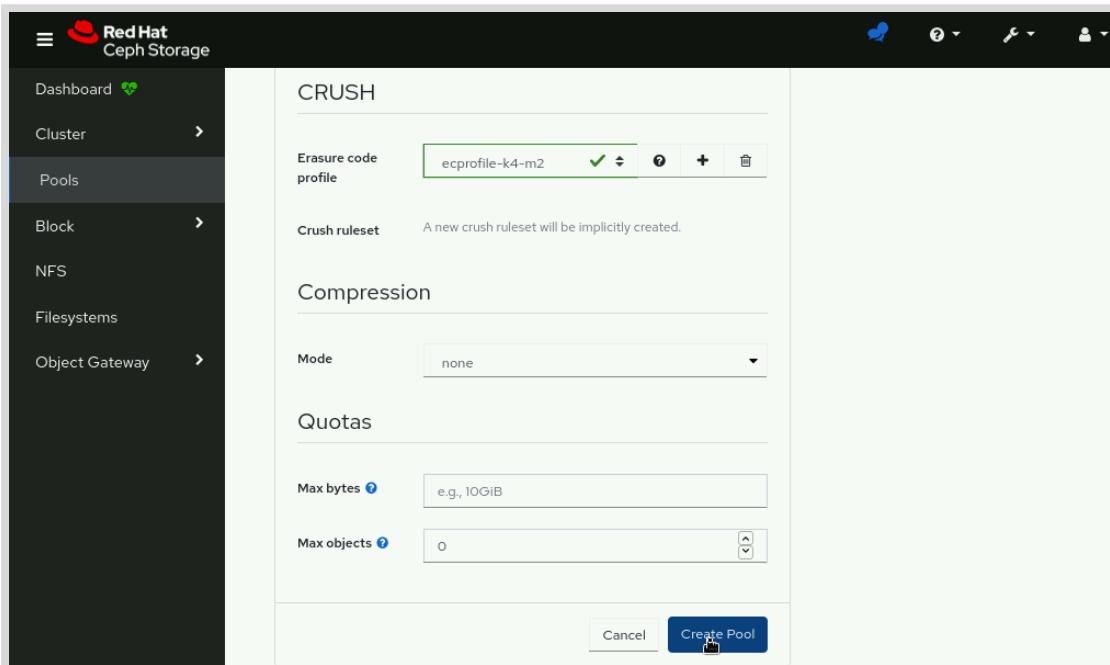


Figure 4.11: Creating an erasure coded pool in the Ceph Dashboard

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 component-pool
```

This concludes the guided exercise.

Managing Ceph Authentication

Objectives

After completing this section, you should be able to describe Cephx and configure user authentication and authorization for Ceph clients.

Authenticating Users

Red Hat Ceph Storage uses the cephx protocol to authorize communication between clients, applications, and daemons in the cluster. The cephx protocol is based on shared secret keys.

The installation process enables cephx by default, so that the cluster requires user authentication and authorization by all client applications.

Ceph uses user accounts for several purposes:

- For internal communication between Ceph daemons.
- For client applications accessing the cluster through the librados library.
- For cluster administrators.

Accounts used by Ceph daemons have names that match their associated daemon: osd.1 or mgr.serverc and are created during the installation.

Accounts used by client applications that use librados have names with the client. prefix. For example, when integrating OpenStack with Ceph, it is common to create a dedicated client.openstack user account. For the Ceph Object Gateway, the installation creates a dedicated client.rgw.hostname user account. Developers creating custom software on top of librados should create dedicated accounts with appropriate capabilities.

Administrator account names also have the client. prefix. These accounts are used when running commands such as ceph and rados. The installer creates the superuser account, client.admin, with capabilities that allow the account to access everything and to modify the cluster configuration. Ceph uses the client.admin account to run administrative commands, unless you explicitly specify a user name with the --name or --id options.

You can set the CEPH_ARGS environment variable to define parameters such as the cluster name or the ID of the user.

```
[ceph: root@node /]# export CEPH_ARGS="--id cephuser"
```

End users of a Ceph-aware application do not have an account on the Ceph cluster. Rather, they access the application, which then accesses Ceph on their behalf. From the Ceph point of view, the application is the client. The application might provide its own user authentication through other mechanisms.

Figure 4.12 provides an overview of how an application can provide its own user authentication.

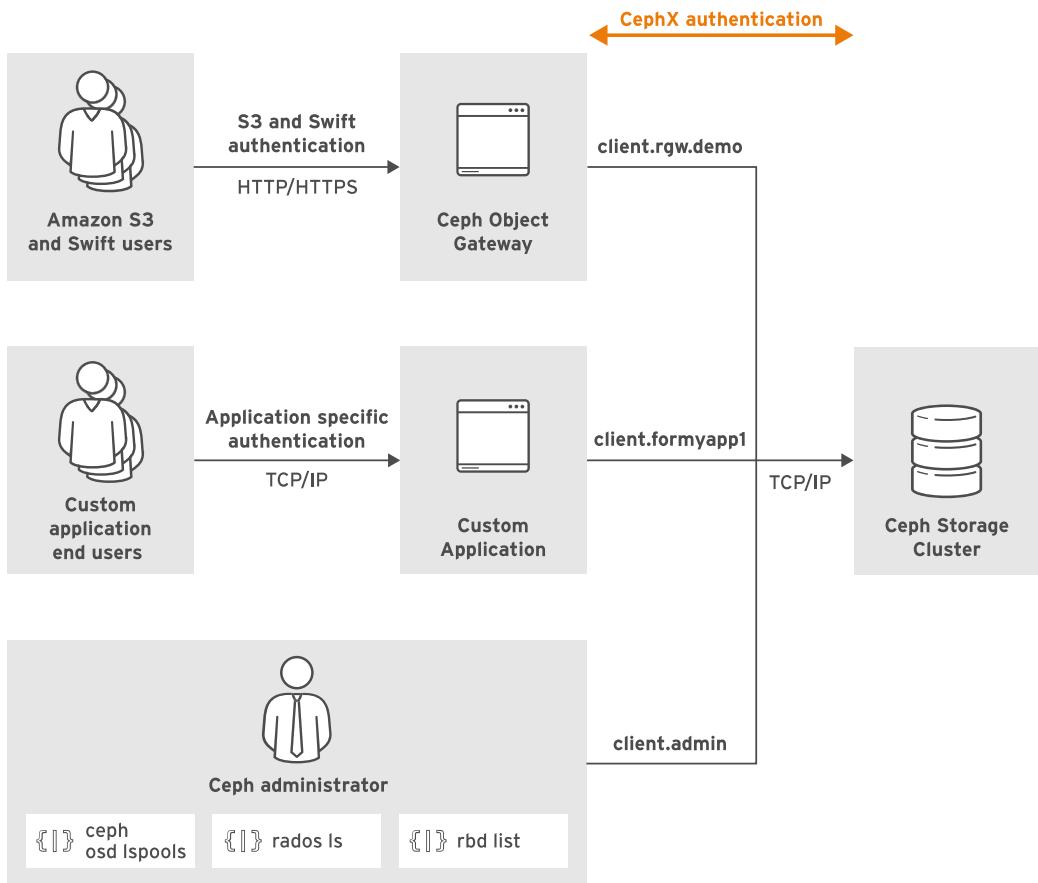


Figure 4.12: User authentication for Ceph applications

The Ceph Object Gateway has its own user database to authenticate Amazon S3 and Swift users, but uses the `client.rgw.hostname` account to access the cluster.

The Key-ring File

For authentication, clients are configured with a Ceph user name and a key-ring file containing the user's secret key. Ceph generates the key-ring file for each user account when it is created. However, you must copy this file to each client system or application server on which it is needed.

On these client systems, librados uses the `keyring` parameter from the `/etc/ceph/ceph.conf` configuration file to locate the key-ring file. Its default value is `/etc/ceph/$cluster.$name.keyring`. For example, for the `client.openstack` account, the key-ring file is `/etc/ceph/ceph.client.openstack.keyring`.

The key-ring file stores the secret key as plain text. Protect the file with appropriate Linux file permissions for access only by authorized Linux users. Only deploy a Ceph user's key-ring file on systems that need it for authentication.

Transmitting Secret Keys

The `cephx` protocol does not transmit the shared secret keys as plain text. Instead, a client requests a session key from a Monitor. The Monitor encrypts the session key with the client's shared secret key and provides the session key to the client. A client decrypts the session key and

requests a ticket from a Monitor to authenticate to cluster daemons. This is similar to the Kerberos protocol, with a cephx key-ring file being comparable to a Kerberos keytab file.

A more detailed discussion of the protocol is available from the upstream Ceph project's documentation at High Availability Authentication [<https://docs.ceph.com/docs/master/architecture/#high-availability-authentication>].

Configuring User Authentication

Using command-line tools such as `ceph`, `rados`, and `rbd`, administrators can specify the user account and the key-ring file by using the `--id` and `--keyring` options. When not specified, commands authenticate as the `client.admin` user.

In this example, the `ceph` command authenticates as `client.operator3` to list the available pools:

```
[ceph: root@node /]# ceph --id operator3 osd ls pools
1 myfirstpool
2 mysecondpool
```



Important

Do not include the `client.` prefix when using the `--id` option. The `--id` option automatically assumes that `client.` prefix. Alternatively, the `--name` option requires the `client.` prefix.

If you store the key-ring file in its default location, you do not need the `--keyring` option. The `cephadm` shell automatically mounts the key-ring from the `/etc/ceph/` directory.

Configuring User Authorization

When you create a new user account, grant cluster permissions sufficient to authorize the user's cluster tasks. Permissions within cephx are known as *capabilities*, and you grant them by daemon type (`mon`, `osd`, `mgr`, or `mds`.)

Use capabilities to restrict or provide access to data in a pool, a pool's namespace, or a set of pools based on application tags. Capabilities also allow the daemons in the cluster to interact with each other.

Cephx Capabilities

Within cephx, for each daemon type, several capabilities are available:

- `r` grants read access. Each user account should have at least read access on the Monitors to be able to retrieve the CRUSH map.
- `w` grants write access. Clients need write access to store and modify objects on OSDs. For Managers (MGRs), `w` grants the right to enable or disable modules.
- `x` grants authorization to execute extended object classes. This allows clients to perform extra operations on objects such as setting locks with `rados lock` or listing RBD images with `rbd list`.
- `*` grants full access.
- `class-read` and `class-write` are subsets of `x`. You typically use them on RBD pools.

This example creates the `formyapp1` user account, and grants the capability to store and retrieve objects from any pool:

```
[ceph: root@node /]# ceph auth get-or-create client.formyapp1 \
mon 'allow r' osd 'allow rw'
```

Using Profiles to Set Capabilities

cephx offers predefined capability profiles. When creating user accounts, utilize profiles to simplify configuration of user access rights.

This example utilizes the `rbd` profile to define the access rights for the new `forrbd` user account. A client application can use this account for block-based access to Ceph storage using a RADOS Block Device.

```
[ceph: root@node /]# ceph auth get-or-create client.forrbd \
mon 'profile rbd' osd 'profile rbd'
```

The `rbd-read-only` profile works the same way but grants read-only access. Ceph utilizes other existing profiles for internal communication between daemons. You cannot create your own profiles, Ceph defines them internally.

The following table lists Ceph capabilities on a default installation.

Capability	Description
allow	Precedes access settings for a daemon.
r	Gives the user read access. Required with monitors to retrieve the CRUSH map.
w	Gives the user write access to objects.
x	Gives the user the capability to call class methods (that is, both read and write) and to conduct authentication operations on monitors.
class-read	Gives the user the capability to call class read methods. Subset of x.
class-write	Gives the user the capability to call class write methods. Subset of x.
*	Gives the user read, write and execute permissions for a particular daemon or pool, and the ability to execute admin commands.
profile osd	Gives a user permissions to connect as an OSD to other OSDs or monitors. Conferred on OSDs to enable OSDs to handle replication heartbeat traffic and status reporting.
profile bootstrap-osd	Gives a user permissions to bootstrap an OSD, so that they have permissions to add keys when bootstrapping an OSD.
profile rbd	Gives a user read-write access to the Ceph Block Devices.
profile rbd-read-only	Gives a user read-only access to the Ceph Block Devices.

Restricting Access

Restrict user OSD permissions such that users can only access the pools they need. This example creates the `formyapp2` user and limits their access to read and write on the `myapp` pool:

```
[ceph: root@node /]# ceph auth get-or-create client.formyapp2 \
mon 'allow r' osd 'allow rw pool=myapp'
```

If you do not specify a pool when you configure capabilities, then Ceph sets them on all existing pools.

The `cephx` mechanism can restrict access to objects by other means:

- By *object name prefix*. The following example restricts access to only those objects whose names start with `pref` in any pool.

```
[ceph: root@node /]# ceph auth get-or-create client.formyapp3 \
mon 'allow r' osd 'allow rw object_prefix pref'
```

- By *namespace*. Implement namespaces to logically group objects within a pool. You can then restrict user accounts to objects belonging to a specific namespace:

```
[ceph: root@node /]# ceph auth get-or-create client.designer \
mon 'allow r' osd 'allow rw namespace=photos'
```

- By *path*. The Ceph File System (CephFS) utilizes this method to restrict access to specific directories. This example creates a new user account, `webdesigner`, that can access only the `/webcontent` directory and its contents:

```
[ceph: root@node /]# ceph fs authorize cephfs client.webdesigner /webcontent rw
[ceph: root@node /]# ceph auth get client.webdesigner
exported keyring for client.webdesigner
[client.webdesigner]
key = AQBrVE9aNwoEGRApYR6m71ECRzULLpp4wEJkw==
caps mds = "allow rw path=/webcontent"
caps mon = "allow r"
caps osd = "allow rw pool=cephfs_data"
```

- By *Monitor command*. This method restricts administrators to a specific list of commands. The following example creates the `operator1` user account and limits its access to two commands:

```
[ceph: root@node /]# ceph auth get-or-create client.operator1 \
mon 'allow r, allow command "auth get-or-create", allow command "auth list"'
```

User Management

To list existing user accounts, run the `ceph auth list` command:

```
[ceph: root@node /]# ceph auth list
...output omitted...
osd.0
key: AQBW6Tha5z60IhAAMQ7nY/4MogYecxKqQxx1sA==
caps: [mgr] allow profile osd
```

```
caps: [mon] allow profile osd
caps: [osd] allow *
client.admin
key: AQCi6Dhajw7pIRAA/ECKwyipx2/raLWjgbklyA==
caps: [mds] allow *
caps: [mgr] allow *
caps: [mon] allow *
caps: [osd] allow *
...output omitted...
```

To get the details of a specific account, use the `ceph auth get` command:

```
[ceph: root@node /]# ceph auth get client.admin
exported keyring for client.admin
[client.admin]
key = AQCi6Dhajw7pIRAA/ECKwyipx2/raLWjgbklyA==
caps mds = "allow *"
caps mgr = "allow *"
caps mon = "allow *"
caps osd = "allow *"
```

You can print the secret key:

```
[ceph: root@node /]# ceph auth print-key client.admin
AQCi6Dhajw7pIRAA/ECKwyipx2/raLWjgbklyA==
```

To export and import user accounts, run the `ceph auth export` and `ceph auth import` commands:

```
[ceph: root@node /]# ceph auth export client.operator1 > ~/operator1.export
[ceph: root@node /]# ceph auth import -i ~/operator1.export
```

Creating New User Accounts

The `ceph auth get-or-create` command creates a new user account and generates its secret key. The command prints this key to `stdout` by default, so it is common to add the `-o` option to save standard output to a key-ring file.

This example creates the `app1` user account with read and write access to all pools, and stores the key-ring file in `/etc/ceph/ceph.client.app1.keyring`:

```
[ceph: root@node /]# ceph auth get-or-create client.app1 \
mon 'allow r' osd 'allow rw' -o /etc/ceph/ceph.client.app1.keyring
```

Authentication requires the key-ring file, so you must copy the file to all client systems that operate with this new user account.

Modifying User Capabilities

Modify the capabilities of a user account with the `ceph auth caps` command.

This example modifies the `app1` user account capabilities on OSDs to allow only read and write access to the `myapp` pool:

```
[ceph: root@node /]# ceph auth caps client.app1 mon 'allow r' \
osd 'allow rw pool=myapp'
updated caps for client.app1
```

The `ceph auth caps` command overwrites existing capabilities. When using this command, you must specify the full set of capabilities for all daemons, not only those you want to modify. Define an empty string to remove all capabilities:

```
[ceph: root@node /]# ceph auth caps client.app1 osd ''
updated caps for client.app1
```

Deleting User Accounts

The `ceph auth del` command deletes a user account:

```
[ceph: root@node /]# ceph auth del client.app1
updated
```

You can then remove the associated key-ring file.



References

For more information, refer to the *Ceph User Management* chapter in the *Red Hat Ceph Storage 5 Administration Guide* at
https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/administration_guide

► Guided Exercise

Managing Ceph Authentication

In this exercise, you will configure user authentication by setting up users for an application that stores and retrieves documents as RADOS objects.

Outcomes

You should be able to configure user authentication and capabilities to store and retrieve objects in the cluster.

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 component-auth
```

This command confirms that the hosts required for this exercise are accessible.

Instructions

- 1. Log in to `clienta` as the `admin` user and switch to the `root` user.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo -i
[root@clienta ~]$
```

- 2. Configure two users for an application with the following capabilities. The first user, `client.docedit`, stores and retrieves documents in the `docs` namespace of the `replpool1` pool. The second user, `client.docget`, only retrieves documents from the `replpool1` pool.



Note

The `tee` command saves the output of the command, instead of using the `-o` option. This technique is used because the `cephadm` container does not retain standard output files after the command exits.

- 2.1. Use the `cephadm` shell to create the `client.docedit` user with read and write capabilities in the `docs` namespace within the `replpool1` pool. Save the associated key-ring file by using the appropriate directory and file name: `/etc/ceph/ceph.client.docedit.keyring`

```
[root@clienta ~]$ cephadm shell -- ceph auth get-or-create client.docedit \
mon 'allow r' osd 'allow rw pool=replpool1 namespace=docs' | sudo tee \
/etc/ceph/ceph.client.docedit.keyring
```

- 2.2. Use the `cephadm` shell to create the `client.docget` user with read capabilities in the `docs` namespace within the `replpool1` pool. Save the associated key-ring file using the appropriate directory and file name: `/etc/ceph/ceph.client.docget.keyring`

```
[root@clienta ~]$ cephadm shell -- ceph auth get-or-create client.docget \
mon 'allow r' osd 'all ow r pool=replpool1 namespace=docs' | sudo tee \
/etc/ceph/ceph.client.docget.keyring
```

- 2.3. Verify that you created both user names correctly.

```
[root@clienta ~]$ cephadm shell -- ceph auth ls | grep -A3 -ie docedit \
-ie docget
installed auth entries:

client.docedit
key: AQARYFNhUVqjLxAAvD/00leu3V93+e9umSTBKQ==
caps: [mon] allow r
caps: [osd] allow rw pool=replpool1 namespace=docs
client.docget
key: AQDByFNhac58MxAA/ukJXL52cpsQLw65zZ+WcQ==
caps: [mon] allow r
caps: [osd] allow r pool=replpool1 namespace=docs
installed auth entries:
```

- 3. Your application is running on `serverd`. Copy the users' key-ring files to that server to allow the application to authenticate with the cluster.

```
[root@clienta ~]$ rsync -v /etc/ceph/ceph.client.docedit.keyring \
serverd:/etc/ceph/
ceph.client.docedit.keyring

sent 170 bytes received 35 bytes 136.67 bytes/sec
total size is 65 speedup is 0.32
[root@clienta ~]$ rsync -v /etc/ceph/ceph.client.docget.keyring \
serverd:/etc/ceph/
ceph.client.docget.keyring

sent 168 bytes received 35 bytes 135.33 bytes/sec
total size is 64 speedup is 0.32
```

- 4. Use the `cephadm` shell with the `--mount` option to mount the `/etc/ceph` directory. Store and retrieve an object to verify that the key-rings are working correctly. The two files should be identical as verified by the `diff` command showing no output.

```
[root@clienta ~]$ cephadm shell --mount /etc/ceph:/etc/ceph
[ceph: root@clienta /]# rados --id docedit -p replpool1 -N docs put \
adoc /etc/hosts
[ceph: root@clienta /]# rados --id docget -p replpool1 -N docs get \
adoc /tmp/test
[ceph: root@clienta /]# diff /etc/hosts /tmp/test
```

- ▶ 5. Your application evolves over time and now the `client.docget` user also needs write access to the `docs` namespace within the `replpool1` pool. This user also needs to store documents in the `docarchive` pool.

Confirm that the `client.docget` user cannot store objects yet in the `docs` namespace within the `replpool1` pool:

```
[ceph: root@clienta /]# rados --id docget -p replpool1 -N docs put \
mywritetest /etc/hosts
error putting mypool/mywritetest: (1) Operation not permitted
```

- ▶ 6. Grant the `client.docget` user `rw` capabilities on the `docs` namespace within the `replpool1` pool, and `rw` capabilities on the non-yet-created `docarchive` pool. Confirm that the `client.docget` user can now store objects in the `docs` namespace.

```
[ceph: root@clienta /]# ceph auth caps client.docget mon 'allow r' \
osd 'allow rw pool=replpool1 namespace=docs, allow rw pool=docarchive'
updated caps for client.docget
[ceph: root@clienta /]# rados --id docget -p replpool1 -N docs put \
mywritetest /etc/hosts
```

You must define the total user capabilities with the `ceph auth caps` command because it overwrites previous definitions. You can define capabilities on pools that do not exist yet, such as the `docarchive` pool.

- ▶ 7. Exit the `cephadm` shell and clean up by deleting the `client.docedit` and the `client.docget` users. Remove the associated key-ring files.

```
[ceph: root@clienta /]# exit
[root@clienta ~]$ rm /etc/ceph/ceph.client.docedit.keyring
[root@clienta ~]$ ssh serverd rm /etc/ceph/ceph.client.docedit.keyring
[root@clienta ~]$ cephadm shell -- ceph auth del client.docedit
updated
[root@clienta ~]$ rm /etc/ceph/ceph.client.docget.keyring
[root@clienta ~]$ ssh serverd rm /etc/ceph/ceph.client.docget.keyring
[root@clienta ~]$ cephadm shell -- ceph auth del client.docget
updated
```

- ▶ 8. Return to workstation as the student user.

```
[root@clienta ~]$ exit
[admin@clienta ~]$ exit
[student@workstation ~]$
```

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 component-auth
```

This concludes the guided exercise.

▶ Lab

Creating Object Storage Cluster Components

In this lab, you will create and manage cluster components and authentication.

Outcomes

You should be able to create and configure BlueStore OSDs and pools, and set up authentication to the cluster.

Before You Begin

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

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

This command confirms that the hosts required for this exercise are accessible.

Instructions

1. Log in to `clienta` as the `admin` user. Create a new OSD daemon by using the `/dev/vde` device on `serverc`. View the details of the OSD. Restart the OSD daemon and verify it starts correctly.
2. Create a replicated pool called `labpool1` with 64 PGs. Set the number of replicas to 3. Set the application type to `rbd`. Set the `pg_auto_scale` mode to `on` for the pool.
3. Create an erasure code profile called `k8m4` with data chunks on 8 OSDs ($k=8$), able to sustain the loss of 4 OSDs ($m=4$), and set `crush_failure_domain=rack`. Create an erasure coded pool called `labpool2` with 64 PGs that uses the `k8m4` profile.
4. Create the `client.rwpool` user account with the capabilities to read and write objects in the `labpool1` pool. This user must not be able to access the `labpool2` pool in any way.
Create the `client.rpool` user account with the capability to only read objects with names containing an `rgb_` prefix from the `labpool1` pool.
Store the key-ring files for these two accounts in the correct location on `clienta`.
Store the `/etc/profile` file as the `my_profile` object in the `labpool1` pool.
5. Return to `workstation` as the `student` user.

Evaluation

Grade your work by running the `lab grade component-review` command from your `workstation` machine. Correct any reported failures and rerun the script until successful.

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

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 component-review
```

This concludes the lab.

► Solution

Creating Object Storage Cluster Components

In this lab, you will create and manage cluster components and authentication.

Outcomes

You should be able to create and configure BlueStore OSDs and pools, and set up authentication to the cluster.

Before You Begin

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

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

This command confirms that the hosts required for this exercise are accessible.

Instructions

1. Log in to `clienta` as the `admin` user. Create a new OSD daemon by using the `/dev/vde` device on `serverc`. View the details of the OSD. Restart the OSD daemon and verify it starts correctly.
 - 1.1. Log in to `serverc` as the `admin` user and use `sudo` to run the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]#
```

- 1.2. Create a new OSD daemon by using the `/dev/vde` device on `serverc`.

```
[ceph: root@clienta /]# ceph orch daemon add osd serverc.lab.example.com:/dev/vde
Created osd(s) 9 on host 'serverc.lab.example.com'
```

- 1.3. View the details of the OSD. The OSD ID might be different in your lab environment.

```
[ceph: root@clienta /]# ceph osd find 9
{
  "osd": 9,
  "addrs": {
    "addrvec": [
      {
        "type": "v2",
        "addr": "172.25.250.12:6816",
        "nonce": 2214147187
      }
    ]
  }
}
```

```

        },
        {
            "type": "v1",
            "addr": "172.25.250.12:6817",
            "nonce": 2214147187
        }
    ],
},
"osd_fsid": "eae3b333-24f3-46fb-83a5-b1de2559166b",
"host": "serverc.lab.example.com",
"crush_location": {
    "host": "serverc",
    "root": "default"
}
}

```

1.4. Restart the OSD daemon.

```
[ceph: root@clienta /]# ceph orch daemon restart osd.9
Scheduled to restart osd.9 on host 'serverc.lab.example.com'
```

1.5. Verify that the OSD starts correctly.

```
[ceph: root@clienta /]# ceph orch ps
...output omitted...
osd.9                               serverc.lab.example.com  running (13m)  2m ago
    21m -           16.2.0-117.el8cp  2142b60d7974  d4773b95c856
...output omitted...
```

2. Create a replicated pool called `labpool1` with 64 PGs. Set the number of replicas to 3. Set the application type to `rbd`. Set the `pg_auto_scale` mode to `on` for the pool.

2.1. Create a replicated pool called `labpool1` with 64 PGs.

```
[ceph: root@clienta /]# ceph osd pool create labpool1 64 64 replicated
pool 'labpool1' created
```

2.2. Set the number of replicas to 3. The pool ID might be different in your lab environment.

```
[ceph: root@clienta /]# ceph osd pool set labpool1 size 3
set pool 6 size to 3
```

2.3. Set the application type to `rbd`.

```
[ceph: root@clienta /]# ceph osd pool application enable labpool1 rbd
enabled application 'rbd' on pool 'labpool1'
```

2.4. Set the `pg_auto_scale` mode to `on` for the pool.

```
[ceph: root@clienta /]# ceph osd pool set labpool1 pg_autoscale_mode on
set pool 6 pg_autoscale_mode to on
```

3. Create an erasure code profile called k8m4 with data chunks on 8 OSDs ($k=8$), able to sustain the loss of 4 OSDs ($m=4$), and set `crush-failure-domain=rack`. Create an erasure coded pool called `labpool2` with 64 PGs that uses the k8m4 profile.
 - 3.1. Create an erasure code profile called k8m4 with data chunks on 8 OSDs ($k=8$), able to sustain the loss of 4 OSDs ($m=4$), and set `crush-failure-domain=rack`.

```
[ceph: root@clienta /]# ceph osd erasure-code-profile set k8m4 k=8 m=4 \
crush-failure-domain=rack
[ceph: root@clienta /]#
```

- 3.2. Create an erasure coded pool called `labpool2` with 64 PGs that use the k8m4 profile.

```
[ceph: root@clienta /]# ceph osd pool create labpool2 64 64 erasure k8m4
pool 'labpool2' created
```

4. Create the `client.rwpool` user account with the capabilities to read and write objects in the `labpool1` pool. This user must not be able to access the `labpool2` pool in any way.
Create the `client.rpool` user account with the capability to only read objects with names containing an `rgb_` prefix from the `labpool1` pool.
Store the key-ring files for these two accounts in the correct location on `clienta`.
Store the `/etc/profile` file as the `my_profile` object in the `labpool1` pool.
 - 4.1. Exit the `cephadm` shell, then interactively use `cephadm shell` to create the two accounts from the `clienta` host system. Create the `client.rwpool` user account with read and write access to the `labpool1` pool.

```
[ceph: root@clienta /]# exit
exit
[admin@clienta ~]$ sudo cephadm shell -- ceph auth get-or-create client.rwpool \
mon 'allow r' osd 'allow rw pool=labpool1' | sudo tee \
/etc/ceph/ceph.client.rwpool.keyring
[client.rwpool]
key = AQAn7FNhDd5u0RAAqZPIq7nU0yDWebk2EXuk0w==
```

Because you explicitly provide the `pool=labpool1` argument, no other pool is accessible by the user. Therefore, the `client.rwpool` user cannot access the `labpool2` pool, matching the requirements.

- 4.2. Create the `client.rpool` user account with read access to objects with names containing an `rgb_` prefix in the `labpool1` pool. Note that there is no equals sign (=) between `object_prefix` and its value.

```
[admin@clienta ~]$ sudo cephadm shell -- ceph auth get-or-create client.rpool \
mon 'allow r' osd 'allow r pool=labpool1 object_prefix my_' | sudo tee \
/etc/ceph/ceph.client.rpool.keyring
[client.rpool]
key = AQAD7VNhV0oWIhAAFTR+F3zuY3087n10aLELVA==
```

- 4.3. Use `sudo` to run a new `cephadm` shell with a bind mount from the host. Use the `rados` command to store the `/etc/profile` file as the `my_profile` object in the `labpool1` pool. Use the `client.rwpool` user account rather than the default `client.admin` account to test the access rights you defined for the user.

```
[admin@clienta ~]$ sudo cephadm shell --mount /etc/ceph:/etc/ceph  
[ceph: root@clienta /]# rados --id rpool -p labpool1 put my_profile /etc/profile
```

- 4.4. Verify that the `client.rpool` user can retrieve the `my_profile` object from the `labpool1` pool.

```
[ceph: root@clienta /]# rados --id rpool -p labpool1 get \  
my_profile /tmp/profile.out  
[ceph: root@clienta /]# diff /etc/profile /tmp/profile.out
```

5. Return to `workstation` as the `student` user.

- 5.1. Return to `workstation` as the `student` user.

```
[ceph: root@clienta /]# exit  
[admin@clienta ~]$ exit  
[student@workstation ~]$
```

Evaluation

Grade your work by running the `lab grade component-review` command from your `workstation` machine. Correct any reported failures and rerun the script until successful.

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

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 component-review
```

This concludes the lab.

Summary

In this chapter, you learned:

- BlueStore is the default storage back end for Red Hat Ceph Storage 5. It stores objects directly on raw block devices and improves performance over the previous FileStore back end.
- BlueStore OSDs use a RocksDB key-value database to manage metadata and store it on a BlueFS partition. Red Hat Ceph Storage 5 uses sharding by default for new OSDs.
- Block.db stores object metadata and the write-ahead log (WAL) stores journals. You can improve OSD performance by placing the block.db and WAL devices on faster storage than the object data.
- You can provision OSDs by using service specification files, by choosing a specific host and device, or automatically with the orchestrator service.
- Pools are logical partitions for storing objects. The available pool types are replicated and erasure coded.
- Replicated pools are the default type of pool, they copy each object to multiple OSDs.
- Erasure coded pools function by dividing object data into chunks (k), calculating coding chunks (m) based on the data chunks, then storing each chunk on separate OSDs. The coding chunks are used to reconstruct object data if an OSD fails.
- A pool namespace allows you to logically partition a pool and is useful for restricting storage access by an application.
- The cephx protocol authenticates clients and authorizes communication between clients, applications, and daemons in the cluster. It is based on shared secret keys.
- Clients can access the cluster when they are configured with a user account name and a key-ring file containing the user's secret key.
- Cephx capabilities provide a way to control access to pools and object data within pools.

Chapter 5

Creating and Customizing Storage Maps

Goal

Manage and adjust the CRUSH and OSD maps to optimize data placement to meet the performance and redundancy requirements of cloud applications.

Objectives

- Administer and update the cluster CRUSH map used by the Ceph cluster.
- Describe the purpose and modification of the OSD maps.

Sections

- Managing and Customizing the CRUSH Map (and Guided Exercise)
- Managing the OSD Map (and Guided Exercise)

Lab

Creating and Customizing Storage Maps

Managing and Customizing the CRUSH Map

Objectives

After completing this section, you should be able to administer and update the cluster CRUSH map used by the Ceph cluster.

CRUSH and Object Placement Strategies

Ceph calculates which OSDs should hold which objects by using a placement algorithm called CRUSH (Controlled Replication Under Scalable Hashing). Objects are assigned to placement groups (PGs) and CRUSH determines which OSDs those placement groups should use to store their objects.

The CRUSH Algorithm

The CRUSH algorithm enables Ceph clients to directly communicate with OSDs; this avoids a centralized service bottleneck. Ceph clients and OSDs use the CRUSH algorithm to efficiently compute information about object locations, instead of having to depend on a central lookup table. Ceph clients retrieve the cluster maps and use the CRUSH map to algorithmically determine how to store and retrieve data. This enables massive scalability for the Ceph cluster by avoiding a single point of failure and a performance bottleneck.

The CRUSH algorithm works to uniformly distribute the data in the object store, manage replication, and respond to system growth and hardware failures. When new OSDs are added or an existing OSD or OSD host fails, Ceph uses CRUSH to rebalance the objects in the cluster among the active OSDs.

CRUSH Map Components

Conceptually, a CRUSH map contains two major components:

A CRUSH hierarchy

This lists all available OSDs and organizes them into a treelike structure of *buckets*.

The CRUSH hierarchy is often used to represent where OSDs are located. By default, there is a root bucket representing the whole hierarchy, which contains a host bucket for each OSD host.

The OSDs are the leaves of the tree, and by default all OSDs on the same OSD host are placed in that host's bucket. You can customize the tree structure to rearrange it, add more levels, and group OSD hosts into buckets representing their location in different server racks or data centers.

At least one CRUSH rule

CRUSH rules determine how placement groups are assigned OSDs from those buckets. This determines where objects for those placement groups are stored. Different pools might use different CRUSH rules from the CRUSH map.

CRUSH Bucket Types

The CRUSH hierarchy organizes OSDs into a tree of different containers, called buckets. For a large installation, you can create a specific hierarchy to describe your storage infrastructure: data centers with rows of racks, racks, hosts, and OSD devices on those hosts. By creating a CRUSH map rule, you can cause Ceph to place an object's replicas on OSDs on separate servers, on servers in different racks, or even on servers in different data centers.

To summarize, buckets are the containers or branches in the CRUSH hierarchy. Devices are OSDs, and are leaves in the CRUSH hierarchy.

Some of the most important bucket attributes are:

- The ID of the bucket. These IDs are negative numbers to distinguish them from storage device IDs.
- The name of the bucket.
- The type of the bucket. The default map defines several types that you can retrieve with the `ceph osd crush dump` command.

Bucket types include `root`, `region`, `datacenter`, `room`, `pod`, `pdu`, `row`, `rack`, `chassis`, and `host`, but you can also add your own types. The bucket at the root of the hierarchy is of the `root` type.

- The algorithm that Ceph uses to select items inside the bucket when mapping PG replicas to OSDs. Several algorithms are available: `uniform`, `list`, `tree`, and `straw2`. Each algorithm represents a trade-off between performance and reorganization efficiency. The default algorithm is `straw2`.

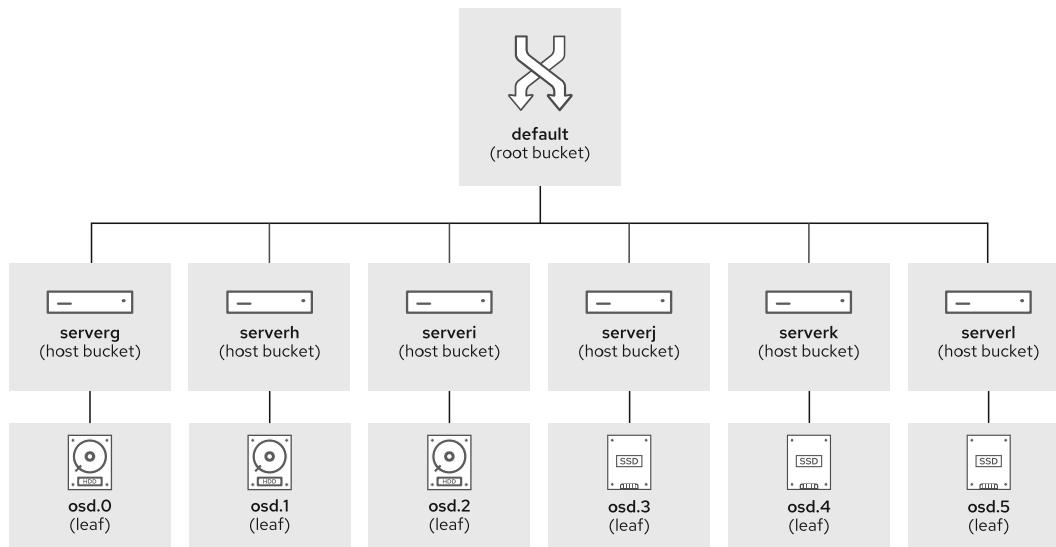


Figure 5.1: CRUSH map default hierarchy example

Customizing Failure and Performance Domains

The CRUSH map is the central configuration mechanism for the CRUSH algorithm. You can edit this map to influence data placement and customize the CRUSH algorithm.

Configuring the CRUSH map and creating separate failure domains allows OSDs and cluster nodes to fail without any data loss occurring. The cluster simply operates in a degraded state until the problem is fixed.

Configuring the CRUSH map and creating separate performance domains can reduce performance bottlenecks for clients and applications that use the cluster to store and retrieve data. For example, CRUSH can create one hierarchy for HDDs and another hierarchy for SSDs.

A typical use case for customizing the CRUSH map is to provide additional protection against hardware failures. You can configure the CRUSH map to match the underlying physical infrastructure, which helps mitigate the impact of hardware failures.

By default, the CRUSH algorithm places replicated objects on OSDs on different hosts. You can customize the CRUSH map so that object replicas are placed across OSDs in different shelves, or on hosts in different rooms, or in different racks with distinct power sources.

Another use case is to allocate OSDs with SSD drives to pools used by applications requiring very fast storage, and OSDs with traditional HDDs to pools supporting less demanding workloads.

The CRUSH map can contain multiple hierarchies that you can select through different CRUSH rules. By using separate CRUSH hierarchies, you can establish separate performance domains. Use case examples for configuring separate performance domains are:

- To separate block storage used by VMs from object storage used by applications.
- To separate "cold" storage, containing infrequently accessed data, from "hot" storage, containing frequently accessed data.

If you examine an actual CRUSH map definition, it contains:

- A list of all available physical storage devices.
- A list of all the infrastructure buckets and the IDs of the storage devices or other buckets in each of them. Remember that a bucket is a container, or a branch, in the infrastructure tree. For example, it might represent a location or a piece of physical hardware.
- A list of CRUSH rules to map PGs to OSDs.
- A list of other CRUSH tunables and their settings.

The cluster installation process deploys a default CRUSH map. You can use the `ceph osd crush dump` command to print the CRUSH map in JSON format. You can also export a binary copy of the map and decompile it into a text file:

```
[ceph: root@node /]# ceph osd getcrushmap -o ./map.bin  
[ceph: root@node /]# crushtool -d ./map.bin -o ./map.txt
```

Customizing OSD CRUSH Settings

The CRUSH map contains a list of all the storage devices in the cluster. For each storage device the following information is available:

- The ID of the storage device.
- The name of the storage device.
- The weight of the storage device, normally based on its capacity in terabytes. For example, a 4 TB storage device has a weight of about 4.0. This is the relative amount of data the device can store, which the CRUSH algorithm uses to help ensure uniform object distribution.

You can set the weight of an OSD with the `ceph osd crush reweight` command. CRUSH tree bucket weights should equal the sum of their leaf weights. If you manually edit the CRUSH map weights, then you should execute the following command to ensure that the CRUSH tree bucket weights accurately reflect the sum of the leaf OSDs within the bucket.

```
[ceph: root@node /]# ceph osd crush reweight-all
reweighted crush hierarchy
```

- The class of the storage device. Multiple types of storage devices can be used in a storage cluster, such as HDDs, SSDs, or NVMe SSDs. A storage device's class reflects this information and you can use that to create pools optimized for different application workloads. OSDs automatically detect and set their device class. You can explicitly set the device class of an OSD with the `ceph osd crush set-device-class` command. Use the `ceph osd crush rm-device-class` to remove a device class from an OSD.

The `ceph osd crush tree` command shows the CRUSH map's current CRUSH hierarchy:

```
[ceph: root@node /]# ceph osd crush tree
ID CLASS WEIGHT  TYPE NAME
-1      1.52031  root default
-3      0.48828  host serverg
  0  hdd 0.48828    osd.0
-5      0.48828  host serverh
  1  hdd 0.48828    osd.1
-7      0.48828  host serveri
  2  hdd 0.48828    osd.2
-9      0.01849  host serverj
  3  ssd 0.01849    osd.3
-11     0.01849  host serverk
  4  ssd 0.01849    osd.4
-13     0.01849  host serverl
  5  ssd 0.01849    osd.5
```

Device classes are implemented by creating a “shadow” CRUSH hierarchy for each device class in use that contains only devices of that class. CRUSH rules can then distribute data over the shadow hierarchy. You can view the CRUSH hierarchy with shadow items with the `ceph osd crush tree --show-shadow` command.

Create a new device class by using the `ceph osd crush class create` command. Remove a device class using the `ceph osd crush class rm` command.

List configured device classes with the `ceph osd crush class ls` command.

Using CRUSH Rules

The CRUSH map also contains the data placement rules that determine how PGs are mapped to OSDs to store object replicas or erasure coded chunks.

The `ceph osd crush rule ls` command lists the existing rules and the `ceph osd crush rule dump rule_name` command prints the details of a rule.

The decompiled CRUSH map also contains the rules and might be easier to read:

```
[ceph: root@node /]# ceph osd getcrushmap -o ./map.bin
[ceph: root@node /]# crushtool -d ./map.bin -o ./map.txt
[ceph: root@node /]# cat ./map.txt
...output omitted...
rule replicated_rule { ①
    id 0 ②
    type replicated
    min_size 1 ③
    max_size 10 ④
    step take default ⑤
    step chooseleaf firstn 0 type host ⑥
    step emit ⑦
}
...output omitted...
```

- ① The name of the rule. Use this name to select the rule when creating a pool with the `ceph osd pool create` command.
- ② The ID of the rule. Some commands use the rule ID instead of the rule name. For example, the `ceph osd pool set pool-name crush_ruleset ID` command, which sets the rule for an existing pool, uses the rule ID.
- ③ If a pool makes fewer replicas than this number, then CRUSH does not select this rule.
- ④ If a pool makes more replicas than this number, then CRUSH does not select this rule.
- ⑤ Takes a bucket name, and begins iterating down the tree. In this example, the iterations start at the bucket called `default`, which is the root of the default CRUSH hierarchy. With a complex hierarchy composed of multiple data centers, you could create a rule for a data center designed to force objects in specific pools to be stored in OSDs in that data center. In that situation, this step could start iterating at the data center bucket.
- ⑥ Selects a set of buckets of the given type (`host`) and chooses a leaf (OSD) from the subtree of each bucket in the set. In this example, the rule selects an OSD from each host bucket in the set, ensuring that the OSDs come from different hosts. The number of buckets in the set is usually the same as the number of replicas in the pool (the pool size):
 - If the number after `firstn` is 0, choose as many buckets as there are replicas in the pool.
 - If the number is greater than zero, and less than the number of replicas in the pool, choose that many buckets. In that case, the rule needs another step to draw buckets for the remaining replicas. You can use this mechanism to force the location of a subset of the object replicas.
 - If the number is less than zero, subtract its absolute value from the number of replicas and choose that many buckets.
- ⑦ Output the results of the rule.

For example, you could create the following rule to select as many OSDs as needed on separate racks, but only from the DC1 data center:

```
rule myrackruleinDC1 {
    id 2
    type replicated
```

```
min_size 1
max_size 10
step take DC1
step chooseleaf firstn 0 type rack
step emit
}
```

Using CRUSH Tunables

You can also modify the CRUSH algorithm's behavior using *tunables*. Tunables adjust and disable or enable features of the CRUSH algorithm. Ceph defines the tunables at the beginning of the decompiled CRUSH map, and you can get their current values by using the following command:

```
[ceph: root@node /]# ceph osd crush show-tunables
{
    "choose_local_tries": 0,
    "choose_local_fallback_tries": 0,
    "choose_total_tries": 50,
    "chooseleaf_descend_once": 1,
    "chooseleaf_vary_r": 1,
    "chooseleaf_stable": 1,
    "straw_calc_version": 1,
    "allowed_bucket_algs": 54,
    "profile": "jewel",
    "optimal_tunables": 1,
    "legacy_tunables": 0,
    "minimum_required_version": "jewel",
    "require_feature_tunables": 1,
    "require_feature_tunables2": 1,
    "has_v2_rules": 1,
    "require_feature_tunables3": 1,
    "has_v3_rules": 0,
    "has_v4_buckets": 1,
    "require_feature_tunables5": 1,
    "has_v5_rules": 0
}
```



Important

Adjusting CRUSH tunables will probably change how CRUSH maps placement groups to OSDs. When that happens, the cluster needs to move objects to different OSDs in the cluster to reflect the recalculated mappings. Cluster performance could degrade during this process.

Rather than modifying individual tunables, you can select a predefined profile with the `ceph osd crush tunables profile` command. Set the value of `profile` to `optimal` to enable the best (optimal) values for the current version of Red Hat Ceph Storage.



Important

Red Hat recommends that all cluster daemons and clients use the same release version.

CRUSH Map Management

The cluster keeps a compiled binary representation of the CRUSH map. You can modify it by:

- Using the `ceph osd crush` command.
- Extracting and decompiling the binary CRUSH map to plain text, editing the text file, recompiling it to binary format, and importing it back into the cluster.

It is usually easier to update the CRUSH map with the `ceph osd crush` command. However, there are less common scenarios which can only be implemented by using the second method.

Customizing the CRUSH Map Using Ceph Commands

This example creates a new bucket:

```
[ceph: root@node /]# ceph osd crush add-bucket name type
```

For example, these commands create three new buckets, one of the `datacenter` type and two of the `rack` type:

```
[ceph: root@node /]# ceph osd crush add-bucket DC1 datacenter
added bucket DC1 type datacenter to crush map
[ceph: root@node /]# ceph osd crush add-bucket rackA1 rack
added bucket rackA1 type rack to crush map
[ceph: root@node /]# ceph osd crush add-bucket rackB1 rack
added bucket rackB1 type rack to crush map
```

You can then organize the new buckets in a hierarchy with the following command:

```
[ceph: root@node /]# ceph osd crush move name type=parent
```

You also use this command to reorganize the tree. For example, the following commands attach the two rack buckets from the previous example to the data center bucket, and attaches the data center bucket to the default root bucket:

```
[ceph: root@node /]# ceph osd crush move rackA1 datacenter=DC1
moved item id -10 name 'rackA1' to location {datacenter=DC1} in crush map
[ceph: root@node /]# ceph osd crush move rackB1 datacenter=DC1
moved item id -11 name 'rackB1' to location {datacenter=DC1} in crush map
[ceph: root@node /]# ceph osd crush move DC1 root=default
moved item id -9 name 'DC1' to location {root=default} in crush map
```

Setting the Location of OSDs

After you have created your custom bucket hierarchy, place the OSDs as leaves on this tree. Each OSD has a location, which is a string defining the full path to the OSD from the root of the tree. For example, the location of an OSD attached to the `rackA1` bucket is:

```
root=default datacenter=DC1 rack=rackA1
```

When Ceph starts, it uses the `ceph-crush-location` utility to automatically verify that each OSD is in the correct CRUSH location. If the OSD is not in the expected location in the CRUSH map, it is automatically moved. By default, this is `root=default host=hostname`.

You can replace the `ceph-crush-location` utility with your own script to change where OSDs are placed in the CRUSH map. To do this, specify the `crush_location_hook` parameter in the `/etc/ceph/ceph.conf` configuration file.

```
...output omitted...
[osd]
crush_location_hook = /path/to/your/script
...output omitted...
```

Ceph executes the script with these arguments: `--cluster cluster-name --id osd-id --type osd`. The script must print the location as a single line on its standard output. The upstream Ceph documentation has an example of a custom script that assumes each system has an `/etc/rack` file containing the name of its rack:

```
#!/bin/sh
echo "root=default rack=$(cat /etc/rack) host=$(hostname -s)"
```

You can set the `crush_location` parameter in the `/etc/ceph/ceph.conf` configuration file to redefine the location for particular OSDs. For example, to set the location for `osd.0` and `osd.1`, add the `crush_location` parameter inside their respective sections in that file:

```
[osd.0]
crush_location = root=default datacenter=DC1 rack=rackA1

[osd.1]
crush_location = root=default datacenter=DC1 rack=rackB1
```

Adding CRUSH Map Rules

This example creates a rule that Ceph can use for replicated pools:

```
[ceph: root@node /]# ceph osd crush rule create-replicated name root
 \
failure-domain-type [class]
```

- `name` is the name of the rule.
- `root` is the starting node in the CRUSH map hierarchy.
- `failure-domain-type` is the bucket type for replication.
- `class` is the class of the devices to use, such as `ssd` or `hdd`. This parameter is optional.

The following example creates the new `inDC2` rule to store replicas in the DC2 data center, and distributes the replicas across racks:

```
[ceph: root@node /]# ceph osd crush rule create-replicated inDC2 DC2 rack
[ceph: root@node /]# ceph osd crush rule ls
replicated_rule
erasure-code
inDC2
```

After you have defined the rule, use it when creating a replicated pool:

```
[ceph: root@node /]# ceph osd pool create myfirstpool 50 50 inDC2
pool 'myfirstpool' created
```

For erasure coding, Ceph automatically creates rules for each erasure coded pool you create. The name of the rule is the name of the new pool. Ceph uses the rule parameters you define in the erasure code profile that you specify when you create the pool.

The following example first creates the new `myprofile` erasure code profile, then creates the `myecpool` pool based on this profile:

```
[ceph: root@node /]# ceph osd erasure-code-profile set myprofile k=2 m=1 \
crush-root=DC2 crush-failu re-domain=rack crush-device-class=ssd
[ceph: root@node /]# ceph osd pool create myecpool 50 50 erasure myprofile
pool 'myecpool' created
[ceph: root@node /]# ceph osd crush rule ls
replicated_rule
erasure-code
myecpool
```

Customizing the CRUSH Map by Decompiling the Binary Version

You can decompile and manually edit the CRUSH map with the following commands:

Command	Action
<code>ceph osd getcrushmap -o <i>binfile</i></code>	Export a binary copy of the current map.
<code>crushtool -d <i>binfile</i> -o <i>textfilepath</i></code>	Decompile a CRUSH map binary into a text file.
<code>crushtool -c <i>textfilepath</i> -o <i>binfile</i></code>	Compile a CRUSH map from text.
<code>crushtool -i <i>binfile</i> --test</code>	Perform dry runs on a binary CRUSH map and simulate placement group creation.
<code>ceph osd setcrushmap -i <i>binfile</i></code>	Import a binary CRUSH map into the cluster.



Note

The `ceph osd getcrushmap` and `ceph osd setcrushmap` commands provide a useful way to back up and restore the CRUSH map for your cluster.

Optimizing Placement Groups

Placement groups (PGs) allow the cluster to store millions of objects in a scalable way by aggregating them into groups. Objects are organized into placement groups based on the object's ID, the ID of the pool, and the number of placement groups in the pool.

During the cluster life cycle, the number of PGs must be adjusted as the cluster layout changes. CRUSH attempts to ensure a uniform distribution of objects among OSDs in the pool, but there are scenarios where the PGs become unbalanced. The placement group autoscaler can be used to optimize PG distribution, and is on by default. You can also manually set the number of PGs per pool, if required.

Objects are typically distributed uniformly, provided that there are one or two orders of magnitude (factors of ten) more placement groups than OSDs in the pool. If there are not enough PGs, then objects might be distributed unevenly. If there is a small number of very large objects stored in the pool, then object distribution might become unbalanced.



Note

PGs should be configured so that there are enough to evenly distribute objects across the cluster. If the number of PGs is set too high, then it increases CPU and memory use significantly. Red Hat recommends approximately 100 to 200 placement groups per OSD to balance these factors.

Calculating the Number of Placement Groups

For a cluster with a single pool, you can use the following formula, with 100 placement groups per OSD:

$$\text{Total PGs} = (\text{OSDs} * 100) / \text{Number of replicas}$$

Red Hat recommends the use of the Ceph Placement Groups per Pool Calculator, <https://access.redhat.com/labs/cephpgc/>, from the Red Hat Customer Portal Labs.

Mapping PGs Manually

Use the `ceph osd pg-upmap-items` command to manually map PGs to specific OSDs. Because older Ceph clients do not support it, you must configure the `ceph osd set-require-min-compat-client` setting to enable the `pg-upmap` command.

```
[ceph: root@node /]# ceph osd set-require-min-compat-client luminous
set require_min_compat_client to luminous
```

The following example remaps the PG 3.25 from OSDs 2 and 0 to 1 and 0:

```
[ceph: root@node /]# ceph pg map 3.25
osdmap e384 pg 3.25 (3.25) -> up [2,0] acting [2,0]
[ceph: root@node /]# ceph osd pg-upmap-items 3.25 2 1
set 3.25 pg_upmap_items mapping to [2->1]
[ceph: root@node /]# ceph pg map 3.25
osdmap e387 pg 3.25 (3.25) -> up [1,0] acting [1,0]
```

Remapping hundreds of PGs this way is not practical. The `osdmaptool` command is useful here. It takes the actual map for a pool, analyses it, and generates the `ceph osd pg-upmap-items` commands to run for an optimal distribution:

1. Export the map to a file. The following command saves the map to the `./om` file:

```
[ceph: root@node /]# ceph osd getmap -o ./om
got osdmap epoch 387
```

2. Use the `--test-map-pgs` option of the `osdmaptool` command to display the actual distribution of PGs. The following command prints the distribution for the pool with the ID of 3:

```
[ceph: root@node /]# osdmaptool ./om --test-map-pgs --pool 3
osdmaptool: osdmap file './om'
pool 3 pg_num 50
#osd count first primary c wt wt
osd.0    34    19 19 0.0184937 1
osd.1    39    14 14 0.0184937 1
osd.2    27    17 17 0.0184937 1
...output omitted...
```

This output shows that `osd.2` has only 27 PGs but `osd.1` has 39.

3. Generate the commands to rebalance the PGs. Use the `--upmap` option of the `osdmaptool` command to store the commands in a file:

```
[ceph: root@node /]# osdmaptool ./om --upmap ./cmds.txt --pool 3
osdmaptool: osdmap file './om'
writing upmap command output to: ./cmds.txt
checking for upmap cleanups
upmap, max-count 100, max deviation 0.01
[ceph: root@node /]# cat ./cmds.txt
ceph osd pg-upmap-items 3.1 0 2
ceph osd pg-upmap-items 3.3 1 2
ceph osd pg-upmap-items 3.6 0 2
...output omitted...
```

4. Execute the commands:

```
[ceph: root@node /]# bash ./cmds.txt
set 3.1 pg_upmap_items mapping to [0->2]
set 3.3 pg_upmap_items mapping to [1->2]
set 3.6 pg_upmap_items mapping to [0->2]
...output omitted...
```



References

For more information, refer to *Red Hat Ceph Storage 5 Strategies Guide* at
https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/storage_strategies_guide/

► Guided Exercise

Managing and Customizing the CRUSH Map

In this exercise, you will view and modify the cluster CRUSH map.

Outcomes

You should be able to create data placement rules to target a specific device class, create a pool by using a specific data placement rule, and decompile and edit the CRUSH map.

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 map-crush
```

This command confirms that the hosts required for this exercise are accessible, backs up the CRUSH map, adds the `ssd` device class, and sets the `mon_allow_pool_delete` setting to `true`.

Instructions

- 1. Log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell. Verify that the cluster returns a `HEALTH_OK` state.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]# ceph health
HEALTH_OK
```

- 2. Create a new CRUSH rule called `onssd` that uses only the OSDs backed by SSD storage. Create a new pool called `myfast` with 32 placement groups that use that rule. Confirm that the pool is using only OSDs that are backed by SSD storage.
- 2.1. List the available device classes in your cluster.

```
[ceph: root@clienta /]# ceph osd crush class ls
[
    "hdd",
    "ssd"
]
```

- 2.2. Display the CRUSH map tree to locate the OSDs backed by SSD storage.

```
[ceph: root@clienta /]# ceph osd crush tree
ID CLASS WEIGHT TYPE NAME
-1          0.08817  root default
-3          0.02939  host serverc
 0  hdd  0.00980      osd.0
 2  hdd  0.00980      osd.2
 1  ssd  0.00980      osd.1
-5          0.02939  host serverd
 3  hdd  0.00980      osd.3
 7  hdd  0.00980      osd.7
 5  ssd  0.00980      osd.5
-7          0.02939  host servere
 4  hdd  0.00980      osd.4
 8  hdd  0.00980      osd.8
 6  ssd  0.00980      osd.6
```

- 2.3. Add a new CRUSH map rule called onssd to target the OSDs with SSD devices.

```
[ceph: root@clienta /]# ceph osd crush rule create-replicated onssd \
default host ssd
```

- 2.4. Use the `ceph osd crush rule ls` command to verify the successful creation of the new rule.

```
[ceph: root@clienta /]# ceph osd crush rule ls
replicated_rule
onssd
```

- 2.5. Create a new replicated pool called `myfast` with 32 placement groups that uses the `onssd` CRUSH map rule.

```
[ceph: root@clienta /]# ceph osd pool create myfast 32 32 onssd
pool 'myfast' created
```

- 2.6. Verify that the placement groups for the pool called `myfast` are only using the OSDs backed by SSD storage. In a previous step, the OSDs are `osd.2`, `osd.5`, and `osd.8`. Retrieve the ID of the pool called `myfast`.

```
[ceph: root@clienta /]# ceph osd lspools
...output omitted...
6 myfast
```

- 2.7. Use the `ceph pg dump pgs_brief` command to list all the PGs in the cluster.

The pool ID is the first number in a PG ID. For example, the PG `6.1b` belongs to the pool whose ID is 6.

```
[ceph: root@clienta /]# ceph pg dump pgs_brief
PG_STAT STATE           UP      UP_PRIMARY ACTING ACTING_PRIMARY
6.1b    active+clean  [6,5,1]        6  [6,5,1]        6
4.19    active+clean  [6,2,5]        6  [6,2,5]        6
```

```

2.1f    active+clean [0,3,8]      0  [0,3,8]      0
3.1e    active+clean [2,6,3]      2  [2,6,3]      2
6.1a  active+clean [6,1,5]    6  [6,1,5]      6
4.18    active+clean [3,2,6]      3  [3,2,6]      3
2.1e    active+clean [2,6,5]      2  [2,6,5]      2
3.1f    active+clean [0,3,4]      0  [0,3,4]      0
6.19  active+clean [1,5,6]    1  [1,5,6]      1
4.1b    active+clean [3,2,8]      3  [3,2,8]      3
2.1d    active+clean [6,7,0]      6  [6,7,0]      6
...output omitted...

```

The pool called `myfast`, whose ID is 6, only uses `osd.1`, `osd.5`, and `osd.6`. These are the only OSDs with SSD drives.

- ▶ 3. Create a new CRUSH hierarchy under `root=default-cl260` that has three rack buckets (`rack1`, `rack2`, and `rack3`), each of which contains one host bucket (`hostc`, `hostd`, and `hoste`).

- 3.1. Create a new CRUSH map hierarchy that matches this infrastructure:

```

default-cl260  (root bucket)
  rack1        (rack bucket)
    hostc       (host bucket)
      osd.1
      osd.5
      osd.6

  rack2        (rack bucket)
    hostd       (host bucket)
      osd.0
      osd.3
      osd.4

  rack3        (rack bucket)
    hoste       (host bucket)
      osd.2
      osd.7
      osd.8

```

You should place the three SSDs (in this example are OSDs 1, 5, and 6) on `hostc`. Because in your cluster OSD numbers can be different, modify the CRUSH map hierarchy accordingly to this requirement.

First, create the buckets with the `ceph osd crush add-bucket` command.

```

[ceph: root@clienta /]# ceph osd crush add-bucket default-cl260 root
added bucket default-cl260 type root to crush map
[ceph: root@clienta /]# ceph osd crush add-bucket rack1 rack
added bucket rack1 type rack to crush map
[ceph: root@clienta /]# ceph osd crush add-bucket hostc host
added bucket hostc type host to crush map
[ceph: root@clienta /]# ceph osd crush add-bucket rack2 rack
added bucket rack2 type rack to crush map
[ceph: root@clienta /]# ceph osd crush add-bucket hostd host
added bucket hostd type host to crush map

```

```
[ceph: root@clienta /]# ceph osd crush add-bucket rack3 rack
added bucket rack3 type rack to crush map
[ceph: root@clienta /]# ceph osd crush add-bucket hoste host
added bucket hoste type host to crush map
```

3.2. Use the `ceph osd crush move` command to build the hierarchy.

```
[ceph: root@clienta /]# ceph osd crush move rack1 root=default-cl260
moved item id -14 name 'rack1' to location {root=default-cl260} in crush map
[ceph: root@clienta /]# ceph osd crush move hostc rack=rack1
moved item id -15 name 'hostc' to location {rack=rack1} in crush map
[ceph: root@clienta /]# ceph osd crush move rack2 root=default-cl260
moved item id -16 name 'rack2' to location {root=default-cl260} in crush map
[ceph: root@clienta /]# ceph osd crush move hostd rack=rack2
moved item id -17 name 'hostd' to location {rack=rack2} in crush map
[ceph: root@clienta /]# ceph osd crush move rack3 root=default-cl260
moved item id -18 name 'rack3' to location {root=default-cl260} in crush map
[ceph: root@clienta /]# ceph osd crush move hoste rack=rack3
moved item id -19 name 'hoste' to location {rack=rack3} in crush map
```

3.3. Display the CRUSH map tree to verify the new hierarchy.

```
[ceph: root@clienta /]# ceph osd crush tree
ID CLASS WEIGHT          TYPE NAME
ID  CLASS  WEIGHT    TYPE NAME
-13           0  root default-cl260
-14           0      rack rack1
-15           0      host hostc
-16           0      rack rack2
-17           0      host hostd
-18           0      rack rack3
-19           0      host hoste
-1        0.08817  root default
-3        0.02939  host serverc
  0   hdd  0.00980      osd.0
  2   hdd  0.00980      osd.2
  1   ssd  0.00980      osd.1
-5        0.02939  host serverd
  3   hdd  0.00980      osd.3
  7   hdd  0.00980      osd.7
  5   ssd  0.00980      osd.5
-7        0.02939  host servere
  4   hdd  0.00980      osd.4
  8   hdd  0.00980      osd.8
  6   ssd  0.00980      osd.6
```

3.4. Place the OSDs as leaves in the new tree.

```
[ceph: root@clienta /]# ceph osd crush set osd.1 1.0 root=default-cl260 \
rack=rack1 host=hostc
set item id 1 name 'osd.1' weight 1 at location
{host=hostc,rack=rack1,root=default-cl260} to crush map
[ceph: root@clienta /]# ceph osd crush set osd.5 1.0 root=default-cl260 \
rack=rack1 host=hostc
set item id 5 name 'osd.5' weight 1 at location
{host=hostc,rack=rack1,root=default-cl260} to crush map
[ceph: root@clienta /]# ceph osd crush set osd.6 1.0 root=default-cl260 \
rack=rack1 host=hostc
set item id 6 name 'osd.6' weight 1 at location
{host=hostc,rack=rack1,root=default-cl260} to crush map
[ceph: root@clienta /]# ceph osd crush set osd.0 1.0 root=default-cl260 \
rack=rack2 host=hostd
set item id 0 name 'osd.0' weight 1 at location
{host=hostd,rack=rack2,root=default-cl260} to crush map
[ceph: root@clienta /]# ceph osd crush set osd.3 1.0 root=default-cl260 \
rack=rack2 host=hostd
set item id 3 name 'osd.3' weight 1 at location
{host=hostd,rack=rack2,root=default-cl260} to crush map
[ceph: root@clienta /]# ceph osd crush set osd.4 1.0 root=default-cl260 \
rack=rack2 host=hostd
set item id 4 name 'osd.4' weight 1 at location
{host=hostd,rack=rack2,root=default-cl260} to crush map
[ceph: root@clienta /]# ceph osd crush set osd.2 1.0 root=default-cl260 \
rack=rack3 host=hoste
set item id 2 name 'osd.2' weight 1 at location
{host=hoste,rack=rack3,root=default-cl260} to crush map
[ceph: root@clienta /]# ceph osd crush set osd.7 1.0 root=default-cl260 \
rack=rack3 host=hoste
set item id 7 name 'osd.7' weight 1 at location
{host=hoste,rack=rack3,root=default-cl260} to crush map
[ceph: root@clienta /]# ceph osd crush set osd.8 1.0 root=default-cl260 \
rack=rack3 host=hoste
set item id 8 name 'osd.8' weight 1 at location
{host=hoste,rack=rack3,root=default-cl260} to crush map
```

3.5. Display the CRUSH map tree to verify the new OSD locations.

```
[ceph: root@clienta /]# ceph osd crush tree
ID  CLASS  WEIGHT  TYPE NAME
-13         9.00000  root default-cl260
-14         3.00000    rack rack1
-15         3.00000    host hostc
  1   ssd  1.00000        osd.1
  5   ssd  1.00000        osd.5
  6   ssd  1.00000        osd.6
-16         3.00000    rack rack2
-17         3.00000    host hostd
  0   hdd  1.00000        osd.0
  3   hdd  1.00000        osd.3
  4   hdd  1.00000        osd.4
-18         3.00000    rack rack3
```

```
-19      3.00000    host hoste
 2  hdd  1.00000          osd.2
 7  hdd  1.00000          osd.7
 8  hdd  1.00000          osd.8
-1          0  root default
-3          0  host serverc
-5          0  host serverd
-7          0  host servere
```

All the OSDs with SSD devices are in the `rack1` bucket and no OSDs are in the default tree.

- ▶ 4. Add a custom CRUSH rule by decompiling the binary CRUSH map and editing the resulting text file to add a new CRUSH rule called `ssd-first`. This rule always selects OSDs backed by SSD storage as the primary OSD, and OSDs backed by HDD storage as secondary OSDs for each placement group.

When the rule is created, compile the map and load it into your cluster. Create a new replicated pool called `testcrush` that uses the rule, and verify that its placement groups are mapped correctly.

Clients accessing the pools that are using this new rule will read data from fast drives because clients always read and write from the primary OSDs.

- 4.1. Retrieve the current CRUSH map by using the `ceph osd getcrushmap` command. Store the binary map in the `/home/ceph/cm-org.bin` file.

```
[ceph: root@clienta /]# ceph osd getcrushmap -o ~/cm-org.bin
...output omitted...
```

- 4.2. Use the `crushtool` command to decompile the binary map to the `~/cm-org.txt` text file. When successful, this command returns no output, but immediately use the `echo $?` command to determine its return code.

```
[ceph: root@clienta /]# crushtool -d ~/cm-org.bin -o ~/cm-org.txt
[ceph: root@clienta /]# echo $?
0
```

- 4.3. Save a copy of the CRUSH map as `~/cm-new.txt`, and add the following rule at the end of the file.

```
[ceph: root@clienta /]# cp ~/cm-org.txt ~/cm-new.txt
[ceph: root@clienta /]# cat ~/cm-new.txt
...output omitted...
rule onssd {
    id 3
    type replicated
    min_size 1
    max_size 10
    step take default class ssd
    step chooseleaf firstn 0 type host
    step emit
}
rule ssd-first {
    id 5
```

```

type replicated
min_size 1
max_size 10
step take rack1
step chooseleaf firstn 1 type host
step emit
step take default-cl260 class hdd
step chooseleaf firstn -1 type rack
step emit
}

# end crush map

```

With this rule, the first replica uses an OSD from `rack1` (backed by SSD storage), and the remaining replicas use OSDs backed by HDD storage from different racks.

4.4. Compile your new CRUSH map.

```
[ceph: root@clienta /]# crushtool -c ~/cm-new.txt -o ~/cm-new.bin
```

4.5. Before applying the new map to the running cluster, use the `crushtool` command with the `--show-mappings` option to verify that the first OSD is always from `rack1`.

```
[ceph: root@clienta /]# crushtool -i ~/cm-new.bin --test --show-mappings \
--rule=5 --num-rep 3
...output omitted...
CRUSH rule 5 x 1013 [5,4,7]
CRUSH rule 5 x 1014 [1,3,7]
CRUSH rule 5 x 1015 [6,2,3]
CRUSH rule 5 x 1016 [5,0,7]
CRUSH rule 5 x 1017 [6,0,8]
CRUSH rule 5 x 1018 [6,4,7]
CRUSH rule 5 x 1019 [1,8,3]
CRUSH rule 5 x 1020 [5,7,4]
CRUSH rule 5 x 1021 [5,7,4]
CRUSH rule 5 x 1022 [1,4,2]
CRUSH rule 5 x 1023 [1,7,4]
```

The first OSD is always 1, 5, or 6, which corresponds to the OSDs with SSD devices from `rack1`.

4.6. Apply the new CRUSH map to your cluster by using the `ceph osd setcrushmap` command.

```
[ceph: root@clienta /]# ceph osd setcrushmap -i ~/cm-new.bin
...output omitted...
```

4.7. Verify that the new `ssd-first` rule is now available.

```
[ceph: root@clienta /]# ceph osd crush rule ls
replicated_rule
onssd
ssd-first
```

- 4.8. Create a new replicated pool called `testcrush` with 32 placement groups and use the `ssd-first` CRUSH map rule.

```
[ceph: root@clienta /]# ceph osd pool create testcrush 32 32 ssd-first
cephpool 'testcrush' created
```

- 4.9. Verify that the first OSDs for the placement groups in the pool called `testcrush` are the ones from `rack1`. These OSDs are `osd.1`, `osd.5`, and `osd.6`.

```
[ceph: root@clienta /]# ceph osd lspools
...output omitted...
6 myfast
7 testcrush
[ceph: root@clienta /]# ceph pg dump pgs_brief | grep ^6
dumped pgs_brief
 7.b      active+clean [1,8,3]      1 [1,8,3]      1
 7.8     active+clean [5,3,7]      5 [5,3,7]      5
 7.9     active+clean [5,0,7]      5 [5,0,7]      5
 7.e     active+clean [1,2,4]      1 [1,2,4]      1
 7.f     active+clean [1,0,8]      1 [1,0,8]      1
 7.c     active+clean [6,0,8]      6 [6,0,8]      6
 7.d     active+clean [1,4,8]      1 [1,4,8]      1
 7.2     active+clean [6,8,0]      6 [6,8,0]      6
 7.3     active+clean [5,3,7]      5 [5,3,7]      5
 7.0     active+clean [5,0,7]      5 [5,0,7]      5
 7.5     active+clean [5,4,2]      5 [5,4,2]      5
...output omitted...
```

- ▶ 5. Use the `pg-upmap` feature to manually remap some secondary OSDs in one of the PGs in the `testcrush` pool.

- 5.1. Use the new `pg-upmap` optimization feature to manually map a PG to specific OSDs. Remap the second OSD of your PG from the previous step to another OSD of your choosing, except 1, 5 or, 6.

```
[ceph: root@clienta /]# ceph osd pg-upmap-items 7.8 3 0
set 7.8 pg_upmap_items mapping to [3->0]
```

- 5.2. Use the `ceph pg map` command to verify the new mapping. When done, log off from `clienta`.

```
[ceph: root@clienta /]# ceph pg map 7.8
osdmap e238 pg 7.8 (7.8) -> up [5,0,7] acting [5,0,7]
```

- ▶ 6. Return to workstation as the student user.

```
[ceph: root@clienta /]# exit  
[admin@clienta ~]$ exit  
[student@workstation ~]$
```

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 map-crush
```

This concludes the guided exercise.

Managing the OSD Map

Objectives

After completing this section, you should be able to describe the purpose and modification of the OSD maps.

Describing the OSD Map

The cluster OSD map contains the address and status of each OSD, the pool list and details, and other information such as the OSD near-capacity limit information. Ceph uses these last parameters to send warnings and to stop accepting write requests when an OSD reaches full capacity.

When a change occurs in the cluster's infrastructure, such as OSDs joining or leaving the cluster, the MONs update the corresponding map accordingly. The MONs maintain a history of map revisions. Ceph identifies each version of each map using an ordered set of incremented integers known as *epochs*.

The `ceph status -f json-pretty` command displays the epoch of each map. Use the `ceph map dump` subcommand to display each individual map, such as `ceph osd dump`.

```
[ceph: root@serverc /]# ceph status -f json-pretty
...output omitted...
{
    "osdmap": {
        "epoch": 478,
        "num_osds": 15,
        "num_up_osds": 15,
        "osd_up_since": 1632743988,
        "num_in_osds": 15,
        "osd_in_since": 1631712883,
        "num_remapped_pgs": 0
    },
...output omitted...
[ceph: root@serverc /]# ceph osd dump
epoch 478
fsid 11839bde-156b-11ec-bb71-52540000fa0c
created 2021-09-14T14:50:39.401260+0000
modified 2021-09-27T12:04:26.832212+0000
flags sortbitwise,recovery_deletes,purged_snapdirs,pglog_hardlimit
crush_version 69
full_ratio 0.95
backfillfull_ratio 0.9
nearfull_ratio 0.85
require_min_compat_client luminous
min_compat_client luminous
require_osd_release pacific
stretch_mode_enabled false
```

```

pool 1 'device_health_metrics' replicated size 3 min_size 2 crush_rule 0
object_hash rjenkins pg_num 1 pgp_num 1 autoscale_mode on last_change 475 flags
hashpspool stripe_width 0 pg_num_min 1 application mgr_devicehealth
...output omitted...
osd.0 up in weight 1 up_from 471 up_thru 471 down_at 470 last_clean_interval
[457,466) [v2:172.25.250.12:6801/1228351148,v1:172.25.250.12:6802/1228351148]
[v2:172.25.249.12:6803/1228351148,v1:172.25.249.12:6804/1228351148] exists,up
cfe311b0-dea9-4c0c-a1ea-42aaac4cb160
...output omitted...

```

Analyzing OSD Map Updates

Ceph updates the OSD map every time an OSD joins or leaves the cluster. An OSD can leave the Ceph cluster either because of an OSD failure or a hardware failure.

Even though the cluster map as a whole is maintained by the MONs, OSDs do not use a leader to manage the OSD map; they propagate the map among themselves. OSDs tag every message they exchange with the OSD map epoch. When an OSD detects that it is lagging behind, it performs a map update with its peer OSD.

In large clusters, where OSD map updates are frequent, it is not practical to always distribute the full map. Instead, receiving OSD nodes perform incremental map updates.

Ceph also tags the messages between OSDs and clients with the epoch. Whenever a client connects to an OSD, the OSD inspects the epoch. If the epoch does not match, then the OSD responds with the correct increment so that the client can update its OSD map. This negates the need for aggressive propagation, because clients learn about the updated map only at the time of next contact.

Updating Cluster Maps with Paxos

To access a Ceph cluster, a client first retrieves a copy of the cluster map from the MONs. All MONs must have the same cluster map for the cluster to function correctly.

MONs use the Paxos algorithm as a mechanism to ensure that they agree on the cluster state. Paxos is a distributed consensus algorithm. Every time a MON modifies a map, it sends the update to the other monitors through Paxos. Ceph only commits the new version of the map after a majority of monitors agree on the update.

The MON submits a map update to Paxos and only writes the new version to the local key-value store after Paxos acknowledges the update. The read operations directly access the key-value store.

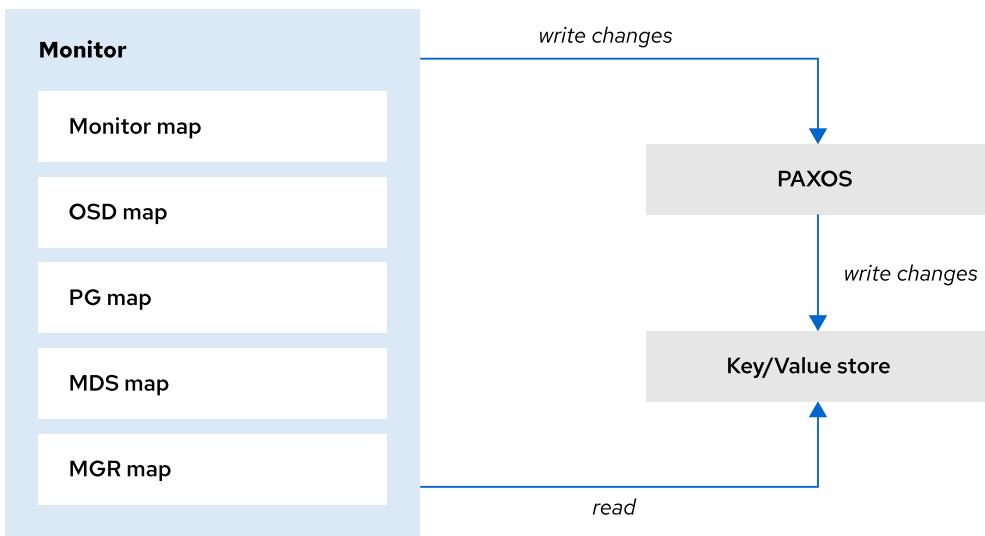


Figure 5.2: Cluster map consistency using Paxos

Propagating the OSD Map

OSDs regularly report their status to the monitors. In addition, OSDs exchange heartbeats so that an OSD can detect the failure of a peer and report that event to the monitors.

When a leader monitor learns of an OSD failure, it updates the map, increments the epoch, and uses the Paxos update protocol to notify the other monitors, at the same time revoking their leases. After a majority of monitors acknowledge the update, and the cluster has a quorum, the leader monitor issues a new lease so that the monitors can distribute the updated OSD map. This method avoids the map epoch ever going backwards anywhere in the cluster, and finding previous leases that are still valid.

OSD Map Commands

Use the following commands to manage the OSD map as an administrator:

Command	Action
<code>ceph osd dump</code>	Dump the OSD map to standard output.
<code>ceph osd getmap -o <i>binfile</i></code>	Export a binary copy of the current map.
<code>osdmaptool --print <i>binfile</i></code>	Display a human-readable copy of the map to standard output.
<code>osdmaptool --export-crush <i>crushbinfile</i> <i>binfile</i></code>	Extract the CRUSH map from the OSD map.
<code>osdmaptool --import-crush <i>crushbinfile</i> <i>binfile</i></code>	Embed a new CRUSH map.
<code>osdmaptool --test-map-pg <i>pgid</i> <i>binfile</i></code>	Verify the mapping of a given PG.



References

For more information, refer to the *Red Hat Ceph Storage 5 Storage Strategies Guide* at
https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/storage_strategies_guide

► Guided Exercise

Managing the OSD Map

In this exercise, you will modify and verify OSD maps for a common use case.

Outcomes

You should be able to display the OSD map and modify the OSD near-full and full ratios.

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 map-osd
```

This command confirms that the hosts required for this exercise are accessible. It resets the `full_ratio` and `nearfull_ratio` settings to the default values, and installs the `ceph-base` package on servera.

Instructions

- ▶ 1. Log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm shell`. Verify that the cluster status is `HEALTH_OK`.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]# ceph health
HEALTH_OK
```

- ▶ 2. Run the `ceph osd dump` command to display the OSD map. Record the current epoch value in your lab environment. Record the value of the `full_ratio` and `nearfull_ratio` settings.

Verify that the status of each OSD is up and in.

```
[ceph: root@clienta /]# ceph osd dump
epoch 478
fsid 11839bde-156b-11ec-bb71-52540000fa0c
created 2021-09-14T14:50:39.401260+0000
modified 2021-09-27T12:04:26.832212+0000
flags sortbitwise,recovery_deletes,purged_snapdirs,pglog_hardlimit
crush_version 69
full_ratio 0.95
backfillfull_ratio 0.9
nearfull_ratio 0.85
require_min_compat_client luminous
min_compat_client luminous
require_osd_release pacific
stretch_mode_enabled false
```

```
pool 1 'device_health_metrics' replicated size 3 min_size 2 crush_rule 0
object_hash rjenkins pg_num 1 pgp_num 1 autoscale_mode on last_change 475 flags
hashpspool stripe_width 0 pg_num_min 1 application mgr_devicehealth
...output omitted...
osd.0 up in weight 1 up_from 471 up_thru 471 down_at 470 last_clean_interval
[457,466) [v2:172.25.250.12:6801/1228351148,v1:172.25.250.12:6802/1228351148]
[v2:172.25.249.12:6803/1228351148,v1:172.25.249.12:6804/1228351148] exists,up
cfe311b0-dea9-4c0c-a1ea-42aaac4cb160
...output omitted...
```

► 3. Set the `full_ratio` and `nearfull_ratio` and verify the values.

- 3.1. Set the `full_ratio` parameter to 0.97 (97%) and `nearfull_ratio` to 0.9 (90%).

```
[ceph: root@clienta /]# ceph osd set-full-ratio 0.97
osd set-full-ratio 0.97
[ceph: root@clienta /]# ceph osd set-nearfull-ratio 0.9
osd set-nearfull-ratio 0.9
```

- 3.2. Verify the `full_ratio` and `nearfull_ratio` values. Compare this epoch value with the value from the previous dump of the OSD map. The epoch has incremented two versions because each `ceph osd set-* -ratio` command produces a new OSD map version.

```
[ceph: root@clienta /]# ceph osd dump
epoch 480
fsid 11839bde-156b-11ec-bb71-52540000fa0c
created 2021-09-14T14:50:39.401260+0000
modified 2021-09-27T12:27:38.328351+0000
flags sortbitwise,recovery_deletes,purged_snapdirs,pglog_hardlimit
crush_version 69
full_ratio 0.97
backfillfull_ratio 0.9
nearfull_ratio 0.9
...output omitted...
```

► 4. Extract and view the OSD map.

- 4.1. Instead of using the `ceph osd dump` command, use the `ceph osd getmap` command to extract a copy of the OSD map to a binary file and the `osdmaptool` command to view the file.

Use the `ceph osd getmap` command to save a copy of the OSD map in the `map.bin` file.

```
[ceph: root@clienta /]# ceph osd getmap -o map.bin
got osdmap epoch 480
```

- 4.2. Use the `osdmaptool --print` command to display the text version of the binary OSD map. The output is similar to the output of the `ceph osd dump` command.

```
[ceph: root@clienta /]# osdmaptool --print map.bin
osdmaptool: osdmap file 'map.bin'
epoch 480
fsid 11839bde-156b-11ec-bb71-52540000fa0c
created 2021-09-14T14:50:39.401260+0000
modified 2021-09-27T12:27:38.328351+0000
flags sortbitwise,recovery_deletes,purged_snapdirs,pglog_hardlimit
crush_version 69
full_ratio 0.97
backfillfull_ratio 0.9
nearfull_ratio 0.9
...output omitted...
```

- ▶ 5. Extract and decompile the current CRUSH map, then compile and import the CRUSH map. You will not change any map settings, but only observe the change in the epoch.

- 5.1. Use the `osdmaptool --export-crush` command to extract a binary copy of the CRUSH map and save it in the `crush.bin` file.

```
[ceph: root@clienta /]# osdmaptool --export-crush crush.bin map.bin
osdmaptool: osdmap file 'map.bin'
osdmaptool: exported crush map to crush.bin
```

- 5.2. Use the `crushtool` command to decompile the binary CRUSH map.

```
[ceph: root@clienta /]# crushtool -d crush.bin -o crush.txt
```

- 5.3. Use the `crushtool` command to compile the CRUSH map using the `crush.txt` file. Send the output to the `crushnew.bin` file.

```
[ceph: root@clienta /]# crushtool -c crush.txt -o crushnew.bin
```

- 5.4. Use the `osdmaptool --import-crush` command to import the new binary CRUSH map into a copy of the binary OSD map.

```
[ceph: root@clienta /]# cp map.bin mapnew.bin
[ceph: root@clienta /]# osdmaptool --import-crush crushnew.bin mapnew.bin
osdmaptool: osdmap file 'mapnew.bin'
osdmaptool: imported 1300 byte crush map from crushnew.bin
osdmaptool: writing epoch 482 to mapnew.bin
```

- ▶ 6. Use the `osdmaptool` command to test the impact of changes to the CRUSH map before applying them in production.

- 6.1. Run the `osdmaptool --test-map-pgs-dump` command to display the mapping between PGs and OSDs. The `osdmaptool` command output might be different in your lab environment.

```
[ceph: root@clienta /]# osdmaptool --test-map-pgs-dump mapnew.bin
osdmaptool: osdmap file 'mapnew.bin'
pool 3 pg_num 32
1.0      [3,7,2] 3
1.1      [7,0,5] 7
1.2      [4,0,8] 4
...output omitted...
```

6.2. Return to workstation as the student user.

```
[ceph: root@clienta /]# exit
[admin@clienta ~]$ exit
[student@workstation ~]$
```

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 map-osd
```

This concludes the guided exercise.

▶ Lab

Creating and Customizing Storage Maps

In this lab, you will modify the CRUSH map, create a CRUSH rule, and set the CRUSH tunables profile.

Outcomes

You should be able to create a new CRUSH hierarchy and move OSDs into it, create a CRUSH rule and configure a replicated pool to use it, and set the CRUSH tunables profile.

Before You Begin

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

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

This command confirms that the hosts required for this exercise are accessible, backs up the CRUSH map, and sets the `mon_allow_pool_delete` setting to `true`.

Instructions

1. Create a new CRUSH hierarchy under `root=review-cl260` that has two data center buckets (`dc1` and `dc2`), two rack buckets (`rack1` and `rack2`), one in each data center, and two host buckets (`hostc` and `hostd`), one in each rack.
Place `osd.1` and `osd.2` into `dc1`, `rack1`, `hostc`.
Place `osd.3` and `osd.4` into `dc2`, `rack2`, `hostd`.
2. Add a CRUSH rule called `replicated1` of type `replicated`. Set the root to `review-cl260` and the failure domain to `datacenter`.
3. Create a new replicated pool called `reviewpool` with 64 PGs that use the new CRUSH rule from the previous step.
4. Set CRUSH tunables to use the `optimal` profile.
5. Return to `workstation` as the student user.

Evaluation

Grade your work by running the `lab grade map-review` command from your `workstation` machine. Correct any reported failures and rerun the script until successful.

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

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 map-review
```

This concludes the lab.

► Solution

Creating and Customizing Storage Maps

In this lab, you will modify the CRUSH map, create a CRUSH rule, and set the CRUSH tunables profile.

Outcomes

You should be able to create a new CRUSH hierarchy and move OSDs into it, create a CRUSH rule and configure a replicated pool to use it, and set the CRUSH tunables profile.

Before You Begin

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

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

This command confirms that the hosts required for this exercise are accessible, backs up the CRUSH map, and sets the `mon_allow_pool_delete` setting to `true`.

Instructions

1. Create a new CRUSH hierarchy under `root=review-cl260` that has two data center buckets (`dc1` and `dc2`), two rack buckets (`rack1` and `rack2`), one in each data center, and two host buckets (`hostc` and `hostd`), one in each rack.

Place `osd.1` and `osd.2` into `dc1`, `rack1`, `hostc`.

Place `osd.3` and `osd.4` into `dc2`, `rack2`, `hostd`.

- 1.1. Log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]#
```

- 1.2. Create the buckets with the `ceph osd crush add-bucket` command.

```
[ceph: root@clienta /]# ceph osd crush add-bucket review-cl260 root
added bucket review-cl260 type root to crush map
[ceph: root@clienta /]# ceph osd crush add-bucket dc1 datacenter
added bucket dc1 type datacenter to crush map
[ceph: root@clienta /]# ceph osd crush add-bucket dc2 datacenter
added bucket dc2 type datacenter to crush map
[ceph: root@clienta /]# ceph osd crush add-bucket rack1 rack
added bucket rack1 type rack to crush map
[ceph: root@clienta /]# ceph osd crush add-bucket rack2 rack
added bucket rack2 type rack to crush map
[ceph: root@clienta /]# ceph osd crush add-bucket hostc host
```

```
added bucket hostc type host to crush map
[ceph: root@clienta /]# ceph osd crush add-bucket hostd host
added bucket hostd type host to crush map
```

- 1.3. Use the `ceph osd crush move` command to build the hierarchy.

```
[ceph: root@clienta /]# ceph osd crush move dc1 root=review-cl260
moved item id -10 name 'dc1' to location {root=review-cl260} in crush map
[ceph: root@clienta /]# ceph osd crush move dc2 root=review-cl260
moved item id -11 name 'dc2' to location {root=review-cl260} in crush map
[ceph: root@clienta /]# ceph osd crush move rack1 datacenter=dc1
moved item id -12 name 'rack1' to location {datacenter=dc1} in crush map
[ceph: root@clienta /]# ceph osd crush move rack2 datacenter=dc2
moved item id -13 name 'rack2' to location {datacenter=dc2} in crush map
[ceph: root@clienta /]# ceph osd crush move hostc rack=rack1
moved item id -14 name 'hostc' to location {rack=rack1} in crush map
[ceph: root@clienta /]# ceph osd crush move hostd rack=rack2
moved item id -15 name 'hostd' to location {rack=rack2} in crush map
```

- 1.4. Place the OSDs as leaves in the new tree and set all OSD weights to 1.0.

```
[ceph: root@clienta /]# ceph osd crush set osd.1 1.0 root=review-cl260 \
datacenter=dc1 rack=rack1 host=hostc
set item id 1 name 'osd.1' weight 1 at location
{datacenter=dc1,host=hostc,rack=rack1,root=review-cl260} to crush map
[ceph: root@clienta /]# ceph osd crush set osd.2 1.0 root=review-cl260 \
datacenter=dc1 rack=rack1 host=hostc
set item id 2 name 'osd.2' weight 1 at location
{datacenter=dc1,host=hostc,rack=rack1,root=review-cl260} to crush map
[ceph: root@clienta /]# ceph osd crush set osd.3 1.0 root=review-cl260 \
datacenter=dc2 rack=rack2 host=hostd
set item id 3 name 'osd.3' weight 1 at location
{datacenter=dc2,host=hostd,rack=rack2,root=review-cl260} to crush map
[ceph: root@clienta /]# ceph osd crush set osd.4 1.0 root=review-cl260 \
datacenter=dc1 rack=rack2 host=hostd
set item id 4 name 'osd.4' weight 1 at location
{datacenter=dc1,host=hostd,rack=rack2,root=review-cl260} to crush map
```

- 1.5. Display the CRUSH map tree to verify the new hierarchy and OSD locations.

```
[ceph: root@clienta /]# ceph osd tree
ID CLASS WEIGHT TYPE NAME STATUS REWEIGHT PRI-AFF
-9        4.00000 root review-cl260
-10       2.00000    datacenter dc1
-12       2.00000          rack rack1
-14       2.00000          host hostc
  1   hdd 1.00000            osd.1      up  1.00000 1.00000
  2   hdd 1.00000            osd.2      up  1.00000 1.00000
-11       2.00000    datacenter dc2
-13       2.00000          rack rack2
-15       2.00000          host hostd
  3   hdd 1.00000            osd.3      up  1.00000 1.00000
  4   hdd 1.00000            osd.4      up  1.00000 1.00000
```

```
-1      0.04898 root default
-3      0.00980    host serverc
0  hdd  0.00980        osd.0          up  1.00000 1.00000
-5      0.00980    host serverd
5  hdd  0.00980        osd.5          up  1.00000 1.00000
-7      0.02939    host servere
6  hdd  0.00980        osd.6          up  1.00000 1.00000
7  hdd  0.00980        osd.7          up  1.00000 1.00000
8  hdd  0.00980        osd.8          up  1.00000 1.00000
```

2. Add a CRUSH rule called `replicated1` of type `replicated`. Set the root to `review-cl260` and the failure domain to `datacenter`.

- 2.1. Use the `ceph osd crush rule create-replicated` command to create the rule.

```
[ceph: root@clienta /]# ceph osd crush rule create-replicated replicated1 \
review-cl260 datacenter
```

- 2.2. Verify that the `replicated1` CRUSH rule was created correctly. Record the CRUSH rule ID, it might be different in your lab environment.

```
[ceph: root@clienta /]# ceph osd crush rule dump | grep -B2 -A 20 replicated1
{
    "rule_id": 1,
    "rule_name": "replicated1",
    "ruleset": 1,
    "type": 1,
    "min_size": 1,
    "max_size": 10,
    "steps": [
        {
            "op": "take",
            "item": -9,
            "item_name": "review-cl260"
        },
        {
            "op": "chooseleaf_firstrn",
            "num": 0,
            "type": "datacenter"
        },
        {
            "op": "emit"
        }
    ]
}
```

3. Create a new replicated pool called `reviewpool` with 64 PGs that use the new CRUSH rule from the previous step.

- 3.1. Create the pool.

```
[ceph: root@clienta /]# ceph osd pool create reviewpool 64 64 \
replicated replicated1
pool 'reviewpool' created
```

- 3.2. Verify that the pool was created correctly. The pool ID and CRUSH rule ID might be different in your lab environment. Compare the CRUSH rule ID with the output of the previous step.

```
[ceph: root@clienta /]# ceph osd pool ls detail | grep reviewpool
pool 5 'reviewpool' replicated size 3 min_size 2 crush_rule 1 object_hash rjenkins
pg_num 64 pgp_num 64 autoscale_mode warn last_change 155 flags hashpspool
stripe_width 0
```

4. Set CRUSH tunables to use the optimal profile.

- 4.1. Set the CRUSH tunable profile to optimal.

```
[ceph: root@clienta /]# ceph osd crush tunables optimal
adjusted tunables profile to optimal
```

5. Return to `workstation` as the `student` user.

- 5.1. Return to `workstation` as the `student` user.

```
[ceph: root@clienta /]# exit
[admin@clienta ~]$ exit
[student@workstation ~]$
```

Evaluation

Grade your work by running the `lab grade map-review` command from your `workstation` machine. Correct any reported failures and rerun the script until successful.

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

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 map-review
```

This concludes the lab.

Summary

In this chapter, you learned:

- The CRUSH algorithm provides a decentralized way for Ceph clients to interact with the Red Hat Ceph Storage cluster, which enables massive scalability.
- The CRUSH map contains two main components: a hierarchy of buckets that organize OSDs into a treelike structure where the OSDs are the leaves of the tree, and at least one CRUSH rule that determines how Ceph assigns PGs to OSDs from the CRUSH tree.
- Ceph provides various command-line tools to display, tune, modify, and use the CRUSH map.
- You can modify the CRUSH algorithm's behavior by using `tunables`, which disable, enable, or adjust features of the CRUSH algorithm.
- The OSD map epoch is the map's revision number and increments whenever a change occurs. Ceph updates the OSD map every time an OSD joins or leaves the cluster and OSDs keep the map synchronized among themselves.

Chapter 6

Providing Block Storage Using RADOS Block Devices

Goal

Configure Red Hat Ceph Storage to provide block storage for clients using RADOS block devices (RBDs).

Objectives

- Provide block storage to Ceph clients using RADOS block devices (RBDs), and manage RBDs from the command line.
- Create and configure RADOS block devices snapshots and clones.
- Export an RBD image from the cluster to an external file and import it into another cluster.

Sections

- Managing RADOS Block Devices (and Guided Exercise)
- Managing RADOS Block Device Snapshots (and Guided Exercise)
- Importing and Exporting RBD Images (and Guided Exercise)

Lab

Providing Block Storage Using RADOS Block Devices

Managing RADOS Block Devices

Objectives

After completing this section, you should be able to provide block storage to Ceph clients using RADOS block devices (RBDs), and manage RBDs from the command line.

Block Storage Using a RADOS Block Device (RBD)

Block devices are the most common long-term storage devices for servers, laptops, and other computing systems. They store data in fixed-size blocks. Block devices include both hard drives, based on spinning magnetic platters, and solid-state drives, based on nonvolatile memory. To use the storage, format a block device with a file system and mount it on the Linux file system hierarchy.

The *RADOS Block Device (RBD)* feature provides block storage from the Red Hat Ceph Storage cluster. RADOS provides virtual block devices stored as *RBD images* in pools in the Red Hat Ceph Storage cluster.

Managing and Configuring RBD Images

As a storage administrator, use the `rbd` command to create, list, retrieve information from, resize, and remove block device images. The following example procedure creates an RBD image:

- Ensure that the `rbd` pool (or custom pool) for your RBD images exists. Use the `ceph osd pool create` command to create a custom pool to store RBD images. After creating the custom pool, initialize it with the `rbd pool init` command.
- Although Ceph administrators can access the pool, Red Hat recommends that you create a more restricted Cephx user for clients by using the `ceph auth` command. Grant the restricted user read/write access to only the needed RBD pool instead of access to the entire cluster.
- Create the RBD image with the `rbd create --size size pool-name/image-name` command. This command uses the default pool name if you do not specify a pool name.

The `rbd_default_pool` parameter specifies the name of the default pool used to store RBD images. Use `ceph config set osd rbd_default_pool value` to set this parameter.

Accessing RADOS Block Device Storage

The kernel RBD client (`krbd`) maps an RBD image to a Linux block device. The `librbd` library provides RBD storage to KVM virtual machines and OpenStack cloud instances. These clients enable bare-metal servers or virtual machines to use the RBD images as normal block-based storage. In an OpenStack environment, OpenStack attaches and maps these RBD images to Linux servers where they can serve as boot devices. Red Hat Ceph Storage disperses the actual storage used by the virtual block devices across the cluster, which provides high performance access using the IP network.

Accessing Ceph Storage with the RBD Kernel Client

Ceph clients can mount an RBD image using the native Linux kernel module, `krbd`. This module maps RBD images to Linux block devices with names such as `/dev/rbd0`.

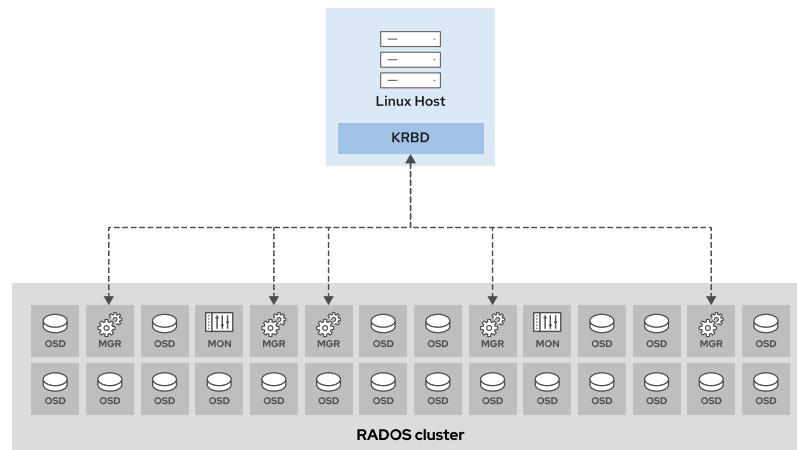


Figure 6.1: Kernel environment access

The `rbd device map` command uses the `krbd` kernel module to map an image. The `rbd map` command is an abbreviated form of the `rbd device map` command. The `rbd device unmap`, or `rbd unmap`, command uses the `krbd` kernel module to unmap a mapped image. The following example command maps the `test` RBD image in the `rbd` pool to the `/dev/rbd0` device on the host client machine:

```
[root@node ~]# rbd map rbd/test
/dev/rbd0
```

A Ceph client system can use the mapped block device, called `/dev/rbd0` in the example, like any other block device. You can format it with a file system, mount it, and unmount it.



Warning

Two clients can map the same RBD image as a block device at the same time. This can be useful for high availability clustering for standby servers, but Red Hat recommends attaching a block device to one client at a time when the block device contains a normal, single-mount file system. Mounting a RADOS block device that contains a normal file system, such as XFS, on two or more clients at the same time can cause file-system corruption and data loss.

The `rbd device list` command, abbreviated `rbd showmapped`, lists the RBD images mapped in the machine.

```
[root@node ~]# rbd showmapped
id  pool  namespace  image  snap  device
0   rbd    test       -      -     /dev/rbd0
```

The `rbd device unmap` command, abbreviated `rbd unmap`, unmaps the RBD image from the client machine.

```
[root@node ~]# rbd unmap /dev/rbd0
```

The `rbd map` and `rbd unmap` commands require root privileges.

Mapping RBD Images Persistently

The `rbdmap` service can automatically map and unmap RBD images to devices when booting and shutting down the system. This service looks for the mapped images with their credentials in the `/etc/ceph/rbdmap` file. The service mounts and unmounts the RBD images using their mount points as they appear in the `/etc/fstab` file.

The following steps configure `rbdmap` to persistently map and unmap an RBD image that already contains a file system:

1. Create the mount point for the file system.
2. Create a single-line entry in the `/etc/ceph/rbdmap` RBD map file. This entry must specify the name of the RBD pool and image. It must also reference the Cephx user who has read/write permissions to access the image and the corresponding key-ring file. Ensure that the key-ring file for the Cephx user exists on the client system.
3. Create an entry for the RBD in the `/etc/fstab` file on the client system. The name of the block device has the following form:

```
/dev/rbd/pool_name/image_name
```

Specify the `noauto` mount option, because the `rbdmap` service, not the Linux `fstab` routines, handles the mounting of the file system.

4. Confirm that the block device mapping works. Use the `rbdmap map` command to mount the devices. Use the `rbdmap unmap` command to unmount them.
5. Enable the `rbdmap` `systemd` service.

Refer to `rbdmap(8)` for more information.

Accessing Ceph Storage with librbd-based Clients

The `librbd` library provides direct access to RBD images for user space applications. It inherits the capabilities of `librados` to map data blocks to objects in the Ceph Object Store, and implements the ability to access RBD images and create snapshots and clones.

Cloud and virtualization solutions, such as OpenStack and `libvirt`, use `librbd` to provide RBD images as block devices to cloud instances and the virtual machines that they manage. For example, RBD images can store QEMU virtual machine images. Using the RBD clone feature, virtualized containers can boot a virtual machine without copying the boot image. The copy-on-write (COW) mechanism copies data from the parent to the clone when it writes to an unallocated object within the clone. The copy-on-read (COR) mechanism copies data from the parent to the clone when it reads from an unallocated object within the clone.

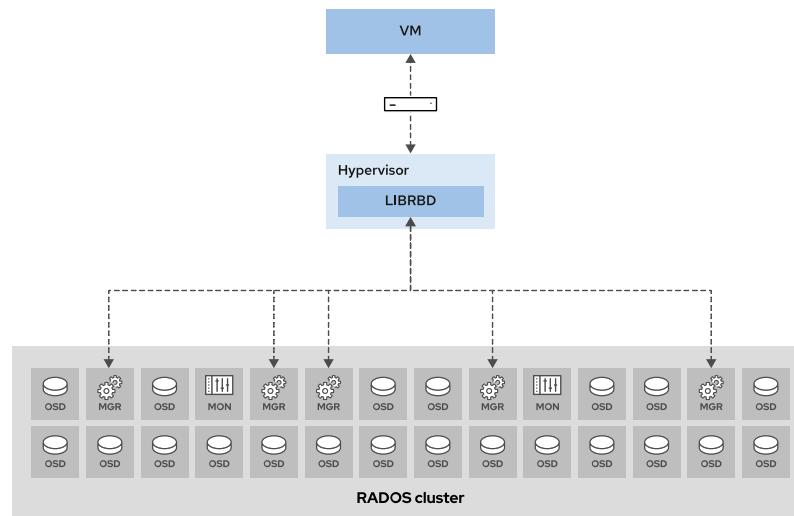


Figure 6.2: Virtual environment access

Because the user space implementation of the Ceph block device (for example, `librbd`) cannot take advantage of the Linux page cache, it performs its own in-memory caching, known as *RBD caching*. RBD caching behaves in a similar manner to the Linux page cache. When the OS implements a barrier mechanism or a flush request, Ceph writes all dirty data to the OSDs. This means that using write-back caching is just as safe as using physical hard disk caching with a VM that properly sends flushes (for example, Linux kernel $\geq 2.6.32$). The cache uses a Least Recently Used (LRU) algorithm, and in write-back mode it can coalesce contiguous requests for better throughput.



Note

The RBD cache is local to the client because it uses RAM on the machine that initiated the I/O requests. For example, if you have Nova compute nodes in your Red Hat OpenStack Platform installation that use `librbd` for their virtual machines, the OpenStack client initiating the I/O request will use local RAM for its RBD cache.

RBD Caching Configurations

Caching Not Enabled

Reads and writes go to the Ceph Object Store. The Ceph cluster acknowledges the writes when the data is written and flushed on all relevant OSD journals.

Cache Enabled (write-back)

Considering two values, unflushed cache bytes U and maximum dirty cache bytes M , writes are acknowledged when $U < M$, or after writing data back to disk until $U < M$.

Write-through Caching

Set the maximum dirty byte to 0 to force write-through mode. The Ceph cluster acknowledges the writes when the data is written and flushed on all relevant OSD journals.

If using write-back mode, then the `librbd` library caches and acknowledges the I/O requests when it writes the data into the local cache of the server. Consider write-through for strategic production servers to reduce the risk of data loss or file system corruption in case of a server failure. Red Hat Ceph Storage offers the following set of RBD caching parameters:

RBD Caching Parameters

Parameter	Description	Default
rbd_cache	Enable RBD caching. Value=true false.	true
rbd_cache_size	Cache size in bytes per RBD image. Value=n.	32 MB
rbd_cache_max_dirty	Maximum dirty bytes allowed per RBD image. Value=n.	24 MB
rbd_cache_target_dirty	Dirty bytes to start preemptive flush per RBD image. Value=n.	16 MB
rbd_cache_max_dirty_age	Maximum page age in seconds before flush. Value=n.	1
rbd_cache_writethrough_until_flush	Start in write-through mode until performing the first flush. Value=true false.	true

Run `ceph config set client parameter value` command or `ceph config set global parameter value` command for client or global, respectively.



Note

When using librbd with Red Hat OpenStack Platform, create separate Cephx user names for OpenStack Cinder, Nova, and Glance. By following this recommended practice, you can create different caching strategies based on the type of RBD images that your Red Hat OpenStack Platform environment accesses.

Tuning the RBD Image Format

RBD images are striped over objects and stored in a RADOS object store. Red Hat Ceph Storage provides parameters that define how these images are striped.

RADOS Block Device Image Layout

All objects in an RBD image have a name that starts with the value contained in the RBD Block Name Prefix field of each RBD image and displayed using the `rbd info` command. After this prefix, there is a period (.), followed by the object number. The value for the object number field is a 12-character hexadecimal number.

```
[root@node ~]# rbd info rbdimage
rbd image 'rbdimage':
  size 10240{nbsp}MB in 2560 objects
  order 22 (4 MiB objects)
  snapshot_count: 0
  id: 867cba5c2d68
  block_name_prefix: rbd_data.867cba5c2d68
  format: 2
  features: layering, exclusive-lock, object-map, fast-diff, deep-flatten
```

```

op_features:
flags:
create_timestamp: Thu Sep 23 18:54:35 2021
access_timestamp: Thu Sep 23 18:54:35 2021
modify_timestamp: Thu Sep 23 18:54:35 2021
[root@node ~]# rados -p rbd ls
rbd_object_map.d3d0d7d0b79e.00000000000000000008
rbd_id.rbdimage
rbd_object_map.d42c1e0a1883
rbd_directory
rbd_children
rbd_info
rbd_header.d3d0d7d0b79e
rbd_header.d42c1e0a1883
rbd_object_map.d3d0d7d0b79e
rbd_trash

```

Ceph block devices allow storing data striped over multiple Object Storage Devices (OSD) in a Red Hat Ceph Storage cluster.

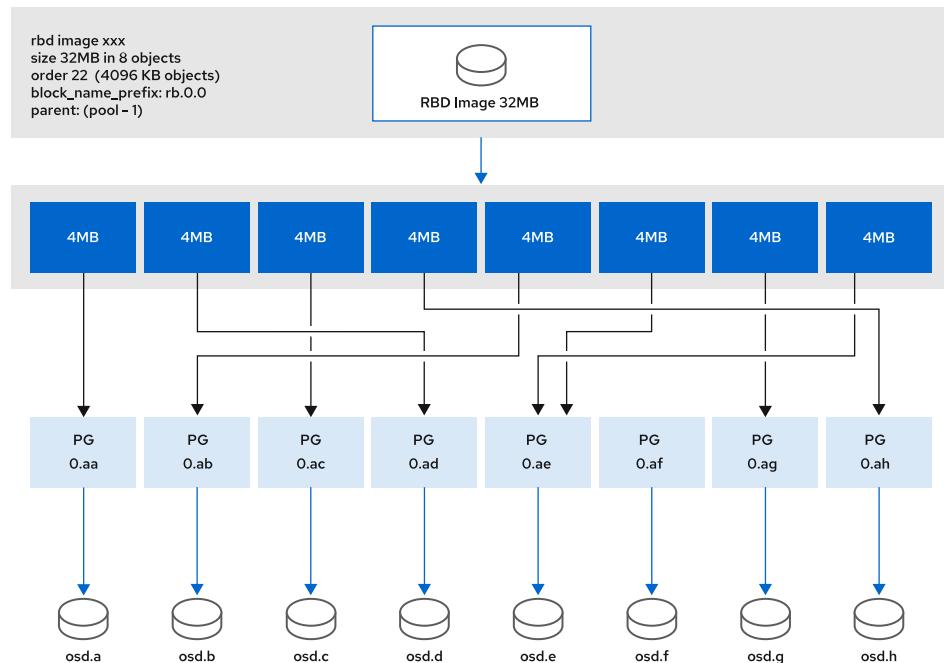


Figure 6.3: RBD layout

RBD Image Order

The *image order* is the size of the objects used for the RBD image. Image order defines a binary shift value based on the `<<` (bitwise left shift) C operator. This operator shifts the left operand bits by the right operand value. For example, `1 << 2 = 4`. Decimal 1 is `0001` in binary, so the result of the `1 << 2 = 4` operation is `0100` in binary, which is 4 in decimal. The value of the image order must be between 12 and 25, where 12 = 4 KiB and 13 = 8 KiB, for example. The default image order is 22, resulting in 4 MiB objects. You can override the default value by using the `--order` option of the `rbd create` command.

You can specify the size of the objects used with the `--object-size` option. This parameter must specify an object size between 4096 bytes (4 KiB) and 33,554,432 bytes (32 MiB), expressed in bytes, K or M (for example, 4096, 8 K or 4 M).

RBD Image Format

Each RBD image has three parameters associated with it:

`image_format`

The RBD image format version. The default value is 2, the most recent version. Version 1 has been deprecated and does not support features such as cloning and mirroring.

`stripe_unit`

The number of consecutive bytes stored in one object, `object_size` by default.

`stripe_count`

The number of RBD image objects that a stripe spans, 1 by default.

For RBD format 2 images, you can change the value of each of those parameters. The settings must align with the following equation:

```
stripe_unit * stripe_count = object_size
```

For example:

```
stripe_unit = 1048576, stripe_count = 4 for default 4 MiB objects
```

Remember that `object_size` must be no less than 4096 bytes and no greater than 33,554,432 bytes. Use the `--object-size` option to specify this value when you create the RBD image. The default `object_size` is 4192304 bytes (4 MiB).



References

`rbd(8)` and `rbdmap(8)` man pages

For more information, refer to the *Red Hat Ceph Storage 5 Block Device Guide* at https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/block_device_guide/index

► Guided Exercise

Managing RADOS Block Devices

In this exercise, you will create and configure RADOS block devices.

Outcomes

You should be able to create and manage RADOS block device images and use them as regular block devices.

Before You Begin

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

This command confirms that the hosts required for this exercise are accessible.

```
[student@workstation ~]$ lab start block-devices
```

Instructions

- 1. Verify that the Red Hat Ceph cluster is in a healthy state.

- 1.1. Log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@clienta  
[admin@clienta ~]$ sudo cephadm shell  
[ceph: root@clienta /]#
```

- 1.2. Verify that the cluster status is `HEALTH_OK`.

```
[ceph: root@clienta /]# ceph health  
HEALTH_OK
```

- 2. Create a replicated pool called `test_pool` with 32 placement groups. Set the application type for the pool to `rbd`.

- 2.1. Create the pool `test_pool`.

```
[ceph: root@clienta /]# ceph osd pool create test_pool 32 32  
pool 'test_pool' created
```

- 2.2. Set `rbd` as the application type for the pool.

```
[ceph: root@clienta /]# rbd pool init test_pool
```

- 2.3. List the configured pools and view the usage and availability for `test_pool`. The ID of `test_pool` might be different in your lab environment.

```
[ceph: root@clienta /]# ceph df
--- RAW STORAGE ---
CLASS      SIZE     AVAIL      USED   RAW USED %RAW USED
hdd       90 GiB    90 GiB   136 MiB  136 MiB      0.15
TOTAL     90 GiB    90 GiB   136 MiB  136 MiB      0.15

--- POOLS ---
POOL          ID   PGS   STORED  OBJECTS      USED %USED MAX AVAIL
...output omitted...
test_pool        6    32      9 B       1     8 KiB      0    28 GiB
```

- ▶ 3. Create a dedicated Ceph user called `client.test_pool.clientb`. Configure the `clienta` node with the user's key-ring file to access the RBD pool containing the RBD images.

- 3.1. Create the `client.test_pool.clientb` user and display the new file.

```
[ceph: root@clienta /]# ceph auth get-or-create client.test_pool.clientb \
mon 'profile rbd' osd 'profile rbd' \
-o /etc/ceph/ceph.client.test_pool.clientb.keyring
```

```
[ceph: root@clienta /]# cat /etc/ceph/ceph.client.test_pool.clientb.keyring
[client.test_pool.clientb]
key = AQAxBE1h79iUNhAAnBWswmX1Wk1Dhlq4a3U61Q==
```

- 3.2. Verify that the `client.test_pool.clientb` user now exists.

```
[ceph: root@clienta /]# ceph auth get client.test_pool.clientb
[client.test_pool.clientb]
key = AQAxBE1h79iUNhAAnBWswmX1Wk1Dhlq4a3U61Q==
caps mon = "profile rbd"
caps osd = "profile rbd"
exported keyring for client.test_pool.clientb
```

- ▶ 4. Open a second terminal window and log in to the `clientb` node as the `admin` user. Copy the key-ring file for the new `test_pool` user. Use the `client.test_pool.clientb` user name when connecting to the cluster.

- 4.1. Log in to `clientb` as the `admin` user and switch to the `root` user.

```
[student@workstation ~]$ ssh admin@clientb
[admin@clientb ~]$ sudo -i
[root@clientb ~]#
```

- 4.2. Install the `ceph-common` package on the `clientb` node.

```
[root@clientb ~]# yum install -y ceph-common
...output omitted...
Complete!
```

- 4.3. Go to the first terminal and copy the Ceph configuration and the key-ring files from the /etc/ceph/ directory on the **clienta** node to the /etc/ceph/ directory on the **clientb** node.

```
[ceph: root@clienta /]# scp \
/etc/ceph/{ceph.conf,ceph.client.test_pool.clientb.keyring} \
root@clientb:/etc/ceph
root@clientb's password: redhat
...output omitted...
```

- 4.4. Go to the second terminal window. Temporarily set the default user ID used for connections to the cluster to **test_pool.clientb**.

```
[root@clientb ~]# export CEPH_ARGS='--id=test_pool.clientb'
```

- 4.5. Verify that you can connect to the cluster.

```
[root@clientb ~]# rbd ls test_pool
```

- ▶ 5. Create a new RADOS Block Device Image and map it to the **clientb** machine.

- 5.1. Create an RBD image called **test** in the **test_pool** pool. Specify a size of 128 megabytes.

```
[root@clientb ~]# rbd create test_pool/test --size=128M
```

```
[root@clientb ~]# rbd ls test_pool
test
```

- 5.2. Verify the parameters of the RBD image.

```
[root@clientb ~]# rbd info test_pool/test
rbd image 'test':
  size 128 MiB in 32 objects
  order 22 (4 MiB objects)
  snapshot_count: 0
  id: 867cba5c2d68
  block_name_prefix: rbd_data.867cba5c2d68
  format: 2
  features: layering, exclusive-lock, object-map, fast-diff, deep-flatten
  op_features:
  flags:
  create_timestamp: Thu Sep 23 18:54:35 2021
  access_timestamp: Thu Sep 23 18:54:35 2021
  modify_timestamp: Thu Sep 23 18:54:35 2021
```

- 5.3. Map the RBD image on the **clientb** node by using the kernel RBD client.

```
[root@clientb ~]# rbd map test_pool/test
/dev/rbd0
```

```
[root@clientb ~]# rbd showmapped
id  pool      namespace  image  snap  device
0   test_pool           test    -     /dev/rbd0
```

- 6. Verify that you can use the RBD image mapped on the `clientb` node like a regular disk block device.

6.1. Format the device with an XFS file system.

```
[root@clientb ~]# mkfs.xfs /dev/rbd0
meta-data=/dev/rbd0          isize=512    agcount=8, agsize=4096 blks
                           =         sectsz=512  attr=2, projid32bit=1
                           =         crc=1      finobt=1, sparse=1, rmapbt=0
                           =         reflink=1
data        =         bsize=4096   blocks=32768, imaxpct=25
                           =         sunit=16    swidth=16 blks
naming      =version 2       bsize=4096   ascii-ci=0, ftype=1
log         =internal log    bsize=4096   blocks=1872, version=2
                           =         sectsz=512  sunit=16 blks, lazy-count=1
realtime   =none            extsz=4096   blocks=0, rtextents=0
Discarding blocks...Done.
```

6.2. Create a mount point for the file system.

```
[root@clientb ~]# mkdir /mnt/rbd
```

6.3. Mount the file system created on the `/dev/rbd0` device.

```
[root@clientb ~]# mount /dev/rbd0 /mnt/rbd
```

6.4. Change the ownership of the mount point.

```
[root@clientb ~]# chown admin:admin /mnt/rbd
```

6.5. Review the file-system usage.

```
[root@clientb ~]# df /mnt/rbd
Filesystem      1K-blocks  Used Available Use% Mounted on
/dev/rbd0        123584   7940   115644   7% /mnt/rbd
```

6.6. Add some content to the file system.

```
[root@clientb ~]# dd if=/dev/zero of=/mnt/rbd/test1 bs=10M count=1
1+0 records in
1+0 records out
10485760 bytes (10 MB, 10 MiB) copied, 0.00838799 s, 1.3 GB/s
```

```
[root@clientb ~]# ls /mnt/rbd
test1
```

6.7. Review the file-system usage.

```
[root@clientb ~]# df /mnt/rbd
Filesystem      1K-blocks  Used Available Use% Mounted on
/dev/rbd0        123584  18180    105404  15% /mnt/rbd
```

6.8. Review the content of the cluster.

```
[root@clientb ~]# ceph df
--- RAW STORAGE ---
CLASS      SIZE     AVAIL     USED   RAW USED %RAW USED
hdd       90 GiB   90 GiB   158 MiB  158 MiB  0.17
TOTAL     90 GiB   90 GiB   158 MiB  158 MiB  0.17

--- POOLS ---
POOL          ID  PGS  STORED OBJECTS     USED %USED MAX AVAIL
...output omitted...
test_pool      6   32   2.5 MiB    14   7.5 MiB    0   28 GiB
```

6.9. Unmount the file system and unmap the RBD image on the clientb node.

```
[root@clientb ~]# umount /mnt/rbd
[root@clientb ~]# rbd unmap /dev/rbd0
[root@clientb ~]# rbd showmapped
```

▶ 7. Configure the client system so that it persistently mounts the test_pool/test RBD image as /mnt/rbd.

7.1. Create a single-line entry for test_pool/test in the /etc/ceph/rbdmap RBD map file. The RBD mount daemon should authenticate as the test_pool.clientb user using the appropriate key-ring file. The resulting file should have the following contents:

```
[root@clientb ~]# cat /etc/ceph/rbdmap
# RbdDevice          Parameters
#poolname/imagename  id=client,keyring=/etc/ceph/ceph.client.keyring
test_pool/test
  id=test_pool.clientb,keyring=/etc/ceph/ceph.client.test_pool.clientb.keyring
```

7.2. Create an entry for /dev/rbd/test_pool/test in the /etc/fstab file. The resulting file should have the following contents:

```
[root@clientb ~]# cat /etc/fstab
UUID=fe1e8b67-e41b-44b8-bcfe-e0ec966784ac /           xfs      defaults
          0 0
UUID=F537-0F4F      /boot/efi           vfat
          defaults,uid=0,gid=0,umask=077,shortname=winnt 0 2
/dev/rbd/test_pool/test /mnt/rbd           xfs noauto 0 0
```

7.3. Verify your RBD map configuration. Use the rbdmap command to map and unmap configured RBD devices.

```
[root@clientb ~]# rbdmap map
[root@clientb ~]# rbd showmapped
id pool      namespace image snap device
0  test_pool        test   -    /dev/rbd0
[root@clientb ~]# rbdmap unmap
[root@clientb ~]# rbd showmapped
```

- 7.4. After you have verified that the RBD mapped devices work, enable the `rbdmap` service. Reboot the `clientb` node to verify that the RBD device mounts persistently.

```
[root@clientb ~]# systemctl enable rbdmap
Created symlink /etc/systemd/system/multi-user.target.wants/rbdmap.service → /usr/
lib/systemd/system/rbdmap.service.
```

```
[root@clientb ~]# reboot
Connection to clientb closed by remote host.
Connection to clientb closed.
```

When the `clientb` node finishes rebooting, log in and verify that it has mounted the RBD device.

```
[student@workstation ~]$ ssh admin@clientb
[admin@clientb ~]$ df /mnt/rbd
Filesystem      1K-blocks  Used Available Use% Mounted on
/dev/rbd0        123584  18180    105404  15% /mnt/rbd
```

- 8. Unmount your file system, unmap and delete the `test_pool/test` RBD image, and delete the temporary objects to clean up your environment.

- 8.1. Unmount the `/mnt/rbd` file system and unmap the RBD image.

```
[admin@clientb ~]$ sudo -i
[root@clientb ~]# rbdmap unmap
[root@clientb ~]# df | grep rbd
[root@clientb ~]# rbd showmapped
```

- 8.2. Remove the RBD entry from the `/etc/fstab` file. The resulting file should contain the following:

```
[root@clientb ~]# cat /etc/fstab
UUID=fe1e8b67-e41b-44b8-bcfe-e0ec966784ac /          xfs      defaults
0 0
UUID=F537-0F4F      /boot/efi           vfat
defaults,uid=0,gid=0,umask=077,shortname=winnt 0 2
```

- 8.3. Remove the RBD map entry for `test_pool/test` from the `/etc/ceph/rbdmap` file. The resulting file should contain the following:

```
[root@clientb ~]# cat /etc/ceph/rbdmap
# RbdDevice          Parameters
#poolname/imagename   id=client,keyring=/etc/ceph/ceph.client.keyring
```

8.4. Delete the test RBD image.

```
[root@clientb ~]# rbd rm test_pool/test --id test_pool.clientb
Removing image: 100% complete...done.
```

8.5. Verify that the `test_pool` RBD pool does not yet contain extra data. The `test_pool` pool should initially contain the three list objects.

```
[root@clientb ~]# rados -p test_pool ls --id test_pool.clientb
rbd_directory
rbd_info
rbd_trash
```

▶ 9. Exit and close the second terminal. In the first terminal, return to `workstation` as the `student` user.

```
[root@clientb ~]# exit
[admin@clientb ~]$ exit
[student@workstation ~]$ exit
```

```
[ceph: root@clienta /]# exit
exit
[root@clienta ~]# exit
[admin@clienta ~]$ exit
[student@workstation ~]$
```

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 block-devices
```

This concludes the guided exercise.

Managing RADOS Block Device Snapshots

Objectives

After completing this section, you should be able to create and configure RADOS block devices snapshots and clones.

Enabling RBD Snapshots and Cloning

Images that use RBD format 2 support several optional features. Use the `rbd feature enable` or `rbd feature disable` commands to enable or disable RBD image features. This example enables the `layering` feature on the `test` image in the `rbd` pool.

```
[root@node ~]# rbd feature enable rbd/test layering
```

To disable the `layering` feature, use the `rbd feature disable` command:

```
[root@node ~]# rbd feature disable rbd/test layering
```

These are some of the available features for an RBD image:

RBD Image Features

Name	Description
<code>layering</code>	Layering support to enable cloning.
<code>striping</code>	Striping v2 support for enhanced performance, supported by <code>librbd</code> .
<code>exclusive-lock</code>	Exclusive locking support.
<code>object-map</code>	Object map support (requires <code>exclusive-lock</code>).
<code>fast-diff</code>	Fast diff command support (requires <code>object-map</code> AND <code>exclusive-lock</code>).
<code>deep-flatten</code>	Flattens all snapshots of the RBD image.
<code>journaling</code>	Journaling support.
<code>data-pool</code>	EC data pool support.

RBD Snapshots

RBD snapshots are read-only copies of an RBD image created at a particular time. RBD snapshots use a COW technique to reduce the amount of storage needed to maintain snapshots. Before applying a write I/O request to an RBD snapshot image, the cluster copies the original data to another area in the placement group of the object affected by the I/O operation. Snapshots do

not consume any storage space when created, but grow in size as the objects that they contain change. RBD images support incremental snapshots.



Important

Use the `fsfreeze` command to suspend access to a file system before taking a snapshot. The `fsfreeze -freeze` command stops access to the file system and creates a stable image on disk. Do not take a file system snapshot when the file system is not frozen because it will corrupt the snapshot's file system. After taking the snapshot, use `fsfreeze --unfreeze` command to resume file system operations and access.

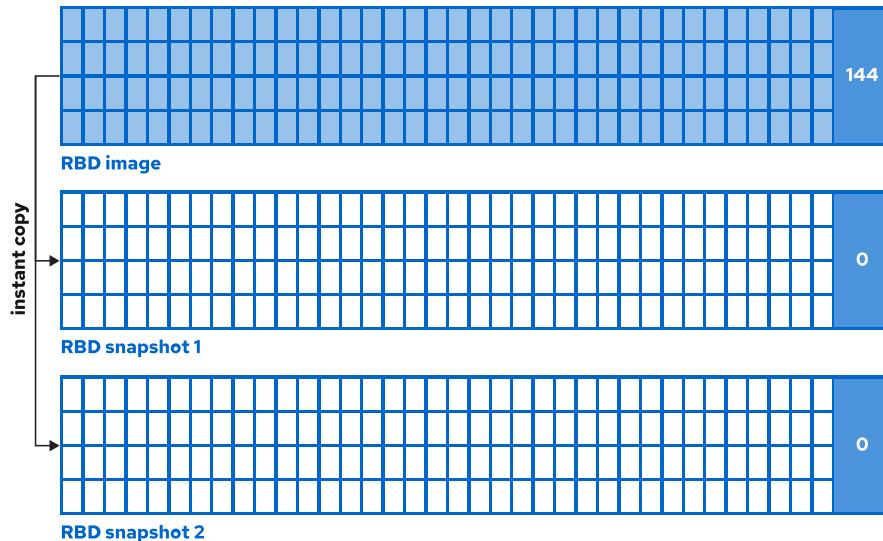


Figure 6.4: RBD snapshots creation

The snapshot COW procedure operates at the *object* level, regardless of the size of the write I/O request made to the RBD image. If you write a single byte to an RBD image that has a snapshot, then Ceph copies the entire affected object from the RBD image into the snapshot area.



Figure 6.5: Writing to an RBD snapshot

Deleting an RBD image fails if snapshots exist for the image. Use the `rbd snap purge` command to delete the snapshots.

Use the `rbd snap create` command to create a snapshot of a Ceph block device.

```
[root@node ~]# rbd snap create pool/image@firstsnap
Creating snap: 100% complete...done.
```

Use the `rbd snap ls` command to list the block device snapshots.

```
[root@node ~]# rbd snap ls pool/image
SNAPID  NAME      SIZE    PROTECTED   TIMESTAMP
        firstsnap  128 MiB          Thu Oct 14 16:55:01 2021
        secondsnap 128 MiB          Thu Oct 14 17:48:59 2021
```

Use the `rbd snap rollback` command to roll back a block device snapshot, overwriting the current version of the image with data from the snapshot.

```
[root@node ~]# rbd snap rollback pool/image@firstsnap
SNAPID  NAME      SIZE    PROTECTED   TIMESTAMP
        firstsnap  128 MiB          Thu Oct 14 16:55:01 2021
        secondsnap 128 MiB          Thu Oct 14 17:48:59 2021
```

Use the `rbd snap rm` command to delete a snapshot for Ceph block devices.

```
[root@node ~]# rbd snap rm pool/image@secondsnap
Removing snap: 100% complete...done.
```

RBD Clones

RBD clones are read/write copies of an RBD image that use a protected RBD snapshot as a base. An RBD clone can also be *flattened*, which converts it into an RBD image independent of its source. The cloning process has three steps:

1. Create a snapshot:

```
[root@node ~]# rbd snap create pool/image@snapshot
Creating snap: 100% complete...done.
```

2. Protect the snapshot from deletion:

```
[root@node ~]# rbd snap protect pool/image@snapshot
```

3. Create the clone using the protected snapshot:

```
[root@node ~]# rbd clone pool/imagename@snapshotname pool/clonename
```

The newly created clone behaves just like a regular RBD image. Clones support COW and COR, with COW as the default. COW copies the parent snapshot data into the clone before applying a write I/O request to the clone.

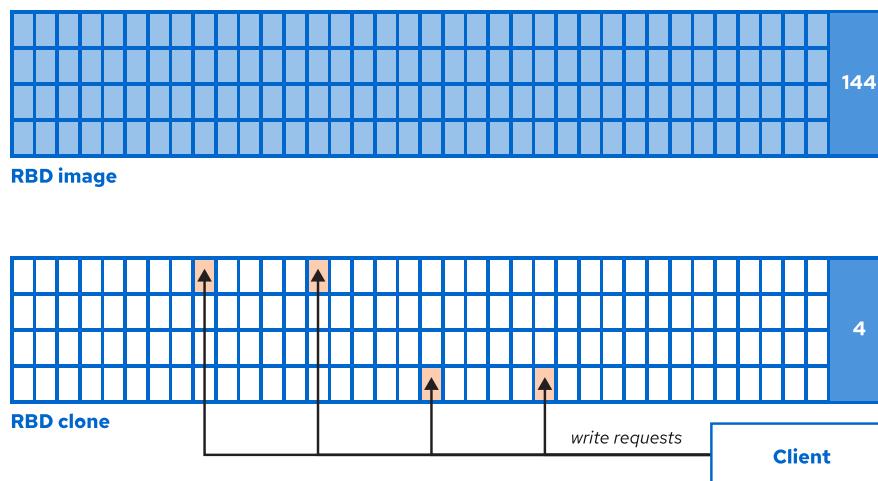


Figure 6.6: RBD clone write operation

You can also enable COR support for RBD clones. Data that is the same for the parent RBD snapshot and the clone is read directly from the parent. This can make reads more expensive if the parent's OSDs have high latency relative to the client. COR copies objects to the clone when they are first read.

If you enable COR, Ceph copies the data from the parent snapshot into the clone before processing a read I/O request, if the data is not already present in the clone. Activate the COR feature by running the `ceph config set client rbd_clone_copy_on_read true` command or the `ceph config set global rbd_clone_copy_on_read true` command for the client or global setting. The original data is not overwritten.

**Important**

If COR is disabled on an RBD clone, every read operation that the clone cannot satisfy results in an I/O request to the parent of the clone.

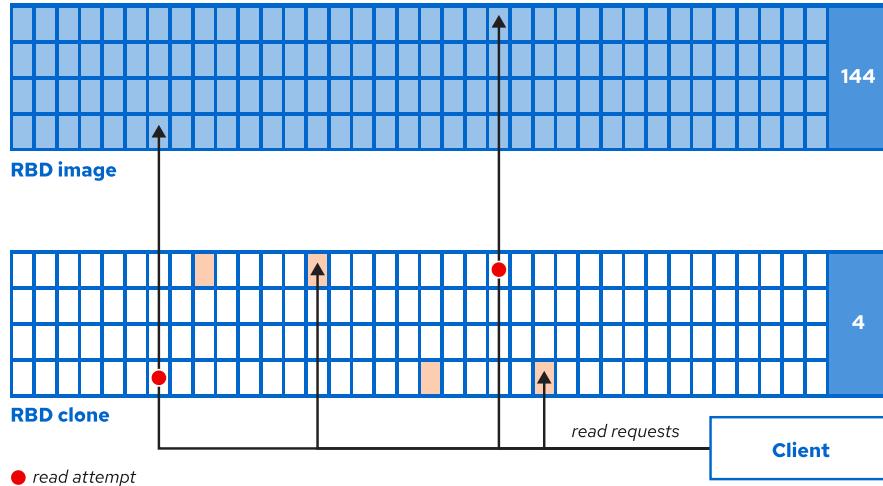


Figure 6.7: RBD clone read operation

The clone COW and COR procedures operate at the object level, regardless of the I/O request size. To read or write a single byte of the RBD clone, Ceph copies the entire object from the parent image or snapshot into the clone.

Use the `rbd` command to manage RBD clones.

RBD commands for clone management	
<code>rbd children [pool-name/]image-name@snapshot-name</code>	List clones
<code>rbd clone [pool-name/]parent-image@snap-name [pool-name/]child-image-name</code>	Create clone
<code>rbd flatten [pool-name/]child-image-name</code>	Flatten clone

When flattening a clone, Ceph copies all missing data from the parent into the clone and then removes the reference to the parent. The clone becomes an independent RBD image and is no longer the child of a protected snapshot.

**Note**

You cannot delete RBD images directly from a pool. Instead, use the `rbd trash mv` command to move an image from a pool to the trash. Delete objects from the trash with the `rbd trash rm` command. You are allowed to move active images that are in use by clones to the trash for later deletion.



References

For more information, refer to the *Snapshot management* chapter in the *Red Hat Ceph Storage 5 Block Device Guide* at
https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/block_device_guide/index#snapshot-management

► Guided Exercise

Managing RADOS Block Device Snapshots

In this exercise, you will create and clone a RADOS block device snapshot.

Outcomes

You should be able to create and manage RADOS block device snapshots, as well as clone and create a child image.

Before You Begin

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

```
[student@workstation ~]$ lab start block-snapshot
```

This command confirms that the hosts required for this exercise are accessible. It also creates an image called `image1` within the `rbd` pool. Finally, this command creates a user and associated key in Red Hat Ceph Storage cluster and copies it to the `clientb` node.

Instructions

- 1. Use the `ceph health` command to verify that the primary cluster is in a healthy state.

- 1.1. Log in to `clienta` as the `admin` user and switch to the `root` user.

```
[student@workstation ~]$ ssh admin@clienta
...output omitted...
[admin@clienta ~]$ sudo -i
[root@clienta ~]#
```

- 1.2. Use the `cephadm shell` to run the `ceph health` command to verify that the primary cluster is in a healthy state.

```
[root@clienta ~]# cephadm shell -- ceph health
HEALTH_OK
```

- 2. Map the `rbd/image1` image as a block device, format it with an XFS file system, and confirm that the `/dev/rbd0` device is writable.

- 2.1. Map the `rbd/image1` image as a block device.

```
[root@clienta ~]# rbd map --pool rbd image1
/dev/rbd0
```

- 2.2. Format the block device with an XFS file system.

```
[root@clienta ~]# mkfs.xfs /dev/rbd0
...output omitted...
```

- 2.3. Confirm that the `/dev/rbd0` device is writable.

```
[root@clienta ~]# blockdev --getro /dev/rbd0
0
```

- 3. Create an initial snapshot called `firshtsnap`. Calculate the provisioned and actual disk usage of the `rbd/image1` image and its associated snapshots by using the `rbd disk-usage` command.

- 3.1. Run the `cephadm` shell. Create an initial snapshot called `firshtsnap`.

```
[root@clienta ~]# cephadm shell
...output omitted...
[ceph: root@clienta /]# rbd snap create rbd/image1@firshtsnap
Creating snap: 100% complete...done.
```

- 3.2. Calculate the provisioned and used size of the `rbd/image1` image and its associated snapshots.

```
[ceph: root@clienta /]# rbd disk-usage --pool rbd image1
NAME          PROVISIONED   USED
image1@firshtsnap    128 MiB  36 MiB
image1           128 MiB  36 MiB
<TOTAL>          128 MiB  72 MiB
```

- 4. Open another terminal window. Log in to `clientb` as the `admin` user and switch to the `root` user. Set the `CEPH_ARGS` variable to the '`--id=rbd.clientb`' value.

```
[student@workstation ~]$ ssh admin@clientb
...output omitted...
[admin@clientb ~]$ sudo -i
[root@clientb ~]# export CEPH_ARGS='--id=rbd.clientb'
```

- 5. On the `clientb` node, map the `image1@firshtsnap` snapshot and verify that the device is writable.

- 5.1. Map the `rbd/image1` image as a block device.

```
[root@clientb ~]# rbd map --pool rbd image1@firshtsnap
/dev/rbd0
[root@clientb ~]# rbd showmapped
id  pool  namespace  image  snap      device
0   rbd     image1     firstsnap  /dev/rbd0
```

- 5.2. Confirm that `/dev/rbd0` is a read-only block device.

```
[root@clientb ~]# blockdev --getro /dev/rbd0
1
```

- 6. On the `clienta` node, exit the `cephadm` shell. Mount the `/dev/rbd0` device in `/mnt/image` directory, copy some data into it, and then unmount it.

6.1. Mount the block device in `/mnt/image` directory.

```
[ceph: root@clienta ~]# exit
[root@clienta ~]# mount /dev/rbd0 /mnt/image
[root@clienta ~]# mount | grep rbd
/dev/rbd0 on /mnt/image type xfs
  (rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=64k,
  sunit=128,swidth=128,noquota)
```

6.2. Copy some data into `/mnt/image` directory.

```
[root@clienta ~]# cp /etc/ceph/ceph.conf /mnt/image/file0
[root@clienta ~]# ls /mnt/image/
file0
```

6.3. Check the disk space usage for the `/dev/rbd0` device.

```
[root@clienta ~]# df /mnt/image/
Filesystem      1K-blocks  Used Available Use% Mounted on
/dev/rbd0        123584   7944    115640   7% /mnt/image
```

- 7. On the `clientb` node, mount the `image1@firstsnap` snapshot in `/mnt/snapshot` directory. Review the disk space usage for the `/dev/rbd0` device and list the directory contents. Unmount the `/mnt/snapshot` directory, and then unmap the `/dev/rbd0` device.

7.1. Mount the block device in `/mnt/snapshot` directory.

```
[root@clientb ~]# mount /dev/rbd0 /mnt/snapshot/
mount: /mnt/snapshot: WARNING: device write-protected, mounted read-only.
```

7.2. Check the disk space usage for the `/dev/rbd0` device and list the directory content.

```
[root@clientb ~]# df /mnt/snapshot/
Filesystem      1K-blocks  Used Available Use% Mounted on
/dev/rbd0        123584   480    123104   1% /mnt/snapshot
[root@clientb ~]# ls -l /mnt/snapshot/
total 0
```

Notice that the `file0` file does not display on the `clientb` node because the file system of the snapshot block device is empty.

Changes to the original block device did not alter the snapshot.

7.3. Unmount the `/mnt/snapshot` directory, and then unmap the `/dev/rbd0` device.

```
[root@clientb ~]# umount /mnt/snapshot  
[root@clientb ~]# rbd unmap --pool rbd image1@firstsnap  
[root@clientb ~]# rbd showmapped
```

- 8. On the `clienta` node, protect the `firstsnap` snapshot and create a clone called `clone1` in the `rbd` pool. Verify that the child image is created.

8.1. Run the `cephadm` shell. Protect the `firstsnap` snapshot.

```
[root@clienta ~]# cephadm shell  
...output omitted...  
[ceph: root@clienta /]# rbd snap protect rbd/image1@firstsnap
```

8.2. Clone the `firstsnap` block device snapshot to create a read or write child image called `clone1` that uses the `rbd` pool.

```
[ceph: root@clienta /]# rbd clone rbd/image1@firstsnap rbd/clone1
```

8.3. List the children of the `firstsnap` snapshot.

```
[ceph: root@clienta /]# rbd children rbd/image1@firstsnap  
rbd/clone1
```

- 9. On the `clientb` node, map the `rbd/clone1` image as a block device, mount it, and then copy some content to the clone.

9.1. Map the `rbd/clone1` image as a block device.

```
[root@clientb ~]# rbd map --pool rbd clone1  
/dev/rbd0
```

9.2. Mount the block device in `/mnt/clone` directory, and then list the directory contents.

```
[root@clientb ~]# mount /dev/rbd0 /mnt/clone  
[root@clientb ~]# ls -l /mnt/clone  
total 0
```

9.3. Add some content to the `/mnt/clone` directory.

```
[root@clientb ~]# dd if=/dev/zero of=/mnt/clone/file1 bs=1M count=10  
...output omitted...  
[root@clientb ~]# ls -l /mnt/clone/  
total 10240  
-rw-r--r--. 1 root root 10485760 Oct 15 00:04 file1
```

- 10. Clean up your environment.

10.1. On the `clientb` node, unmount the file system and unmap the RBD image.

```
[root@clientb ~]# umount /mnt/clone  
[root@clientb ~]# rbd unmap --pool rbd clone1  
[root@clientb ~]# rbd showmapped  
[root@clientb ~]# unset CEPH_ARGS
```

- 10.2. On the `clienta` node, exit the `cephadm` shell. Unmount the file system, and then unmap the RBD image.

```
[ceph: root@clienta ~]# exit  
[root@clienta ~]# umount /mnt/image  
[root@clienta ~]# rbd unmap --pool rbd image1  
[root@clienta ~]# rbd showmapped
```

- 11. Exit and close the second terminal. Return to `workstation` as the `student` user.

```
[root@clientb ~]# exit  
[admin@clientb ~]$ exit  
[student@workstation ~]$ exit
```

```
[root@clienta ~]# exit  
[admin@clienta ~]$ exit  
[student@workstation ~]$
```

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 block-snapshots
```

This concludes the guided exercise.

Importing and Exporting RBD Images

Objectives

After completing this section, you should be able to export an RBD image from the cluster to an external file and import it into another cluster.

Importing and Exporting RBD Images

The RBD export and import mechanism allows you to maintain operational copies of RBD images that are fully functional and accessible, either within the same cluster or by using a separate cluster. You can use these copies for a variety of use cases, including:

- Testing new versions using realistic volumes of data
- Running quality assurance processes using realistic volumes of data
- Implementing business continuity scenarios
- Decoupling backup processes from the production block devices

The RADOS block device feature provides the ability to export and import entire RBD images or only RBD image changes between two points in time.

Exporting RBD Images

Red Hat Ceph Storage provides the `rbd export` command to export an RBD image to a file. This command exports an RBD image or an RBD image snapshot to the specified destination file. The `rbd export` command has the following syntax:

```
rbd export [--export-format {1|2}] (image-spec | snap-spec) [dest-path]
```

The `--export-format` option specifies the format of the exported data, allowing you to convert earlier RBD format 1 images to newer, format 2 images. The following example exports an RBD image called `test` to the `/tmp/test.dat` file.

```
[ceph: root@node /]# rbd export rbd/test /tmp/test.dat
```

Importing RBD Images

Red Hat Ceph Storage provides the `rbd import` command to import an RBD image from a file. The command creates a new image and imports the data from the specified source path. The `rbd import` command has the following syntax:

```
rbd import [--export-format {1|2}] [--image-format format-id] [--object-size size-in-B/K/M] [--stripe-unit size-in-B/K/M --stripe-count num] [--image-feature feature-name]... [--image-shared] src-path [image-spec]
```

The `--export-format` option specifies the data format of the data to be imported. When importing format 2 exported data, use the `--stripe-unit`, `--stripe-count`, `--object-size`, and `--image-feature` options to create the new RBD format 2 image.

**Note**

The `--export-format` parameter value must match for the related `rbd export` and the `rbd import` commands.

Exporting and Importing Changes to RBD Images

Red Hat Ceph Storage provides the `rbd export-diff` and the `rbd import-diff` commands to export and import changes made between two points in time on an RBD image. The syntax is the same as for the `rbd export` and `rbd import` commands.

The endpoint in time can be:

- The current content of an RBD image, such as `poolname/imagename`.
- A snapshot of an RBD image, such as `poolname/imagename@snapname`.

The start time can be:

- The creation date and time of the RBD image. For example, without using the `--from-snap` option.
- A snapshot of an RBD image, such as is obtained by using the `--from-snap snapname` option.

If you specify a start point snapshot, the command exports the changes after you created that snapshot. If you do not specify a snapshot, the command exports all changes since the creation of the RBD image, which is the same as a regular RBD image export operation.

**Note**

The `import-diff` operation performs the following validity checks:

- If the `export-diff` is relative to a start snapshot, this snapshot must also exist in the target RBD image.
- If the `export-diff` is performed specifying an end snapshot, the same snapshot name is created in the target RBD image after the data is imported.

Piping the Export and Import Processes

Specifying the dash (-) character as the target file of an export operation causes the output to go to standard output (`stdout`).

You can also use the dash character (-) to specify either `stdout` or standard input (`stdin`) as the export target or import source.

You can pipe two commands into a single command.

```
[ceph: root@node /]# rbd export rbd/img1 - | rbd import - bup/img1
```

The `rbd merge-diff` command merges the output of two continuous incremental `rbd export-diff` image operations into one single target path. The command can only process two incremental paths at one time.

```
[ceph: root@node /]# rbd merge-diff first second merged
```

To merge more than two continuous incremental paths in a single command, pipe one `rbd export-diff` output to another `rbd export-diff` command. Use the dash character (-) as the target in the command before the pipe, and as the source in the command after the pipe.

For example, you can merge three incremental diffs into a single merged target on one command line. The snapshot end time of the earlier `export-diff` command must be equal to the snapshot start time of the later `export-diff` command.

```
[ceph: root@node /]# rbd merge-diff first second - | rbd merge-diff - third merged
```

The `rbd merge-diff` command only supports RBD images with `stripe-count` set to 1.



References

`rbd(8)` man pages.

► Guided Exercise

Importing and Exporting RBD Images

In this exercise, you will export an RBD image, then import the image into another cluster.

Outcomes

You should be able to:

- Export an entire RBD image.
- Import an entire RBD image.
- Export the changes applied to an RBD image between two points in time.
- Import the changes applied to an RBD image into another RBD image.

Before You Begin

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

```
[student@workstation ~]$ lab start block-import
```

This command confirms that the hosts required for this exercise are accessible. It also ensures that `clienta` has the necessary RBD client authentication keys.

Instructions

- 1. Open two terminals and log in to `clienta` and `serverf` as the `admin` user. Verify that both clusters are reachable and have a `HEALTH_OK` status.
- 1.1. Open a terminal window. Log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell. Verify that the primary cluster is in a healthy state.

```
[student@workstation ~]$ ssh admin@clienta
...output omitted...
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]# ceph health
HEALTH_OK
```

- 1.2. Open another terminal window. Log in to `serverf` as the `admin` user and use `sudo` to run the `cephadm` shell. Verify that the secondary cluster is in a healthy state.

```
[student@workstation ~]$ ssh admin@serverf
...output omitted...
[admin@serverf ~]$ sudo cephadm shell
[ceph: root@serverf /]# ceph health
HEALTH_OK
```

- 2. Create a pool called `rbd`, and then enable the `rbd` client application for the Ceph Block Device and make it usable by the RBD feature.

- 2.1. In the primary cluster, create a pool called rbd with 32 placement groups. Enable the rbd client application for the Ceph Block Device and make it usable by the RBD feature.

```
[ceph: root@clienta /]# ceph osd pool create rbd 32
pool 'rbd' created
[ceph: root@clienta /]# ceph osd pool application enable rbd rbd
enabled application 'rbd' on pool 'rbd'
[ceph: root@clienta /]# rbd pool init -p rbd
```

- 2.2. In the secondary cluster, create a pool called rbd with 32 placement groups. Enable the rbd client application for the Ceph Block Device and make it usable by the RBD feature.

```
[ceph: root@serverf /]# ceph osd pool create rbd 32
pool 'rbd' created
[ceph: root@serverf /]# ceph osd pool application enable rbd rbd
enabled application 'rbd' on pool 'rbd'
[ceph: root@serverf /]# rbd pool init -p rbd
```

- 3. Create an RBD image called rbd/test in your primary Ceph cluster. Map it as a block device, format it with an XFS file system, mount it in /mnt/rbd directory, copy some data into it, and then unmount it.

- 3.1. Create the RBD image. Exit the cephadm shell and switch to the root user. Map the image, and then format it with an XFS file system.

```
[ceph: root@clienta /]# rbd create test --size 128 --pool rbd
[ceph: root@clienta /]# exit
exit
[admin@clienta ~]$ sudo -i
[root@clienta ~]# rbd map --pool rbd test
/dev/rbd0
[root@clienta ~]# mkfs.xfs /dev/rbd0
...output omitted...
```

- 3.2. Mount /dev/rbd0 to the /mnt/rbd directory and copy a file to it.

```
[root@clienta ~]# mount /dev/rbd0 /mnt/rbd
[root@clienta ~]# mount | grep rbd
/dev/rbd0 on /mnt/rbd type xfs (rw,relatime,seclabel,attr2,inode64,...)
[root@clienta ~]# cp /etc/ceph/ceph.conf /mnt/rbd/file0
[root@clienta ~]# ls /mnt/rbd
file0
```

- 3.3. Unmount your file system to ensure that the system flushes all data to the Ceph cluster.

```
[root@clienta ~]# umount /mnt/rbd
```

- ▶ 4. Create a backup copy of the primary rbd/test block device. Export the entire rbd/test image to a file called /mnt/export.dat. Copy the export.dat file to the secondary cluster.

- 4.1. In the primary cluster, run the cephadm shell using the --mount argument to bind mount the /home/admin/rbd-export/ directory.

```
[root@clienta ~]# cephadm shell --mount /home/admin/rbd-export/
...output omitted...
[ceph: root@clienta /]#
```

- 4.2. Export the entire rbd/test image to a file called /mnt/export.dat. Exit the cephadm shell.

```
[ceph: root@clienta /]# rbd export rbd/test /mnt/export.dat
Exporting image: 100% complete...done.
[ceph: root@clienta /]# exit
exit
[root@clienta ~]$
```

- 4.3. Copy the export.dat file to the secondary cluster in the /home/admin/rbd-import/ directory.

```
[root@clienta ~]# rsync -avP /home/admin/rbd-export/export.dat \
serverf:/home/admin/rbd-import
...output omitted...
```

- ▶ 5. In the secondary cluster, import the /mnt/export.dat file containing the exported rbd/test RBD image into the secondary cluster. Confirm that the import was successful by mapping the imported image to a block device, mounting it, and inspecting its contents.

- 5.1. Exit the current cephadm shell. Use sudo to run the cephadm shell with the --mount argument to bind mount the /home/admin/rbd-import/ directory.

```
[ceph: root@serverf /]# exit
[admin@serverf ~]$ sudo cephadm shell --mount /home/admin/rbd-import/
[ceph: root@serverf /]#
```

- 5.2. List the contents of the backup cluster's empty rbd pool. Use the rbd import command to import the RBD image contained in the /mnt/export.dat file into the backup cluster, referring to it as rbd/test.

```
[ceph: root@serverf /]# rbd --pool rbd ls
[ceph: root@serverf /]# rbd import /mnt/export.dat rbd/test
Importing image: 100% complete...done.
[ceph: root@serverf /]# rbd du --pool rbd test
NAME  PROVISIONED  USED
test    128 MiB   32 MiB
```

- 5.3. Exit the cephadm shell, and then switch to the root user. Map the backup cluster's imported RBD image and mount the file system it contains. Confirm that its contents are the same as those originally created on the primary cluster's RBD image.

```
[ceph: root@serverf /]# exit
exit
[admin@serverf ~]$ sudo -i
[root@serverf ~]# rbd map --pool rbd test
/dev/rbd0
[root@serverf ~]# mount /dev/rbd0 /mnt/rbd
[root@serverf ~]# mount | grep rbd
/dev/rbd0 on /mnt/rbd type xfs (rw,relatime,seclabel,attr2,inode64,...)
[root@serverf ~]# df -h /mnt/rbd
Filesystem      Size  Used Avail Use% Mounted on
/dev/rbd0       121M   7.8M  113M   7% /mnt/rbd
[root@serverf ~]# ls -l /mnt/rbd
total 4
-rw-r--r-- 1 admin users 177 Sep 30 22:02 file0
[root@serverf ~]# cat /mnt/rbd/file0
# minimal ceph.conf for c315020c-21f0-11ec-b6d6-52540000fa0c
[global]
fsid = c315020c-21f0-11ec-b6d6-52540000fa0c
mon_host = [v2:172.25.250.12:3300/0,v1:172.25.250.12:6789/0]
```

5.4. Unmount the file system and unmap the RBD image.

```
[root@serverf ~]# umount /mnt/rbd
[root@serverf ~]# rbd unmap /dev/rbd0
```

- ▶ 6. In this part of the exercise, you will create a pair of snapshots of `rbd/test` on your primary cluster and export the changes between those snapshots as an incremental diff image. You will then import the changes from the incremental diff into your copy of the `rbd/test` image on your secondary cluster.
 - 6.1. In the primary cluster, run the `cephadm shell` and create an initial snapshot called `rbd/test@firstsnap`. Calculate the provisioned and actual disk usage of the `rbd/test` image and its associated snapshots.

```
[root@clienta ~]# cephadm shell
...output omitted...
[ceph: root@clienta /]# rbd snap create rbd/test@firstsnap
Creating snap: 100% complete...done.
[ceph: root@clienta /]# rbd du --pool rbd test
NAME      PROVISIONED  USED
test@firstsnap    128 MiB  36 MiB
test          128 MiB  36 MiB
<TOTAL>        128 MiB  72 MiB
[ceph: root@clienta /]# exit
exit
[root@clienta ~]#
```

- 6.2. In the secondary cluster, run the `cephadm shell`, create an initial snapshot called `rbd/test@firstsnap`. Calculate the provisioned and actual disk usage of the `rbd/test` image and its associated snapshots.

```
[root@serverf ~]$ cephadm shell
...output omitted...
[ceph: root@serverf /]# rbd snap create rbd/test@firstsnap
Creating snap: 100% complete...done.
[ceph: root@serverf /]# rbd du --pool rbd test
NAME      PROVISIONED  USED
test@firstsnap    128 MiB  32 MiB
test          128 MiB  32 MiB
<TOTAL>        128 MiB  64 MiB
[ceph: root@serverf /]# exit
exit
[root@clientf ~]#
```

- 6.3. In the primary cluster, mount the file system on the `/dev/rbd0` device, mapped from the `rbd/test` image, to change the RBD image. Make changes to the file system to effect changes to the RBD image. Unmount the file system when you are finished.

```
[root@clienta ~]# mount /dev/rbd0 /mnt/rbd
[root@clienta ~]# dd if=/dev/zero of=/mnt/rbd/file1 bs=1M count=5
...output omitted...
[root@clienta ~]# ls -l /mnt/rbd/
total 5124
-rw-r--r--. 1 admin users      177 Sep 30 22:02 file0
-rw-r--r--. 1 admin users 5242880 Sep 30 23:15 file1
[root@clienta ~]$ umount /mnt/rbd
```

- 6.4. In the primary cluster, run the `cephadm` shell and note that the amount of data used in the image of the primary cluster increased. Create a new snapshot called `rbd/test@secondsnap` to delimit the ending time window of the changes that you want to export. Note the adjustments made to the reported used data.

```
[root@clienta ~]# cephadm shell
...output omitted...
[ceph: root@clienta /]# rbd du --pool rbd test
NAME      PROVISIONED  USED
test@firstsnap    128 MiB  36 MiB
test          128 MiB  40 MiB
<TOTAL>        128 MiB  76 MiB
[ceph: root@clienta /]# rbd snap create rbd/test@secondsnap
Creating snap: 100% complete...done.
[ceph: root@clienta /]# rbd du --pool rbd test
NAME      PROVISIONED  USED
test@firstsnap    128 MiB  36 MiB
test@secondsnap  128 MiB  40 MiB
test          128 MiB  12 MiB
<TOTAL>        128 MiB  88 MiB
```

- 6.5. In the primary cluster, exit the current `cephadm` shell. Run the `cephadm` shell with the `--mount` argument to bind mount the `/home/admin/rbd-import/` directory.

```
[ceph: root@clienta /]# exit
exit
[root@clienta ~]# cephadm shell --mount /home/admin/rbd-export/
...output omitted...
[ceph: root@clienta /]#
```

- 6.6. Export the changes between the snapshots of the primary cluster's rbd/test image to a file called /mnt/export-diff.dat. Exit the cephadm shell, and copy the export-diff.dat file to the secondary cluster in the /home/admin/rbd-import/ directory.

```
[ceph: root@clienta /]# rbd export-diff --from-snap firstsnap \
rbd/test@secondsnap /mnt/export-diff.dat
Exporting image: 100% complete...done.
[ceph: root@clienta /]# exit
exit
[root@clienta ~]$ rsync -avP /home/admin/rbd-export/export-diff.dat \
serverf:/home/admin/rbd-import
...output omitted...
```

- 6.7. In the secondary cluster, run the cephadm shell using the --mount argument to mount the /home/admin/rbd-import/ directory. Use the rbd import-diff command to import the changes to the secondary cluster's copy of the rbd/test image by using the /mnt/export-diff.dat file. This eliminates the need to save the exported image to a file as an intermediate step. Inspect the information about the remote RBD image. Exit the cephadm shell.

```
[root@serverf ~]# cephadm shell --mount /home/admin/rbd-import
[ceph: root@serverf /]# rbd du --pool rbd test
NAME      PROVISIONED   USED
test@firstsnap    128 MiB  32 MiB
test          128 MiB  32 MiB
<TOTAL>        128 MiB  64 MiB
[ceph: root@serverf /]# rbd import-diff /mnt/rbd-import/export-diff.dat rbd/test
Importing image diff: 100% complete...done.
[ceph: root@serverf /]# rbd du --pool rbd test
NAME      PROVISIONED   USED
test@firstsnap    128 MiB  32 MiB
test@secondsnap    128 MiB  32 MiB
test          128 MiB  8 MiB
<TOTAL>        128 MiB  72 MiB
[ceph: root@serverf /]# exit
exit
[root@serverf ~]#
```



Note

The end snapshot is present on the secondary cluster's RBD image. The rbd import-diff command automatically creates it.

- 6.8. Verify that the backup cluster's image is identical to the primary cluster's RBD image.

```
[root@serverf ~]# rbd map --pool rbd test
/dev/rbd0
[root@serverf ~]# mount /dev/rbd0 /mnt/rbd
[root@serverf ~]# df /mnt/rbd
Filesystem      1K-blocks  Used Available Use% Mounted on
/dev/rbd0        123584 13064   110520  11% /mnt/rbd
[root@serverf ~]# ls -l /mnt/rbd
total 5124
-rw-r--r--. 1 admin users    177 Sep 30 22:02 file0
-rw-r--r--. 1 admin users 5242880 Sep 30 23:15 file1
[root@serverf ~]# umount /mnt/rbd
```

► 7. Clean up your environment.

- 7.1. In the primary cluster, unmap the RBD image. Run the `cephadm shell`, purge all existing snapshots on the RBD image of both clusters, and then delete the RBD image.

```
[root@clienta ~]# rbd unmap /dev/rbd0
```

```
[root@clienta ~]# cephadm shell
...output omitted...
[ceph: root@clienta /]# rbd --pool rbd snap purge test
Removing all snapshots: 100% complete...done.
[ceph: root@clienta /]# rbd rm test --pool rbd
Removing image: 100% complete...done.
[ceph: root@clienta /]# exit
exit
[root@clienta ~]#
```

- 7.2. In the secondary cluster, unmap the RBD image. Run the `cephadm shell`, purge all existing snapshots on the RBD image of both clusters, and then delete the RBD image.

```
[root@serverf ~]# rbd unmap /dev/rbd0
```

```
[root@serverf ~]# cephadm shell
...output omitted...
[ceph: root@serverf /]# rbd --pool rbd snap purge test
Removing all snapshots: 100% complete...done.
[ceph: root@serverf /]# rbd rm test --pool rbd
Removing image: 100% complete...done.
[ceph: root@serverf /]# exit
exit
[root@serverf ~]$
```

► 8. Exit and close the second terminal. Return to `workstation` as the `student` user.

```
[root@serverf ~]# exit  
[admin@serverf ~]$ exit  
[student@workstation ~]$ exit
```

```
[root@clienta ~]# exit  
[admin@clienta ~]$ exit  
[student@workstation ~]$
```

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 block-import
```

This concludes the guided exercise.

► Lab

Providing Block Storage Using RADOS Block Devices

In this lab you will configure Red Hat Ceph Storage to provide block storage to clients using RADOS block devices (RBDs). You will import and export RBD images to and from the Ceph cluster.

Outcomes

You should be able to:

- Create and prepare an RBD pool.
- Create, manage, and use RBD images.
- Export and import RBD images.

Before You Begin

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

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

This command verifies the status of the cluster and creates the `rbd` pool if it does not already exist.

Instructions

Perform the following steps on your `clienta` admin node, which is a client node to the primary 3-node Ceph storage cluster.

1. Log in to `clienta` as the `admin` user. Create a pool called `rbd260`, enable the `rbd` client application for the Ceph block device, and make it usable by the RBD feature.
2. Create a 128 MiB RADOS block device image called `prod260` in the `rbd260` pool. Verify your work.
3. Map the `prod260` RBD image in the `rbd260` pool to a local block device file by using the kernel RBD client. Format the device with an XFS file system. Mount the file system on the `/mnt/prod260` image and copy the `/etc/resolv.conf` file to the root of this new file system. When done, unmount and unmap the device.
4. Create a snapshot of the `prod260` RBD image in the `rbd260` pool and name it `beforeprod`.
5. Export the `prod260` RBD image from the `rbd260` pool to the `/root/prod260.xfs` file. Import that image file into the `rbd` pool on your primary 3-node Ceph cluster, and name the imported image `img260` in that pool.
6. Configure the client system so that it persistently mounts the `rbd260/prod260` RBD image as `/mnt/prod260`. Authenticate as the `admin` Ceph user using existing keys found in the `/etc/ceph/ceph.client.admin.keyring` file.

7. Return to `workstation` as the `student` user.

Evaluation

Grade your work by running the `lab grade block-review` command from your `workstation` machine. Correct any reported failures and rerun the script until successful.

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

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 block-review
```

This concludes the lab.

► Solution

Providing Block Storage Using RADOS Block Devices

In this lab you will configure Red Hat Ceph Storage to provide block storage to clients using RADOS block devices (RBDs). You will import and export RBD images to and from the Ceph cluster.

Outcomes

You should be able to:

- Create and prepare an RBD pool.
- Create, manage, and use RBD images.
- Export and import RBD images.

Before You Begin

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

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

This command verifies the status of the cluster and creates the `rbd` pool if it does not already exist.

Instructions

Perform the following steps on your `clienta` admin node, which is a client node to the primary 3-node Ceph storage cluster.

1. Log in to `clienta` as the `admin` user. Create a pool called `rbd260`, enable the `rbd` client application for the Ceph block device, and make it usable by the RBD feature.
 - 1.1. Log in to `clienta`, as the `admin` user and use `sudo` to run the `cephadm` shell. Verify that the primary cluster is in a healthy state.

```
[student@workstation ~]$ ssh admin@clienta
...output omitted...
[admin@clienta ~]$ sudo cephadm shell
...output omitted...
[ceph: root@clienta /]# ceph health
HEALTH_OK
```

- 1.2. Create a pool called `rbd260` with 32 placement groups. Enable the `rbd` client application for the Ceph Block Device and make it usable by the RBD feature.

```
[ceph: root@clienta /]# ceph osd pool create rbd260 32
pool 'rbd260' created
[ceph: root@clienta /]# ceph osd pool application enable rbd260 rbd
enabled application 'rbd' on pool 'rbd260'
[ceph: root@clienta /]# rbd pool init -p rbd260
```

- 1.3. List the rbd260 pool details to verify your work. The pool ID might be different in your lab environment.

```
[ceph: root@clienta /]# ceph osd pool ls detail | grep rbd260
pool 7 'rbd260' replicated size 3 min_size 2 crush_rule 0 object_hash
rjenkins pg_num 32 pgp_num 32 autoscale_mode on last_change 203 flags
hashpspool, selfmanaged_snaps stripe_width 0 application rbd
```

2. Create a 128 MiB RADOS block device image called prod260 in the rbd260 pool. Verify your work.

- 2.1. Create the 128 MiB prod260 RBD image in the rbd260 pool.

```
[ceph: root@clienta /]# rbd create prod260 --size 128 --pool rbd260
```

- 2.2. List the images in the rbd260 pool to verify the result.

```
[ceph: root@clienta /]# rbd ls rbd260
prod260
```

3. Map the prod260 RBD image in the rbd260 pool to a local block device file by using the kernel RBD client. Format the device with an XFS file system. Mount the file system on the /mnt/prod260 image and copy the /etc/resolv.conf file to the root of this new file system. When done, unmount and unmap the device.

- 3.1. Exit the cephadm shell, then switch to the root user. Install the ceph-common package on the clienta node. Map the prod260 image from the rbd260 pool using the kernel RBD client.

```
[ceph: root@clienta /]# exit
exit
[admin@clienta ~]# sudo -i
[root@clienta ~]# yum install -y ceph-common
...output omitted...
Complete!
[root@clienta ~]# rbd map --pool rbd260 prod260
/dev/rbd0
```

- 3.2. Format the /dev/rbd0 device with an XFS file system and mount the file system on the device. Change the user and group ownership of the root directory of the new file system to admin.

```
[root@clienta ~]# mkfs.xfs /dev/rbd0
...output omitted...
[root@clienta ~]# mount /dev/rbd0 /mnt/prod260
[root@clienta ~]# chown admin:admin /mnt/prod260
```

- 3.3. Copy the /etc/resolv.conf file to the root of the /mnt/prod260 file system, and then list the contents to verify the copy.

```
[root@clienta ~]# cp /etc/resolv.conf /mnt/prod260
[root@clienta ~]# ls /mnt/prod260/
resolv.conf
```

- 3.4. Unmount and unmap the /dev/rbd0 device.

```
[root@clienta ~]# umount /dev/rbd0
[root@clienta ~]# rbd unmap --pool rbd260 prod260
```

4. Create a snapshot of the prod260 RBD image in the rbd260 pool and name it beforeprod.

- 4.1. Run the cephadm shell. Create the beforeprod snapshot of the prod260 image in the rbd260 pool.

```
[root@clienta ~]# cephadm shell
...output omitted...
[ceph: root@clienta /]# rbd snap create rbd260/prod260@beforeprod
Creating snap: 100% complete...done.
```

- 4.2. List the snapshots of the prod260 RBD image in the rbd260 pool to verify your work.

```
[ceph: root@clienta /]# rbd snap list --pool rbd260 prod260
SNAPID  NAME      SIZE    PROTECTED   TIMESTAMP
  4  beforeprod  128 MiB          Mon Oct  4 17:11:57 2021
```



Note

The snapshot ID and the time stamp are different in your lab environment.

5. Export the prod260 RBD image from the rbd260 pool to the /root/prod260.xfs file. Import that image file into the rbd pool on your primary 3-node Ceph cluster, and name the imported image img260 in that pool.

- 5.1. Export the prod260 RBD image from the rbd260 pool to a file called /root/prod260.xfs.

```
[ceph: root@clienta /]# rbd export rbd260/prod260 /root/prod260.xfs
Exporting image: 100% complete...done.
```

- 5.2. Retrieve the size of the /home/admin/prod260.xfs file to verify the export.

```
[ceph: root@clienta /]# ls -lh /root/prod260.xfs  
-rw-r--r-- 1 root root 128M Oct  4 17:39 /root/prod260.xfs
```

- 5.3. Import the /root/prod260.xfs file as the img260 RBD image in the rbd pool.

```
[ceph: root@clienta /]# rbd import /root/prod260.xfs rbd/img260  
Importing image: 100% complete...done.
```

- 5.4. List the images in the rbd pool to verify the import. Exit from the cephadm shell.

```
[ceph: root@clienta /]# rbd --pool rbd ls  
img260  
[ceph: root@clienta /]# exit  
exit  
[root@clienta ~]#
```



Note

The rbd ls command might display images from previous exercises.

6. Configure the client system so that it persistently mounts the rbd260/prod260 RBD image as /mnt/prod260. Authenticate as the admin Ceph user using existing keys found in the /etc/ceph/ceph.client.admin.keyring file.

- 6.1. Create an entry for the rbd260/prod260 image in the /etc/ceph/rbdmap RBD map file. The resulting file should have the following contents:

```
[root@clienta ~]# cat /etc/ceph/rbdmap  
# RbdDevice          Parameters  
#poolname/imagename   id=client,keyring=/etc/ceph/ceph.client.keyring  
rbd260/prod260        id=admin,keyring=/etc/ceph/ceph.client.admin.keyring
```

- 6.2. Create an entry for the /dev/rbd/rbd260/prod260 image in the /etc/fstab file. The resulting file should have the following contents:

```
[root@clienta ~]# cat /etc/fstab  
UUID=d47ead13-ec24-428e-9175-46aefa764b26      /      xfs      defaults 0 0  
UUID=7B77-95E7  /boot/efi      vfat  
defaults,uid=0,gid=0,umask=077,shortname=winnt 0 2  
/dev/rbd/rbd260/prod260  /mnt/prod260    xfs      noauto  0 0
```

- 6.3. Use the rbdmap command to validate your RBD map configuration.

```
[root@clienta ~]# rbdmap map  
[root@clienta ~]# rbd showmapped  
id  pool      namespace  image  snap  device  
0   rbd260     prod260   -      /dev/rbd0  
[root@clienta ~]# rbdmap unmap  
[root@clienta ~]# rbd showmapped
```

- 6.4. After you have confirmed that the RBD mapped devices work, enable the `rbdmap` service. Reboot the `clienta` node to confirm that the RBD device mounts persistently.

```
[root@clienta ~]# systemctl enable rbdmap
Created symlink /etc/systemd/system/multi-user.target.wants/rbdmap.service → /usr/
lib/systemd/system/rbdmap.service.
[root@clienta ~]# reboot
Connection to clienta closed by remote host.
Connection to clienta closed.
```

- 6.5. After rebooting, log in to the `clienta` node as the `admin` user. Confirm that the system has mounted the RBD device.

```
[student@workstation ~]$ ssh admin@clienta
...output omitted...
[admin@clienta ~]$ df -h /mnt/prod260
Filesystem      Size  Used Avail Use% Mounted on
/dev/rbd0       121M  7.8M  113M   7% /mnt/prod260
```

7. Return to `workstation` as the `student` user.

- 7.1. Return to `workstation` as the `student` user.

```
[admin@clienta ~]$ exit
[student@workstation ~]$
```

Evaluation

Grade your work by running the `lab grade block-review` command from your `workstation` machine. Correct any reported failures and rerun the script until successful.

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

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 block-review
```

This concludes the lab.

Summary

In this chapter, you learned:

- The `rbd` command manages RADOS block device pools, images, snapshots, and clones.
- The `rbd map` command uses the `krbd` kernel module to map RBD images to Linux block devices. Configuring the `rbdmap` service can map these images persistently.
- RBD has an export and import mechanism for maintaining copies of RBD images that are fully functional and accessible.
- The `rbd export-diff` and the `rbd import-diff` commands export and import RBD image changes made between two points in time.

Chapter 7

Expanding Block Storage Operations

Goal

Expand block storage operations by implementing remote mirroring and the iSCSI Gateway.

Objectives

- Configure an RBD mirror to replicate an RBD block device between two Ceph clusters for disaster recovery purposes.
- Configure the Ceph iSCSI Gateway to export RADOS Block Devices using the iSCSI protocol, and configure clients to use the iSCSI Gateway.

Sections

- Configuring RBD Mirrors (and Guided Exercise)
- Providing iSCSI Block Storage (and Quiz)

Lab

Expanding Block Storage Operations

Configuring RBD Mirrors

Objectives

After completing this section, you should be able to configure an RBD mirror to replicate an RBD block device between two Ceph clusters for disaster recovery purposes.

RBD Mirroring

Red Hat Ceph Storage supports *RBD mirroring* between two storage clusters. This allows you to automatically replicate RBD images from one Red Hat Ceph Storage cluster to another remote cluster. This mechanism mirrors the source (primary) RBD image and the target (secondary) RBD image over the network using an asynchronous mechanism. If the cluster containing the primary RBD image becomes unavailable, then you can fail over to the secondary RBD image from the remote cluster and restart the applications that use it.

When failing over from the source RBD image to the mirror RBD image, you must *demote* the source RBD image and *promote* the target RBD image. A demoted image becomes locked and unavailable. A promoted image becomes available and accessible in read/write mode.

The RBD mirroring features requires the *rbd-mirror* daemon. The *rbd-mirror* daemon pulls the image updates from the remote peer cluster and applies them to the local cluster image.

Supported Mirroring Configurations

RBD mirroring supports two configurations:

One-way mirroring or active-passive

In one-way mode, the RBD images of one cluster are available in read/write mode and the remote cluster contains mirrors. The mirroring agent runs on the remote cluster. This mode enables the configuration of multiple secondary clusters.

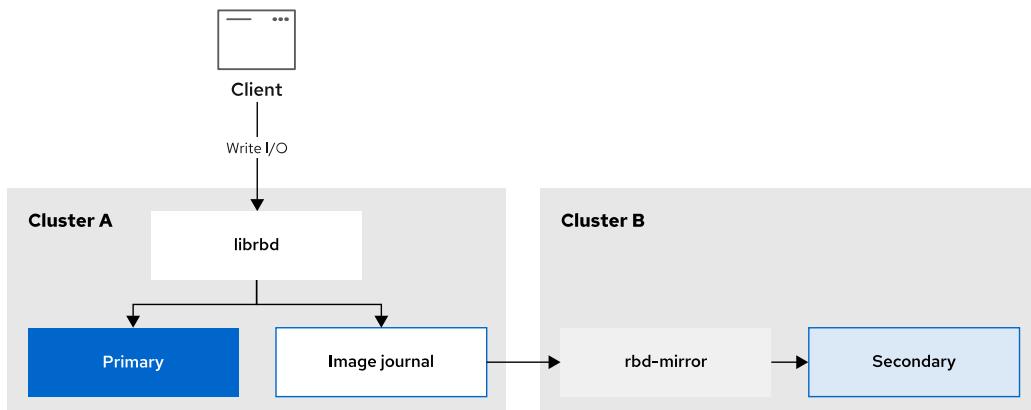


Figure 7.1: One-way mirroring

Two-way mirroring or active-active

In two-way mode, Ceph synchronizes the source and target pairs (primary and secondary). This mode allows replication between only two clusters, and you must configure the mirroring agent on each cluster.

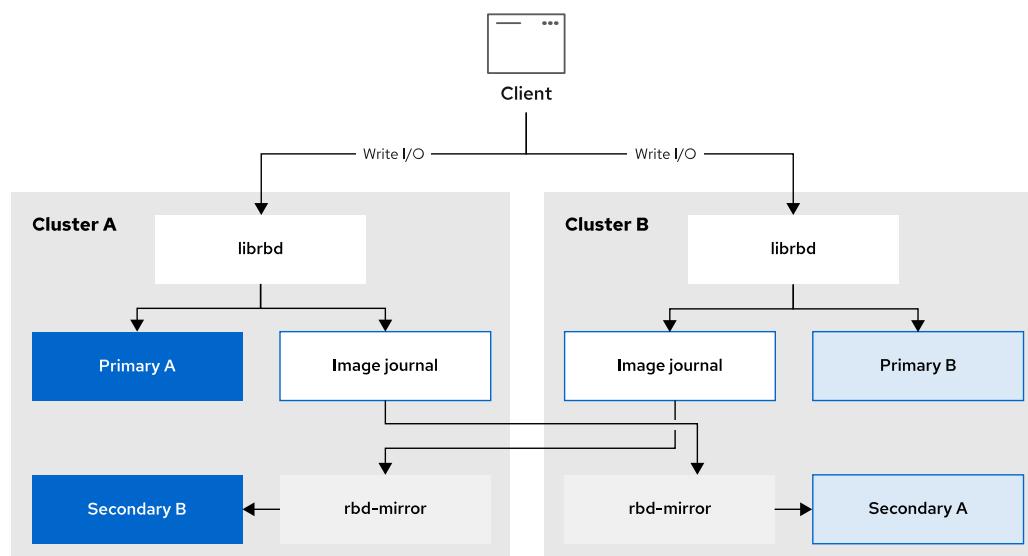


Figure 7.2: Two-way mirroring

Supported Mirroring Modes

RBD mirroring supports two modes: pool mode and image mode.

Pool Mode

In pool mode, Ceph automatically enables mirroring for each RBD image created in the mirrored pool. When you create an image in the pool on the source cluster, Ceph creates a secondary image on the remote cluster.

Image Mode

In image mode, mirroring can be selectively enabled for individual RBD images within the mirrored pool. In this mode, you have to explicitly select the RBD images to replicate between the two clusters.

RBD Image Mirroring Modes

The RBD images asynchronously mirrored between two Red Hat Ceph Storage clusters have the following modes:

Journal-based mirroring

This mode uses the RBD journaling image feature to ensure point-in-time and crash-consistent replication between two Red Hat Ceph Storage clusters. Every writes to the RBD image is first recorded to the associated journal before modifying the actual image. The remote cluster reads from this journal and replays the updates to its local copy of the image.

Snapshot-based mirroring

Snapshot-based mirroring uses periodically scheduled or manually created RBD image mirror snapshots to replicate crash-consistent RBDs images between two Red Hat Ceph Storage clusters. The remote cluster determines any data or metadata updates between two mirror snapshots and copies the deltas to the image's local copy. The RBD fast-diff image

feature enables the quick determination of updated data blocks without the need to scan the full RBD image. The complete delta between two snapshots must be synced prior to use during a failover scenario. Any partially applied set of deltas will be rolled back at the moment of failover.

Managing Replication

Image resynchronization

In case of an inconsistent state between the two peer clusters, the `rbd-mirror` daemon does not attempt to mirror the image that is causing the inconsistency, use `rbd mirror image resync` to resynchronize an image.

```
[ceph: root@node /]# rbd mirror image resync mypool/myimage
```

Enabling and disabling image mirroring

Use `rbd mirror image enable` or `rbd mirror image disable` to enable or disable mirroring on the whole pool in image mode on both peer storage clusters.

```
[ceph: root@node /]# rbd mirror image enable mypool/myimage
```

Using snapshot-based mirroring

To use the *snapshot-based mirroring* convert journal-based mirroring to snapshot-based mirroring by disabling mirroring and enabling snapshot.

```
[ceph: root@node /]# rbd mirror image disable mypool/myimage
Mirroring disabled
```

```
[ceph: root@node /]# rbd mirror image enable mypool/myimage snapshot
Mirroring enabled
```

Configuring RBD Mirroring

As a storage administrator, you can improve redundancy by mirroring data images between Red Hat Ceph Storage clusters. Ceph block device mirroring provides protection against data loss, such as a site failure.

To achieve RBD mirroring, and enable the `rbd-mirror` daemon to discover its peer cluster, you must have a registered peer and a created user account. Red Hat Ceph Storage 5 automates this process by using the `rbd mirror pool peer bootstrap create` command.



Important

Each instance of the `rbd-mirror` daemon must connect to both the local and remote Ceph clusters simultaneously. Also, the network must have sufficient bandwidth between the two data centers to handle the mirroring workload.

Configuring RBD Mirroring Step-by-Step

The `rbd-mirror` daemon does not require the source and destination clusters to have unique internal names; both can and should call themselves `ceph`. The `rbd mirror pool peer bootstrap` command utilizes the `--site-name` option to describe the clusters used by the `rbd-mirror` daemon.

The following list outlines the steps required to configure mirroring between two clusters, called prod and backup:

1. Create a pool with the same name in both clusters, prod and backup.
2. Create or modify the RBD image with the `exclusive-lock`, and `journaling` features enabled.
3. Enable pool-mode or image-mode mirroring on the pool.
4. In the prod cluster, bootstrap the storage cluster peer and save the bootstrap token
5. Deploy a `rbd-mirror` daemon.
 - For one-way replication the `rbd-mirror` daemon runs only on the backup cluster.
 - For two-way replication the `rbd-mirror` daemon runs on both clusters.
6. In the backup cluster, import the bootstrap token.
 - For one-way replication, use the `--direction rx-only` argument.

Step by Step One-way Pool Mode Example

In this example, you see the step-by-step instructions needed to configure one-way mirroring with the prod and backup clusters.

```
[admin@node ~]$ ssh admin@prod-node
[admin@prod-node ~]# sudo cephadm shell --mount /home/admin/token/
[ceph: root@prod-node /]# ceph osd pool create rbd 32 32
pool 'rbd' created
[ceph: root@prod-node /]# ceph osd pool application enable rbd rbd
enabled application 'rbd' on pool 'rbd'
[ceph: root@prod-node /]# rbd pool init -p rbd
[ceph: root@prod-node /]# rbd create my-image \
--size 1024 \
--pool rbd \
--image-feature=exclusive-lock,journaling
[ceph: root@prod-node /]# rbd mirror pool enable rbd pool
[ceph: root@prod-node /]# rbd --image my-image info
rbd image 'my-image':
  size 1 GiB in 256 objects
  order 22 (4 MiB objects)
  snapshot_count: 0
  id: acf674690a0c
  block_name_prefix: rbd_data.acf674690a0c
  format: 2
  features: exclusive-lock, journaling
  op_features:
  flags:
  create_timestamp: Wed Oct  6 22:07:41 2021
  access_timestamp: Wed Oct  6 22:07:41 2021
  modify_timestamp: Wed Oct  6 22:07:41 2021
  journal: acf674690a0c
  mirroring state: enabled
  mirroring mode: journal
  mirroring global id: d1140b2e-4809-4965-852a-2c21d181819b
```

```

    mirroring primary: true
[ceph: root@prod-node /]# rbd mirror pool peer bootstrap create \
--site-name prod rbd > /mnt/bootstrap_token_prod
[ceph: root@prod-node /]# exit
exit
[root@prod-node ~]# rsync -avP /home/admin/token/bootstrap_token_prod \
backup-node:/home/admin/toke n/bootstrap_token_prod
...output omitted...
[root@prod-node ~]# exit
logout
[admin@node ~]$ ssh admin@backup-node
[root@backup-node ~]# cephadm shell --mount /home/admin/token/
[ceph: root@backup-node /]# ceph osd pool create rbd 32 32
pool 'rbd' created
[ceph: root@backup-node /]# ceph osd pool application enable rbd rbd
enabled application 'rbd' on pool 'rbd'
[ceph: root@backup-node /]# rbd pool init -p rbd
[ceph: root@backup-node /]# ceph orch apply rbd-mirror \
--placement=backup-node.example.com
Scheduled rbd-mirror update...
[ceph: root@backup-node /]# rbd mirror pool peer bootstrap import \
--site-name backup --direction rx-only r bd /mnt/bootstrap_token_prod
[ceph: root@backup-node /]# rbd -p rbd ls
my-image

```

The backup cluster displays the following pool information and status.

```

[ceph: root@backup-node /]# rbd mirror pool info rbd
Mode: pool
Site Name: backup

Peer Sites:

UUID: 5e2f6c8c-a7d9-4c59-8128-d5c8678f9980
Name: prod
Direction: rx-only
Client: client.rbd-mirror-peer
[ceph: root@backup-node /]# rbd mirror pool status
health: OK
daemon health: OK
image health: OK
images: 1 total
    1 replaying

```

The prod cluster displays the following pool information and status.

```

[ceph: root@prod-node /]# rbd mirror pool info rbd
Mode: pool
Site Name: prod

Peer Sites:

UUID: 6c5f860c-b683-44b4-9592-54c8f26ac749

```

```
Name: backup
Mirror UUID: 7224d1c5-4bd5-4bc3-aa19-e3b34efd8369
Direction: tx-only
[ceph: root@prod-node /]# rbd mirror pool status
health: UNKNOWN
daemon health: UNKNOWN
image health: OK
images: 1 total
    1 replaying
```



Note

In one-way mode, the source cluster is not aware of the state of the replication. The RBD mirroring agent in the target cluster updates the status information.

Failover Procedure

If the primary RBD image becomes unavailable, then you can use the following steps to enable access to the secondary RBD image:

- Stop access to the primary RBD image. This means stopping all applications and virtual machines that are using the image.
- Use the `rbd mirror image demote pool-name/image-name` command to demote the primary RBD image.
- Use the `rbd mirror image promote pool-name/image-name` command to promote the secondary RBD image.
- Resume access to the RBD image. Restart the applications and virtual machines.



Note

When a failover after a non-orderly shutdown occurs, you must promote the non-primary images from a Ceph Monitor node in the backup storage cluster. Use the `--force` option because the demotion cannot propagate to the primary storage cluster



References

`rbd(8)` man page

For more information, refer to the *Mirroring Ceph block devices* chapter in the *Block Device Guide for Red Hat Ceph Storage 5* at

https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/block_device_guide/index

► Guided Exercise

Configuring RBD Mirrors

In this exercise you will configure pool-mode and image-mode RBD mirroring between two Ceph clusters.

Outcomes

You should be able to:

- Configure one-way, pool-mode RBD mirroring between two clusters.
- Verify the status of the mirroring process between two clusters.

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 mirror-mirrors
```

This command confirms that the hosts required for this exercise are accessible. Your Ceph clusters and configuration will not be modified by this `lab start` command.

Instructions

- 1. Open two terminals and log in to `clienta` and `serverf` as the `admin` user. Verify that both clusters are reachable and have a `HEALTH_OK` status.
- 1.1. Open a terminal window and log in to `clienta`, as the `admin` user and switch to the `root` user. Run a `cephadm` shell. Verify the health of your production cluster.

```
[student@workstation ~]$ ssh admin@clienta
...output omitted...
[admin@clienta ~]$ sudo -i
[root@clienta ~]# cephadm shell
[ceph: root@clienta /]# ceph status
...output omitted...
cluster:
  id:      ff97a876-1fd2-11ec-8258-52540000fa0c
  health:  HEALTH_OK

  services:
    mon: 4 daemons, quorum serverc.lab.example.com,servere,serverd,clienta (age
15m)
    mgr: serverc.lab.example.com.btgxor(active, since 15m), standbys:
servere.fmyxwv, clienta.soxncl, serverd.ufqxxk
    osd: 9 osds: 9 up (since 15m), 9 in (since 47h)
    rgw: 2 daemons active (2 hosts, 1 zones)

  data:
```

```
pools: 5 pools, 105 pgs
objects: 190 objects, 5.3 KiB
usage: 147 MiB used, 90 GiB / 90 GiB avail
pgs: 105 active+clean
```



Important

Ensure that the monitor daemons displayed in the `services` section match those of your 3-node production cluster plus the client.

- 1.2. Open another terminal window and log in to `serverf` as the `admin` user and switch to the `root` user. Verify the health of your backup cluster.

```
[student@workstation ~]$ ssh admin@serverf
...output omitted...
[admin@serverf ~]$ sudo -i
[root@serverf ~]# cephadm shell
[ceph: root@clientf /]# ceph status
...output omitted...
cluster:
  id: 3c67d550-1fd3-11ec-a0d5-52540000fa0f
  health: HEALTH_OK

  services:
    mon: 1 daemons, quorum serverf.lab.example.com (age 18m)
    mgr: serverf.lab.example.com.qfmyuk(active, since 18m)
    osd: 5 osds: 5 up (since 18m), 5 in (since 47h)
    rgw: 1 daemon active (1 hosts, 1 zones)

  data:
    pools: 5 pools, 105 pgs
    objects: 189 objects, 4.9 KiB
    usage: 82 MiB used, 50 GiB / 50 GiB avail
    pgs: 105 active+clean
```



Important

Ensure that the monitor daemon displayed in the `services` section matches that of your single-node backup cluster.

- ▶ 2. Create a pool called `rbd` in the production cluster with 32 placement groups. In the backup cluster, configure a pool to mirror the data from the `rbd` pool in the production cluster to the backup cluster. Pool-mode mirroring *always* mirrors data between two pools that have the same name in both clusters.
- 2.1. In the production cluster, create a pool called `rbd` with 32 placement groups. Enable the `rbd` client application for the Ceph Block Device and make it usable by the RBD feature.

```
[ceph: root@clienta /]# ceph osd pool create rbd 32 32
pool 'rbd' created
[ceph: root@clienta /]# ceph osd pool application enable rbd rbd
enabled application 'rbd' on pool 'rbd'
[ceph: root@clienta /]# rbd pool init -p rbd
```

- 2.2. In the backup cluster, create a pool called **rbd** with 32 placement groups. Enable the **rbd** client application for the Ceph Block Device and make it usable by the RBD feature.

```
[ceph: root@serverf /]# ceph osd pool create rbd 32 32
pool 'rbd' created
[ceph: root@serverf /]# ceph osd pool application enable rbd rbd
enabled application 'rbd' on pool 'rbd'
[ceph: root@serverf /]# rbd pool init -p rbd
```

- 3. In the production cluster, create a test RBD image and verify it. Enable pool-mode mirroring on the pool.

- 3.1. Create an RBD image called **image1** in the **rbd** pool in the production cluster. Specify a size of 1024 MB. Enable the **exclusive-lock** and **journaling** RBD image features.

```
[ceph: root@clienta /]# rbd create image1 \
--size 1024 \
--pool rbd \
--image-feature=exclusive-lock,journaling
```

- 3.2. List the images, and show the information about the **image1** image in the **rbd** pool.

```
[ceph: root@clienta /]# rbd -p rbd ls
image1
[ceph: root@clienta /]# rbd --image image1 info
rbd image 'image1':
  size 1 GiB in 256 objects
  order 22 (4 MiB objects)
  snapshot_count: 0
  id: acb0966ee3a0
  block_name_prefix: rbd_data.acb0966ee3a0
  format: 2
  features: exclusive-lock, journaling
  op_features:
  flags:
  create_timestamp: Wed Sep 29 21:14:20 2021
  access_timestamp: Wed Sep 29 21:14:20 2021
  modify_timestamp: Wed Sep 29 21:14:20 2021
  journal: acb0966ee3a0
  mirroring state: disabled
```

- 3.3. Enable pool-mode mirroring on the **rbd** pool, and verify it.

```
[ceph: root@clienta /]# rbd mirror pool enable rbd pool
[ceph: root@clienta /]# rbd --image image1 info
rbd image 'image1':
  size 1 GiB in 256 objects
  order 22 (4 MiB objects)
  snapshot_count: 0
  id: acb0966ee3a0
  block_name_prefix: rbd_data.acb0966ee3a0
  format: 2
  features: exclusive-lock, journaling
  op_features:
  flags:
  create_timestamp: Wed Sep 29 21:14:20 2021
  access_timestamp: Wed Sep 29 21:14:20 2021
  modify_timestamp: Wed Sep 29 21:14:20 2021
  journal: acb0966ee3a0
  mirroring state: enabled
  mirroring mode: journal
  mirroring global id: a4610478-807b-4288-9581-241f651d63c3
  mirroring primary: true
```

- ▶ 4. In the production cluster, create a `/root/mirror/` directory. Run the `cephadm` shell by using the `--mount` argument to mount the `/root/mirror/` directory. Bootstrap the storage cluster peer and create Ceph user accounts, then save the token in the `/mnt/bootstrap_token_prod` file in the container. Copy the bootstrap token file to the backup storage cluster.
- 4.1. On the `clienta` node, exit the `cephadm` shell. Create the `/root/mirror/` directory, then run the `cephadm` shell to bind mount the `/root/mirror` directory.

```
[ceph: root@clienta /]# exit
[root@clienta ~]# mkdir /root/mirror
[root@clienta ~]# cephadm shell --mount /root/mirror/
...output omitted...
[ceph: root@clienta /]#
```

- 4.2. Bootstrap the storage cluster peer and save the output in the `/mnt/bootstrap_token_prod` file. Name the production cluster `prod`.

```
[ceph: root@clienta /]# rbd mirror pool peer bootstrap create \
--site-name prod rbd > /mnt/bootstrap_token_prod
```

- 4.3. Exit the `cephadm` shell to the `clienta` host system. Copy the bootstrap token file to the backup storage cluster in the `/root` directory.

```
[ceph: root@clienta /]# exit
exit
[root@clienta ~]# rsync -avP /root/mirror/bootstrap_token_prod \
serverf:/root/bootstrap_token_prod
...output omitted...
```

- 5. In the backup cluster, run the `cephadm` shell with a bind mount of the `/root/bootstrap_token_prod` file. Deploy a `rbd-mirror` daemon in the `serverf` node. Import the bootstrap token. Verify that the RBD image is present.

- 5.1. On the `serverf` node, exit the `cephadm` shell. Run the `cephadm` shell again to bind mount the `/root/mirror` directory.

```
[ceph: root@serverf /]# exit  
[root@serverf ~]# cephadm shell --mount /root/bootstrap_token_prod  
...output omitted...  
[ceph: root@serverf /]#
```

- 5.2. Deploy a `rbd-mirror` daemon, use the argument `--placement` to set the `serverf.lab.example.com` node, and then verify it.

```
[ceph: root@serverf /]# ceph orch apply rbd-mirror \  
--placement=serverf.lab.example.com  
Scheduled rbd-mirror update...  
[ceph: root@serverf /]# ceph orch ls  
NAME          RUNNING  REFRESHED  AGE  PLACEMENT  
...output omitted...  
rbd-mirror      1/1    1s ago     6s  serverf.lab.example.com  
...output omitted...
```

- 5.3. Import the bootstrap token located in the `/mnt/bootstrap_token_prod` file. Name the backup cluster `bup`.

```
[ceph: root@serverf /]# rbd mirror pool peer bootstrap import \  
--site-name bup --direction rx-only rbd /mnt/bootstrap_token_prod
```



Important

Ignore the known error containing the following text: auth: unable to find a keyring on ...

- 5.4. Verify that the RBD image is present.

```
[ceph: root@serverf /]# rbd -p rbd ls  
image1
```

- 6. Display the pool information and status in both Ceph clusters.

- 6.1. In the production cluster, run the `cephadm` shell. Display the pool information and status.

```
[root@clienta ~]# cephadm shell  
[ceph: root@clienta /]# rbd mirror pool info rbd  
Mode: pool  
Site Name: prod  
  
Peer Sites:
```

```
UUID: deacabfb-545f-4f53-9977-ce986d5b93b5
Name: bup
Mirror UUID: bec08767-04c7-494e-b01e-9c1a75f9aa0f
Direction: tx-only
[ceph: root@clienta /]# rbd mirror pool status
health: UNKNOWN
daemon health: UNKNOWN
image health: OK
images: 1 total
    1 replaying
```

6.2. In the backup cluster, display the pool information and status.

```
[ceph: root@serverf /]# rbd mirror pool info rbd
Mode: pool
Site Name: bup

Peer Sites:

UUID: 591a4f58-3ac4-47c6-a700-86408ec6d585
Name: prod
Direction: rx-only
Client: client.rbd-mirror-peer
[ceph: root@serverf /]# rbd mirror pool status
health: OK
daemon health: OK
image health: OK
images: 1 total
    1 replaying
```

- 7. Clean up your environment. Delete the RBD image from the production cluster and verify that it is absent from both clusters.

7.1. In the production cluster, remove the `image1` block device from the `rbd` pool.

```
[ceph: root@clienta /]# rbd rm image1 -p rbd
Removing image: 100% complete...done.
```

7.2. In the production cluster, list block devices in the `rbd` pool.

```
[ceph: root@clienta /]# rbd -p rbd ls
```

7.3. In the backup cluster, list block devices in the `rbd` pool.

```
[ceph: root@serverf /]# rbd -p rbd ls
```

- 8. Exit and close the second terminal. Return to `workstation` as the `student` user.

```
[ceph: root@serverf /]# exit  
[root@serverf ~]# exit  
[admin@serverf ~]$ exit  
[student@workstation ~]$ exit
```

```
[ceph: root@clienta /]# exit  
[root@clienta ~]# exit  
[admin@clienta ~]$ exit  
[student@workstation ~]$
```

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 mirror-mirrors
```

This concludes the guided exercise.

Providing iSCSI Block Storage

Objectives

After completing this section, you should be able to configure the Ceph iSCSI Gateway to export RADOS Block Devices using the iSCSI protocol, and configure clients to use the iSCSI Gateway.

Describing the Ceph iSCSI Gateway

Red Hat Ceph Storage 5 can provide highly available iSCSI access to RADOS block device images stored in the cluster. The iSCSI protocol allows clients (initiators) to send SCSI commands to storage devices (targets) over TCP/IP networks. Each initiator and target is uniquely identified by an iSCSI qualified name (IQN). Clients with standard iSCSI initiators can access cluster storage without requiring native Ceph RBD client support.

The Linux I/O target kernel subsystem runs on every iSCSI gateway to support the iSCSI protocol. Previously called LIO, the iSCSI target subsystem is now called *TCM*, or the Target Core Mod. The TCM subsystem utilizes a user-space pass-through (TCMU) to interact with the Ceph *librbd* library to expose RBD images to iSCSI clients.

iSCSI-specific OSD Tuning

Object Storage Devices (OSDs) and Monitors (MONs) do not require any iSCSI-specific server settings. To limit client SCSI time outs, reduce the delay timeout setting that the cluster uses to detect a failing OSD.

In the `cephadm` shell, run the `ceph tell <daemon_type>. <id> config set` command to set the timeout parameters.

```
[root@node ~]# ceph config set osd osd_heartbeat_interval 5
[root@node ~]# ceph config set osd osd_heartbeat_grace 20
[root@node ~]# ceph config set osd osd_client_watch_timeout 15
```

Deploying an iSCSI Gateway

You can deploy the iSCSI gateway on dedicated nodes or collocated with the OSDs. Meet the following prerequisites before deploying a Red Hat Ceph Storage iSCSI gateway:

- Install the iSCSI gateway nodes with Red Hat Enterprise Linux 8.3 or later.
- Have an operational cluster running Red Hat Ceph Storage 5 or later.
- Have 90 MiB of RAM available for each RBD image exposed as a target on iSCSI gateway nodes.
- Open TCP ports 3260 and 5000 on the firewall on each Ceph iSCSI node.
- Create a new RADOS block device or use an existing, available device.

Create a Configuration File

To deploy iSCSI gateway nodes, use the `cephadm` shell to create a configuration file called `iscsi-gateway.yaml` in the `/etc/ceph/` directory. The file should display as follows:

```

service_type: iscsi
service_id: iscsi
placement:
  hosts:
    - serverc.lab.example.com
    - servere.lab.example.com
spec:
  pool: iscsipool1
  trusted_ip_list: "172.25.250.12,172.25.250.14"
  api_port: 5000
  api_secure: false
  api_user: admin
  api_password: redhat

```

Apply the Specification File and Deploy the iSCSI Gateway

Run the `ceph orch apply` command to implement the configuration by using the `-i` option to use a specification file.

```
[ceph: root@node /]# ceph orch apply -i /etc/ceph/iscsi-gateway.yaml
Scheduled iscsi.iscsi update...
```

List the gateways and verify that they are present.

```
[ceph: root@node /]# ceph dashboard iscsi-gateway-list
{"gateways": {"serverc.lab.example.com": {"service_url": "http://
admin:redhat@172.25.250.12:5000"}, "servere.lab.example.com": {"service_url": "
"http://admin:redhat@172.25.250.14:5000"}}}
```

Open a web browser and log in to the Ceph Dashboard as a user with administrative privileges. In the Ceph Dashboard web UI, click **Block** → **iSCSI** to display the **iSCSI Overview** page.

Name	State	# Targets	# Sessions
serverc.lab.example.com	up	0	0
servere.lab.example.com	up	0	0

Images

Pool	Image	Backstore	Read Bytes	Write Bytes	Read Ops	Write Ops	A/O Since
No data to display							

Figure 7.3: iSCSI gateway Dashboard page

After the Ceph Dashboard is configured to access the iSCSI gateway APIs, use it to manage iSCSI targets. Use the Ceph Dashboard to create, view, edit, and delete iSCSI targets.

Configure an iSCSI Target

Use the Ceph Dashboard or the `ceph-iscsi gwcli` utility to configure an iSCSI target.

The *Using gwcli to add more iSCSI gateways* section of the *Block Device Guide for Red Hat Ceph Storage 5* provides detailed instructions on how to manage iSCSI targets using the `ceph-iscsi gwcli` utility.

These are example steps to configure an iSCSI target from the Ceph Dashboard.

1. Log in to the Dashboard.
2. On the navigation menu, click **Block** → **iSCSI**.
3. Click the **Targets** tab.
4. Select **Create** from the Create list.
5. In the *Create Target* window, set the following parameters:
 - a. Modify the Target IQN (optional).
 - b. Click **+Add portal** and select the first of at least two gateways.
 - c. Click **+Add image** and select an image for the target to export.
 - d. Click **Create Target**.

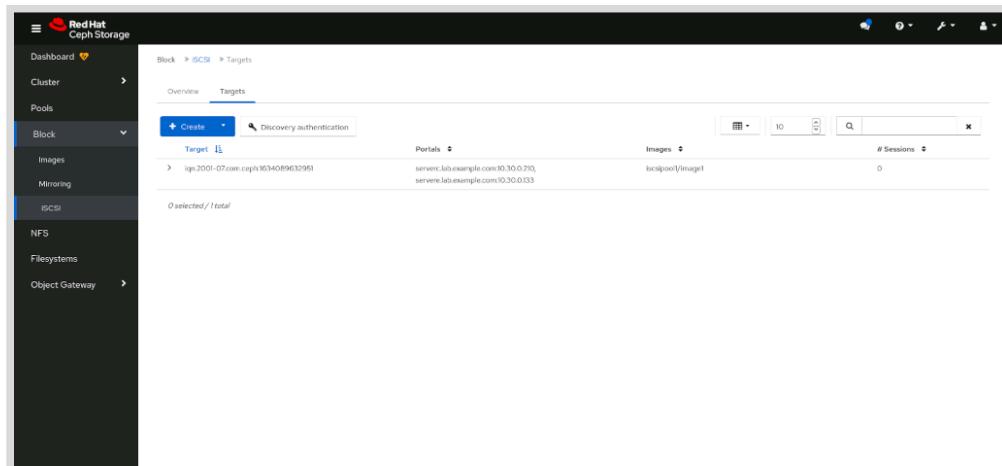


Figure 7.4: iSCSI Targets page

Configure an iSCSI Initiator

Configuring an iSCSI initiator to communicate with the Ceph iSCSI gateway is the same as for any industry-standard iSCSI gateway. For RHEL 8, install the `iscsi-initiator-utils` and the `device-mapper-multipath` packages. The `iscsi-initiator-utils` package contains the utilities required to configure an iSCSI initiator. When using more than one iSCSI gateway, configure clients for multipath support to failover between gateways by using the cluster's iSCSI targets.

**Note**

A system might be able to access the same storage device through multiple different communication paths, whether those are using Fibre Channel, SAS, iSCSI, or some other technology. Multipathing allows you to configure a virtual device that can use any of these communication paths to access your storage. If one path fails, then the system automatically switches to use one of the other paths instead.

If deploying a single iSCSI gateway for testing, skip the multipath configuration.

These example steps configure an iSCSI initiator to use multipath support and to log in to an iSCSI target. Configure your client's Challenge-Handshake Authentication Protocol (CHAP) user name and password to log in to the iSCSI targets.

1. Install the iSCSI initiator tools.

```
[root@node ~]# yum install iscsi-initiator-utils
```

2. Configure multipath I/O.

- Install the multipath tools.

```
[root@node ~]# yum install device-mapper-multipath
```

- Enable and create a default multipath configuration.

```
[root@node ~]# mpathconf --enable --with_multipathd y
```

- Add the following to the /etc/multipath.conf file.

```
devices {
    device {
        vendor           "LIO-ORG"
        hardware_handler "1 alua"
        path_grouping_policy "failover"
        path_selector    "queue-length 0"
        fallback         60
        path_checker     tur
        prio             alua
        prio_args        exclusive_pref_bit
        fast_io_fail_tmo 25
        no_path_retry    queue
    }
}
```

- Restart the multipathd service.

```
[root@node ~]# systemctl reload multipathd
```

3. If required for your configuration, set CHAP authentication.

Chapter 7 | Expanding Block Storage Operations

- Update the CHAP user name and password to match your iSCSI gateway configuration in the `/etc/iscsi/iscsid.conf` file.

```
node.session.auth.authmethod = CHAP
node.session.auth.username = user
node.session.auth.password = password
```

4. Discover and log in to the iSCSI portal, and then view targets and their multipath configuration.
 - Discover the iSCSI portal.

```
[root@node ~]# iscsiadm -m discovery -t st -p 10.30.0.210
10.30.0.210:3260,1 iqn.2001-07.com.ceph:1634089632951
10.30.0.133:3260,2 iqn.2001-07.com.ceph:1634089632951
```

- Log in to the iSCSI portal.

```
[root@node ~]# iscsiadm -m node -T iqn.2001-07.com.ceph:1634089632951 -l
Logging in to [iface: default, target: iqn.2001-07.com.ceph:1634089632951, portal:
10.30.0.210,3260]
Logging in to [iface: default, target: iqn.2001-07.com.ceph:1634089632951, portal:
10.30.0.133,3260]
Login to [iface: default, target: iqn.2001-07.com.ceph:1634089632951, portal:
10.30.0.210,3260] successful.
Login to [iface: default, target: iqn.2001-07.com.ceph:1634089632951, portal:
10.30.0.133,3260] successful.
```

- Verify any attached SCSI targets.

```
[root@node ~]# lsblk
NAME      MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
sda        8:0     0   1G  0 disk
└─mpatha 253:0    0   1G  0 mpath
sdb        8:16    0   1G  0 disk
└─mpatha 253:0    0   1G  0 mpath
vda       252:0    0 10G  0 disk
```

- Use the `multipath` command to show devices set up in a failover configuration with a priority group for each path.

```
[root@node ~]# multipath -ll
mpatha (3600140537b026aa91c844138da53ffe7) dm-0 LIO-ORG,TCMU device
size=1.0G features='1 queue_if_no_path' hwhandler='1 alua' wp=rw
|-+- policy='queue-length 0' prio=50 status=active
| `-- 2:0:0:0 sda 8:0 active ready running
`-- policy='queue-length 0' prio=10 status=enabled
  `-- 3:0:0:0 sdb 8:16 active ready running
```

**Note**

If logging in to an iSCSI target through a single iSCSI gateway, then the system creates a physical device for the iSCSI target (for example, /dev/sdX). If logging in to an iSCSI target through multiple iSCSI gateways with Device Mapper multipath, then the Device Mapper creates a multipath device (for example, /dev/mapper/mpatha).

**References**

For more information, refer to the *The Ceph iSCSI Gateway* chapter in the *Block Device Guide for Red Hat Ceph Storage 5* at

https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/block_device_guide/index#the-ceph-iscsi-gateway

For more information, refer to the *Management of iSCSI functions using the Ceph dashboard* section in the *_Dashboard Guide for Red Hat Ceph Storage 5* at

https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/dashboard_guide/index#management-of-iscsi-functions-on-the-ceph-dashboard

Further information, refer to the Device Mapper Multipath configuration for Red Hat Enterprise Linux 8 is available at *Configuring Device Mapper Multipath* at

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_device_mapper_multipath/

► Quiz

Providing iSCSI Block Storage

Choose the correct answers to the following questions:

► 1. **Which of the following statements best describes iSCSI?**

- a. An iSCSI target provides a POSIX file system to clients (initiators) over Fibre Channel.
- b. iSCSI allows initiators to send SCSI commands to storage devices (targets) over a TCP/IP network.
- c. The iSCSI protocol is only used by Linux.
- d. A dedicated proprietary hardware array containing SCSI drives is necessary to provide iSCSI storage devices.

► 2. **What are two of the requirements for deploying the Ceph iSCSI Gateway? (Choose two.)**

- a. Red Hat Enterprise Linux 8.3 or later.
- b. At least two nodes on which to deploy the Ceph iSCSI Gateway.
- c. 90 MiB of RAM per RBD image exposed as a target.
- d. A dedicated network between the initiators and the iSCSI Gateways.
- e. A 10 GbE network between the iSCSI Gateways and the Red Hat Ceph Storage cluster nodes.

► 3. **Which two of the following methods is used to expose an RBD image as an iSCSI target? (Choose two.)**

- a. Use the `targetcli` command from the `targetcli` package.
- b. Use the `Storage` page in the RHEL Web Console.
- c. Use the `Block → iSCSI` page in the Ceph Dashboard.
- d. Use the `mpathconf` command from the `mpathconf` package..
- e. Use the `gwcli` command from the `ceph-iscsi` package.

► 4. **Which package must be present on iSCSI initiator systems that connect to a target provided by an Ceph iSCSI gateway?**

- a. `ceph-iscsi`
- b. `iscsi-initiator-utils`
- c. `ceph-common`
- d. `storaged-iscsi`

► Solution

Providing iSCSI Block Storage

Choose the correct answers to the following questions:

► 1. **Which of the following statements best describes iSCSI?**

- a. An iSCSI target provides a POSIX file system to clients (initiators) over Fibre Channel.
- b. iSCSI allows initiators to send SCSI commands to storage devices (targets) over a TCP/IP network.
- c. The iSCSI protocol is only used by Linux.
- d. A dedicated proprietary hardware array containing SCSI drives is necessary to provide iSCSI storage devices.

► 2. **What are two of the requirements for deploying the Ceph iSCSI Gateway? (Choose two.)**

- a. Red Hat Enterprise Linux 8.3 or later.
- b. At least two nodes on which to deploy the Ceph iSCSI Gateway.
- c. 90 MiB of RAM per RBD image exposed as a target.
- d. A dedicated network between the initiators and the iSCSI Gateways.
- e. A 10 GbE network between the iSCSI Gateways and the Red Hat Ceph Storage cluster nodes.

► 3. **Which two of the following methods is used to expose an RBD image as an iSCSI target? (Choose two.)**

- a. Use the targetcli command from the targetcli package.
- b. Use the Storage page in the RHEL Web Console.
- c. Use the Block → iSCSI page in the Ceph Dashboard.
- d. Use the mpathconf command from the mpathconf package..
- e. Use the gwcli command from the ceph-iscsi package.

► 4. **Which package must be present on iSCSI initiator systems that connect to a target provided by an Ceph iSCSI gateway?**

- a. ceph-iscsi
- b. iscsi-initiator-utils
- c. ceph-common
- d. storaged-iscsi

▶ Lab

Expanding Block Storage Operations

In this lab you will configure pool-mode RBD mirroring between two Red Hat Ceph clusters, demote the image on the primary cluster, and promote the image on the secondary cluster.

Outcomes

You should be able to configure two-way pool-mode RBD mirroring between two clusters.

Before You Begin

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

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

The `lab` command confirms that the hosts required for this exercise are accessible. It creates the `rbd` pool in the primary, and secondary clusters. It also creates an image in primary cluster, called `myimage` with `exclusive-lock` and `journaling` features enabled. Finally, this command creates the `/home/admin/mirror-review` directory in the primary cluster.

Instructions

1. Log in to `clienta` as the `admin` user. Run the `cephadm` shell with a bind mount of the `/home/admin/mirror-review/` directory. Verify that the primary cluster is in a healthy state. Verify that the `rbd` pool is created successfully.
2. Deploy the `rbd-mirror` daemon in the primary and secondary clusters.
3. Enable pool-mode mirroring on the `rbd` pool and verify it. Verify that the `journaling` feature on the `myimage` image is enabled.
4. Register the storage cluster peer to the pool, and then copy the bootstrap token file to the secondary cluster.
5. In the secondary cluster, import the bootstrap token located in the `/home/admin/mirror-review/` directory. Verify that the RBD image is present.
6. Verify the mirroring status in both clusters. Note which is the primary image.
7. Demote the primary image and promote the secondary image, and then verify the change.
8. Return to `workstation` as the `student` user.

Evaluation

Grade your work by running the `lab grade mirror-review` command from your `workstation` machine. Correct any reported failures and rerun the script until successful.

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

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 mirror-review
```

This concludes the lab.

► Solution

Expanding Block Storage Operations

In this lab you will configure pool-mode RBD mirroring between two Red Hat Ceph clusters, demote the image on the primary cluster, and promote the image on the secondary cluster.

Outcomes

You should be able to configure two-way pool-mode RBD mirroring between two clusters.

Before You Begin

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

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

The `lab` command confirms that the hosts required for this exercise are accessible. It creates the `rbd` pool in the primary, and secondary clusters. It also creates an image in primary cluster, called `myimage` with `exclusive-lock` and `journaling` features enabled. Finally, this command creates the `/home/admin/mirror-review` directory in the primary cluster.

Instructions

1. Log in to `clienta` as the `admin` user. Run the `cephadm` shell with a bind mount of the `/home/admin/mirror-review/` directory. Verify that the primary cluster is in a healthy state. Verify that the `rbd` pool is created successfully.
 - 1.1. Log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell with a bind mount. Use the `ceph health` command to verify that the primary cluster is in a healthy state.

```
[student@workstation ~]$ ssh admin@clienta
...output omitted...
[admin@clienta ~]$ sudo cephadm shell --mount /home/admin/mirror-review/
[ceph: root@clienta /]# ceph health
HEALTH_OK
```

- 1.2. Verify that the `rbd` pool and the `myimage` image are created.

```
[ceph: root@clienta /]# ceph osd lspools
1 device_health_metrics
2 .rgw.root
3 default.rgw.log
4 default.rgw.control
5 default.rgw.meta
6 rbd
```

2. Deploy the rbd-mirror daemon in the primary and secondary clusters.

- 2.1. On the primary cluster, deploy an rbd-mirror daemon in the serverc.lab.example.com node.

```
[ceph: root@clienta /]# ceph orch apply rbd-mirror \
--placement=serverc.lab.example.com
Scheduled rbd-mirror update...
```

- 2.2. Open another terminal window. Log in to serverf as the admin user and use sudo to run a cephadm shell. Use the ceph health command to verify that the primary cluster is in a healthy state.

```
[student@workstation ~]$ ssh admin@serverf
...output omitted...
[admin@serverf ~]$ sudo cephadm shell
...output omitted...
[ceph: root@serverf /]# ceph health
HEALTH_OK
```

- 2.3. Deploy an rbd-mirror daemon in the serverf.lab.example.com node.

```
[ceph: root@serverf /]# ceph orch apply rbd-mirror \
--placement=serverf.lab.example.com
Scheduled rbd-mirror update...
```

3. Enable pool-mode mirroring on the rbd pool and verify it. Verify that the journaling feature on the myimage image is enabled.

- 3.1. On the primary cluster, enable pool-mode mirroring on the rbd pool and verify it.

```
[ceph: root@clienta /]# rbd mirror pool enable rbd pool
[ceph: root@clienta /]# rbd mirror pool info rbd
Mode: pool
Site Name: 2ae6d05a-229a-11ec-925e-52540000fa0c

Peer Sites: none
```

- 3.2. On the primary cluster, verify the journaling feature on the myimage image.

```
[ceph: root@clienta /]# rbd --image myimage info
rbd image 'myimage':
  size 512 MiB in 128 objects
  order 22 (4 MiB objects)
  snapshot_count: 0
  id: 8605767b2168
  block_name_prefix: rbd_data.8605767b2168
  format: 2
  features: exclusive-lock, journaling
  op_features:
  flags:
  create_timestamp: Thu Oct 21 13:47:22 2021
  access_timestamp: Thu Oct 21 13:47:22 2021
```

```
modify_timestamp: Thu Oct 21 13:47:22 2021
journal: 8605767b2168
mirroring state: enabled
mirroring mode: journal
mirroring global id: 33665293-baba-4678-b9f2-ec0b8d1513ea
mirroring primary: true
```

4. Register the storage cluster peer to the pool, and then copy the bootstrap token file to the secondary cluster.
 - 4.1. Bootstrap the storage cluster peer and save the output in the /mnt/bootstrap_token_primary file. Name the production cluster primary.

```
[ceph: root@clienta /]# rbd mirror pool peer bootstrap create \
--site-name primary rbd > /mnt/bootstrap_token_primary
```

- 4.2. Exit the cephadm shell to the clienta host. Copy the bootstrap token file to the backup storage cluster in the /home/admin directory.

```
[ceph: root@clienta /]# exit
exit
[admin@clienta ~]$ sudo rsync -avP /home/admin/mirror-review \
serverf:/home/admin/
...output omitted...
```

5. In the secondary cluster, import the bootstrap token located in the /home/admin/mirror-review/ directory. Verify that the RBD image is present.
 - 5.1. Exit the cephadm shell to the serverf host. Use sudo to run the cephadm shell with a bind mount for the /home/admin/mirror-review/ directory.

```
[ceph: root@serverf /]# exit
exit
[admin@serverf ~]$ sudo cephadm shell --mount /home/admin/mirror-review/
```

- 5.2. Import the bootstrap token located in /mnt/bootstrap_token_primary file. Name the backup cluster secondary.

```
[ceph: root@serverf /]# rbd mirror pool peer bootstrap import \
--site-name secondary rbd /mnt/bootstrap_token_primary
```



Important

Ignore the known error containing the following text: auth: unable to find a keyring on ...

- 5.3. Verify that the RBD image is present.

```
[ceph: root@serverf /]# rbd --pool rbd ls
myimage
```

The image could take a few minutes to replicate and display in the list.

6. Verify the mirroring status in both clusters. Note which is the primary image.

6.1. On the primary cluster, run the `cephadm shell` and verify the mirroring status.

```
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]# rbd mirror image status rbd/myimage
myimage:
  global_id: 33665293-baba-4678-b9f2-ec0b8d1513ea
  state: up+stopped
  description: local image is primary
  service: serverc.yrgdmc on serverc.lab.example.com
  last_update: 2021-10-21 16:58:20
  peer_sites:
    name: secondary
    state: up+replaying
    description: replaying,
    {"bytes_per_second":0.0,"entries_behind_primary":0,"entries_per_second":0.0,
     "non_primary_position": {"entry_tid":3,"object_number":3,"tag_tid":1},
     "primary_position": {"entry_tid":3,"object_number":3,"tag_tid":1}}
    last_update: 2021-10-21 16:58:23
```

6.2. On the secondary cluster, verify the mirroring status.

```
[ceph: root@serverf /]# rbd mirror image status rbd/myimage
myimage:
  global_id: 33665293-baba-4678-b9f2-ec0b8d1513ea
  state: up+replaying
  description: replaying,
  {"bytes_per_second":0.0,"entries_behind_primary":0,"entries_per_second":0.0,
   "non_primary_position": {"entry_tid":3,"object_number":3,"tag_tid":1},
   "primary_position": {"entry_tid":3,"object_number":3,"tag_tid":1}}
  service: serverf.kclptt on serverf.lab.example.com
  last_update: 2021-10-21 16:58:23
  peer_sites:
    name: primary
    state: up+stopped
    description: local image is primary
    last_update: 2021-10-21 16:58:20
```

7. Demote the primary image and promote the secondary image, and then verify the change.

7.1. On the primary cluster, demote the image and verify the change.

```
[ceph: root@clienta /]# rbd mirror image demote rbd/myimage
Image demoted to non-primary
```

7.2. On the secondary cluster, promote the image and verify the change.

```
[ceph: root@serverf /]# rbd mirror image promote rbd/myimage
Image promoted to primary
```

7.3. On the primary cluster, verify the change.

```
[ceph: root@clienta /]# rbd mirror image status rbd/myimage
myimage:
  global_id: 33665293-baba-4678-b9f2-ec0b8d1513ea
  state: up+replaying
  description: replaying,
  {"bytes_per_second":1.2,"entries_behind_primary":0,"entries_per_second":0.05,
  "non_primary_position":{"entry_tid":0,"object_number":0,"tag_tid":3},
  "primary_position":{"entry_tid":0,"object_number":0,"tag_tid":3}}
  service: serverc.yrgdmc on serverc.lab.example.com
  last_update: 2021-10-21 17:27:20
  peer_sites:
    name: secondary
    state: up+stopped
    description: local image is primary
    last_update: 2021-10-21 17:27:23
```

- 7.4. On the secondary cluster, verify the change. Note that the primary image is now in the secondary cluster, on the `serverf` server.

```
[ceph: root@serverf /]# rbd mirror image status rbd/myimage
myimage:
  global_id: 33665293-baba-4678-b9f2-ec0b8d1513ea
  state: up+stopped
  description: local image is primary
  service: serverf.kclptt on serverf.lab.example.com
  last_update: 2021-10-21 17:28:23
  peer_sites:
    name: primary
    state: up+replaying
    description: replaying,
    {"bytes_per_second":0.0,"entries_behind_primary":0,"entries_per_second":0.0,
    "non_primary_position":{"entry_tid":0,"object_number":0,"tag_tid":3},
    "primary_position":{"entry_tid":0,"object_number":0,"tag_tid":3}}
    last_update: 2021-10-21 17:28:20
```

8. Return to `workstation` as the `student` user.

- 8.1. Exit and close the second terminal. Return to `workstation` as the `student` user.

```
[ceph: root@serverf /]# exit
[root@serverf ~]# exit
[admin@serverf ~]$ exit
[student@workstation ~]$ exit
```

```
[ceph: root@clienta /]# exit
[root@clienta ~]# exit
[admin@clienta ~]$ exit
[student@workstation ~]$
```

Evaluation

Grade your work by running the `lab grade mirror-review` command from your `workstation` machine. Correct any reported failures and rerun the script until successful.

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

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 mirror-review
```

This concludes the lab.

Summary

In this chapter, you learned:

- RBD mirroring supports automatic or selective mirroring of images using pool mode or image mode.
- The RBD mirror agent can replicate pool data between two Red Hat Ceph Storage clusters, in either one-way or two-way mode, to facilitate disaster recovery.
- Deploying an iSCSI gateway publishes RBD images as iSCSI targets for network-based block storage provisioning.

Chapter 8

Providing Object Storage Using a RADOS Gateway

Goal

Configure Red Hat Ceph Storage to provide object storage for clients using a RADOS Gateway (RGW).

Objectives

- Deploy a RADOS Gateway to provide clients with access to Ceph object storage.
- Configure the RADOS Gateway with multisite support to allow objects to be stored in two or more geographically diverse Ceph storage clusters.

Sections

- Deploying an Object Storage Gateway (and Guided Exercise)
- Configuring a Multisite Object Storage Deployment (and Guided Exercise)

Lab

Providing Object Storage Using a RADOS Gateway

Deploying an Object Storage Gateway

Objectives

After completing this section, you should be able to deploy a RADOS Gateway to provide clients with access to Ceph object storage.

Introducing Object Storage

Object storage stores data as discrete items, each individually called an *object*. Unlike files in a file system, objects are not organized in a tree of directories and subdirectories. Instead, objects are stored in a flat namespace. Each object is retrieved by using the object's unique *object ID*, also known as an *object key*.

Applications do not use normal file-system operations to access object data. Instead, applications access a REST API to send and receive objects. Red Hat Ceph Storage supports the two most common object APIs, Amazon S3 (Simple Storage Service) and OpenStack Swift (OpenStack Object Storage).

Amazon S3 calls the flat namespace for object storage a *bucket* while OpenStack Swift calls it a *container*. Because a namespace is flat, neither buckets nor containers can be nested. Ceph typically uses the term bucket, as does this lecture.

A single user account can be configured for access to multiple buckets on the same storage cluster. Buckets can each have different access permissions and be used to store objects for different use cases.

The advantage of object storage is that it is easy to use, expand, and scale. Because each object has a unique ID, it can be stored or retrieved without the user knowing the object's location. Without the directory hierarchy, relationships between objects are simplified.

Objects, similar to files, contain a binary data stream and can grow to arbitrarily large sizes. Objects also contain *metadata* about the object data, and natively support extended metadata information, typically in the form of key-value pairs. You can also create your own metadata keys and store custom information in the object as key values.

Introducing the RADOS Gateway

The RADOS Gateway, also called the Object Gateway (RGW), is a service that provides access to the Ceph cluster for clients using standard object storage APIs. The RADOS Gateway supports both the Amazon S3 and OpenStack Swift APIs.

The core daemon, `radosgw`, is built on top of the `librados` library. The daemon provides a web service interface, based on the Beast HTTP, WebSocket, and networking protocol library, as a front-end to handle API requests.

The `radosgw` is a client to Red Hat Ceph Storage that provides object access to other client applications. Client applications use standard APIs to communicate with the RADOS Gateway, and the RADOS Gateway uses `librados` module calls to communicate with the Ceph cluster.

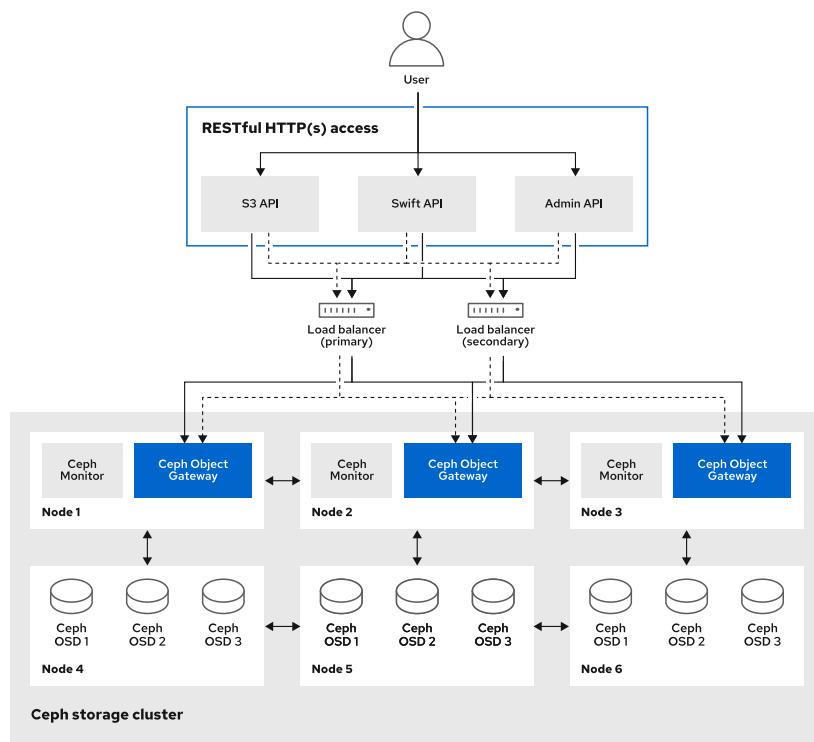


Figure 8.1: RADOS Gateway service architecture

The RADOS Gateway provides the `radosgw-admin` utility for creating users for using the gateway. These users can only access the gateway, and are not `cdephx` users with direct access to the storage cluster. RADOS Gateway clients authenticate using these gateway user accounts when submitting Amazon S3 or OpenStack Swift API requests. After a gateway user is authenticated by the RADOS Gateway, the gateway uses `cephx` credentials to authenticate to the storage cluster to handle the object request. Gateway users can also be managed by integrating an external LDAP-based authentication service.

The RADOS Gateway service automatically creates pools on a per-zone basis. These pools use placement group values from the configuration database and use the default CRUSH hierarchy. The default pool settings might not be optimal for a production environment.

The RADOS Gateway creates multiple pools for the default zone.

- **.rgw.root** - Stores information records
- **.default.rgw.control** - Used as the control pool
- **.default.rgw.meta** - Stores user_keys and other critical metadata
- **.default.rgw.log** - Contains logs of all bucket/container and object actions such as create, read, and delete
- **.default.rgw.buckets.index** - Stores indexes of the buckets
- **.default.rgw.buckets.data** - Stores bucket data
- **.default.rgw.buckets.non-ec** - Used for multipart object metadata uploads

You can manually create pools with custom settings. Red Hat recommends using the zone name as a prefix for manually created pools, as in `.<zone-name>.rgw.control`. For example, using `.us-east-1.rgw.buckets.data` as a pool name when `us-east-1` is the zone name.

Supporting Static Web Hosting with RADOS Gateway

RADOS Gateway supports static website hosting in S3 buckets, which can be more efficient than using virtual machines for website hosting. This is suitable for websites that only use static elements such as XHTML or HTML files, or CSS.

Deploying RADOS Gateway instances for static web hosting has restrictions.

- Instances cannot also be used for S3 or Swift API access.
- Instances should have domain names that are different from and do not overlap those of the standard S3 and Swift API gateway instances.
- Instances should use public-facing IP addresses that are different from the standard S3 and Swift API gateway instances.

RADOS Gateway Deployment

The `cephadm` tool deploys RADOS Gateway services as a collection of daemons that manage a single cluster or a multisite deployment. Use the `client. rgw.*` section in the centralized configuration database to define parameters and characteristics for new RADOS Gateway daemons.

Use the Ceph Orchestrator to deploy or remove RADOS Gateway services. Use the Ceph Orchestrator with either the command-line interface or a service specification file.

```
[ceph: root@node /]# ceph orch apply rgw <service-name> [--realm=<realm>] \
[--zone=<zone>] [--port=< port>] --placement="<num-daemons> [<host1> ...]" \
[--unmanaged]
```

In this example, the Ceph Orchestrator deploys the `my_rgw_service` RADOS Gateway service with two daemons in a single cluster, and presents the service on port 80.

```
[ceph: root@node /]# ceph orch apply rgw my_rgw_service
```

If no configuration is defined in the `client. rgw.*` section or passed to the Ceph orchestrator at build time, then the deployment uses these default settings.

The following example YAML file contains common parameters defined for a RADOS Gateway deployment.

```
service_type: rgw
service_name: rgw_service_name
placement:
  count: 2
  hosts:
    - node01
    - node02
spec:
  rgw_frontend_port: 8080
  rgw_realm: realm_name
  rgw_zone: zone_name
  ssl: true
  rgw_frontend_ssl_certificate: |
    -----BEGIN PRIVATE KEY-----
```

```
...output omitted...
-----END PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
...output omitted...
-----END CERTIFICATE-----
networks:
- 172.25.200.0/24
```

In this example, a RGW service is created with similar parameters than the previous one, but now using the CLI.

```
[ceph: root@node /]# ceph orch apply rgw rgw_service_name --realm=realm_name \
--zone=zone_name --port 8080 --placement="2 node01 node 02" --ssl
```

Notice that in the service specification file, the parameter names for the realm, zone, and port are different than the used by the CLI. Some parameters such as the network to be used by RGW instances or the ssl certificate content can only be defined by using the service specification file.

The count parameter sets the number of RGW instances to be created on each server defined in the hosts parameter. If you create more than one instance, then the Ceph orchestrator sets the port of the first instance to the specified rgw_frontend_port or port value from. For each subsequent instance, the port value is increased by 1. Using the previous YAML file example, the service deployment creates:

- Two RGW instances in the node01 server, one with port 8080, another with port 8081.
- Two RGW instances in the node02 server, one with port 8080, another with port 8081.

Each instance has its own unique port enabled for access and creates the same responses to requests. Configure high availability for the RADOS Gateway by deploying a load-balancer service that presents a single service IP address and port.



Note

The Ceph orchestrator service names the daemons by using the format
rgw.<realm>.<zone>.<host>.<random-string>

Customizing the Service Configuration

Configure the Beast front-end web port for the RADOS Gateway by using the port option in the rgw_frontends parameter in the cluster configuration client.rgw section. View the current configuration with the ceph config command.

```
[ceph: root@node /]# ceph config get client.rgw rgw_frontends
beast port=7480
```

When using Transport Layer Security/Secure Socket Layer (TLS/SSL), the ports are defined using an s character at the end of the port number, such as port=443s. The port option supports a dual-port configuration using the plus character (+), so that users can access the RADOS Gateway on either of two different ports.

For example, a rgw_frontends configuration can enable the RADOS Gateway to listen on the 80/TCP port, and with TLS/SSL support on the 443/TCP port.

```
[ceph: root@node /]# ceph config get client.rgw rgw_frontends
beast port=80+443s
```

Using the Beast Front-end

The RADOS Gateway provides Beast embedded HTTP servers as a front-end. A Beast front-end uses the `Boost .Beast` library for HTTP parsing and the `Boost .Asio` library for asynchronous network I/O.

Beast Configuration Options

Configure the Beast web server to use TLS by configuring certificates from a Certificate Authority (CA) with the hostname of the RGW instances and matching secret keys.

Beast configuration options are passed to the embedded web server in the Ceph configuration file or from the configuration database. If a value is not specified, the default value is empty.

port and ssl_port

Sets the listening port number for the IPv4 and IPv6 protocols and can be specified multiple times, as in `port=80 port=8000`.

endpoint and ssl_endpoint

Sets the listening address in the form `address[:port]`, and can be specified multiple times, as in `endpoint=[::1] endpoint=192.168.0.100:8000`.

ssl_certificate

Specifies the path to the SSL certificate file used for SSL-enabled endpoints.

ssl_private_key

Specifies an SSL private key, but if a value is not provided, then the file specified by `ssl_certificate` is used as the private key.

tcp_nodelay

Sets performance optimization parameters in some environments.



Important

Red Hat recommends use of HAProxy and keepalived services to configure TLS/SSL access in production environments.

High Availability Proxy and Encryption

When the RADOS Gateway service load increases, you can deploy more RGW instances to support the workload. You can add instances in a single zone group deployment, but consider that each RGW instance has its own IP address and it can be difficult to balance requests to different instances in a single zone.

Instead, configure HAProxy and keepalived to balance the load across RADOS Gateway servers. HAProxy presents only one IP address and it balances the requests to all RGW instances. Keepalived ensures that the proxy nodes maintain the same presented IP address, independent of node availability.

**Note**

Configure at least two separated hosts for HAProxy and keepalived services to maintain high availability.

You can configure the HAProxy service to use HTTPS. To enable HTTPS, generate SSL keys and certificates for the configuration. If you do not have a certificate from a Certificate Authority, then use a self-signed certificate.

Server-side Encryption

You can enable server-side encryption to allow sending requests to the RADOS Gateway service using unsecured HTTP when it is not possible to send encrypted requests over SSL. Currently, the server-side encryption scenario is only supported when using the Amazon S3 API.

There are two options to configure server-side encryption for the RADOS Gateway, customer-provided keys or a key management service.

Customer-provided Keys

This option is implemented according to the Amazon SSE-C specification. Each read or write request to the RADOS Gateway service contains an encryption key provided by the user via the S3 client.

An object can only be encrypted with one key and users use different keys to encrypt different objects. It is the user's responsibility to track the keys used to encrypt each object.

Key Management Service

You can configure a key management service to securely store keys for the RADOS Gateway service. When a key management service is configured, the RADOS Gateway service retrieves keys on demand to encrypt or decrypt objects.

Figure 8.2 demonstrates the encryption flow between RADOS Gateway and an example HashiCorp Vault key management service.

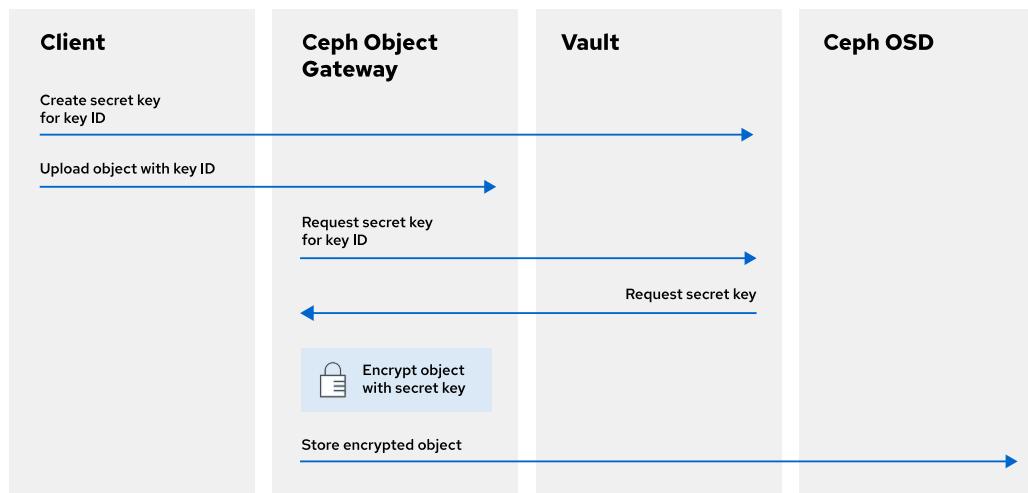


Figure 8.2: HashiCorp key management integration

Currently, HashiCorp Vault and OpenStack Barbican are the tested key management service implementations for RADOS Gateway.



Note

Integration with OpenStack Barbican is in technology preview and is not yet supported for production environments.



References

For more information, refer to *Red Hat RADOS Gateway for Production Guide* at
https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/object_gateway_guide/

For more information, refer to *The Beast front-end web server* at
https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/object_gateway_guide/index#the-beast-front-end-web-server_rgw

For more information, refer to *The HashiCorp Vault* at
https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/object_gateway_guide/index#the-hashicorp-vault

► Guided Exercise

Deploying an Object Storage Gateway

In this exercise, you will deploy a RADOS Gateway and verify client access.

Outcomes

You should be able to deploy a Ceph RADOS Gateway by using the Ceph orchestrator.

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-radosgw
```

This command confirms that the hosts required for this exercise are accessible.

Instructions

- 1. Log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell. Verify the health of the cluster.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]# ceph health
HEALTH_OK
```

- 2. View the cluster services. Verify that there are no `rgw` services running.

```
[ceph: root@clienta /]# ceph orch ls
NAME          RUNNING  REFRESHED  AGE   PLACEMENT
alertmanager    1/1     92s ago   2d    count:1
crash          4/4     4m ago    2d    *
grafana         1/1     92s ago   2d    count:1
mgr            4/4     4m ago    2d    ...;servere.lab.example.com
mon            4/4     4m ago    2d    ...;servere.lab.example.com
node-exporter    4/4     4m ago    2d    *
osd.default_drive_group 9/12    93s ago   2d    server*
prometheus      1/1     92s ago   2d    count:1
[ceph: root@clienta /]# ceph orch ls --service-type rgw
No services reported
```

- 3. Create the `rgw_service.yaml` file. Configure the service to start two RGW instances in each of the `serverd` and `servere` hosts. The ports of the RGW instances must start from port 8080. Your file should look like this example.

```
[ceph: root@clienta /]# cat rgw_service.yaml
service_type: rgw
service_id: myrealm.myzone
service_name: rgw.myrealm.myzone
placement:
  count: 4
  hosts:
    - serverd.lab.example.com
    - servere.lab.example.com
spec:
  rgw_frontend_port: 8080
```

- 4. Use the Ceph orchestrator to create an RGW service with the `rgw_service.yaml` file. View the cluster and RGW service status. Verify that there are two daemons per host.

- 4.1. Use Ceph orchestrator to create the RGW service with the `rgw_service.yaml` file.

```
[ceph: root@clienta /]# ceph orch apply -i rgw_service.yaml
Scheduled rgw.myrealm.myzone update...
```

- 4.2. View the cluster status and find the status of the new RGW service daemons.

```
[ceph: root@clienta /]# ceph status
cluster:
  id: 2ae6d05a-229a-11ec-925e-52540000fa0c
  health: HEALTH_OK

  services:
    mon: 4 daemons, quorum serverc.lab.example.com,clienta,serverd,servere (age
        4m)
    mgr: serverc.lab.example.com.aiqepd(active, since 10m), standbys:
        clienta.nncugs, serverd.klrkci
    osd: 9 osds: 9 up (since 8m), 9 in (since 9m)
    rgw: 4 daemons active (2 hosts, 1 zones)
...output omitted...
```

- 4.3. Verify that the orchestrator created two running daemons per node.

```
[ceph: root@clienta /]# ceph orch ps --daemon-type rgw
NAME                      HOST          STATUS
REFRESHED    AGE   PORTS ...
rgw.myrealm.myzone.serverd.tknapl serverd.lab.example.com  running (14s)  0s ago
  14s  *:8080 ...
rgw.myrealm.myzone.serverd.xpabfe serverd.lab.example.com  running (6s)   0s ago
  6s  *:8081 ...
rgw.myrealm.myzone.servere.lwusbq servere.lab.example.com  running (18s)  0s ago
  17s  *:8080 ...
rgw.myrealm.myzone.servere.uyginy servere.lab.example.com  running (10s)  0s ago
  10s  *:8081 ...
```

- 5. Log in to the `serverd` node and view the running containers. Filter the running container processes to find the RGW container. Verify that the Beast embedded web server is accessible on port 8080 and also on port 8081.
- 5.1. Exit the `cephadm` shell. Log in to `serverd` as the `admin` user and switch to the `root` user. List the running containers, filtered to find the RGW container.

```
[ceph: root@clienta /]# exit
[admin@clienta ~]$ ssh admin@serverd
admin@serverd's password: redhat
...output omitted...
[admin@serverd ~]$ sudo -i
[root@serverd ~]# podman ps -a --format "{{.ID}} {{.Names}}" | grep rgw
7e99b444305d ceph-2ae6d05a-229a-11ec-925e-52540000fa0c-rgw-myrealm-myzone-serverd-xpabfe
dba722fb413c ceph-2ae6d05a-229a-11ec-925e-52540000fa0c-rgw-myrealm-myzone-serverd-tknapl
```

- 5.2. Verify that the Beast embedded web server is accessible on port 8080, and also on port 8081. If the gateway is working, you will receive a tagged response.

```
[root@serverd ~]# curl http://serverd:8080
<?xml version="1.0" encoding="UTF-8"?><ListAllMyBucketsResult xmlns="http://
s3.amazonaws.com/doc/2006-03-01/"><Owner><ID>anonymous</ID><DisplayName></
DisplayName></Owner><Buckets></Buckets></ListAllMyBucketsResult>
[root@serverd ~]# curl http://serverd:8081
<?xml version="1.0" encoding="UTF-8"?><ListAllMyBucketsResult xmlns="http://
s3.amazonaws.com/doc/2006-03-01/"><Owner><ID>anonymous</ID><DisplayName></
DisplayName></Owner><Buckets></Buckets></ListAllMyBucketsResult>
```

- 6. Return to `workstation` as the `student` user.

```
[root@serverd ~]# exit
[admin@serverd ~]$ exit
[admin@clienta ~]$ exit
[student@workstation ~]$
```

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-radosgw
```

This concludes the guided exercise.

Configuring a Multisite Object Storage Deployment

Objectives

After completing this section, you should be able to configure the RADOS Gateway with multisite support to allow objects to be stored in two or more geographically diverse Ceph storage clusters.

RADOS Gateway Multisite Deployment

The Ceph RADOS Gateway supports *multisite* deployments within a global namespace, which allows the RADOS Gateway to automatically replicate object data between multiple Red Hat Ceph Storage clusters. A common supported use case is active/active replication between geographically separate clusters for disaster recovery purposes.

The latest multisite configuration simplifies failover and fallback procedures, supports active/active replication configuration between clusters, and incorporates new features such as a simpler configuration and support for namespaces.

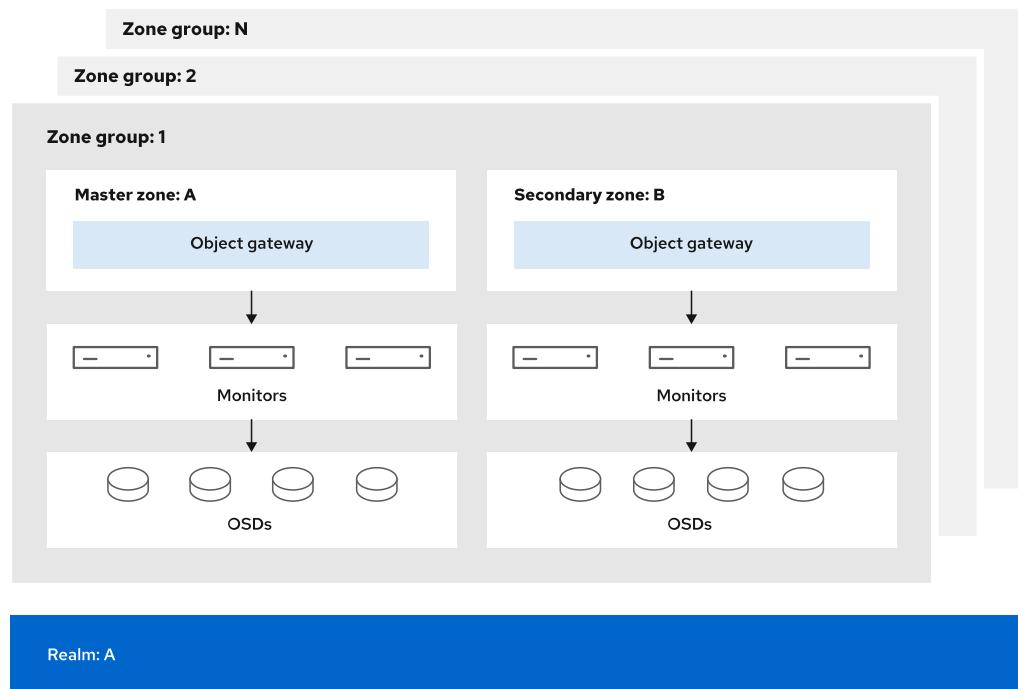


Figure 8.3: RADOS Gateway multisite diagram

Multisite Components

The multisite components and definitions are listed below.

zone

A zone is backed by its own Red Hat Ceph Storage cluster. Each zone has one or more RADOS Gateways associated with it.

zone group

A zone group is a set of one or more zones. Data stored in one zone in the zone group is replicated to all other zones in the zone group. One zone in every zone group is designated as the master zone for that group. The other zones in the zone group are secondary zones.

realm

A *realm* represents the global namespace for all objects and buckets in the multisite replication space. A realm contains one or more zone groups, each of which contains one or more zones. One zone group in the realm is designated as the master zone group, and the others are secondary zone groups. All RADOS Gateways in the environment pull their configuration from the RADOS Gateway in the master zone group and master zone.

Because the master zone in the master zone group handles all metadata updates, operations such as creating users must occur in the master zone.

**Important**

You can execute metadata operations in a secondary zone, but it is not recommended because the metadata will not be synchronized over the realm. This behavior can lead to metadata fragmentation and configuration inconsistency between zones.

This architecture can be structured in several ways:

- A **single zone** configuration has one zone group and one zone in the realm. One or more (possibly load-balanced) RADOS Gateways are backed by one Red Hat Ceph Storage cluster.
- A **multizone** configuration has one zone group but multiple zones. Each zone is backed by one or more RADOS Gateways and an independent Red Hat Ceph Storage cluster. Data stored in one zone is replicated to all zones in the zone group. This can be used for disaster recovery if one zone suffers a catastrophic failure.
- A **multizone group** configuration has multiple zone groups, each with one or more zones. You can use a multizone group to manage the geographic location of RADOS Gateways within one or more zones in a region.
- A **multiregion** configuration allows the same hardware to be used to support multiple object namespaces that are common across zone groups and zones.

A minimal RADOS Gateway multisite deployment requires two Red Hat Ceph Storage clusters, and a RADOS Gateway for each cluster. They exist in the same realm and are assigned to the same master zone group. One RADOS Gateway is associated with the master zone in that zone group. The other is associated with a separate secondary zone in that zone group. This is a basic multizone configuration.

Change Coordination with Periods and Epochs

Each realm has an associated period, and each period has an associated epoch. A period is used to track the configuration state of the realm, zone groups, and zones at a particular time. Epochs are version numbers to track configuration changes for a particular realm period. Each period has a unique ID, contains realm configuration, and knows the previous period ID.

When you update the configuration of the master zone, the RADOS Gateway service updates the period. This new period becomes the current period of the realm, and the epoch of this period increases its value by one. For other configuration changes, only the epoch is incremented; the period does not change.

Multisite Synchronization Process

The RADOS Gateway synchronizes metadata and data operations between all master and secondary zone group sets. Metadata operations relate to buckets: creating, deleting, enabling and disabling versioning, and managing users. The meta master, which is in the master zone of the master zone group, manages metadata updates. Data operations are those relevant to objects.

When a multisite configuration is active, the RADOS Gateway performs an initial full synchronization between the master and secondary zones. Subsequent updates are incremental. When the RADOS Gateway writes data to any zone within a zone group, it synchronizes this data between all of the zones in other zone groups. When the RADOS Gateway synchronizes data, all active gateways update the data log and notify the other gateways. When the RADOS Gateway synchronizes metadata because of a bucket or user operation, the master updates the metadata log and notifies the other RADOS Gateways.

Configuring Multisite RGW Deployments

You can deploy, configure, and remove multisite Ceph RADOS Gateway instances by using the Ceph orchestrator command-line interface or by using a service specification file.

Multisite Configuration Example

The following example configures a realm with one zone group containing two zones, one acting as the master zone and the other as a secondary zone. Each zone has one RADOS Gateway instance associated with it.

Configure the Master Zone

These example steps configure the RADOS Gateway instance in the master zone.

1. Create a realm.

```
[ceph: root@node01 /]# radosgw-admin realm create --default --rgw-realm=gold
```

2. Create a master zone group.

```
[ceph: root@node01 /]# radosgw-admin zonegroup create --rgw-zonegroup=us \
--master --default --endpoints=http://node01:80
```

3. Create a master zone.

```
[ceph: root@node01 /]# radosgw-admin zone create --rgw-zone=datacenter01 \
--master --rgw-zonegroup=us --endpoints=http://node01:80 \
--access-key=12345 --secret=67890 --default
```

4. Create a system user.

```
[ceph: root@node01 /]# radosgw-admin user create --uid=sysadm \
--display-name="SysAdmin" --access-key=12345 --secret=67890 --system
```

5. Commit the changes.

```
[ceph: root@node01 /]# radosgw-admin period update --commit
```

6. Create the RADOS Gateway service for the master zone.

```
[ceph: root@node /]# ceph orch apply rgw gold-service --realm=gold \
--zone=datacenter01 --placement="1 node01"
```

7. Update the zone name in the configuration database.

```
[ceph: root@node01 /]# ceph config set client.rgw rgw_zone datacenter01
```

Configure the Secondary Zone

These example steps configure the RADOS Gateway instance on the secondary zone.

1. Pull the realm configuration.

```
[ceph: root@node02 /]# radosgw-admin realm pull --rgw-realm=gold \
--url=http://node01:80 --access-key=12345 --secret=67890 --default
```

2. Pull the period.

```
[ceph: root@node02 /]# radosgw-admin period pull --url=http://node01:8000 \
--access-key=12345 --secret=67890
```

3. Create a secondary zone.

```
[ceph: root@node /]# radosgw-admin zone create --rgw-zone=datacenter02 \
--rgw-zonegroup=us --endpoints=http://node02:80 \
--access-key=12345 --secret=67890 --default
```

Use the `--read-only` option to set the zone as read-only when adding it to the zone group.

4. Commit the changes.

```
[ceph: root@node02 /]# radosgw-admin period update --commit
```

5. Create the RADOS Gateway service for the secondary zone.

```
[ceph: root@node02 /]# ceph orch apply rgw gold-service --realm=gold \
--zone=datacenter02 --placement="1 node02"
```

6. Update the zone name in the configuration database.

```
[ceph: root@node02 /]# ceph config set client.rgw rgw_zone datacenter02
```

Use the `radosgw-admin sync status` command to verify the synchronization status.

Managing Zone Failover

In a multisite deployment, when the master zone is unavailable, the secondary zones can continue to serve read and write requests. However, because the master zone is not available, you cannot

create new buckets and users. If the master zone does not recover immediately, promote one of the secondary zones as a replacement for the master zone.

To promote a secondary zone, modify the zone and zone group and commit the period update.

1. Point the master zone to the secondary zone (`datacenter02`).

```
[ceph: root@node02 /]# radosgw-admin zone modify --master --rgw-zone=datacenter02
```

2. Update the master zone group after changing the role of the zone.

```
[ceph: root@node02 /]# radosgw-admin zonegroup modify --rgw-zonegroup=us \
--endpoints=http://node02:80
```

3. Commit the changes.

```
[ceph: root@node02 /]# radosgw-admin period update --commit
```

Metadata Search Capabilities

Very large object stores are becoming common. Users still require smart access to objects so that they can extract additional information from the object metadata. The metadata associated with the objects provides insightful information about the objects. RADOS Gateway supports Elasticsearch to query metadata in an object store, and to index objects based on user-defined custom metadata for objects.

RADOS Gateway supports Elasticsearch as a component of the `multimode` architecture. Elasticsearch can manage the metadata for all objects in a zone group. A zone group can contain zones that store objects, while other zones store the metadata for those objects.

The following command defines the `metadata-zone` zone as a metadata zone managed by Elasticsearch:

```
[ceph: root@node /]# radosgw-admin zone modify --rgw-zone=metadata-zone \
--tier-type=elasticsearch \
--tier-config=endpoint=http://node03:9200,num_shards=10,num_replicas=1
```

- The `--tier-type` option sets the zone type to `elasticsearch`.
- The `--tier-config` option defines the configuration for the Elasticsearch zone.
- The `endpoint` parameter defines the endpoint to access the Elasticsearch server.
- The `num_shards` parameter defines the number of shards used by Elasticsearch.
- The `num_replicas` parameter specifies the number of replicas used by Elasticsearch.

RADOS Gateway Multisite Monitoring

You can monitor RADOS Gateway performance and usage statistics in the Ceph Storage Dashboard. To integrate the RADOS Gateway with the Ceph Storage Dashboard, you must add the login credentials of an RGW user with the `system` flag in a JSON format file.

Use the `ceph dashboard set-rgw-api-access-key` and `ceph dashboard set-rgw-api-secret-key` commands to provide the access key and secret key, when adding an RGW system user to the dashboard.

```
[ceph: root@node /]# cat access_key
{'myzone.node.tdncax': 'ACCESS_KEY'}
[ceph: root@node /]# cat secret_key
{'myzone.node.tdncax': 'SECRET_KEY'}
[ceph: root@node /]# ceph dashboard set-rgw-api-access-key -i access_key
Option RGW_API_ACCESS_KEY updated
[ceph: root@node /]# ceph dashboard set-rgw-api-secret-key -i secret_key
Option RGW_API_SECRET_KEY updated
```

To view details of the Ceph RADOS Gateway service, log in to the Dashboard and click **Object Gateway**. You are presented with a choice of **Daemons**, **Users**, or **Buckets**.

In the **Daemons** submenu, the dashboard shows a list of RGW daemons.

ID	Hostname	Zone Group	Zone	Version
cl260-1.serverc.tdncax	serverc.lab.example.com	classroom	us-east-1	16.2.0-117.el8qe

Figure 8.4: RADOS Gateway Daemon list

To view details and performance statistics, click a daemon name.

Name	Description	Value
objecter-0x562628f0ale0.op_r	Read operations	0
objecter-0x562628f0ale0.op_rw	Read-modify-write operations	0
objecter-0x562628f0ale0.op_w	Write operations	0.1999818818294775
objecter.op_active	Operations active	0
objecter.op_r	Read operations	5.399510809395893
objecter.op_rw	Read-modify-write operations	0
objecter.op_w	Write operations	1.999818818294775
rgw.cache_hit	Cache hits	1119898538245074
rgw.cache_miss	Cache miss	0
rgw.failed_req	Aborted requests	0

Figure 8.5: RADOS Gateway Daemon Performance

To view the overall performance of the service, click **Overall Performance**.

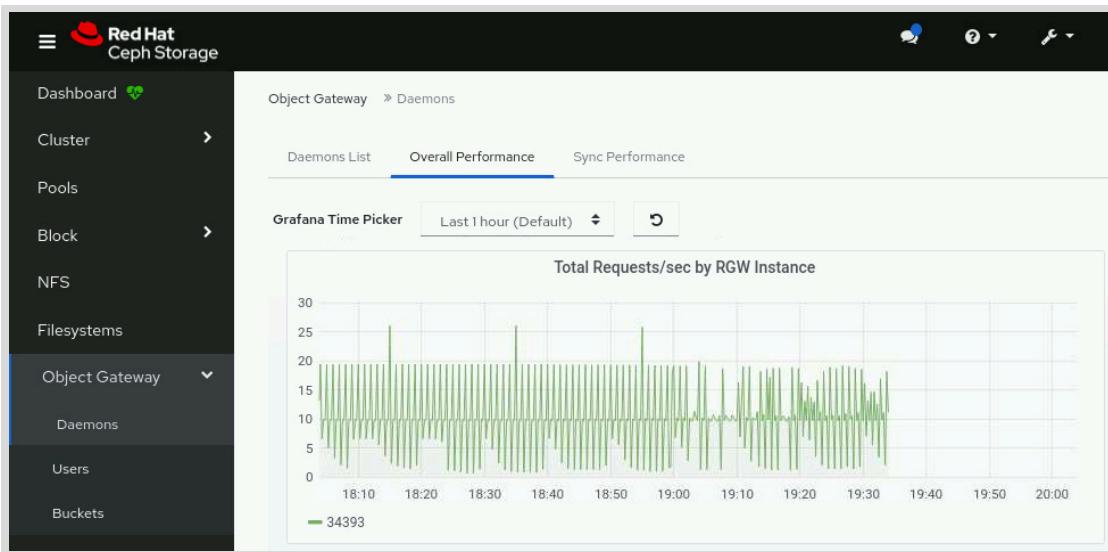


Figure 8.6: RADOS Gateway Overall Performance



References

RGW Metadata Search

<http://ceph.com/rgw/new-luminous-rgw-metadata-search/>

For more information, refer to the *Multi-site Configuration and Administration* chapter in the *Object Gateway Guide* at

https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/object_gateway_guide/index#multisite-configuration-and-administration

► Guided Exercise

Configuring a Multisite Object Storage Deployment

In this exercise, you will configure the RADOS Gateway with multisite support and verify the configuration.

Outcomes

You should be able to deploy a Ceph RADOS Gateway and configure multisite replication by using `serverc` in the primary cluster as site `us-east-1` and `serverf` as site `us-east-2`.

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-multisite
```

This command confirms that the hosts required for this exercise are accessible.

Instructions

- ▶ 1. Open two terminals and log in to both `serverc` and `serverf` as the `admin` user. Verify that both clusters are reachable and have a `HEALTH_OK` status.
 - 1.1. Open a terminal window. Log in to `serverc` as the `admin` user and use `sudo` to run the `cephadm` shell. Verify that the primary cluster is in a healthy state.

```
[student@workstation ~]$ ssh admin@serverc
...output omitted...
[admin@serverc ~]$ sudo cephadm shell
[ceph: root@serverc /]# ceph health
HEALTH_OK
```

- 1.2. Open another terminal window. Log in to `serverf` as the `admin` user and use `sudo` to run the `cephadm` shell. Verify that the secondary cluster is in a healthy state.

```
[student@workstation ~]$ ssh admin@serverf
...output omitted...
[admin@serverf ~]$ sudo cephadm shell
[ceph: root@serverf /]# ceph health
HEALTH_OK
```

- ▶ 2. On the `serverc` node, configure the `us-east-1` site. Create a realm, zone group, zone, and a replication user. Set the realm and zone as defaults for the site. Commit the configuration and review the period id. Use the names provided in the table:

Option	Name
--realm	cl260
--zonegroup	classroom
--zone	us-east-1
--uid	repl.user

- 2.1. Create a realm called `cl260`. Set the realm as the default.

```
[ceph: root@serverc /]# radosgw-admin realm create --rgw-realm=cl260 --default
{
    "id": "9eef2ff2-5fb1-4398-a69b-eeb3d9610638",
    "name": "cl260",
    "current_period": "031c3bfb-6626-47ef-a523-30b4313499d9",
    "epoch": 1
}
```

- 2.2. Create a zone group called `classroom`. Configure the `classroom` zone group with an endpoint pointing to the RADOS Gateway running on the `serverc` node. Set the `classroom` zone group as the default.

```
[ceph: root@serverc /]# radosgw-admin zonegroup create --rgw-zonegroup=classroom \
--endpoints=http://serverc:80 --master --default
{
    "id": "d3524ffb-8a3c-45f1-ac18-23db1bc99071",
    "name": "classroom",
    "api_name": "classroom",
    "is_master": "true",
    "endpoints": [
        "http://serverc:80"
    ],
    ...output omitted...
    "realm_id": "9eef2ff2-5fb1-4398-a69b-eeb3d9610638",
    ...output omitted...
}
```

- 2.3. Create a master zone called `us-east-1`. Configure the `us-east-1` zone with an endpoint pointing to `http://serverc:80`. Use `replication` as the access key and `secret` as the secret key. Set the `us-east-1` zone as the default.

```
[ceph: root@serverc /]# radosgw-admin zone create --rgw-zonegroup=classroom \
--rgw-zone=us-east-1 --endpoints=http://serverc:80 --master --default \
--access-key=replication --secret=secret
{
    "id": "4f1863ca-1fca-4c2d-a7b0-f693ddd14882",
    "name": "us-east-1",
    "domain_root": "us-east-1.rgw.meta:root",
    "control_pool": "us-east-1.rgw.control",
    ...output omitted...
```

```

"system_key": {
    "access_key": "replication",
    "secret_key": "secret"
},
"placement_pools": [
    {
        "key": "default-placement",
        "val": {
            "index_pool": "us-east-1.rgw.buckets.index",
            "storage_classes": {
                "STANDARD": {
                    "data_pool": "us-east-1.rgw.buckets.data"
                }
            },
            "data_extra_pool": "us-east-1.rgw.buckets.non-ec",
            "index_type": 0
        }
    }
],
"realm_id": "9eef2ff2-5fb1-4398-a69b-eeb3d9610638",
"notif_pool": "us-east-1.rgw.log:notif"
}

```

- 2.4. Create a system user called `repl.user` to access the zone pools. The keys for the `repl.user` user must match the keys configured for the zone.

```

[ceph: root@serverc /]# radosgw-admin user create --uid="repl.user" --system \
--display-name="Replica t ion User" --secret=secret --access-key=replication
{
    "user_id": "repl.user",
    "display_name": "Replication User",
    "email": "",
    "suspended": 0,
    "max_buckets": 1000,
    "subusers": [],
    "keys": [
        {
            "user": "repl.user",
            "access_key": "replication",
            "secret_key": "secret"
        }
    ...
    ...output omitted...
}

```

- 2.5. Every realm has an associated current period, holding the current state of zone groups and storage policies. Commit the realm configuration changes to the period. Note the period id associated with the current configuration.

```

[ceph: root@serverc /]# radosgw-admin period update --commit
{
    "id": "7cdc83cf-69d8-478e-b625-d5250ac4435b",
    "epoch": 1,
    "predecessor_uuid": "031c3bfb-6626-47ef-a523-30b4313499d9",
    "sync_status": [],
    "period_map": {

```

```

"id": "7cdc83cf-69d8-478e-b625-d5250ac4435b",
"zonegroups": [
{
    "id": "d3524ffb-8a3c-45f1-ac18-23db1bc99071",
    "name": "classroom",
    "api_name": "classroom",
    "is_master": "true",
    "endpoints": [
        "http://serverc:80"
    ],
    ...output omitted...
},
{
    "master_zonegroup": "d3524ffb-8a3c-45f1-ac18-23db1bc99071",
    "master_zone": "4f1863ca-1fca-4c2d-a7b0-f693ddd14882",
    ...output omitted...
},
{
    "realm_id": "9eef2ff2-5fb1-4398-a69b-eeb3d9610638",
    "realm_name": "cl260",
    "realm_epoch": 2
}
]
}

```

- ▶ 3. Create a new RADOS Gateway service called `cl260-1` in the `cl260` realm and `us-east-1` zone, and with a single RGW daemon on the `serverc` node. Verify that the RGW daemon is up and running. Update the zone name in the configuration database.

3.1. Create a new RADOS Gateway service called `cl260-1`.

```

[ceph: root@serverc /]# ceph orch apply rgw cl260-1 --realm=cl260 \
--zone=us-east-1 --placement="1 se rverc.lab.example.com"
Scheduled rgw.cl260-1 update...
[ceph: root@serverc /]# ceph orch ps --daemon-type rgw
NAME                      HOST          STATUS      REFRESHED
AGE  PORTS ...
rgw.cl260-1.serverc.sxsntj  serverc.lab.example.com  running (6m)  6m ago      6m
*:80 ...

```

3.2. Update the zone name in the configuration database.

```
[ceph: root@serverc /]# ceph config set client.rgw rgw_zone us-east-1
```

- ▶ 4. On the `serverf` node, pull the realm and period configuration in from the `serverc` node. Use the credentials for `replic1.user` to authenticate. Verify the current period id is the same as for the `serverc` node.

4.1. On the second terminal, pull the realm configuration from the `serverc` node.

```

[ceph: root@serverf /]# radosgw-admin realm pull --url=http://serverc:80 \
--access-key=replication --secret-key=secret
{
    "id": "9eef2ff2-5fb1-4398-a69b-eeb3d9610638",
    "name": "cl260",
    "current_period": "7cdc83cf-69d8-478e-b625-d5250ac4435b",
    "epoch": 2
}

```

4.2. Pull the period configuration from the serverc node.

```
[ceph: root@serverf /]# radosgw-admin period pull --url=http://serverc:80 \
--access-key=replication --secret-key=secret
{
    "id": "7cdc83cf-69d8-478e-b625-d5250ac4435b",
    "epoch": 1,
    "predecessor_uuid": "031c3bfb-6626-47ef-a523-30b4313499d9",
    "sync_status": [],
    "period_map": {
        "id": "7cdc83cf-69d8-478e-b625-d5250ac4435b",
        "zonegroups": [
            {
                "id": "d3524ffb-8a3c-45f1-ac18-23db1bc99071",
                "name": "classroom",
                "api_name": "classroom",
                "is_master": "true",
                "endpoints": [
                    "http://serverc:80"
                ],
                "hostnames": [],
                "hostnames_s3website": [],
                "master_zone": "4f1863ca-1fca-4c2d-a7b0-f693ddd14882",
                "zones": [
                    {
                        "id": "4f1863ca-1fca-4c2d-a7b0-f693ddd14882",
                        "name": "us-east-1",
                        "endpoints": [
                            "http://serverc:80"
                        ],
...
.output omitted...
        "master_zonegroup": "d3524ffb-8a3c-45f1-ac18-23db1bc99071",
        "master_zone": "4f1863ca-1fca-4c2d-a7b0-f693ddd14882",
...
.output omitted...
        "realm_id": "9eef2ff2-5fb1-4398-a69b-eeb3d9610638",
        "realm_name": "cl260",
        "realm_epoch": 2
    }
}
```

4.3. View the current period id for us-east-2 zone.

```
[ceph: root@serverf /]# radosgw-admin period get-current
{
    "current_period": "7cdc83cf-69d8-478e-b625-d5250ac4435b"
}
```

- 5. On the serverf node, configure the us-east-2 site. Set the c1260 realm and classroom zone group as the defaults and create the us-east-2 zone. Commit the site configuration and review the period id. Update the zone name in the configuration database.

5.1. Set the cl260 realm and classroom zone group as default.

```
[ceph: root@serverf /]# radosgw-admin realm default --rgw-realm=cl260
[ceph: root@serverf /]# radosgw-admin zonegroup default --rgw-zonegroup=classroom
```

- 5.2. Create a zone called us-east-2. Configure the us-east-2 zone with an endpoint pointing to `http://serverf:80`.

```
[ceph: root@serverf /]# radosgw-admin zone create --rgw-zonegroup=classroom \
--rgw-zone=us-east-2 --endpoints=http://serverf:80 --access-key=replication \
--secret-key=secret --default
{
  "id": "3879a186-cc0c-4b42-8db1-7624d74951b0",
  "name": "us-east-2",
  "domain_root": "us-east-2.rgw.meta:root",
  "control_pool": "us-east-2.rgw.control",
  ...output omitted...
  "system_key": {
    "access_key": "replication",
    "secret_key": "secret"
  },
  "placement_pools": [
    {
      "key": "default-placement",
      "val": {
        "index_pool": "us-east-2.rgw.buckets.index",
        "storage_classes": {
          "STANDARD": {
            "data_pool": "us-east-2.rgw.buckets.data"
          }
        },
        "data_extra_pool": "us-east-2.rgw.buckets.non-ec",
        "index_type": 0
      }
    }
  ],
  "realm_id": "9eef2ff2-5fb1-4398-a69b-eeb3d9610638",
  "notif_pool": "us-east-2.rgw.log:notif"
}
```

- 5.3. Commit the site configuration.

```
[ceph: root@serverf /]# radosgw-admin period update --commit --rgw-zone=us-east-2
{
  "id": "7cdc83cf-69d8-478e-b625-d5250ac4435b",
  "epoch": 2,
  "predecessor_uuid": "031c3bfb-6626-47ef-a523-30b4313499d9",
  "sync_status": [],
  "period_map": {
    "id": "7cdc83cf-69d8-478e-b625-d5250ac4435b",
    "zonegroups": [
      {
        "id": "d3524ffb-8a3c-45f1-ac18-23db1bc99071",
        "name": "classroom",
      }
    ]
  }
}
```

```

    "api_name": "classroom",
    "is_master": "true",
    "endpoints": [
        "http://serverc:80"
    ],
    "hostnames": [],
    "hostnames_s3website": [],
    "master_zone": "4f1863ca-1fca-4c2d-a7b0-f693ddd14882",
    "zones": [
        {
            "id": "3879a186-cc0c-4b42-8db1-7624d74951b0",
            "name": "us-east-2",
            "endpoints": [
                "http://serverf:80"
            ],
            ...
            ...
        },
        {
            "id": "4f1863ca-1fca-4c2d-a7b0-f693ddd14882",
            "name": "us-east-1",
            "endpoints": [
                "http://serverc:80"
            ],
            ...
            ...
        }
    ],
    ...
    ...
    "master_zonename": "d3524ffb-8a3c-45f1-ac18-23db1bc99071",
    "master_zone": "4f1863ca-1fca-4c2d-a7b0-f693ddd14882",
    ...
    ...
    "realm_id": "9eef2ff2-5fb1-4398-a69b-eeb3d9610638",
    "realm_name": "cl260",
    "realm_epoch": 2
}

```

5.4. Update the zone name in the configuration database.

```
[ceph: root@serverf /]# ceph config set client.rgw rgw_zone us-east-2
```

- ▶ 6. Create a new RADOS Gateway service called `cl260-2` in the `cl260` realm and `us-east-2` zone, and with a single RGW daemon on the `serverf` node. Verify that the RGW daemon is up and running. View the period associated with the current configuration. Verify the sync status of the site.

6.1. Create the RADOS Gateway service on the `serverf` node.

```
[ceph: root@serverf /]# ceph orch apply rgw cl260-2 --realm=cl260 \
--zone=us-east-2 --placement="1 se rverf.lab.example.com"
Scheduled rgw.east update...
[ceph: root@serverf /]# ceph orch ps --daemon-type rgw
NAME                      HOST                      STATUS          REFRESHED   AGE
PORTS  ...
rgw.east.serverf.zgkgem  serverf.lab.example.com  running (37m)  6m ago     37m
*:80  ...
```

6.2. View the period associated with the current configuration.

```
[ceph: root@serverf /]# radosgw-admin period get-current
{
    "current_period": "7cdc83cf-69d8-478e-b625-d5250ac4435b"
}
```

6.3. Verify the sync status.

```
[ceph: root@serverf /]# radosgw-admin sync status
realm 9eef2ff2-5fb1-4398-a69b-eeb3d9610638 (cl260)
zonegroup d3524ffb-8a3c-45f1-ac18-23db1bc99071 (classroom)
zone 3879a186-cc0c-4b42-8db1-7624d74951b0 (us-east-2)
metadata sync syncing
    full sync: 0/64 shards
    incremental sync: 64/64 shards
metadata is caught up with master
data sync source: 4f1863ca-1fca-4c2d-a7b0-f693ddd14882 (us-east-1)
    syncing
    full sync: 0/128 shards
    incremental sync: 128/128 shards
data is caught up with source
```

- 7. Exit and close the second terminal. Return to workstation as the student user.

```
[root@serverf ~]# exit
[admin@serverf ~]$ exit
[student@workstation ~]$ exit
```

```
[root@serverc ~]# exit
[admin@serverc ~]$ exit
[student@workstation ~]$
```

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-multisite
```

This concludes the guided exercise.

▶ Lab

Providing Object Storage Using a RADOS Gateway

In this lab, you will configure Ceph to provide object storage to clients.

Outcomes

You should be able to deploy and configure the Ceph RADOS Storage service.

Before You Begin

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

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

This command ensures that the lab environment is created and ready for the lab exercise.

Instructions

1. Log in to `serverc` as the `admin` user. Create a realm called `prod` and set it as the default. Create a master zone group called `us-west` with the endpoint set to `http://serverc:8080`. Set the zone group as the default.
2. Create a master zone called `us-west-1` and set the endpoint to `http://serverc:8080`. Set the zone as the default. Use `admin` as the access key and `secure` as the secret key.
3. Create a system user called `admin.user` with `admin` as the access key and `secure` as the secret key.
4. Commit the configuration. Save the period ID value in the `/home/admin/period-id.txt` file on the `serverc` node. Do not include quotes, double quotes, or characters other than the period ID in UUID format.
5. Deploy a RADOS Gateway service called `prod-object` with two instances running on port 8080, one on the `serverc` node and the second on the `servere` node.
6. Return to `workstation` as the `student` user.

Evaluation

Grade your work by running the `lab grade object-review` command from your `workstation` machine. Correct any reported failures and rerun the script until successful.

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

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-review
```

This concludes the lab.

► Solution

Providing Object Storage Using a RADOS Gateway

In this lab, you will configure Ceph to provide object storage to clients.

Outcomes

You should be able to deploy and configure the Ceph RADOS Storage service.

Before You Begin

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

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

This command ensures that the lab environment is created and ready for the lab exercise.

Instructions

1. Log in to `serverc` as the `admin` user. Create a realm called `prod` and set it as the default. Create a master zone group called `us-west` with the endpoint set to `http://serverc:8080`. Set the zone group as the default.

- 1.1. Log in to `serverc` as the `admin` user and use `sudo` to run the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@serverc
[admin@serverc ~]$ sudo cephadm shell
[ceph: root@serverc /]#
```

- 1.2. Create a realm called `prod` and set it as the default.

```
[ceph: root@serverc /]# radosgw-admin realm create --rgw-realm=prod --default
```

- 1.3. Create a master zone group called `us-west`.

```
[ceph: root@serverc /]# radosgw-admin zonegroup create --rgw-zonegroup=us-west \
--endpoints=http://serverc:8080 --master --default
```

2. Create a master zone called `us-west-1` and set the endpoint to `http://serverc:8080`. Set the zone as the default. Use `admin` as the access key and `secure` as the secret key.

```
[ceph: root@serverc /]# radosgw-admin zone create --rgw-zonegroup=us-west \
--rgw-zone=us-west-1 --endpoints=http://serverc:8080 --master \
--access-key=admin --secret=secure --default
```

3. Create a system user called `admin.user` with `admin` as the access key and `secure` as the secret key.

```
[ceph: root@serverc /]# radosgw-admin user create --uid="admin.user" --system \
--display-name="Admin User" --access-key=admin --secret=secure
```

4. Commit the configuration. Save the period ID value in the `/home/admin/period-id.txt` file on the `serverc` node. Do not include quotes, double quotes, or characters other than the period ID in UUID format.

```
[ceph: root@serverc /]# radosgw-admin period update --commit
[ceph: root@serverc /]# radosgw-admin period get-current
{
    "current_period": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee"
}
[ceph: root@serverc /]# exit
[admin@serverc ~]$ cat /home/admin/period-id.txt
aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee
```

5. Deploy a RADOS Gateway service called `prod-object` with two instances running on port 8080, one on the `serverc` node and the second on the `servere` node.

```
[admin@serverc ~]$ sudo cephadm shell
[ceph: root@serverc /]# ceph orch apply rgw prod-object --realm=prod \
--zone=us-west-1 --port 8080 \
--placement="2 serverc.lab.example.com servere.lab.example.com"
```

6. Return to `workstation` as the `student` user.

```
[ceph: root@serverc /]# exit
[admin@serverc ~]$ exit
[student@workstation ~]$
```

Evaluation

Grade your work by running the `lab grade object-review` command from your `workstation` machine. Correct any reported failures and rerun the script until successful.

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

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-review
```

This concludes the lab.

Summary

In this chapter, you learned:

- The RADOS Gateway is a service that connects to a Red Hat Ceph Storage cluster, and provides object storage to applications using a REST API.
- You can deploy the RADOS Gateway by using the Ceph orchestrator command-line interface or by using a service specification file.
- You can use HAProxy and keepalived services to load balance the RADOS Gateway service.
- The RADOS Gateway supports multisite configuration, which allows RADOS Gateway objects to be replicated between separate Red Hat Ceph Storage clusters.
- Objects written to a RADOS Gateway for one zone are replicated to all other zones in the zone group.
- Metadata and configuration updates must occur in the master zone of the master zone group.

Chapter 9

Accessing Object Storage Using a REST API

Goal

Configure the RADOS Gateway to provide access to object storage using REST APIs.

Objectives

- Configure the RADOS Gateway to provide access to object storage compatible with the Amazon S3 API, and manage objects stored using that API.
- Configure the RADOS Gateway to provide access to object storage compatible with the Swift API, and manage objects stored using that API.

Sections

- Providing Object Storage Using the Amazon S3 API (and Guided Exercise)
- Providing Object Storage Using the Swift API (and Guided Exercise)

Lab

Accessing Object Storage Using a REST API

Providing Object Storage Using the Amazon S3 API

Objectives

After completing this section, you should be able to configure the RADOS Gateway to provide access to object storage compatible with the Amazon S3 API, and manage objects stored using that API.

Amazon S3 API in RADOS Gateway

The Amazon S3 API enables developers to manage object storage resources using an Amazon S3 compatible interface. Applications implemented with the S3 API can inter-operate with other S3-compatible object storage services besides the RADOS Gateway, and migrate storage from other locations to your Ceph storage cluster. In a hybrid cloud environment, you can configure your applications to use different authentication keys, regions, and vendor services to mix private enterprise and public cloud resources and storage locations seamlessly using the same API.

The Amazon S3 interface defines the namespace in which objects are stored as a *bucket*. To access and manage objects and buckets using the S3 API, applications use RADOS Gateway users for authentication. Each user has an *access key* that identifies the user and a *secret key* that authenticates the user.

There are object and metadata size limits to consider when using the Amazon S3 API:

- An object size is between a minimum of OB and a maximum of 5 TB.
- The maximum size is 5GB in a single upload operation.
- Upload objects larger than 100MB by using the *multipart upload* capability.
- The maximum metadata size is 16,000 bytes in a single HTTP request.

Creating a User for the Amazon S3 API

Create the needed RADOS Gateway users first, then use them to authenticate Amazon S3 clients with the gateway. Use the `radosgw-admin user create` command to create RADOS Gateway users.

When creating RADOS Gateway users, both the `--uid` and `--display-name` options are required, and specify a unique account name and a human friendly display name. Use the `--access-key` and `--secret` options to specify a custom AWS account and secret key for the RADOS user.

```
[ceph: root@node /]# radosgw-admin user create --uid=testuser \
--display-name="Test User" --email= test@example.com \
--access-key=12345 --secret=67890
...output omitted...
"keys": [
{
    "user": "testuser",
    "access_key": "12345",
```

```
        "secret_key": "67890"
    }
...output omitted...
```

If the access key and secret key are not specified, the `radosgw-admin` command automatically generates them and displays them in the output.

```
[ceph: root@node /]# radosgw-admin user create --uid=s3user \
--display-name="Amazon S3 API user"
...output omitted...
"keys": [
{
    "user": "s3user",
    "access_key": "8PI2D9ARWNGJI99K8TOS",
    "secret_key": "brKaQdhyR022znVVVdDLuAEafRjbrAorr0GoXN1"
}
...output omitted...
```



Important

When the `radosgw-admin` command automatically generates the access key and secret, either key might include a JSON escape character (\). Clients might not handle this character correctly. Either regenerate or manually specify the keys to avoid this issue.

Managing Ceph Object Gateway Users

To regenerate only the secret key of an existing user, use the `radosgw-admin key create` command with the `--gen-secret` option.

```
[ceph: root@node /]# radosgw-admin key create --uid=s3user \
--access-key="8PI2D9ARWNGJI99K8TOS" --gen-secret
...output omitted...
"keys": [
{
    "user": "s3user",
    "access_key": "8PI2D9ARWNGJI99K8TOS",
    "secret_key": "MFVxrGNMBjK007JscLFbEyrEmJFnLl43PHSwpLC"
}
...output omitted...
```

To add an access key to an existing user, use the `--gen-access-key` option. Creating additional keys is convenient for granting the same user access to multiple applications that require different or unique keys.

```
[ceph: root@node /]# radosgw-admin key create --uid=s3user --gen-access-key
{
...output omitted...
"keys": [
{
    "user": "s3user",
    "access_key": "8PI2D9ARWNGJI99K8TOS",
```

```

        "secret_key": "MFVxrGNMBjK007JscLFbEyrEmJFnLl43PHSwpLC"
    },
    {
        "user": "s3user",
        "access_key": "GPYJGPS0NURDY7SG0LL0",
        "secret_key": "T7jcG5YgEqqPxWMkdCTBsY0DM3rgI0mqkmtjRlcX"
    }
...output omitted...

```

To remove an access key and related secret key from a user, use the `radosgw-admin key rm` command with the `--access-key` option. This is useful for removing single application access without impacting access with other keys.

```

[ceph: root@node /]# radosgw-admin key rm --uid=s3user \
--access-key=8PI2D9ARWNGJI99K8TOS
{
    "keys": [
        {
            "user": "s3user",
            "access_key": "GPYJGPS0NURDY7SG0LL0",
            "secret_key": "T7jcG5YgEqqPxWMkdCTBsY0DM3rgI0mqkmtjRlcX"
        }
    ]
}

```

Temporarily disable and enable RADOS Gateway users by using the `radosgw-admin user suspend` and `radosgw-admin user enable` commands. When suspended, a user's subusers are also suspended and unable to interact with the RADOS Gateway service.

You can modify user information such as email, display name, keys and access control level. The access control levels are: `read`, `write`, `readwrite`, and `full`. The `full` access level includes the `readwrite` level and the access control management capability.

```
[ceph: root@node /]# radosgw-admin user modify --uid=johndoe --access=full
```

To remove a user and also delete their objects and buckets, use the `--purge-data` option.

```
[ceph: root@node /]# radosgw-admin user rm --uid=s3user --purge-data
```

Set *quotas* to limit the amount of storage a user or bucket can consume. Set the quota parameters first, then enable the quota. To disable a quota, set a negative value for the quota parameter.

Bucket quotas apply to all buckets owned by a specific UUID, regardless of the user accessing or uploading to those buckets.

In this example, the quota for the `app1` user is set to a maximum of 1024 objects. The user quota is then enabled.

```

[ceph: root@node /]# radosgw-admin quota set --quota-scope=user --uid=app1 \
--max-objects=1024
[ceph: root@node /]# radosgw-admin quota enable --quota-scope=user --uid=app1

```

Similarly, apply quotas to buckets by setting the `--quota-scope` option to `bucket`. In this example, the `loghistory` bucket is set for a maximum size of 1024 bytes.

```
[ceph: root@node /]# radosgw-admin quota set --quota-scope=bucket \
--uid=loghistory --max-objects=1024B
[ceph: root@node /]# radosgw-admin quota enable --quota-scope=bucket \
--uid=loghistory
```

Global quotas affect all buckets in the cluster.

```
[ceph: root@node /]# radosgw-admin global quota set --quota-scope bucket \
--max-objects 2048
[ceph: root@node /]# radosgw-admin global quota enable --quota-scope bucket
```

To be implemented in a zone and period configuration, use the `radosgw-admin period update --commit` command to commit the change. Alternately, restart the RGW instances to implement the quota.

Use the following commands to retrieve user information and statistics:

Action	Command
Retrieve user information.	<code>radosgw-admin user info --uid=uid</code>
Retrieve user statistics.	<code>radosgw-admin user stats --uid=uid</code>

Storage administrators monitor usage statistics to determine the storage consumption or user bandwidth usage. Monitoring can also help find inactive applications or inappropriate user quotas.

Use the `radosgw-admin user stats` and `radosgw-admin user info` commands to view user information and statistics.

```
[ceph: root@node /]# radosgw-admin user info --uid=uid
[ceph: root@node /]# radosgw-admin user stats --uid=uid
```

Use the `radosgw-admin usage show` command to show the usage statistics of a user between specific dates.

```
[ceph: root@node /]# radosgw-admin usage show --uid=uid --start-date=start \
--end-date=end
```

View the statistics for all users by using the `radosgw-admin usage show` command. Use these overall statistics to help understand object storage patterns and plan the deployment of new instances for scaling the RADOS gateway service.

```
[ceph: root@node /]# radosgw-admin usage show --show-log-entries=false
```

Accessing S3 Objects Using RADOS Gateway

The Amazon S3 API supports several bucket URL formats, including either `http://server.example.com/bucket/` or `http://bucket.server.example.com/`. Some clients, such as the `s3cmd` command, only support the second URL format. RADOS Gateway does not enable this format by default. To enable the second URL format, set the `rgw_dns_name` parameter for your DNS suffix.

```
[ceph: root@node /]# ceph config set client.rgw rgw_dns_name dns_suffix
```

where *dns_suffix* is the fully qualified domain name to be used to create your bucket's name.

In addition to configuring `rgw_dns_name`, you must configure your DNS server with a wildcard DNS record for that domain that points to the RADOS Gateway IP address. Syntax for implementing wildcard DNS entries varies for different DNS servers.

Using Amazon S3 API Clients

Commands from the `awscli` package support bucket and object management by using the S3 API. You can create a bucket by using the `aws mb` command. This example command creates the bucket called `demobucket`.

```
[ceph: root@node /]# aws s3 mb s3://demobucket
```

Upload objects to a bucket using the `aws cp` command. This example command uploads an object called `demoobject` to the `demobucket` bucket, using the local file `/tmp/demoobject`.

```
[ceph: root@node /]# aws --acl=public-read-write s3 cp /tmp/demoobject \
s3://demobucket/demoobject
```

The `radosgw-admin` command supports bucket operations, such as the `radosgw-admin bucket list` and the `radosgw-admin bucket rm` commands.



Note

There are multiple S3 public clients available, such as `awscli`, `cloudberry`, `cyberduck`, and `curl`, which provide access to object storage supporting the S3 API.

S3 Bucket Versioning, Life Cycle, and Policies

S3 bucket versioning supports storing multiple versions of an object in a bucket. RADOS Gateway supports versioned buckets, adding a version identifier to objects uploaded to the bucket. The bucket owner configures the bucket as a versioned bucket.

RADOS Gateway also supports S3 API object expiration by using rules defined for a set of bucket objects. Each rule has a prefix, which selects the objects, and a number of days after which objects become unavailable.

RADOS Gateway supports only a subset of the Amazon S3 API policy language applied to buckets. No policy support is available for users, groups, or roles. Bucket policies are managed through standard S3 operations rather than using the `radosgw-admin` command.

An S3 policy uses JSON format to define the following elements:

- The `Resource` key defines the resources which permissions the policy modifies. The policy uses the Amazon Resource Name (ARN) associated with the resources to identify it.
- The `Actions` key defines the operations allowed or denied for a resource. Each resource has a set of operations available.
- The `Effect` key indicates if the policy allows or denies the action previously defined for a resource. By default, a policy denies the access to a resource.

- The `Principal` key defines the user to whom the policy allows or denies access to a resource.

```
{
  "Version": "2021-03-10",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": ["arn:aws:iam::testaccount:user/testuser"]
      },
      "Action": "s3>ListBucket",
      "Resource": [
        "arn:aws:s3:::testbucket"
      ]
    }
  ]
}
```

Support for S3 MFA Delete

The RADOS Gateway service supports S3 MFA Delete by using Time-based, One-time Password (TOTP) passwords as an authentication factor. This feature adds security against inappropriate and unauthorized data removal. You can configure buckets to require a TOTP one-time token in addition to standard S3 authentication to delete data. To delete an object, the bucket owner must include a request header with the authentication device's serial number and the authentication code in HTTPS to permanently delete an object version or change the versioning state of the bucket. Without the header, the request fails.

Creating New IAM Policies and Roles Using REST APIs

REST APIs for IAM (identity and access management) roles and user policies are now available in the same namespace as S3 APIs and can be accessed using the same endpoint as S3 APIs in the Ceph Object Gateway. This feature allows end users to create new IAM policies and roles by using REST APIs.

Support for Amazon S3 Resources in Ceph Object Gateway

AWS provides the Security Token Service (STS) to allow secure federation with existing OpenID Connect/ OAuth 2.0 compliant identity services such as Keycloak. STS is a standalone REST service that provides temporary tokens for an application or user to access an S3 endpoint after the user authenticates against an identity provider. Previously, users without permanent Amazon Web Services (AWS) credentials could not access S3 resources through Ceph Object Gateway.

Ceph Object Gateway implements a subset of STS APIs that provide temporary credentials for identity and access management. These temporary credentials can be used to make subsequent S3 calls, which will be authenticated by the STS engine in RGW. Permissions of the temporary credentials can be further restricted via an IAM policy passed as a parameter to the STS APIs. Ceph Object Gateway supports STS `AssumeRoleWithWebIdentity`.



References

For more information, refer to the *Static Web Hosting* section in the *Red Hat Ceph Storage 5 Object Gateway Guide* at

https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/object_gateway_guide/basic-configuration#static-web-hosting

For more information, refer to the *Security* section in the *Red Hat Ceph Storage 5 Object Gateway Guide* at

https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/object_gateway_guide/security

For more information, refer to the *Ceph Object Gateway and the S3 API* chapter in the *Red Hat Ceph Storage 5 Developer Guide* at

https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/developer_guide/ceph-object-gateway-and-the-s3-api

► Guided Exercise

Providing Object Storage Using the Amazon S3 API

In this exercise, you will configure the RADOS Gateway and access the gateway using the Amazon S3 API.

Outcomes

You should be able to configure the Ceph Object Gateway to allow access to Ceph object storage via the Amazon S3 API.

Before You Begin

```
[student@workstation ~]$ lab start api-s3
```

Instructions

- 1. Log in to `clienta` as the `admin` user.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$
```

- 2. Create an Amazon S3 API user called `operator`. Use `12345` as the S3 access key and `67890` as the secret key.

```
[admin@clienta ~]$ sudo cephadm shell -- radosgw-admin user create \
--uid="operator" --display-name="S3 Operator" --email="operator@example.com" \
--access-key="12345" --secret="67890"
{
    "user_id": "operator",
    "display_name": "S3 Operator",
    "email": "operator@example.com",
    "suspended": 0,
    "max_buckets": 1000,
    "subusers": [],
    "keys": [
        {
            "user": "operator",
            "access_key": "12345",
            "secret_key": "67890"
        }
    ],
    "swift_keys": [],
    "caps": [],
    "op_mask": "read, write, delete",
    ...output omitted...
```

- ▶ 3. Configure the AWS CLI tool to use operator credentials. Enter 12345 as the access key and 67890 as the secret key.

```
[admin@clienta ~]$ aws configure --profile=ceph
AWS Access Key ID [None]: 12345
AWS Secret Access Key [None]: 67890
Default region name [None]: Enter
Default output format [None]: Enter
```

- ▶ 4. Create a bucket called testbucket. List the created bucket.

```
[admin@clienta ~]$ aws --profile=ceph \
--endpoint=http://serverc:80 s3 mb s3://testbucket
make_bucket: testbucket
[admin@clienta ~]$ aws --profile=ceph --endpoint=http://serverc:80 s3 ls
2021-10-05 21:51:37 testbucket
```

- ▶ 5. Create a 10 MB file called 10MB.bin. Upload the file to testbucket.

```
[admin@clienta ~]$ dd if=/dev/zero of=/tmp/10MB.bin bs=1024K count=10
10+0 records in
10+0 records out
10485760 bytes (10 MB, 10 MiB) copied, 0.00894909 s, 1.2 GB/s
[admin@clienta ~]$ aws --profile=ceph --endpoint=http://serverc:80 \
--acl=public-read-write s3 cp /tmp/10MB.bin s3://testbucket/10MB.bin
upload: ../../tmp/10MB.bin to s3://testbucket/10MB.bin
```

- ▶ 6. Verify that the S3 object is accessible using path-style URLs.

```
[admin@clienta ~]$ wget -O /dev/null http://serverc:80/testbucket/10MB.bin
--2021-10-05 22:03:37-- http://serverc/testbucket/10MB.bin
Resolving serverc (serverc)... 172.25.250.12
Connecting to serverc (serverc)|172.25.250.12|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10485760 (10M) [application/octet-stream]

Saving to: '/dev/null'

10MB.bin 100%[=====] 10.00M --.-KB/s in 0.02s
```

- ▶ 7. Use the radosgw-admin command to view the metadata of the testbucket bucket.

```
[admin@clienta ~]$ sudo cephadm shell -- radosgw-admin bucket list
[
  "testbucket"
]

[admin@clienta ~]$ sudo cephadm shell -- radosgw-admin metadata \
  get bucket:testbucket
{
  "key": "bucket:testbucket",
```

```
"ver": {
    "tag": "_2d3Y6puJve1TnYs0pwHc0Go",
    "ver": 1
},
"mtime": "2021-10-06T01:51:37.514627Z",
"data": {
    "bucket": {
        "name": "testbucket",
        "marker": "cb16a524-d938-4fa2-837f-d1f2011676e2.54360.1",
        "bucket_id": "cb16a524-d938-4fa2-837f-d1f2011676e2.54360.1",
        "tenant": "",
        "explicit_placement": {
            "data_pool": "",
            "data_extra_pool": "",
            "index_pool": ""
        }
    },
    "owner": "operator",
    "creation_time": "2021-10-06T01:51:37.498002Z",
    "linked": "true",
    "has_bucket_info": "false"
}
}
```

- 8. Return to `workstation` as the `student` user.

```
[admin@clienta ~]$ exit
[student@workstation ~]$
```

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 api-s3
```

This concludes the guided exercise.

Providing Object Storage Using the Swift API

Objectives

After completing this section, you should be able to configure the RADOS Gateway to provide access to object storage compatible with the Swift API, and manage objects stored using that API.

OpenStack Swift Support in a RADOS Gateway

The OpenStack Swift API enables developers to manage object storage resources using a Swift compatible interface. Applications implemented with the S3 API can inter-operate with other Swift-compatible object storage services besides the RADOS Gateway, and migrate storage from other locations to your Ceph storage cluster. In a hybrid cloud environment, you can configure your applications to mix private enterprise OpenStack or standalone Swift resources and public cloud OpenStack resources and storage locations seamlessly using the same API.

The OpenStack Swift API is an alternative to the Amazon S3 API to access objects stored in the Red Hat Ceph Storage cluster through a RADOS Gateway. There are important differences between the OpenStack Swift and Amazon S3 APIs.

OpenStack Swift refers to the namespace in which objects are stored as a *container*.

The OpenStack Swift API has a different user model than the Amazon S3 API. To authenticate with a RADOS Gateway using the OpenStack Swift API, you must configure *subusers* for your RADOS Gateway user accounts.

Creating a Subuser for OpenStack Swift

The Amazon S3 API authorization and authentication model has a single-tier design. A single user account might have multiple access keys and secrets, which the user can use to provide different types of access.

The OpenStack Swift API, however, has a multi-tier design, built to accommodate *tenants* and assigned *users*. A Swift tenant owns the storage and its containers used by a service. Swift users are assigned to the service and have different levels of access to the storage owned by the tenant.

To accommodate the OpenStack Swift API authentication and authorization model, RADOS Gateway has the concept of subusers. This model allows Swift API tenants to be handled as RADOS Gateway users, and Swift API users to be handled as RADOS Gateway subusers. The Swift API `tenant:user` tuple maps to RADOS Gateway authentication system as a `user:subuser`. A subuser is created for each Swift user, and it is associated with a RADOS Gateway user and an access key.

To create a subuser, use the `radosgw-admin subuser create` command as follows:

```
[ceph: root@node /]# radosgw-admin subuser create --uid=username \
--subuser=username:swift --access=full
```

The `--access` option sets the user permissions (read, write, read/write, full), and `--uid` specifies the existing associated RADOS Gateway user. Use the `radosgw-admin key create` command

with the `--key-type=swift` option to create a Swift authentication key associated with the subuser.

```
[ceph: root@node /]# radosgw-admin key create --subuser=username:swift \
--key-type=swift
```

When a Swift client communicates with a RADOS Gateway, then the latter acts as both the data server and the Swift authentication daemon (using the `/auth` URL path). The RADOS Gateway supports both Internal Swift (version 1.0) and OpenStack Keystone (version 2.0) authentication. The secret specified using `-K` is the secret created with the Swift key.



Note

Ensure your command-line parameters are not being overridden or influenced by any operating system environment variables. If you are using Auth version 1.0, then use the `ST_AUTH`, `ST_USER`, and `ST_KEY` environment variables. If you are using Auth version 2.0, then use the `OS_AUTH_URL`, `OS_USERNAME`, `OS_PASSWORD`, `OS_TENANT_NAME`, and `OS_TENANT_ID` environment variables.

In the Swift API, a container is a collection of objects. An object in the Swift API is a *binary large object* (blob) of data stored in Swift.

To verify RADOS Gateway accessibility using the Swift API, use the `swift post` command to create a container.

```
[root@node ~]$ swift -A http://host/auth -U username:swift -K secret \
post container-name
```

To upload a file to a container, use the `swift upload` command:

```
[root@node ~]$ swift -A http://host/auth -U username:swift -K secret \
upload container-name file-name
```

If you use the absolute path to define the file location, then the object's name contains the path to file, including the slash character `/`. For example, the following command uploads the `/etc/hosts` file to the services bucket.

```
[root@node ~]$ swift -A http://host/auth/ -U user:swift -K secret \
upload services /etc/hosts
```

In this example, the uploaded object name is `etc/hosts`. You can define the object name by using the `--object-name` option.

To download a file, use the `download` command.

```
[root@node ~]$ swift -A http://host/auth -U username:swift -K secret \
download container-name object-name
```

Managing Ceph Object Gateway Subusers

Modify the access level for a subuser by using the `radosgw-admin subuser modify` command. The access level sets the user permissions to read, write, read/write, or full.

```
[root@node ~]$ radosgw-admin subuser modify --subuser=uid:_subuserid_ \
--access=access-level
```

Remove subusers using the `radosgw-admin subuser rm` command. The `--purge-data` option purges all data associated to the subuser and the `--purge-keys` option purges all the subuser keys.

```
[root@node ~]$ radosgw-admin subuser rm --subuser=uid:_subuserid_ \
[--purge-data] [--purge-keys]
```

You can manage the subuser keys by using the `radosgw-admin key` command. This example creates a subuser key.

```
[root@node ~]$ radosgw-admin key create --subuser=uid:_subuserid_ \
--key-type=swift [--access-key=access-key] [--secret-key=secret-key]
```

The key-type option only admits the values `swift` or `s3`. Use the `--access-key` option if you want to manually specify an S3 access key, and use the `--secret-key` option if you want to manually specify an S3 or Swift secret key. If the access key and secret key are not specified, the `radosgw-admin` command automatically generates them and displays them in the output. Alternately, use the `--gen-access-key` option to generate only a random access key, or the `--gen-secret` option to generate only a random secret.

To remove a subuser key, use the `radosgw-admin key rm` command.

```
[root@node ~]$ radosgw-admin key rm --subuser=uid:subuserid
```

Swift Container Object Versioning and Expiration

The Swift API supports object versioning for a container, providing the ability to keep multiple versions of an object in a container. Object versioning avoids accidental object overwrites and deletions, and archives previous objects versions. The Swift API creates a new object version in the versioned container only when the object content changes.

To enable versioning on a container, set the value of a container flag to be the name of the container which stores the versions. Set the flag when creating new containers or by updating the metadata on existing containers.



Note

You should use a different archive container for each container to be versioned. Enabling versioning on an archive container is not recommended.

The Swift API supports two header keys for this versioning flag, either `X-History-Location` or `X-Versions-Location`, which determines how the Swift API handles object `DELETE` operations.

With the `X-History-Location` flag set, you receive a `404 Not Found` error after deleting the object inside the container. Swift copies the object to the archive container and removes the original copy in the versioned container. You can recover the object from the archive container.

With the `X-Versions-Location` flag set, Swift removes the current object version in the versioned container. Then, Swift copies the most recent object version in the archive container to the versioned container, and deletes that most recent object version from the archive container. To completely remove an object from a versioned container with the `X-Versions-Location` flag set, you must remove the object as many times as there are object versions available in the archive container.

Set only one of these flags at the same time on an OpenStack Swift container. If the container's metadata contains both flags, then a `400 Bad Request` error is issued.

RADOS Gateway supports the Swift API object versioning feature. To activate this feature in the RADOS Gateway, set `rgw_swift_versioning_enabled` to `true` in the `[client.radosgw.radosgw-name]` section in the `/etc/ceph/ceph.conf` configuration file.

RADOS Gateway also supports using the `X-Delete-At` and `X-Delete-After` headers when adding objects using the Swift API. At the time specified by the header, RADOS Gateway stops serving that object, and removes it shortly after.

Multitenancy Support in Swift

The OpenStack Swift API supports the use of tenants to isolate buckets and users. The Swift API associates every new bucket created by a user with a tenant. This feature allows you to use the same bucket name on different tenants, because tenants isolate resources. For backward compatibility, the Swift API uses a generic, no-name tenant for containers which don't have an associated tenant.

Configure Swift API tenants in RADOS Gateway with the `radosgw-admin` command. This command requires a tenant to create the user provided using the `--tenant` option.

```
[root@node ~]$ radosgw-admin user create --tenant testtenant \
    --uid testuser --display-name "Swift User" --subuser testswift:testuser \
    --key-type swift --access full
```

Any further reference to the subuser must include the tenant.

```
[root@node ~]$ radosgw-admin --subuser 'testtenant$testswift:testuser' \
    --key-type swift --secret redhat
```



References

For more information, refer to the *Configuration Reference* chapter in the *Object Gateway Configuration and Administration Guide* at
https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/object_gateway_guide/index#rgw-configuration-reference-rgw

► Guided Exercise

Providing Object Storage Using the Swift API

In this exercise, you will configure the RADOS Gateway and access the gateway using the Swift API.

Outcomes

You should be able to configure the Ceph Object Gateway to allow access to Ceph object storage via the Swift API.

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 api-swift
```

This command confirms that the hosts required for this exercise are accessible. It installs the `awscli` package on `clienta` and creates the `operator` user and the `testbucket` bucket.

Instructions

- ▶ 1. Log in to `clienta` as the `admin` user.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$
```

- ▶ 2. Create a Swift subuser called `operator:swift` on the Ceph Object Gateway for accessing object storage using the Swift API.

You are creating a new subuser attached to the `operator` user that was created by the `lab start` command. The `operator` user can currently access Ceph object storage using the S3 API. With this additional configuration, the `operator` user can create objects and buckets using the S3 API, and then use the Swift API to manage the same objects.

Use `opswif` as the Swift secret key. Assign full permissions to the user.

```
[admin@clienta ~]$ sudo cephadm shell -- radosgw-admin subuser create \
--uid="operator" --subuse r="operator:swift" --access="full" --secret="opswif"
{
    "user_id": "operator",
    "display_name": "S3 Operator",
    "email": "operator@example.com",
    "suspended": 0,
    "max_buckets": 1000,
    "subusers": [
        {
            "id": "operator:swift",
            "key": "opswif"
        }
    ]
}
```

```
        "permissions": "full-control"
    }
],
"keys": [
{
    "user": "operator",
    "access_key": "12345",
    "secret_key": "67890"
},
],
"swift_keys": [
{
    "user": "operator:swift",
    "secret_key": "opswift"
}
],
"caps": [],
"op_mask": "read, write, delete",
"default_placement": "",
"default_storage_class": "",
...output omitted...
```

- 3. Install the `swift` client on `clienta`. Verify access using the `operator:swift` subuser credentials. Use `-K` to specify the Swift secret key for the user.

3.1. Install the `swift` client on `clienta`.

```
[admin@clienta ~]$ sudo pip3 install --upgrade python-swiftclient
...output omitted...
```

3.2. Verify the bucket status. Your output might be different in your lab environment.

```
[admin@clienta ~]$ swift -A http://serverc:80/auth/1.0 \
-U operator:swift -K opswift stat
                        Account: v1
                        Containers: 1
                        Objects: 0
                        Bytes: 0
Objects in policy "default-placement-bytes": 0
Bytes in policy "default-placement-bytes": 0
Containers in policy "default-placement": 1
Objects in policy "default-placement": 0
Bytes in policy "default-placement": 0
                        X-Timestamp: 1633518961.75503
                        X-Account-Bytes-Used-Actual: 0
                        X-Trans-Id: tx000000000000000000000000a-00615d8571-
ace6-default
                        X-Openstack-Request-Id: tx000000000000000000000000a-00615d8571-
ace6-default
                        Accept-Ranges: bytes
                        Content-Type: text/plain; charset=utf-8
                        Connection: Keep-Alive
```

▶ 4. Create and list the Swift containers.

4.1. List the currently available containers.

```
[admin@clienta ~]$ swift -A http://serverc:80/auth/1.0 \
-U operator:swift -K opswift list
testbucket
```

4.2. Create a Swift container called testcontainer.

```
[admin@clienta ~]$ swift -A http://serverc:80/auth/1.0 \
-U operator:swift -K opswift post testcontainer
[admin@clienta ~]$ swift -A http://serverc:80/auth/1.0 \
-U operator:swift -K opswift list
testbucket
testcontainer
```

▶ 5. Create a 10 MB file called swift.dat. Upload the file to testcontainer.

```
[admin@clienta ~]$ dd if=/dev/zero of=/tmp/swift.dat bs=1024K count=10
10+0 records in
10+0 records out
10485760 bytes (10 MB) copied, 0.00926399 s, 1.1 GB/s
[admin@clienta ~]$ swift -A http://serverc:80/auth/1.0 \
-U operator:swift -K opswift upload testcontainer /tmp/swift.dat
tmp/swift.dat
```

▶ 6. View the statistics for testcontainer.

```
[admin@clienta ~]$ swift -A http://serverc:80/auth/1.0 \
-U operator:swift -K opswift stat testcontainer
      Account: v1
      Container: testcontainer
      Objects: 1
      Bytes: 10485760
      Read ACL:
      Write ACL:
      Sync To:
      Sync Key:
      X-Timestamp: 1615999768.12521
X-Container-Bytes-Used-Actual: 10485760
      X-Storage-Policy: default-placement
      X-Storage-Class: STANDARD
      Last-Modified: Wed, 06 Oct 2021 16:50:13 GMT
      X-Trans-Id: tx00000000000000000000000000000013-0060523354-853f-default
X-Openstack-Request-Id: tx00000000000000000000000000000013-0060523354-853f-default
      Accept-Ranges: bytes
      Content-Type: text/plain; charset=utf-8
      Connection: Keep-Alive
```

▶ 7. View the statistics for the operator:swift subuser.

- 8. Return to workstation as the student user.

```
[admin@clienta ~]$ exit  
[student@workstation ~]$
```

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 api-swift
```

This concludes the guided exercise.

▶ Lab

Accessing Object Storage Using a REST API

In this lab, you will configure Ceph to provide object storage to clients, using both the Amazon S3 and OpenStack Swift APIs.

Outcomes

You should be able to:

- Create buckets and containers using the Amazon S3 and OpenStack Swift APIs.
- Upload and download objects using the Amazon S3 and OpenStack Swift APIs.

Before You Begin

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

This command ensures that the lab environment is created and ready for the lab exercise.

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

This command confirms that the hosts required for this exercise are accessible and configures a multisite RADOS Gateway service.

Instructions



Important

This lab runs on a multisite RADOS Gateway deployment.

To ensure that metadata operations, such as user and bucket creation, occur on the master zone and are synced across the multisite service, perform all metadata operations on the `serverc` node. Other normal operations, such as uploading or downloading objects from the RADOS Gateway service, can be performed on any cluster node that has access to the service endpoint.

1. On the `serverc` node, create a user for the S3 API and a subuser for the Swift API. Create the S3 user with the name `S3 Operator`, UID `operator`, access key `12345`, and secret key `67890`. Grant full access to the `operator` user.
Create the Swift subuser of the `operator` user with the name `operator:swift` and the secret ``opswif``. Grant full access to the subuser.
2. Configure the AWS CLI tool to use the `operator` user credentials. Create a bucket called `log-artifacts`. The RADOS Gateway service is running on the default port on the `serverc` node.
3. Create a container called `backup-artifacts`. The RADOS Gateway service is on the default port on the `serverc` node.

4. Create a 10MB file called `log-object-10MB.bin` in the `/tmp` directory. Upload the `log-object-10MB.bin` file to the `log-artifacts` bucket. On the `serverf` node, download the `log-object-10MB.bin` from the `log-artifacts` bucket.
5. On the `serverf` node, create a 20MB file called `backup-object20MB.bin` in the `/tmp` directory. Upload the `backup-object20MB.bin` file to the `backup-artifacts` bucket, using the service default port. View the status of the `backup-artifacts` bucket and verify that the `Objects` field has the value of 1.
6. On the `serverc` node, download the `backup-object-20MB.bin` file to the `/home/admin` directory.
7. Return to `workstation` as the `student` user.

Evaluation

Grade your work by running the `lab grade api-review` command from your `workstation` machine. Correct any reported failures and rerun the script until successful.

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

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 api-review
```

This concludes the lab.

► Solution

Accessing Object Storage Using a REST API

In this lab, you will configure Ceph to provide object storage to clients, using both the Amazon S3 and OpenStack Swift APIs.

Outcomes

You should be able to:

- Create buckets and containers using the Amazon S3 and OpenStack Swift APIs.
- Upload and download objects using the Amazon S3 and OpenStack Swift APIs.

Before You Begin

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

This command ensures that the lab environment is created and ready for the lab exercise.

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

This command confirms that the hosts required for this exercise are accessible and configures a multisite RADOS Gateway service.

Instructions



Important

This lab runs on a multisite RADOS Gateway deployment.

To ensure that metadata operations, such as user and bucket creation, occur on the master zone and are synced across the multisite service, perform all metadata operations on the `serverc` node. Other normal operations, such as uploading or downloading objects from the RADOS Gateway service, can be performed on any cluster node that has access to the service endpoint.

1. On the `serverc` node, create a user for the S3 API and a subuser for the Swift API. Create the S3 user with the name `S3 Operator`, `UID operator`, access key `12345`, and secret key `67890`. Grant full access to the `operator` user.
Create the Swift subuser of the `operator` user with the name `operator:swift` and the secret ``opswift``. Grant full access to the subuser.
 - 1.1. Log in to `serverc` as the `admin` user.

```
[student@workstation ~]$ ssh admin@serverc
[admin@serverc ~]$
```

- 1.2. Create an Amazon S3 API user called `S3 Operator` with the UID of `operator`. Assign an access key of `12345` and a secret of `67890`, and grant the user full access.

```
[admin@serverc ~]$ sudo cephadm shell -- radosgw-admin user create \
--uid="operator" --access="full" --display-name="S3 Operator" \
--access_key="12345" --secret="67890"
...output omitted...
```

- 1.3. Create a Swift subuser called `operator:swift`. Set `opswift` as the subuser secret and grant full access.

```
[admin@serverc ~]$ sudo cephadm shell -- radosgw-admin subuser create \
--uid="operator" --subuse r="operator:swift" --access="full" --secret="opswift"
...output omitted...
```

2. Configure the AWS CLI tool to use the `operator` user credentials. Create a bucket called `log-artifacts`. The RADOS Gateway service is running on the default port on the `serverc` node.
 - 2.1. Configure the AWS CLI tool to use operator credentials. Enter `12345` as the access key and `67890` as the secret key.

```
[admin@serverc ~]$ aws configure --profile=ceph
AWS Access Key ID [None]: 12345
AWS Secret Access Key [None]: 67890
Default region name [None]: Enter
Default output format [None]: Enter
```

- 2.2. Create a bucket called `log-artifacts`.

```
[admin@serverc ~]$ aws --profile=ceph --endpoint=http://serverc:80 s3 mb \
s3://log-artifacts
make_bucket: log-artifacts
```

- 2.3. Verify that the AWS bucket exists.

```
[admin@serverc ~]$ aws --profile=ceph --endpoint=http://serverc:80 s3 ls
2021-11-03 06:00:39 log-artifacts
```

3. Create a container called `backup-artifacts`. The RADOS Gateway service is on the default port on the `serverc` node.

- 3.1. Create a Swift container called `backup-artifacts`.

```
[admin@serverc ~]$ swift -V 1.0 -A http://serverc:80/auth/v1 -U operator:swift \
-K opswift post backup-artifacts
```

- 3.2. Verify that the container exists.

```
[admin@serverc ~]$ swift -V 1.0 -A http://serverc:80/auth/v1 -U operator:swift \
-K opswift list
backup-artifacts
log-artifacts
```

4. Create a 10MB file called `log-object-10MB.bin` in the `/tmp` directory. Upload the `log-object-10MB.bin` file to the `log-artifacts` bucket. On the `serverf` node, download the `log-object-10MB.bin` from the `log-artifacts` bucket.

- 4.1. Create a 10 MB file called `log-object-10MB.bin` in the `/tmp` directory.

```
[admin@serverc ~]$ dd if=/dev/zero of=/tmp/log-object-10MB.bin bs=1024K count=10
10+0 records in
10+0 records out
10485760 bytes (10 MB, 10 MiB) copied, 0.00498114 s, 2.1 GB/s
```

- 4.2. Upload the file `log-object-10MB.bin` to the `log-artifacts` bucket.

```
[admin@serverc ~]$ aws --profile=ceph --endpoint=http://serverc:80 \
--acl=public-read-write s3 cp /tmp/log-object-10MB.bin \
s3://log-artifacts/log-object-10MB.bin
...output omitted...
```

- 4.3. Log in to `serverf` as the `admin` user. Download the `log-object-10MB.bin` from the `log-artifacts` bucket.

```
[admin@serverc ~]$ ssh admin@serverf
admin@serverf's password: redhat
[admin@serverf ~]$ wget http://serverc:80/log-artifacts/log-object-10MB.bin
--2021-10-20 21:28:58--  http://serverc/log-artifacts/log-object-10MB.bin
Resolving serverc (serverc)... 172.25.250.12
Connecting to serverc (serverc)|172.25.250.12|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10485760 (10M) [application/octet-stream]
Saving to: 'log-object-10MB.bin'

log-object-10MB.bin      100%[=====] 10.00M  --.-KB/
s    in 0.01s

2021-10-20 21:28:58 (727 MB/s) - 'log-object-10MB.bin' saved [10485760/10485760]
```

5. On the `serverf` node, create a 20MB file called `backup-object20MB.bin` in the `/tmp` directory. Upload the `backup-object20MB.bin` file to the `backup-artifacts` bucket, using the service default port. View the status of the `backup-artifacts` bucket and verify that the `Objects` field has the value of 1.

- 5.1. Create a 20 MB file called `backup-object-20MB.bin` in the `/tmp` directory.

```
[admin@serverf ~]$ dd if=/dev/zero of=/tmp/backup-object-20MB.bin \
bs=2048K count=10
10+0 records in
10+0 records out
20971520 bytes (21 MB, 20 MiB) copied, 0.010515 s, 2.0 GB/s
```

- 5.2. Upload the backup-object20MB.bin file to the backup-artifacts bucket.

```
[admin@serverf ~]$ swift -V 1.0 -A http://serverf:80/auth/v1 -U operator:swift \
-K opswift upload backup-artifacts /tmp/backup-object-20MB.bin --object-name \
backup-object-20MB.bin
...output omitted...
```

- 5.3. View the statistics for the backup-artifacts bucket and verify that it contains the uploaded object.

```
[admin@serverf ~]$ swift -V 1.0 -A http://serverf:80/auth/v1 -U operator:swift \
-K opswift stat backup-artifacts
...output omitted...
```

6. On the serverc node, download the backup-object-20MB.bin file to the /home/admin directory.

```
[admin@serverf ~]$ exit
Connection to serverf closed.
[admin@serverc ~]$ swift -V 1.0 -A http://serverf:80/auth/v1 -U operator:swift \
-K opswift download backup-artifacts backup-object-20MB.bin
```

7. Return to workstation as the student user.

```
[admin@serverc ~]$ exit
[student@workstation ~]$
```

Evaluation

Grade your work by running the `lab grade api-review` command from your workstation machine. Correct any reported failures and rerun the script until successful.

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

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 api-review
```

This concludes the lab.

Summary

In this chapter, you learned:

- You can access the RADOS Gateway by using clients that are compatible with the Amazon S3 API or the OpenStack Swift API.
- The RADOS Gateway can be configured to use either of the bucket name formats supported by the Amazon S3 API.
- To support authentication by using the OpenStack Swift API, Swift users are represented by RADOS Gateway subusers.
- You can define deletion policies for Amazon S3 buckets, and object versioning for Swift containers, to manage the behavior of deleted objects.

Chapter 10

Providing File Storage with CephFS

Goal

Configure Red Hat Ceph Storage to provide file storage for clients using the Ceph File System (CephFS).

Objectives

- Provide file storage on the Ceph cluster by deploying the Ceph File System (CephFS).
- Configure CephFS, including snapshots, replication, memory management, and client access.

Sections

- Deploying Shared File Storage (and Guided Exercise)
- Managing Shared File Storage (and Guided Exercise)

Lab

Providing File Storage with CephFS

Deploying Shared File Storage

Objectives

After completing this section, you should be able to provide file storage on the Ceph cluster by deploying the Ceph File System (CephFS).

The Ceph File System and MDS

The *Ceph File System (CephFS)* is a POSIX-compliant file system that is built on top of RADOS, Ceph's distributed object store. File-based storage organizes your data as a traditional file system, with a directory tree hierarchy. Implementing the Ceph File System requires a running Ceph storage cluster and at least one *Ceph Metadata Server (MDS)* to manage the CephFS metadata, separately from file data, which reduces complexity and improves reliability. Similar to RBD and RGW, the CephFS daemon is implemented as a native interface to `librados`.

File, Block, and Object Storage

File-based storage organizes your data as a traditional file system. Data is saved as files with a name and associated metadata, such as modification time stamps, an owner, and access permissions. File-based storage uses a directory tree hierarchy to organize how files are stored.

Block-based storage provides a storage volume that operates similar to a disk device, organized into equally sized chunks. Typically, block-based storage volumes are either formatted with a file system, or applications such as databases directly access and write to them.

With object-based storage, you can store arbitrary data and metadata as a unit that is labeled with a unique identifier in a flat storage pool. Rather than accessing data as blocks or in a file-system hierarchy, you use an API to store and retrieve objects. Fundamentally, the Red Hat Ceph Storage RADOS cluster is an object store.

The Metadata Server

The Metadata Server (MDS) manages metadata for CephFS clients. This daemon provides information that CephFS clients need to access RADOS objects, such as providing file locations within the file-system tree. MDS manages the directory hierarchy and stores file metadata, such as the owner, time stamps, and permission modes, in a RADOS cluster. MDS is also responsible for access caching and managing client caches to maintain cache coherence.

MDS daemons operate in two modes: active and standby. An active MDS manages the metadata on the CephFS file system. A standby MDS serves as a backup, and switches to the active mode if the active MDS becomes unresponsive. CephFS shared file systems require an active MDS service. You should deploy at least one standby MDS in your cluster to ensure high availability.

If you do not create enough MDS pools to match the number of configured standby daemons, then the Ceph cluster displays a WARN health status. The recommended solution is to create more MDS pools to provide a pool for each daemon. However, a temporary solution is to set the number of standby pools to 0, which disables the Ceph MDS standby check through the `ceph fs set fs-name standby_count_wanted 0` command.

CephFS clients first contact a MON to authenticate and retrieve the cluster map. Then, the client queries an active MDS for file metadata. The client uses the metadata to access the objects that comprise the requested file or directory by communicating directly with the OSDs.

MDS features and configuration options are described in the following list:

MDS Ranks

MDS ranks define how the metadata workload is distributed over the MDS daemons. The number of ranks, which is defined by the `max_mds` configuration setting, is the maximum number of MDS daemons that can be active at a time. MDS daemons start without a rank and the MON daemon is responsible for assigning them a rank.

Subvolumes and Subvolume Groups

CephFS subvolumes are an abstraction for independent CephFS file system directory trees. When creating subvolumes, you can specify more fine-grained rights management, such as the UID, GID, file mode, size, and the subvolume group for your subvolume. Subvolume groups are abstractions at a directory level across a set of subvolumes.



Note

You can create snapshots of subvolumes, but Red Hat Ceph Storage 5 does not support creating snapshots of subvolume groups. You can list and remove existing snapshots of subvolume groups.

File System Affinity

Configure your CephFS file system to prefer one MDS over another MDS. For example, you can configure to prefer an MDS that runs on a faster server over another MDS that runs on an older server. This file system affinity is configured through the `mds_join_fs` option.

MDS Cache Size Limits

Limit the size of the MDS cache by limiting the maximum memory to use with the `mds_cache_memory_limit` option, or by defining the maximum number of inodes with the `mds_cache_size` option.

Quotas

Configure your CephFS file system to restrict the number of bytes or files that are stored by using quotas. Both the FUSE and kernel clients support checking quotas when mounting a CephFS file system. These clients are also responsible for stopping writing data to the CephFS file system when the user reaches the quota limit. Use the `setfattr` command's `ceph.quota.max_bytes` and `ceph.quota.max_files` options to set the limits.

New CephFS Capabilities

Red Hat Ceph Storage 5 removes limitations from earlier versions.

- Red Hat Ceph Storage 5 supports more than one active MDS in a cluster, which can increase metadata performance. To remain highly available, you can configure additional standby MDSes to take over from any active MDS that fails.
- Red Hat Ceph Storage 5 supports more than one CephFS file system in a cluster. Deploying more than one CephFS file system requires running more MDS daemons.

Deploying CephFS

To implement a CephFS file system, create the required pools, create the CephFS file system, deploy the MDS daemons, and then mount the file system. You can manually create the pools,

create the CephFS file system, and deploy the MDS daemons, or use the `ceph fs volume create` command, which does all these steps automatically. The first option gives the system administrator more control over the process, but with more steps than the simpler `ceph fs volume create` command.

Creating CephFS with the Volume Method

Use `ceph fs volume` to directly create the CephFS volume. This command creates pools that are associated to the CephFS, creates the CephFS volume, and also starts the MDS service on the hosts.

```
[ceph: root@server /]# ceph fs volume create fs-name \
--placement="number-of-hosts list-of-hosts"
```

Creating CephFS with Placement Specification

For more control over the deploying process, manually create the pools that are associated to the CephFS, start the MDS service on the hosts, and create the CephFS file system.

Creating the Data and Metadata Pools

A CephFS file system requires at least two pools, one to store CephFS data, and another to store CephFS metadata. The default names for these two pools are `cephfs_data` and `cephfs_metadata`. To create a CephFS file system, first create the two pools.

```
[ceph: root@server /]# ceph osd pool create cephfs_data
[ceph: root@server /]# ceph osd pool create cephfs_metadata
```

This example creates two pools with standard parameters. Because the metadata pool stores file location information, consider a higher replication level for this pool to avoid data errors that render your data inaccessible.

By default, Ceph uses replicated data pools. However, erasure-coded data pools are now also supported for CephFS file systems. Create an erasure-coded pool with the `ceph osd pool` command:

```
[ceph: root@server /]# ceph osd pool create pool-name erasure
```

Creating CephFS and Deploying MDS service

When the data and metadata pools are available, use the `ceph fs new` command to create the file system, as follows:

```
[ceph: root@server /]# ceph fs new fs-name metadata-pool data-pool
```

To add an existing erasure pool as a data pool in your CephFS file system, use `ceph fs add_data_pool`.

```
[ceph: root@server /]# ceph fs add_data_pool fs-name data-pool
```

You can then deploy the MDS service:

```
[ceph: root@server /]# ceph orch apply mds fs-name \
--placement="number-of-hosts list-of-hosts"
```

Creating CephFS with the Service Specification

Use the Ceph Orchestrator to deploy the MDS service with the service specification. First, manually create the two required pools. Then, create a YAML file with the service details:

```
service_type: mds
service_id: fs-name
placements:
  hosts:
    - host-name-1
    - host-name-2
    - ...
```

Use the YAML service specification to deploy the MDS service with the `ceph orch apply` command:

```
[ceph: root@server /]# ceph orch apply -i file-name.yml
```

Finally, create the CephFS file system with the `ceph fs new` command.

Mounting a File System with CephFS

You can mount CephFS file systems with either of the available clients:

- The kernel client
- The FUSE client

The kernel client requires a Linux kernel version 4 or later, which is available starting with RHEL 8. For previous kernel versions, use the FUSE client instead.

The two clients have unique advantages and disadvantages. Not all features are supported in both clients. For example, the kernel client does not support quotas, but can be faster. The FUSE client supports quotas and ACLs. You must enable ACLs to use them with the CephFS file system mounted with the FUSE client.

Common CephFS Client Configuration

To mount a CephFS-based file system with either client, verify the following prerequisites on the client host.

- Install the `ceph-common` package. For the FUSE client, also install the `ceph-fuse` package.
- Verify that the Ceph configuration file exists (`/etc/ceph/ceph.conf` by default).
- Authorize the client to access the CephFS file system.
- Extract the new authorization key with the `ceph auth get` command and copy it to the `/etc/ceph` folder on the client host.
- When using the FUSE client as a non-root user, add `user_allow_other` in the `/etc/fuse.conf` configuration file.

Mounting CephFS with the FUSE Client

When the prerequisites are met, use the FUSE client to mount and unmount a CephFS file system:

```
[root@node ~]# ceph-fuse [mount-point] [options]
```

To provide the key ring for a specific user, use the `--id` option.

You must authorize the client to access the CephFS file system, by using the `ceph fs authorize` command:

```
[ceph: root@server /]# ceph fs authorize fs-name client-name path permissions
```

With the `ceph fs authorize` command, you can provide fine-grained access control for different users and folders in the CephFS file system. You can set different options for folders in a CephFS file system:

- **r**: Read access to the specified folder. Read access is also granted to the subfolders, if no other restriction is specified.
- **w**: Write access to the specified folder. Write access is also granted to the subfolders, if no other restriction is specified.
- **p**: Clients require the **p** option in addition to **r** and **w** capabilities to use layouts or quotas.
- **s**: Clients require the **s** option in addition to **r** and **w** capabilities to create snapshots.

This example allows one user to read the root folder, and also provides read, write, and snapshot permissions to the `/directory` folder.

```
[ceph: root@server /]# ceph fs authorize mycephfs client.user / r /directory rws
```

By default, the CephFS FUSE client mounts the root directory (/) of the accessed file system. You can mount a specific directory with the `ceph-fuse -r directory` command.



Note

When you try to mount a specific directory, this operation fails if the directory does not exist in the CephFS volume.

When more than one CephFS file system is configured, the CephFS FUSE client mounts the default CephFS file system. To use a different file system, use the `--client_fs` option.

To persistently mount your CephFS file system by using the FUSE client, you can add the following entry to the `/etc/fstab` file:

```
host-name:_port_ mount-point fuse.ceph
ceph.id=myuser,ceph.client_mountpoint=mountpoint,_netdev 0 0
```

Use the `umount` command to unmount the file system:

```
[root@node ~]# umount mount-point
```

Mounting CephFS with the Kernel Client

When using the CephFS kernel client, use the following command to mount the file system:

```
[root@node ~]# mount -t ceph [device]:[path] [mount-point] \
-o [key-value] [other-options]
```

You must authorize the client to access the CephFS file system, with the `ceph fs authorize` command. Extract the client key with the `ceph auth get` command, and then copy the key to the `/etc/ceph` folder on the client host.

With the CephFS kernel client, you can mount a specific subdirectory from a CephFS file system. This example mounts a directory called `/dir/dir2` from the root of a CephFS file system:

```
[root@node ~]# mount -t ceph mon1:/dir1/dir2 mount-point
```

You can specify a list of several comma-separated MONs to mount the device. The standard port (6789) is the default, or you can add a colon and a nonstandard port number after the name of each MON. Recommended practice is to specify more than one MON in case that some are offline when the file system is mounted.

These other options are available when using the CephFS kernel client:

CephFS Kernel Client Mount Options

Option name	Description
<code>name=name</code>	The Cephx client ID to use. The default is <code>guest</code> .
<code>fs=fs-name</code>	The name of the CephFS file system to mount. When no value is provided, it uses the default file system.
<code>secret=secret_value</code>	Value of the secret key for this client.
<code>secretfile=secret_key_file</code>	The path to the file with the secret key for this client.
<code>rsize=bytes</code>	Specify the maximum read size in bytes.
<code>wsize=bytes</code>	Specify the maximum write size in bytes. The default is none.

To persistently mount your CephFS file system by using the kernel client, you can add the following entry to the `/etc/fstab` file:

```
mon1,mon2:/ mount_point ceph name=user1,secretfile=/root/secret,_netdev 0 0
```

Use the `umount` command to unmount the file system:

```
[root@node ~]# umount mount_point
```

Removing CephFS

You can remove a CephFS if needed. However, first back up all your data, because removing your CephFS file system destroys all the stored data on that file system.

The procedure to remove a CephFS is first to mark it as down, as follows:

```
[ceph: root@server /]# ceph fs set fs-name down true
```

Then, you can remove it with the next command:

```
[ceph: root@server /]# ceph fs rm fs-name --yes-i-really-mean-it
```

User Space Implementation of an NFS Server

Red Hat Ceph Storage 5 provides access to Ceph storage from an NFS client, with NFS Ganesha. NFS Ganesha is a user space NFS file server that supports multiple protocols, such as NFSv3, NFSv4.0, NFSv4.1, and pNFS. NFS Ganesha uses a File System Abstraction Layer (FSAL) architecture, to support and share files from multiple file systems or lower-level storage, such as Ceph, Samba, Gluster, and Linux file systems such as XFS.

In Red Hat Ceph Storage, NFS Ganesha shares files with the NFS 4.0 or later protocol. This requirement is necessary for proper feature functioning by the CephFS client, the OpenStack Manila File Sharing service, and other Red Hat products that are configured to access the NFS Ganesha service.

The following list outlines the advantages of a user space NFS server:

- The server does not implement system calls.
- Caching is defined and used more efficiently.
- Service failover and restarting are faster and easier to implement.
- User space services can be clustered easily for high availability.
- You can use distributed lock management (DLM) to allow multiple client protocols.
- Debugging of server issues is simpler, so you do not need to create kernel dumps.
- Resource management and performance monitoring are simpler.

You can deploy NFS Ganesha in an active-active configuration on top of an existing CephFS file system through the ingress service. The main goal of this active-active configuration is for load balancing, and scaling to many instances that handle higher loads. Thus, if one node fails, then the cluster redirects all the workload to the rest of the nodes.

System administrators can deploy the NFS Ganesha daemons via the CLI or manage them automatically if either the Cephadm or Rook orchestrators are enabled.

The following list outlines the advantages to having an ingress service on top of an existing NFS service:

- A virtual IP to access the NFS server.
- Migration of the NFS service to another node if one node fails, providing shorter failover times.
- Load balancing across the NFS nodes.



Note

The ingress implementation is not yet completely developed. It can deploy multiple Ganesha instances and balance the load between them, but failover between hosts is not yet fully implemented. This feature is expected to be available in future releases.

You can use multiple active-active NFS Ganesha services with Pacemaker for high availability. The Pacemaker component is responsible for all cluster-related activities, such as monitoring cluster membership, managing the services and resources, and fencing cluster members.

As prerequisites, create a CephFS file system and install the `nfs-ganesha`, `nfs-ganesha-ceph`, `nfs-ganesha-rados-grace`, and `nfs-ganesha-rados-urls` packages on the Ceph MGR nodes.

After the prerequisites are satisfied, enable the Ceph MGR NFS module:

```
[ceph: root@server /]# ceph mgr module enable nfs
```

Then, create the NFS Ganesha cluster:

```
[ceph: root@server /]# ceph nfs cluster create cluster-name "node-list"
```

The *node-list* is a comma-separated list where the daemon containers are deployed.

Next, export the CephFS file system:

```
[ceph: root@server /]# ceph nfs export create cephfs fs-name  
\\  
cluster-name pseudo-path
```

The *pseudo-path* parameter is the pseudo root path.

Finally, mount the exported CephFS file system on a client node.

```
[root@node ~]# mount -t nfs -o port=ganesha-port node-name:pseudo-path path
```

MDS Autoscaler

CephFS shared file systems require at least one active MDS service for correct operation, and at least one standby MDS to ensure high availability. The MDS autoscaler module ensures the availability of enough MDS daemons.

This module monitors the number of ranks and the number of standby daemons, and adjusts the number of MDS daemons that the orchestrator spawns.

To enable the MDS autoscaler module, use the following command:

```
[ceph: root@server /]# ceph mgr module enable mds_autoscaler
```

Replicating CephFS on Another Ceph Cluster

Red Hat Ceph Storage 5 supports CephFS multi-site configuration for geo-replication. Thus, you can replicate the CephFS file system on another Red Hat Ceph Storage cluster. With this feature, you can fail over to the secondary CephFS file system and restart the applications that use it. The CephFS file system mirroring feature requires the *cephfs-mirror* package.



Note

Both the source and target clusters must use Red Hat Ceph Storage version 5 or later.

The CephFS mirroring feature is snapshot-based. The first snapshot synchronization requires bulk transfer of the data from the source cluster to the remote cluster. Then, for the following synchronizations, the mirror daemon identifies the modified files between local snapshots and synchronizes those files in the remote cluster. This synchronization method is faster than other methods that require bulk transfer of the data to the remote cluster, because it does not need to

Chapter 10 | Providing File Storage with CephFS

query the remote cluster (file differences are calculated between local snapshots) and needs only to transfer the updated files to the remote cluster.

The CephFS mirroring module is disabled by default. To configure a snapshot mirror for CephFS, you must enable the `mirroring` module on the source and remote clusters:

```
[ceph: root@server /]# ceph mgr module enable mirroring
```

Then, you can deploy the CephFS mirroring daemon on the source cluster:

```
[ceph: root@source /]# ceph orch apply cephfs-mirror [node-name]
```

The previous command deploys the CephFS mirroring daemon on `node-name` and creates the Ceph user `cephfs-mirror`. For each CephFS peer, you must create a user on the target cluster:

```
[ceph: root@target /]# ceph fs authorize fs-name client_ / rwp*
```

Then, you can enable mirroring on the source cluster. Mirroring must be enabled for a specific file system.

```
[ceph: root@source /]# ceph fs snapshot mirror enable fs-name
```

The next step is to prepare the target peer. You can create the peer bootstrap in the target node with the next command:

```
[ceph: root@target /]# ceph fs snapshot mirror peer_bootstrap create \
fs-name peer-name site-name
```

You can use the `site-name` string to identify the target storage cluster. When the target peer is created, you must import into the source cluster the bootstrap token from creating the peer on the target cluster:

```
[ceph: root@source /]# ceph fs snapshot mirror peer_bootstrap import \
fs-name bootstrap-token
```

Finally, configure a directory for snapshot mirroring on the source cluster with the following command:

```
[ceph: root@source /]# ceph fs snapshot mirror add fs-name path
```



References

`mount .ceph(8)`, `ceph-fuse(8)`, `ceph(8)`, `rados(8)`, and `cephfs-mirror(8)`
man pages

For more information, refer to the *Red Hat Ceph Storage 5 File System Guide* at
https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/file_system_guide/index

For more information regarding CephFS deployment, refer to the *Deployment of the Ceph File System* chapter in the *Red Hat Ceph Storage 5 File System Guide* at
https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/file_system_guide/index#deployment-of-the-ceph-file-system

For more information regarding CephFS over NFS protocol, refer to the *Exporting Ceph File System Namespaces over the NFS Protocol* chapter in the
https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/file_system_guide/index#exporting-ceph-file-system-namespaces-over-the-nfs-protocol_fs

► Guided Exercise

Deploying Shared File Storage

In this exercise, you configure a shared file client access with CephFS.

Outcomes

You should be able to deploy a Metadata Server (MDS) and mount a CephFS file system with the kernel client and the Ceph-Fuse client. You should be able to save the file system as persistent storage.

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 fileshare-deploy
```

Instructions

- The `serverc`, `serverd`, and `servere` nodes are an operational 3-node Ceph cluster. All three nodes operate as a MON, a MGR, and an OSD host with at least one colocated OSD.
 - The `clienta` node is your admin node server and you will use it to install the MDS on `serverc`.
- 1. Log in to `clienta` as the `admin` user. Deploy the `serverc` node as an MDS. Verify that the MDS is operating and that the `ceph_data` and `ceph_metadata` pools for CephFS are created.

1.1. Log in to `clienta` as the `admin` user, and use `sudo` to run the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]#
```

1.2. Create the two required CephFS pools. Name these pools `mycephfs_data` and `mycephfs_metadata`.

```
[ceph: root@clienta /]# ceph osd pool create mycephfs_data
pool 'mycephfs_data' created
[ceph: root@clienta /]# ceph osd pool create mycephfs_metadata
pool 'mycephfs_metadata' created
```

1.3. Create the CephFS file system with the name `mycephfs`. Your pool numbers might differ in your lab environment.

```
[ceph: root@clienta /]# ceph fs new mycephfs mycephfs_metadata mycephfs_data
new fs with metadata pool 7 and data pool 6
```

1.4. Deploy the MDS service on `serverc.lab.example.com`.

```
[ceph: root@clienta /]# ceph orch apply mds mycephfs \
--placement="1 serverc.lab.example.com"
Scheduled mds.mycephfs update...
```

1.5. Verify that the MDS service is active.

```
[ceph: root@clienta /]# ceph mds stat
mycephfs:1 {0=mycephfs.serverc.mycctv=up:active}
[ceph: root@clienta /]# ceph status
  cluster:
    id:        472b24e2-1821-11ec-87d7-52540000fa0c
    health:   HEALTH_OK

  services:
    mon: 4 daemons, quorum serverc.lab.example.com,serverd,servere,clienta (age
          29m)
    mgr: servere.xnprpz(active, since 30m), standbys: clienta.jahhir,
          serverd.qbvejy, serverc.lab.example.com.xgbgpo
    mds: 1/1 daemons up
    osd: 15 osds: 15 up (since 28m), 15 in (since 28m)
    rgw: 2 daemons active (2 hosts, 1 zones)

  data:
    volumes: 1/1 healthy
    pools:   7 pools, 169 pgs
    objects: 212 objects, 7.5 KiB
    usage:   215 MiB used, 150 GiB / 150 GiB avail
    pgs:     169 active+clean
```

1.6. List the available pools. Verify that `mycephfs_data` and `mycephfs_metadata` are listed.

```
[ceph: root@clienta /]# ceph df
--- RAW STORAGE ---
CLASS      SIZE    AVAIL     USED    RAW USED %RAW USED
hdd       90 GiB  90 GiB  129 MiB  129 MiB      0.14
TOTAL     90 GiB  90 GiB  129 MiB  129 MiB      0.14

--- POOLS ---
POOL              ID  PGS  STORED  OBJECTS  USED  %USED  MAX  AVAIL
device_health_metrics  1   1    0 B     0         0 B    0      28 GiB
.rgw.root           2   32   1.3 KiB   4        48 KiB   0      28 GiB
default.rgw.log      3   32   3.6 KiB  177     408 KiB   0      28 GiB
default.rgw.control  4   32   0 B     8         0 B    0      28 GiB
default.rgw.meta      5   8    0 B     0         0 B    0      28 GiB
mycephfs_data        6   32   0 B     0         0 B    0      28 GiB
mycephfs_metadata    7   32   2.3 KiB  22      96 KiB   0      28 GiB
```

- ▶ 2. Mount the new CephFS file system on the `/mnt/mycephfs` directory as a kernel client on the `clienta` node. Verify normal operation by creating two folders `dir1` and `dir2` on the

file system. Create an empty file called `atestfile` in the `dir1` directory and a 10 MB file called `ddtest` in the same directory.

- 2.1. Exit the `cephadm` shell. Switch to the `root` user. Verify that the Ceph client key ring is present in the `/etc/ceph` folder on the client node.

```
[ceph: root@clienta /]# exit
exit
[admin@clienta ~]$ sudo -i
[root@clienta ~]# ls -l /etc/ceph
total 12
-rw-r--r--. 1 root root 63 Sep 17 21:42 ceph.client.admin.keyring
-rw-r--r--. 1 root root 177 Sep 17 21:42 ceph.conf
-rw-----. 1 root root 82 Sep 17 21:42 podman-auth.json
```

- 2.2. Install the `ceph-common` package on the client node.

```
[root@clienta ~]# yum install ceph-common
...output omitted...
```

- 2.3. Create a mount point called `/mnt/mycephfs` and mount the new CephFS file system.

```
[root@clienta ~]# mkdir /mnt/mycephfs
[root@clienta ~]# mount.ceph serverc.lab.example.com:/ /mnt/mycephfs \
-o name=admin
```

- 2.4. Verify that the mount is successful.

```
[root@clienta ~]# df /mnt/mycephfs
Filesystem      1K-blocks  Used Available Use% Mounted on
172.25.250.12:/   29822976     0  29822976   0% /mnt/mycephfs
```

- 2.5. Create two directories called `dir1` and `dir2`, directly underneath the mount point. Ensure that they are available.

```
[root@clienta ~]# mkdir /mnt/mycephfs/dir1
[root@Clienta ~]# mkdir /mnt/mycephfs/dir2
[root@clienta ~]# ls -al /mnt/mycephfs/
total 0
drwxr-xr-x. 4 root  root  2 Sep 28 06:04 .
drwxr-xr-x. 3 root  root  22 Sep 28 05:49 ..
drwxr-xr-x. 2 root  root  0 Sep 28 06:04 dir1
drwxr-xr-x. 2 root  roots  0 Sep 28 06:04 dir2
```

- 2.6. Create an empty file called `atestfile` in the `dir1` directory. Then, create a 10 MB file called `ddtest` in the same directory.

```
[root@clienta ~]# touch /mnt/mycephfs/dir1/atestfile
[root@clienta ~]# dd if=/dev/zero of=/mnt/mycephfs/dir1/ddtest \
bs=1024 count=10000
10000+0 records in
10000+0 records out
10240000 bytes (10 MB, 9.8 MiB) copied, 0.0208212 s, 492 MB/s
```

- 2.7. Unmount the CephFS file system.

```
[root@clienta ~]# umount /mnt/mycephfs
```

- 3. Run the `ceph fs status` command and inspect the size of the used data in the `mycephfs_data` pool. The larger size is reported because the CephFS file system replicates across the three Ceph nodes.

```
[root@clienta ~]# cephadm shell -- ceph fs status
Inferring fsid 472b24e2-1821-11ec-87d7-52540000fa0c
Inferring config /var/lib/ceph/472b24e2-1821-11ec-87d7-52540000fa0c/mon.clienta/
config
Using recent ceph image registry.redhat.io/rhceph/rhceph-5-
rhel8@sha256:6306...a47ff
mycephfs - 0 clients
=====
RANK STATE MDS ACTIVITY DNS INOS DIRS CAPS
0 active mycephfs.serverc.mycctv Reqs: 0 /s 14 17 14 0
POOL TYPE USED AVAIL
mycephfs_metadata metadata 152k 28.4G
mycephfs_data data 29.2M 28.4G
MDS version: ceph version 16.2.0-117.el8cp
(0e34bb74700060ebfaa22d99b7d2cdc037b28a57) pacific (stable)
```

- 4. Create a `restricteduser` user, which has read access to the root folder, and read and write permissions on the `dir2` folder. Use this new user to mount again the CephFS file system on `clienta` and check the permissions.

- 4.1. Create the `restricteduser` user with read permission on the root folder, and read and write permissions on the `dir2` folder. Use the `cephadm shell --mount` option to copy the user key-ring file to the `/etc/ceph` folder on `clienta`.

```
[root@clienta ~]# cephadm shell --mount /etc/ceph/
[ceph: root@clienta /]# ceph fs authorize mycephfs client.restricteduser \
/ r /dir2 rw
[client.restricteduser]
key = AQBc315hI7PaBRAA9/9fdmj+wjblk+izstA0aQ==
[ceph: root@clienta /]# ceph auth get client.restricteduser \
-o /mnt/ceph.client.restricteduser.keyring
[ceph: root@clienta /]# exit
exit
```

- 4.2. Use the kernel client to mount the `mycephfs` file system with this user.

```
[root@clienta ~]# mount.ceph serverc.lab.example.com:/ /mnt/mycephfs \
-o name=restricteduser,fs=mycephfs
```

- 4.3. Test the user permissions in the different folders and files.

```
[root@clienta ~]# tree /mnt
/mnt
└── mycephfs
    ├── dir1
    │   ├── a3rdfile
    │   └── ddtest
    └── dir2

3 directories, 2 files
[root@clienta ~]# touch /mnt/mycephfs/dir1/restricteduser_file1
touch: cannot touch '/mnt/mycephfs/dir1/restricteduser_file1': Permission denied
[root@clienta ~]# touch /mnt/mycephfs/dir2/restricteduser_file2
[root@clienta ~]# ls /mnt/mycephfs/dir2
restricteduser_file2
[root@clienta ~]# rm /mnt/mycephfs/dir2/restricteduser_file2
```

- 4.4. Unmount the CephFS file system.

```
[root@clienta ~]# umount /mnt/mycephfs
```

► 5. Install the ceph-fuse package and mount to a new directory called cephfuse.

- 5.1. Create a directory called /mnt/mycephfuse to use as a mount point for the Fuse client.

```
[root@clienta ~]# mkdir /mnt/mycephfuse
```

- 5.2. Install the ceph-fuse package, which is not installed by default.

```
[root@clienta ~]# yum install ceph-fuse
...output omitted...
```

- 5.3. Use the installed Ceph-Fuse driver to mount the file system.

```
[root@clienta ~]# ceph-fuse -n client.restricteduser \
--client_fs mycephfs /mnt/mycephfuse
2021-09-28T06:29:06.205-0400 7fc4f1fdd200 -1 init, newargv = 0x56415adcb160
newargc=15
ceph-fuse[39038]: starting ceph client
ceph-fuse[39038]: starting fuse
```

- 5.4. Run the tree command on the /mnt directory to see its data.

```
[root@clienta ~]# tree /mnt
/mnt
└── mycephfs
    └── mycephfuse
        ├── dir1
        │   └── atestfile
        └── ddtest
            └── dir2

4 directories, 2 files
```

5.5. Unmount the Ceph-Fuse file system.

```
[root@clienta ~]# umount /mnt/mycephfuse
```

► 6. Use the FUSE client to persistently mount the CephFS in the /mnt/mycephfuse folder.

6.1. Configure the /etc/fstab file to mount the file system at startup.

```
[root@clienta ~]# cat /etc/fstab
...output omitted...
serverc.lab.example.com:/ /mnt/mycephfuse fuse.ceph ceph.id=restricteduser,_netdev
0 0
```

6.2. Mount again the /mnt/mycephfuse folder with the `mount -a` command. Verify with the `df` command.

```
[root@clienta ~]# mount -a
2021-09-28T06:33:28.715-0400 7fb5a5e02200 -1 init, newargv = 0x55b327cbaa80
newargc=17
ceph-fuse[39201]: starting ceph client
ceph-fuse[39201]: starting fuse
[admin@clienta ~]$ df
Filesystem      1K-blocks     Used Available Use% Mounted on
...output omitted...
ceph-fuse       49647616   12288  49635328   1% /mnt/mycephfuse
```

6.3. Unmount the /mnt/mycephfuse folder.

```
[root@clienta ~]# umount /mnt/mycephfuse
```

6.4. Return to workstation as the student user.

```
[root@clienta ~]# exit
[admin@clienta ~]$ exit
[student@workstation ~]$
```



Warning

Run the `lab finish` script on the `workstation` server so that the `clienta` node can be safely rebooted without mount conflicts.

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 fileshare-deploy
```

This concludes the guided exercise.

Managing Shared File Storage

Objectives

After completing this section, you should be able to configure CephFS, including snapshots, replication, memory management, and client access.

CephFS Administration

Use the following commands to manage CephFS file systems:

Action	Command
Create a file system.	<code>ceph fs new <i>fs-name</i> <i>meta-pool</i> <i>data-pool</i></code>
List existing file systems.	<code>ceph fs ls</code>
Remove a file system.	<code>ceph fs rm <i>fs-name</i> [--yes-i-really-mean-it]</code>
Force MDS to fail status.	<code>ceph mds fail <i>gid/name/role</i></code>
Declare an MDS to be repaired, triggering a fallback.	<code>ceph mds repaired <i>role</i></code>

CephFS provides tools to inspect and repair MDS journals (`cephfs-journal-tool`) or MDS tables (`cephfs-table-tool`), and to inspect and rebuild metadata (`cephfs-data-scan`).

Mapping a File to an Object

For troubleshooting, it is useful to determine which OSDs store a file's objects. Directories or zero-length files might have any associated objects in a data pool.

This example retrieves object mapping information for a file within Ceph:

- Retrieve the inode number for the file.

```
[ceph: root@server /]# stat -c %i filepath
1099511627776
```

- Convert the inode number to hexadecimal. Use the `%x` formatting output of the `printf` command.

```
[ceph: root@server /]# printf '%x\n' 1099511627776
100000000000
```

This example combines these first two steps:

```
[ceph: root@server /]# printf '%x\n' $(stat -c %i filepath)
```

- Search for the hexadecimal ID in the RADOS object list. A large file might return multiple objects.

```
[ceph: root@server /]# rados -p cephfs_data ls | grep 10000000000
10000000000.00000000
```

- Retrieve the mapping information for the returned objects.

```
[ceph: root@server /]# ceph osd map cephfs_data 10000000000.00000000
osdmap e95 pool 'cephfs_data' (3) object '10000000000.00000000' -> pg 3.f0b56f30
(3.30) -> up ([1,2], p1) acting ([1,2], p1)
```

Interpret this output as saying that the e95 map epoch of the OSD map for the `cephfs_data` pool (ID 3) maps the `10000000000.00000000` object to placement group 3.30, which is on OSD 1 and OSD 2, and OSD 1 is primary. If the OSDs in `up` and `acting` status are not the same, then it implies that the cluster is rebalancing or has another issue.

Controlling the RADOS Layout of Files

The RADOS layout controls how files are mapped to objects. These settings are stored in virtual extended attributes (xattrs) in CephFS. You can adjust settings to control the size of the used objects and where they are stored.

Layout attributes are initially set on the directory at the top of the CephFS file system. You can manually set layout attributes on other directories or files. When you create a file, it inherits layout attributes from its parent directory. If layout attributes are not set in its parent directory, then the closest ancestor directory with layout attributes is used.

The layout attributes for a file, such as these examples, use the `ceph.file.layout` prefix.

File Layout Attributes

Attribute name	Description
<code>ceph.file.layout.pool</code>	The pool where Ceph stores the file's data objects (normally <code>cephfs_data</code>).
<code>ceph.file.layout.stripe_unit</code>	The size (in bytes) of a block of data that is used in the RAID 0 distribution of a file.
<code>ceph.file.layout.stripe_count</code>	The number of consecutive stripe units that constitute a RAID 0 "stripe" of file data.
<code>ceph.file.layout.object_size</code>	File data is split into RADOS objects of this size in bytes (4194304 bytes, or 4 MiB, by default).
<code>ceph.file.layout.pool_namespace</code>	The namespace that is used, if any.

The `ceph.dir.layout` prefix identifies the layout attributes for a directory.

Directory Layout Attributes

Attribute name	Description
ceph.dir.layout.pool	This attribute specifies the pool where Ceph stores the directory's data objects (normally <code>cephfs_data</code>).
ceph.dir.layout.stripe_unit	This attribute specifies the size (in bytes) of a block of data for the RAID 0 distribution of a directory.
ceph.dir.layout.stripe_count	This attribute specifies the number of consecutive stripe units that constitute a RAID 0 "stripe" of directory data.
ceph.dir.layout.object_size	Directory data is split into RADOS objects of this size (4194304 bytes, or 4 MiB, by default).
ceph.dir.layout.pool_namespace	This attribute specifies the name space that is used, if any.

The `getfattr` command displays the layout attributes for a file or directory:

```
[ceph: root@server /]# getfattr -n ceph.file.layout file-path
# file: file-path
ceph.file.layout="stripe_unit=4194304 stripe_count=1 object_size=4194304
pool=cephfs_data"
[ceph: root@server /]# getfattr -n ceph.dir.layout directory-path
# file : directory-path
ceph.dir.layout="stripe_unit=4194304 stripe_count=1 object_size=4194304
pool=cephfs_data"
```

The `setfattr` command modifies the layout attributes:

```
[ceph: root@server /]# setfattr -n ceph.file.layout.attribute -v value file
[ceph: root@server /]# setfattr -n ceph.dir.layout.attribute -v value directory
```



Important

Layout attributes are set when data is initially saved to a file. If the parent directory's layout attributes change after the file is created, then the file's layout attributes do not change. Additionally, a file's layout attributes can be changed only if it is empty.

Usage and Statistics

You can use virtual extended attributes for information about CephFS file system use. The `getfattr` command, when used with the `ceph` attribute namespace on a directory, returns a list of recursive statistics for that directory.

```
[ceph: root@server /]# getfattr -d -m ceph.dir.* directory-path
file: directory-path
ceph.dir.entries="1"
ceph.dir.files="0"
ceph.dir.rbytes="424617209"
ceph.dir.rctime="1341146808.804098000"
ceph.dir.rentries="39623"
ceph.dir.rfiles="37362"
ceph.dir.rsubdirs="2261"
ceph.dir.subdirs="1"
```

The statistics provide detailed information.

CephFS Statistics

Attribute name	Description
ceph.dir.entries	Number of direct descendants
ceph.dir.files	Number of regular files in the directory
ceph.dir.rbytes	Total file size in the subtree (the directory and all its descendants)
ceph.dir.rctime	Most recent creation time in the subtree (in seconds since the epoch, 1970-01-01 00:00:00 UTC)
ceph.dir.rentries	Number of descendants in the subtree
ceph.dir.rfiles	Number of regular files in the subtree
ceph.dir.rsubdirs	Number of directories in the subtree
ceph.dir.subdirs	Number of directories in the directory

Managing Snapshots

CephFS enables asynchronous snapshots by default when deploying Red Hat Ceph Storage 5. These snapshots are stored in a hidden directory called `.snap`. In earlier Red Hat Ceph Storage versions, snapshots were disabled by default, as they were an experimental feature.

Creating Snapshots

Use `cephfs set` to enable snapshot creation for an existing CephFS file system.

```
[ceph: root@server /]# ceph fs set fs-name allow_new_snaps true
```

To create a snapshot, first mount the CephFS file system on your client node. Use the `-o fs=_fs-name` option to mount a CephFS file system when you have more than one. Then, create a subdirectory inside the `.snap` directory. The snapshot name is the new subdirectory name. This snapshot contains a copy of all the current files in the CephFS file system.

```
[root@node ~]# mount.ceph server.example.com:/ /mnt/mycephfs
[root@node ~]# mkdir /mnt/mycephfs/.snap/snap-name
```

Authorize the client to make snapshots for the CephFS file system with the `s` option.

```
[ceph: root@target /]# ceph fs authorize fs-name client path rws
```

To restore a file, copy it from the snapshot directory to another normal directory.

```
[root@node ~]# cp -a .snap/snap-name/file-name .
```

To fully restore a snapshot from the `.snap` directory tree, replace the normal entries with copies from the chosen snapshot.

```
[root@node ~]# rm -rf *
[root@node ~]# cp -a .snap/snap-name/* .
```

To discard a snapshot, remove the corresponding directory in `.snap`. The `rmdir` command succeeds even if the snapshot directory is not empty, without needing to use a recursive `rm` command.

```
[root@node ~]# rmdir .snap/snap-name
```

Scheduling Snapshots

You can use CephFS to schedule snapshots. The `snapshot-schedule` module manages the scheduled snapshots. You can use this module to create and delete snapshot schedules. Snapshot schedule information is stored in the CephFS metadata pool.

To create a snapshot schedule, first enable the `snapshot-schedule` module on the MGR node.

```
[ceph: root@server /]# ceph mgr module enable snap_schedule
```

Then, add the new snapshot schedule.

```
[ceph: root@server /]# ceph fs snap-schedule add fs-path time-period [start-time]
```

If an earlier version than Python 3.7 is installed, then the `start-time` string must use the format `%Y-%m-%dT%H:%M:%S`. For Python version 3.7 or later, you can use more flexible date parsing. For example, to create a snapshot schedule to create a snapshot for the `/volume` folder every hour, you can use the `ceph fs snap-schedule add` command.

```
[ceph: root@server /]# ceph fs snap-schedule add /volume 1h
Schedule set for path /volume
```

On the client node, review the snapshots in the `.snap` folder on your mounted CephFS:

```
[root@node ~]# ls /mnt/mycephfs/.snap  
scheduled-2021-10-06-08_00_00  
scheduled-2021-10-06-09_00_00  
scheduled-2021-10-06-10_00_00
```

You can list the snapshot schedules for a path with the `list` option:

```
[ceph: root@server /]# ceph fs snap-schedule list fs-path
```

Use the `status` option to verify the details for the snapshot schedules.

```
[ceph: root@server /]# ceph fs snap-schedule status fs-path
```

Remove a snapshot schedule by specifying the path.

```
[ceph: root@server /]# ceph fs snap-schedule remove fs-path
```

You can activate and deactivate snapshot schedules through the `activate` and `deactivate` options. When you add a snapshot schedule, it is activated by default if the path exists. However, if the path does not exist, then it is set as inactive, so you can activate it later when you create the path.

```
[ceph: root@server /]# ceph fs snap-schedule activate/deactivate
```



References

`getfattr(1)`, and `setfattr(1)` man pages

For more information, refer to the *Red Hat Ceph Storage 5 File System Guide* at https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/file_system_guide/index

For more information regarding snapshots, refer to the *Ceph File System snapshots* chapter in the *Red Hat Ceph Storage 5 File System Guide* at https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/file_system_guide/index#ceph-file-system-snapshots

► Guided Exercise

Managing Shared File Storage

In this exercise, you configure shared file client access with CephFS.

Outcomes

You should be able to manage CephFS and create snapshots.

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 fileshare-manage
```

Instructions

- The `serverc`, `serverd`, and `servere` nodes are an operational 3-node Ceph cluster. All three nodes operate as a MON, a MGR, and an OSD host with at least one colocated OSD.
- The `clienta` node is set up as your admin node server and you use it to install the Metadata Server (MDS) on `serverc`.
- The `mycephfs` CephFS file system is mounted on the `/mnt/mycephfs` folder with the key ring for the `admin` user.

► 1. Log in to `clienta` as the `admin` user.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$
```

- 2. Create a `/mnt/mycephfs/dir1` directory. Create an empty file `/mnt/mycephfs/dir1/ddtest`. Modify the `ceph.dir.layout.stripe_count` layout attribute for the `/mnt/mycephfs/dir1` directory. Verify that the attribute change does not affect existing files within the directory, but that any newly created files inherit it. The `attr` package is preinstalled for you.
- 2.1. Use the `getfattr` command to create and verify the layout of the `/mnt/cephfs/dir1` directory. This command should not return any layout attribute that is associated with this directory.

```
[admin@clienta ~]$ mkdir /mnt/mycephfs/dir1
[admin@clienta ~]$ touch /mnt/mycephfs/dir1/ddtest
[admin@clienta ~]$ getfattr -n ceph.dir.layout /mnt/mycephfs/dir1
/mnt/mycephfs/dir1: ceph.dir.layout: No such attribute
```

**Note**

If a directory has no layout attributes, then it inherits the layout of its parent. You must set the layout attributes of a directory before you can view them. By default, layout attributes are set on only the top-level directory of the mounted CephFS file system (on the mount point).

- 2.2. Use the `setfattr` command to change the layout of the `/mnt/mycephfs/dir1` directory.

```
[admin@clienta ~]$ setfattr -n ceph.dir.layout.stripe_count \
-v 2 /mnt/mycephfs/dir1
```

- 2.3. Verify the layout of the `/mnt/mycephfs/dir1` directory again. Layout attributes should now be available for this directory. Settings other than the one that you specified are inherited from the closest parent directory where attributes were set.

```
[admin@clienta ~]$ getfattr -n ceph.dir.layout /mnt/mycephfs/dir1
getfattr: Removing leading '/' from absolute path names
# file: mnt/mycephfs/dir1
ceph.dir.layout="stripe_unit=4194304 stripe_count=2 object_size=4194304
pool=mycephfs_data"
```

- 2.4. Verify the layout for the `/mnt/mycephfs/dir1/ddtest` file and notice that the layout does not change for this file, which existed before the layout change.

```
[admin@clienta ~]$ getfattr -n ceph.file.layout /mnt/mycephfs/dir1/ddtest
getfattr: Removing leading '/' from absolute path names
# file: mnt/mycephfs/dir1/ddtest
ceph.file.layout="stripe_unit=4194304 stripe_count=1 object_size=4194304
pool=mycephfs_data"
```

- 2.5. Create a file called `anewfile` under the `/mnt/cephfs/dir1/` directory. Notice how the `stripe_count` for the layout of this file matches the new layout of the `/mnt/cephfs/dir1/` directory. The new file inherits the layout attributes from its parent directory at creation time.

```
[admin@clienta ~]$ touch /mnt/mycephfs/dir1/anewfile
[admin@clienta ~]$ getfattr -n ceph.file.layout /mnt/mycephfs/dir1/anewfile
getfattr: Removing leading '/' from absolute path names
# file: mnt/mycephfs/dir1/anewfile
ceph.file.layout="stripe_unit=4194304 stripe_count=2 object_size=4194304
pool=mycephfs_data"
```

- 2.6. Modify the layout of the `/mnt/mycephfs/dir1/anewfile` file so that the `stripe_count` value is 3. You can modify a specific file layout if the file is empty. Ensure that the new value for `stripe_count` is correct.

```
[admin@clienta ~]$ setfattr -n ceph.file.layout.stripe_count \
-v 3 /mnt/mycephfs/dir1/anewfile
[admin@clienta ~]$ getfattr -n ceph.file.layout /mnt/mycephfs/dir1/anewfile
getfattr: Removing leading '/' from absolute path names
# file: mnt/mycephfs/dir1/anewfile
ceph.file.layout="stripe_unit=4194304 stripe_count=3 object_size=4194304
pool=mycephfs_data"
```

- 2.7. Ensure that the /mnt/mycephfs/dir1/anewfile file is not empty. Verify that you cannot modify a specific file's layout attributes if the file is not empty.

```
[admin@clienta ~]$ echo "Not empty" > /mnt/mycephfs/dir1/anewfile
[admin@clienta ~]$ setfattr -n ceph.file.layout.stripe_count \
-v 4 /mnt/mycephfs/dir1/anewfile
setfattr: /mnt/mycephfs/dir1/anewfile: Directory not empty
```

- 2.8. Clear the configured layout attributes for the /mnt/mycephfs/dir1 directory. Verify the new layout settings (that are inherited from the top of the CephFS file system) by creating a file called a3rdfile.

```
[admin@clienta ~]$ setfattr -x ceph.dir.layout /mnt/mycephfs/dir1
[admin@clienta ~]$ touch /mnt/mycephfs/dir1/a3rdfile
[admin@clienta ~]$ getfattr -n ceph.file.layout /mnt/mycephfs/dir1/a3rdfile
getfattr: Removing leading '/' from absolute path names
# file: mnt/mycephfs/dir1/a3rdfile
ceph.file.layout="stripe_unit=4194304 stripe_count=1 object_size=4194304
pool=mycephfs_data"
```

- ▶ 3. Create a snapshot for your CephFS file system. Mount the CephFS file system on /mnt/mycephfs as the user `restricteduser`.

- 3.1. Unmount the /mnt/mycephfs folder and mount again the CephFS file system to that folder as the user `restricteduser`.

```
[admin@clienta ~]$ sudo umount /mnt/mycephfs
[admin@clienta ~]$ sudo mount.ceph serverc.lab.example.com:/ \
/mnt/mycephfs -o name=restricteduser
```

- 3.2. Navigate to the hidden .snap folder in /mnt/mycephfs.

```
[admin@clienta ~]$ cd /mnt/mycephfs/.snap
```

- 3.3. Create a `mysnapshot` folder, which is the snapshot name. As the user `restricteduser` does not currently have snapshot permissions, you must grant snapshot permissions to this user. After you set the permissions, you must remount the CephFS file system to use the new permissions.

```
[admin@clienta .snap]$ mkdir mysnapshot
mkdir: cannot create directory 'mysnapshot': Permission denied
[admin@clienta .snap]$ sudo cephadm shell
[ceph: root@clienta /]# ceph auth get client.restricteduser
```

```
[client.restricteduser]
key = AQBc315hI7PaBRAA9/9fdmj+wjblk+izstA0aQ==
caps mds = "allow r fsname=mycephfs, allow rw fsname=mycephfs path=/dir2"
caps mon = "allow r fsname=mycephfs"
caps osd = "allow rw tag cephfs data=mycephfs"
exported keyring for client.restricteduser
[ceph: root@clienta /]# ceph auth caps client.restricteduser \
mds 'allow rws fsname=mycephfs' \
mon 'allow r fsname=mycephfs' \
osd 'allow rw tag cephfs data=mycephfs'
updated caps for client.restricteduser
[ceph: root@clienta /]# exit
exit
[admin@clienta .snap]$ cd
[admin@clienta ~]$ sudo umount /mnt/mycephfs
[admin@clienta ~]$ sudo mount.ceph serverc.lab.example.com:/ \
/mnt/mycephfs -o name=restricteduser
[admin@clienta ~]$ cd /mnt/mycephfs/.snap
[admin@clienta .snap]$ mkdir mysnapshot
```

- 3.4. Check that the files in the snapshot are the same as the files in the mounted CephFS file system.

```
[admin@clienta .snap]$ tree /mnt/mycephfs
/mnt/mycephfs
└── dir1
    ├── a3rdfile
    ├── anewfile
    └── ddtest

1 directory, 3 files
[admin@clienta .snap]$ tree /mnt/mycephfs/.snap/mysnapshot
/mnt/mycephfs/.snap/mysnapshot
└── dir1
    ├── a3rdfile
    ├── anewfile
    └── ddtest

1 directory, 3 files
```

- ▶ 4. Schedule to create an hourly snapshot of your CephFS file system's root folder.

- 4.1. Use sudo to run the `cephadm` shell. Enable the snapshot module.

```
[admin@clienta .snap]$ sudo cephadm shell
[ceph: root@clienta /]# ceph mgr module enable snap_schedule
```

- 4.2. Schedule to create the snapshot every hour.

```
[ceph: root@clienta /]# ceph fs snap-schedule add / 1h
Schedule set for path /
```

- 4.3. Check that your snapshot schedule is correctly created and in an active state.

```
[ceph: root@clienta /]# ceph fs snap-schedule status /
{"fs": "mycephfs", "subvol": null, "path": "/", "rel_path": "/", "schedule": "1h", "retention": {}, "start": "2021-10-06T00:00:00", "created": "2021-10-06T08:59:16", "first": "2021-10-06T09:00:00", "last": "2021-10-06T09:00:00", "last_pruned": null, "created_count": 1, "pruned_count": 0, "active": true}
```

- 4.4. Exit from the `cephadm` shell. Check that your snapshot is correctly created in your `.snap` folder.

```
[ceph: root@clienta /]# exit
exit
[admin@clienta .snap]$ ls
mysnapshot  scheduled-2021-10-06-09_00_00
[admin@clienta .snap]$ tree
.
├── mysnapshot
│   └── dir1
│       ├── a3rdfile
│       ├── anewfile
│       └── ddtest
└── scheduled-2021-10-06-09_00_00
    └── dir1
        ├── a3rdfile
        ├── anewfile
        └── ddtest

4 directories, 6 files
```

Creating the scheduled snapshot might take time. As you scheduled it every hour, it might take up to one hour to be triggered. You do not have to wait until the snapshot is created.

- 5. Return to `workstation` as the `student` user.

```
[admin@clienta .snap]$ exit
[student@workstation ~]$
```



Warning

Run the `lab finish` script on the `workstation` server so that the `clienta` node can be safely rebooted without mount conflicts.

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 fileshare-manage
```

This concludes the guided exercise.

▶ Lab

Providing File Storage with CephFS

In this lab, you provide file storage by using the kernel client and deploying a Ceph Metadata Server (MDS).

Outcomes

You should be able to deploy an MDS and use the kernel client to mount the CephFS file system.

- The `serverc`, `serverd`, and `servere` nodes are an operational 3-node Ceph cluster. All three nodes operate as a MON, a MGR, and an OSD host with at least one colocated OSD.
- The `clienta` node is set up as your admin node server and you use it to install the MDS on `serverc`.

Before You Begin

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

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

Instructions

1. Log in to `clienta` as the `admin` user. Create the `ceph_data` and `ceph_metadata` pools for CephFS. Create the `mycephfs` CephFS file system. From `clienta`, deploy the MDS to `serverc`. Verify that the MDS is up and active. Verify that the ceph health is OK.
2. On the `clienta` node, create the `/mnt/cephfs-review` mount point and mount the CephFS file system as a kernel client.
3. Create a 10 MB test file called `cephfs.test1`. Verify that the created data is replicated across all three nodes by showing 30 MB in the `cephfs_data` pool.
4. Return to `workstation` as the `student` user.

Evaluation

Grade your work by running the `lab grade fileshare-review` command from your `workstation` machine. Correct any reported failures and rerun the script until successful.

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

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 fileshare-review
```

This concludes the lab.

► Solution

Providing File Storage with CephFS

In this lab, you provide file storage by using the kernel client and deploying a Ceph Metadata Server (MDS).

Outcomes

You should be able to deploy an MDS and use the kernel client to mount the CephFS file system.

- The `serverc`, `serverd`, and `servere` nodes are an operational 3-node Ceph cluster. All three nodes operate as a MON, a MGR, and an OSD host with at least one colocated OSD.
- The `clienta` node is set up as your admin node server and you use it to install the MDS on `serverc`.

Before You Begin

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

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

Instructions

- Log in to `clienta` as the `admin` user. Create the `ceph_data` and `ceph_metadata` pools for CephFS. Create the `mycephfs` CephFS file system. From `clienta`, deploy the MDS to `serverc`. Verify that the MDS is up and active. Verify that the ceph health is OK.

- Log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]#
```

- Create the two required CephFS pools. Name these pools `cephfs_data` and `cephfs_metadata`.

```
[ceph: root@clienta /]# ceph osd pool create cephfs_data
pool 'cephfs_data' created
[ceph: root@clienta /]# ceph osd pool create cephfs_metadata
pool 'cephfs_metadata' created
```

- Create the CephFS file system with the name `mycephfs`. Your pool numbers might differ in your lab environment.

```
[ceph: root@clienta /]# ceph fs new mycephfs cephfs_metadata cephfs_data
new fs with metadata pool 7 and data pool 6
```

- 1.4. Deploy the MDS service on `serverc.lab.example.com`.

```
[ceph: root@clienta /]# ceph orch apply mds mycephfs \
--placement="1 serverc.lab.example.com"
Scheduled mds.mycephfs update...
```

- 1.5. Verify that the MDS service is active. It can take some time until the MDS service is shown.

```
[ceph: root@clienta /]# ceph mds stat
mycephfs:1 {0=mycephfs.serverc.mycctv=up:active}
```

- 1.6. Verify that the cluster health is OK.

```
[ceph: root@clienta /]# ceph status
cluster:
  id:      ff97a876-1fd2-11ec-8258-52540000fa0c
  health:  HEALTH_OK

  services:
    mon: 4 daemons, quorum serverc.lab.example.com,servere,serverd,clienta (age
2h)
    mgr: serverc.lab.example.com.btgxor(active, since 2h), standbys:
          clienta.soxncl, servere.fmyxwv, serverd.ufqxxx
    mds: 1/1 daemons up
    osd: 9 osds: 9 up (since 2h), 9 in (since 36h)
    rgw: 2 daemons active (2 hosts, 1 zones)

  data:
    volumes: 1/1 healthy
    pools:   7 pools, 169 pgs
    objects: 212 objects, 7.5 KiB
    usage:   162 MiB used, 90 GiB / 90 GiB avail
    pgs:     169 active+clean

  io:
    client:  1.1 KiB/s wr, 0 op/s rd, 3 op/s wr
```

2. On the `clienta` node, create the `/mnt/cephfs-review` mount point and mount the CephFS file system as a kernel client.

- 2.1. Exit the `cephadm` shell. Verify that the Ceph client key ring is present in the `/etc/ceph` folder on the client node.

```
[ceph: root@clienta /]# exit
exit
[admin@clienta ~]$ sudo ls -l /etc/ceph
total 12
-rw-r--r-- 1 root root 63 Sep 27 16:42 ceph.client.admin.keyring
-rw-r--r-- 1 root root 177 Sep 27 16:42 ceph.conf
-rw----- 1 root root 82 Sep 27 16:42 podman-auth.json
```

- 2.2. Install the `ceph-common` package in the client node.

```
[admin@clienta ~]$ sudo yum install -y ceph-common
...output omitted...
Complete!
```

- 2.3. Create the /mnt/cephfs-review mount point directory. Mount the new CephFS file system as a kernel client.

```
[admin@clienta ~]$ sudo mkdir /mnt/cephfs-review
[admin@clienta ~]$ sudo mount.ceph serverc.lab.example.com:/ /mnt/cephfs-review \
-o name=admin
```

- 2.4. Change the ownership of the top-level directory of the mounted file system to user and group admin.

```
[admin@clienta ~]$ sudo chown admin:admin /mnt/cephfs-review
```

3. Create a 10 MB test file called cephfs.test1. Verify that the created data is replicated across all three nodes by showing 30 MB in the cephfs_data pool.

- 3.1. Use the dd command to create one 10 MB file, and then verify that it triples across the OSD nodes.

```
[admin@clienta ~]$ dd if=/dev/zero of=/mnt/cephfs-review/cephfs.test1 \
bs=1M count=10
10+0 records in
10+0 records out
10485760 bytes (10 MB, 10 MiB) copied, 0.0291862 s, 359 MB/s
[admin@clienta ~]$ sudo cephadm shell -- ceph fs status
Inferring fsid ff97a876-1fd2-11ec-8258-52540000fa0c
Inferring config /var/lib/ceph/ff97a876-1fd2-11ec-8258-52540000fa0c/mon.clienta/
config
Using recent ceph image registry.redhat.io/rhceph/rhceph-5-
rhel8@sha256:6306...47ff
mycephfs - 1 clients
=====
RANK STATE MDS ACTIVITY DNS INOS DIRS CAPS
0 active mycephfs.serverc.nsihbi Reqs: 0 /s 11 14 12 2
POOL TYPE USED AVAIL
cephfs_metadata metadata 120k 28.4G
cephfs_data data 30.0M 28.4G
MDS version: ceph version 16.2.0-117.el8cp
(0e34bb74700060ebfaa22d99b7d2cdc037b28a57) pacific (stable)
```

4. Return to workstation as the student user.

```
[admin@clienta ~]$ exit
[student@workstation ~]$
```

Evaluation

Grade your work by running the `lab grade fileshare-review` command from your workstation machine. Correct any reported failures and rerun the script until successful.

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

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 fileshare-review
```

This concludes the lab.

Summary

In this chapter, you learned:

- You can distinguish the different characteristics for file-based, block-based, and object-based storage.
- CephFS is a POSIX-compliant file system that is built on top of RADOS to provide file-based storage.
- CephFS requires at least one Metadata Server that is separate from file data.
- Deploying CephFS requires multiple steps:
 - Create two pools, one for CephFS data and another for CephFS metadata.
 - Start the MDS service on the hosts.
 - Create a CephFS file system.
- You can mount CephFS file systems with either of the two available clients:
 - The kernel client, which does not support quotas but is faster.
 - The FUSE client, which supports quotas as ACLs but is slower.
- NFS Ganesha is a user space NFS file server for accessing Ceph storage.
- CephFS supports multisite geo-replication with snapshots.
- You can determine which OSDs store a file's objects.
- You can modify the RADOS layout to control how files are mapped to objects.
- CephFS enables asynchronous snapshots by creating a folder in the hidden `.snap` folder.
- You can schedule snapshots for your CephFS file system.

Chapter 11

Managing a Red Hat Ceph Storage Cluster

Goal

Manage an operational Ceph cluster using tools to check status, monitor services, and properly start and stop all or part of the cluster. Perform cluster maintenance by replacing or repairing cluster components, including MONs, OSDs, and PGs.

Objectives

- Administer and monitor a Red Hat Ceph Storage cluster, including starting and stopping specific services or the full cluster, and querying cluster health and utilization.
- Perform common cluster maintenance tasks, such as adding or removing MONs and OSDs, and recovering from various component failures.

Sections

- Performing Cluster Administration and Monitoring (and Guided Exercise)
- Performing Cluster Maintenance Operations (and Guided Exercise)

Lab

Managing a Red Hat Ceph Storage Cluster

Performing Cluster Administration and Monitoring

Objectives

After completing this section, you should be able to administer and monitor a Red Hat Ceph Storage cluster, including starting and stopping specific services or the full cluster, and querying cluster health and utilization.

Defining the Ceph Manager (MGR)

The role of the Red Hat Ceph Storage Manager (MGR) is to collect cluster statistics.

Client I/O operations continue normally while MGR nodes are down, but queries for cluster statistics fail. Deploy at least two MGRs for each cluster to provide high availability. MGRs are typically run on the same hosts as MON nodes, but it is not required.

The first MGR daemon that is started in a cluster becomes the active MGR and all other MGRs are on standby. If the active MGR does not send a beacon within the configured time interval, a standby MGR takes over. You can configure the `mon_mgr_beacon_grace` setting to change the beacon time interval if needed. The default value is 30 seconds.

Use the `ceph mgr fail <MGR_NAME>` command to manually failover from the active MGR to a standby MGR.

Use the `ceph mgr stat` command to view the status of the MGRs.

```
[ceph: root@node /]# ceph mgr stat
{
    "epoch": 32,
    "available": true,
    "active_name": "mgr1",
    "num_standby": 3
}
```

Ceph MGR Modules

The Ceph MGR has a modular architecture. You can enable or disable modules as needed. The MGR collects cluster statistical data and can send the data to external monitoring and management systems.

View the modules that are available and enabled by using the `ceph mgr module ls` command.

View published addresses for specific modules, such as the Dashboard module URL, by using the `ceph mgr services` command.

The Ceph Dashboard Module

The Ceph Dashboard provides cluster management and monitoring through a browser-based user interface. The Dashboard enables viewing cluster statistics and alerts, and performing selected cluster management tasks. The Ceph Dashboard requires an active MGR daemon with the Dashboard MGR module enabled.

The Dashboard relies on the *Prometheus* and *Grafana* services to display collected monitoring data and to generate alerts. Prometheus is an open source monitoring and alerting tool. Grafana is an open source statistical graphing tool.

The Dashboard supports alerts based on Ceph metrics and configured thresholds. The Prometheus AlertManager component configures, gathers, and triggers the alerts. Alerts are displayed in the Dashboard as notifications. You can view details of recent alerts and mute alerts.

Monitoring Cluster Health

You can use the `ceph health` command to quickly verify the state of the cluster. This command returns one of the following states:

- `HEALTH_OK` indicates that the cluster is operating normally.
- `HEALTH_WARN` indicates that the cluster is in a warning condition. For example, an OSD is down, but there are enough OSDs working properly for the cluster to function.
- `HEALTH_ERR` indicates that the cluster is in an error condition. For example, a full OSD could have an impact on the functionality of the cluster.

If the Ceph cluster is in a warning or an error state, the `ceph health detail` command provides additional details.

```
[ceph: root@node /]# ceph health detail
```

The `ceph -w` command displays additional real-time monitoring information about the events happening in the Ceph cluster.

```
[ceph: root@node /]# ceph -w
```

This command provides the status of cluster activities, such as the following details:

- Data rebalancing across the cluster
- Replica recovery across the cluster
- Scrubbing activity
- OSDs starting and stopping

To monitor the `cephadm` log, use the `ceph -w cephadm` command. Use the `ceph log last cephadm` to view the most recent log entries.

```
[ceph: root@node /]# ceph -w cephadm
```

Managing Ceph Services

Containerized services are controlled by `systemd` on the container host system. Run `systemctl` commands on the container host system to start, stop, or restart cluster daemons.

Cluster daemons are referred to by the type of `$daemon` and the daemon `$id`. The type of `$daemon` is `mon`, `mgr`, `mds`, `osd`, `rgw`, `rbd-mirror`, `crash`, or `cephfs-mirror`.

The daemon `$id` for MON, MGR, and RGW is the host name. The daemon `$id` for OSD is the OSD ID. The daemon `$id` for MDS is the file system name followed by the host name.

Use the `ceph orch ps` command to list all cluster daemons. Use the `--daemon_type=DAEMON` option to filter for a specific daemon type.

NAME	HOST	STATUS	REFRESHED	AGE	PORTS	VERSION	IMAGE ID
CONTAINER ID							
osd.0	node1	running (14h)	2m ago	2d	-	16.2.0-117.el8cp	2142b60d7974 7b1e76ef06d1
osd.1	node1	running (14h)	2m ago	2d	-	16.2.0-117.el8cp	2142b60d7974 85fb30af4ec2
osd.2	node1	running (13h)	2m ago	2d	-	16.2.0-117.el8cp	2142b60d7974 bb66b3b6107c
osd.3	node2	running (14h)	2m ago	2d	-	16.2.0-117.el8cp	2142b60d7974 1f63f7fb88f4
osd.4	node2	running (14h)	2m ago	2d	-	16.2.0-117.el8cp	2142b60d7974 3f1c54eeee927
osd.5	node2	running (14h)	2m ago	2d	-	16.2.0-117.el8cp	2142b60d7974 366d5208c73f
osd.6	node3	running (14h)	2m ago	2d	-	16.2.0-117.el8cp	2142b60d7974 5e5f9cde6c55
osd.7	node3	running (14h)	2m ago	2d	-	16.2.0-117.el8cp	2142b60d7974 0824d7e78aa0
osd.8	node3	running (14h)	2m ago	2d	-	16.2.0-117.el8cp	2142b60d7974 f85c8af8996d

To stop, start, or restart a daemon on a host, use `systemctl` commands and the daemon name. To list the names of all daemons on a cluster host, run the `systemctl list-units` command and search for `ceph`.

The cluster `fsid` is in the daemon name. Some service names end in a random six character string to distinguish individual services of the same type on the same host.

[root@node ~]# systemctl list-units 'ceph*'		
UNIT	ACTIVE SUB DESCRIPTION	LOAD
ceph-ff97a876-1fd2-11ec-8258-52540000fa0c@crash.clienta.service	active running Ceph crash.clienta...	loaded
ceph-ff97a876-1fd2-11ec-8258-52540000fa0c@mgr.clienta.soxncl.service	active running Ceph mgr.clienta.soxncl...	loaded
ceph-ff97a876-1fd2-11ec-8258-52540000fa0c@mon.clienta.service	active running Ceph mon.clienta...	loaded
ceph-ff97a876-1fd2-11ec-8258-52540000fa0c@node-exporter.clienta.service	active running Ceph node-exporter...	loaded
ceph-ff97a876-1fd2-11ec-8258-52540000fa0c.target	active active Ceph cluster...	loaded
ceph.target	active active All Ceph...	loaded
<i>...output omitted...</i>		

Use the `ceph.target` command to manage all the daemons on a cluster node.

```
[root@node ~]# systemctl restart ceph.target
```

You can also use the `ceph orch` command to manage cluster services. First, obtain the service name by using the `ceph orch ls` command. For example, find the service name for cluster OSDs and restart the service.

```
[ceph: root@node /]# ceph orch ls
NAME          RUNNING  REFRESHED  AGE   PLACEMENT
alertmanager    1/1     2s ago    2d    count:1
crash          4/4     3s ago    2d    *
grafana         1/1     2s ago    2d    count:1
mds.fs1          3/3     3s ago   114m  node1;node2;node3;count:3
mgr             4/4     3s ago    2d    node1;node2;node3;node4
mon             4/4     3s ago    2d    node1;node2;node3;node4
node-exporter    4/4     3s ago    2d    *
osd.default_drive_group  8/12    3s ago    2d    server*
prometheus      1/1     2s ago    2d    count:1
rgw.realm.zone  2/2     3s ago    2d    node3;node4

[ceph: root@node /]# ceph orch restart osd.default_drive_group
Scheduled to restart osd.0 on host 'node1'
Scheduled to restart osd.1 on host 'node1'
Scheduled to restart osd.2 on host 'node1'
Scheduled to restart osd.3 on host 'node2'
Scheduled to restart osd.5 on host 'node2'
Scheduled to restart osd.7 on host 'node2'
Scheduled to restart osd.4 on host 'node3'
Scheduled to restart osd.6 on host 'node3'
Scheduled to restart osd.8 on host 'node3'
```

You can manage an individual cluster daemon by using the `ceph orch daemon` command.

```
[ceph: root@node /]# ceph orch daemon restart osd.1
```

Powering Down or Restarting the Cluster

Ceph supports cluster flags to control the behavior of the cluster. You must set some flags when restarting the cluster or performing cluster maintenance. You can use cluster flags to limit the impact of a failed cluster component or to prevent cluster performance issues.

Use the `ceph osd set` and `ceph osd unset` commands to manage these flags:

noup

Do not automatically mark a starting OSD as up. If the cluster network is experiencing latency issues, OSDs can mark each other down on the MON, then mark themselves up. This scenario is called *flapping*. Set the `noup` and `nodown` flags to prevent flapping.

nodown

The `nodown` flag tells the Ceph MON to mark a stopping OSD with the `down` state. Use the `nodown` flag when performing maintenance or a cluster shutdown. Set the `nodown` flag to prevent flapping.

noout

The `noout` flag tells the Ceph MON not to remove any OSDs from the CRUSH map, which prevents CRUSH from automatically rebalancing the cluster when OSDs are stopped. Use the `noout` flag when performing maintenance on a subset of the cluster. Clear the flag after the OSDs are restarted.

noin

The `noin` flag prevent booting OSDs from being marked with the `in` state. The flag prevents data from being automatically allocated to that specific OSD.

norecover

The `norecover` flag prevents recovery operations from running. Use the `norecover` flag when performing maintenance or a cluster shutdown.

nobackfill

The `nobackfill` flag prevents backfill operations from running. Use the `nobackfill` flag when performing maintenance or a cluster shutdown. Backfilling is discussed later in this section.

norebalance

The `norebalance` flag prevents rebalancing operations from running. Use the `norebalance` flag when performing maintenance or a cluster shutdown.

noscrub

The `noscrub` flag prevents scrubbing operations from running. Scrubbing will be discussed later in this section.

nodeep-scrub

The `nodeep-scrub` flag prevents any deep-scrubbing operation from running. Deep-scrubbing is discussed later in this section.

Cluster Power Down

Perform the following steps to shut down the entire cluster:

- Prevent clients from accessing the cluster.
- Ensure that the cluster is in a healthy state (HEALTH_OK) and that all PGs are in an `active +clean` state before proceeding.
- Bring down CephFS.
- Set the `noout`, `norecover`, `norebalance`, `nobackfill`, `nodown`, and `pause` flags.
- Shut down all Ceph Object Gateways (RGW) and iSCSI Gateways.
- Shut down OSD nodes one by one.
- Shut down MON and MGR nodes one by one.
- Shut down the admin node.

Cluster Power Up

Perform the following steps to power on the cluster:

- Power up cluster nodes in the following order: admin node, MON and MGR nodes, OSD nodes, MDS nodes.
- Clear the `noout`, `norecover`, `norebalance`, `nobackfill`, `nodown` and `pause` flags.
- Bring up Ceph Object Gateways and iSCSI Gateways.
- Bring up CephFS.

Monitoring the Cluster

View the MON quorum status with the `ceph mon stat` or the `ceph quorum_status -f json-pretty` commands.

```
[ceph: root@node /]# ceph mon stat
```

```
[ceph: root@node /]# ceph quorum_status -f json-pretty
```

You can also view the status of MONs in the Dashboard.

Viewing Daemon Logs

To view daemon logs, use the `journalctl -u $daemon@$id` command. To show only recent journal entries, use the `-f` option. For example, this example views logs for that host's OSD 10 daemon.

```
[root@node ~]$ journalctl -u \
ceph-ff97a876-1fd2-11ec-8258-52540000fa0c@osd.10.service
```

Ceph containers write to individual log files for each daemon. Enable logging for each specific Ceph daemon by configuring the daemon's `log_to_file` setting to `true`. This example enables logging for MON nodes.

```
[ceph: root@node /]# ceph config set mon log_to_file true
```

Monitoring OSDs

If the cluster is not healthy, Ceph displays a detailed status report containing the following information:

- Current status of the OSDs (up/down/out/in)
- OSD near capacity limit information (nearfull/full)
- Current status of the placement groups (PGs)

The `ceph status` and `ceph health` commands report space-related warning or error conditions. The various `ceph osd` subcommands report OSD usage details, status, and location information.

Analyzing OSD Usage

The `ceph osd df` command displays OSD usage statistics. Use the `ceph osd df tree` command to display the CRUSH tree in the command output.

```
[ceph: root@node /]# ceph osd df
ID CLASS WEIGHT  REWEIGHT SIZE    RAW USE DATA     OMAP      META     AVAIL     %USE
  VAR   PGS STATUS
 0   hdd 0.00980  1.00000 10 GiB 1.0 GiB  28 MiB  20 KiB 1024 MiB 9.0 GiB 10.28
    1.00  41      up
 1   hdd 0.00980  1.00000 10 GiB 1.0 GiB  29 MiB  40 KiB 1024 MiB 9.0 GiB 10.29
    1.00  58      up
 2   hdd 0.00980  1.00000 10 GiB 1.0 GiB  28 MiB  20 KiB 1024 MiB 9.0 GiB 10.28
    1.00  30      up
 3   hdd 0.00980  1.00000 10 GiB 1.0 GiB  28 MiB  20 KiB 1024 MiB 9.0 GiB 10.28
    1.00  43      up
 4   hdd 0.00980  1.00000 10 GiB 1.0 GiB  28 MiB  20 KiB 1024 MiB 9.0 GiB 10.28
    1.00  46      up
 5   hdd 0.00980  1.00000 10 GiB 1.0 GiB  28 MiB  20 KiB 1024 MiB 9.0 GiB 10.28
    1.00  40      up
 6   hdd 0.00980  1.00000 10 GiB 1.0 GiB  29 MiB  44 KiB 1024 MiB 9.0 GiB 10.28
    1.00  44      up
```

```

7  hdd 0.00980 1.00000 10 GiB 1.0 GiB 28 MiB 44 KiB 1024 MiB 9.0 GiB 10.28
  1.00 38      up
8  hdd 0.00980 1.00000 10 GiB 1.0 GiB 28 MiB 44 KiB 1024 MiB 9.0 GiB 10.28
  1.00 47      up
          TOTAL 90 GiB 9.2 GiB 255 MiB 274 KiB 9.0 GiB 81 GiB 10.28

```

The following table describes each column of the command output:

Output column	Description
ID	The OSD ID.
CLASS	The type of devices that the OSD uses (HDD, SSD, or NVMe).
WEIGHT	The weight of the OSD in the CRUSH map. By default, this is set to the OSD capacity in TB and is changed by using the <code>ceph osd crush reweight</code> command. The weight determines how much data CRUSH places onto the OSD relative to other OSDs. For example, two OSDs with the same weight receive roughly the same number of I/O requests and store approximately the same amount of data.
REWEIGHT	Either the default reweight value or the actual value set by the <code>ceph osd reweight</code> command. You can reweight an OSD to temporarily override the CRUSH weight.
SIZE	The total OSD storage capacity.
RAW USE	The utilized OSD storage capacity.
DATA	OSD capacity used by user data.
OMAP	The BlueFS storage that is used to store object map (OMAP) data, which are the key-value pairs stored in RocksDB.
META	The total BlueFS space allocated, or the value of the <code>bluestore_bluefs_min</code> setting, whichever is larger. This is the internal BlueStore metadata, which is calculated as the total space allocated to BlueFS minus the estimated OMAP data size.
AVAIL	Free space available on the OSD.
%USE	The percentage of storage capacity used on the OSD.
VAR	The variation above or below the average OSD utilization.
PGS	The number of placement groups on the OSD.
STATUS	The status of the OSD.

Use the `ceph osd perf` command to view OSD performance statistics.

```
[ceph: root@node /]# ceph osd perf
```

Interpreting OSD Status

An OSD daemon can be in one of four states, based on the combination of these two flags:

- **down** or **up** - indicating whether the daemon is running and communicating with the MONs.
- **out** or **in** - indicating whether the OSD is participating in cluster data placement.

The state of an OSD in normal operation is up and in.

If an OSD fails and the daemon goes offline, the cluster might report it as down and in for a short period of time. This is intended to give the OSD a chance to recover on its own and rejoin the cluster, avoiding unnecessary recovery traffic.

For example, a brief network interruption might cause the OSD to lose communication with the cluster and be temporarily reported as down. After a short interval controlled by the `mon_osd_down_out_interval` configuration option (five minutes by default), the cluster reports the OSD as down and out. At this point, the placement groups assigned to the failed OSD are migrated to other OSDs.

If the failed OSD then returns to the up and in states, the cluster reassigned placement groups based on the new set of OSDs and by rebalancing the objects in the cluster.



Note

Use the `ceph osd set noout` and `ceph osd unset noout` commands to enable or disable the `noout` flag on the cluster. However, the `ceph osd out osdid` command tells the Ceph cluster to ignore an OSD for data placement and marks the OSD with the `out` state.

OSDs verify each other's status at regular time intervals (six seconds by default). They report their status to the MONs every 120 seconds, by default. If an OSD is down, the other OSDs or the MONs do not receive heartbeat responses from that down OSD.

The following configuration settings manage OSD heartbeats:

Configuration option	Description
<code>osd_heartbeat_interval</code>	Number of seconds between OSD peer checks.
<code>osd_heartbeat_grace</code>	Number of seconds before an unresponsive OSD moves to the <code>down</code> state.
<code>mon_osd_min_down_reporters</code>	Number of peers reporting that an OSD is down before a MON considers it to be down.
<code>mon_osd_min_down_reports</code>	Number of times an OSD is reported to be down before a MON considers it to be down.
<code>mon_osd_down_out_subtree_limit</code>	Prevents a CRUSH unit type (such as a host) from being automatically marked as <code>out</code> when it fails.
<code>osd_mon_report_interval_min</code>	A newly booted OSD has to report to a MON within this number of seconds.
<code>osd_mon_report_interval_max</code>	Maximum number of seconds between reports from an OSD to a MON.

Configuration option	Description
<code>osd_mon_heartbeat_interval</code>	Ceph monitor heartbeat interval.
<code>mon_osd_report_timeout</code>	The time-out (in seconds) before the MON marks an OSD as down if it does not report.

Monitoring OSD Capacity

Red Hat Ceph Storage provides configuration parameters to help prevent data loss due to a lack of storage space in the cluster. You can set these parameters to provide an alert when OSDs are low on storage space.

When the value of the `mon_osd_full_ratio` setting is reached or exceeded, the cluster stops accepting write requests from clients and enters the `HEALTH_ERR` state. The default full ratio is 0.95 (95%) of the available storage space in the cluster. Use the full ratio to reserve enough space so that if OSDs fail, there is enough space left that automatic recovery succeeds without running out of space.

The `mon_osd_nearfull_ratio` setting is a more conservative limit. When the value of the `mon_osd_nearfull_ratio` limit is reached or exceeded, the cluster enters the `HEALTH_WARN` state. This is intended to alert you to the need to add OSDs to the cluster or fix issues before you reach the full ratio. The default near full ratio is 0.85 (85%) of the available storage space in the cluster.

The `mon_osd_backfillfull_ratio` setting is the threshold at which cluster OSDs are considered too full to begin a backfill operation. The default backfill full ratio is 0.90 (90%) of the available storage space in the cluster.

Use the `ceph osd set-full-ratio`, `ceph osd set-nearfull-ratio`, and `ceph osd set-backfillfull-ratio` commands to configure these settings.

```
[ceph: root@node /]# ceph osd set-full-ratio .85
```

```
[ceph: root@node /]# ceph osd set-nearfull-ratio .75
```

```
[ceph: root@node /]# ceph osd set-backfillfull-ratio .80
```



Note

The default ratio settings are appropriate for small clusters, such as the one used in this lab environment. Production clusters typically require lower ratios.

Different OSDs might be at `full` or `nearfull` depending on exactly what objects are stored in which placement groups. If you have some OSDs `full` or `nearfull` and others with plenty of space remaining, analyze your placement group distribution and CRUSH map weights.

Monitoring Placement Groups

Every placement group (PG) has a status string assigned to it that indicates its health state. When all placement groups are in the `active+clean` state, the cluster is healthy. A PG status

of scrubbing or deep-scrubbing can also occur in a healthy cluster and does not indicate a problem.

Placement group *scrubbing* is a background process that verifies data consistency by comparing an object's size and other metadata with its replicas on other OSDs and reporting inconsistencies. *Deep scrubbing* is a resource-intensive process that compares the contents of data objects by using a bitwise comparison and recalculates checksums to identify bad sectors on the drive.



Note

Although scrubbing operations are critical to maintain a healthy cluster, they have a performance impact, particularly deep scrubbing. Schedule scrubbing to avoid peak I/O times. Temporarily prevent scrub operations with the `noscrub` and `nodeep-scrub` cluster flags.

Placement groups can have the following states:

PG state	Description
<code>creating</code>	PG creation is in progress.
<code>peering</code>	The OSDs are being brought into agreement about the current state of the objects in the PG.
<code>active</code>	Peering is complete. The PG is available for read and write requests.
<code>clean</code>	The PG has the correct number of replicas and there are no stray replicas.
<code>degraded</code>	The PG has objects with an incorrect number of replicas.
<code>recovering</code>	Objects are being migrated or synchronized with replicas.
<code>recovery_wait</code>	The PG is waiting for local or remote reservations.
<code>undersized</code>	The PG is configured to store more replicas than there are OSDs available to the placement group.
<code>inconsistent</code>	Replicas of this PG are not consistent. One or more replicas in the PG are different, indicating some form of corruption of the PG.
<code>replay</code>	The PG is waiting for clients to replay operations from a log after an OSD crash.
<code>repair</code>	The PG is scheduled for repair.
<code>backfill</code> , <code>backfill_wait</code> , <code>backfill_toofull</code>	A backfill operation is waiting, occurring, or unable to complete due to insufficient storage.
<code>incomplete</code>	The PG is missing information from its history log about writes that might have occurred. This could indicate that an OSD has failed or is not started.
<code>stale</code>	The PG is in an unknown state (OSD report time-out).

PG state	Description
inactive	The PG has been inactive for too long.
unclean	The PG has been unclean for too long.
remapped	The acting set has changed, and the PG is temporarily remapped to a different set of OSDs while the primary OSD recovers or backfills.
down	The PG is offline.
splitting	The PG is being split; the number of PGs is being increased.
scrubbing, deep-scrubbing	A PG scrub or deep-scrub operation is in progress.

When an OSD is added to a placement group, the PG enters the `peering` state to ensure that all nodes agree about the state of the PG. If the PG can handle read and write requests after peering completes, then it reports an `active` state. If the PG also has the correct number of replicas for all of its objects, then it reports a `clean` state. The normal PG operating state after writes are complete is `active+clean`.

When an object is written to the PG's primary OSD, the PG reports a `degraded` state until all replica OSDs acknowledge that they have also written the object.

The `backfill` state means that data is being copied or migrated to rebalance PGs across OSDs. If a new OSD is added to the PG, it is gradually backfilled with objects to avoid excessive network traffic. Backfilling occurs in the background to minimize the performance impact on the cluster. The `backfill_wait` state indicates that a backfill operation is pending. The `backfill` state indicates that a backfill operation is in progress. The `backfill_too_full` state indicates that a backfill operation was requested, but could not be completed due to insufficient storage capacity.

A PG marked as `inconsistent` might have replicas that are different from the others, detected as a different data checksum or metadata size on one or more replicas. A clock skew in the Ceph cluster and corrupted object content can also cause an `inconsistent` PG state.

Identifying Stuck Placement Groups

The placement groups transition into degraded or peering states after a failure. If a placement group remains in one of these states for a long period, then the MON marks the placement group as `stuck`. A stuck PG might be in one or more of the following states:

- An `inactive` PG might be having a peering problem.
- An `unclean` PG might be having problems recovering after a failure.
- A `stale` PG has no OSDs reporting, which might indicate that all OSDs are `down` and `out`.
- An `undersized` PG does not have enough OSDs to store the configured number of replicas.



Note

The MONs use the `mon_pg_stuck_threshold` parameter to decide if a PG has been in an error state for too long. The default value for the threshold is 300 seconds.

Ceph marks a PG as `stale` when all OSDs that have copies of a specific PG are in `down` and `out` states. To return from a `stale` state, an OSD must be revived to have a PG copy available and for

PG recovery to begin. If the situation remains unresolved, the PG is inaccessible and I/O requests to the PG hang.

By default, Ceph performs an automatic recovery. If recovery fails for any PGs, the cluster status continues to display `HEALTH_ERR`.

Ceph can declare that an OSDs or a PG is lost, which might result in data loss. To determine the affected OSDs, first retrieve an overview of cluster status with the `ceph health detail` command. Then, use the `ceph pg dump_stuck option` command to inspect the state of PGs.



Note

If many PGs remain in the peering state, the `ceph osd blocked-by` command displays the OSD that is preventing the OSD peering.

Inspect the PG using either the `ceph pg dump | grep pgid` or the `ceph pg query pgid` command. The OSDs hosting the PG are displayed in square brackets ([]).

To mark a PG as lost, use the `ceph pg pgid mark_unfound_lost revert|delete` command. To mark an OSD as lost, use the `ceph osd lost osdid --yes-i-really-mean-it` command. The state of the OSD must be down and out.

Upgrading the Cluster

Use the `ceph orch upgrade` command to upgrade your Red Hat Ceph Storage 5 cluster.

First, update `cephadm` by running the `cephadm-ansible` preflight playbook with the `upgrade_ceph_packages` option set to `true`.

```
[root@node ~]# ansible-playbook -i /etc/ansible/hosts/ cephadm-preflight.yml \
--extra-vars "ce ph_origin=rhcs upgrade_ceph_packages=true"
```

Then run the `ceph orch upgrade start --ceph-version VERSION` command using the name of the new version.

```
[ceph: root@node /]# ceph orch upgrade start --ceph-version 16.2.0-117.el8cp
```

Run the `ceph status` command to view the progress of the upgrade.

```
[ceph: root@node /]# ceph status
...output omitted...
progress:
  Upgrade to 16.2.0-115.el8cp (1s)
  [.....]
```

Do not mix clients and cluster nodes that use different versions of Red Hat Ceph Storage in the same cluster. Clients include RADOS gateways, iSCSI gateways, and other applications that use `librados`, `librbd`, or `libceph`.

Use the `ceph versions` command after a cluster upgrade to verify that matching versions are installed.

```
[ceph: root@node /]# ceph versions
{
    "mon": {
        "ceph version 16.2.0-117.el8cp (0e34bb74700060ebfaa22d99b7d2cdc037b28a57)
pacific (stable)": 4
    },
    "mgr": {
        "ceph version 16.2.0-117.el8cp (0e34bb74700060ebfaa22d99b7d2cdc037b28a57)
pacific (stable)": 4
    },
    "osd": {
        "ceph version 16.2.0-117.el8cp (0e34bb74700060ebfaa22d99b7d2cdc037b28a57)
pacific (stable)": 9
    },
    "mds": {
        "ceph version 16.2.0-117.el8cp (0e34bb74700060ebfaa22d99b7d2cdc037b28a57)
pacific (stable)": 3
    },
    "rgw": {
        "ceph version 16.2.0-117.el8cp (0e34bb74700060ebfaa22d99b7d2cdc037b28a57)
pacific (stable)": 2
    },
    "overall": {
        "ceph version 16.2.0-117.el8cp (0e34bb74700060ebfaa22d99b7d2cdc037b28a57)
pacific (stable)": 22
    }
}
```

Using the Balancer Module

Red Hat Ceph Storage provides a MGR module called `balancer` that automatically optimizes the placement of PGs across OSDs to achieve a balanced distribution. This module can also be run manually.

The `balancer` module does not run if the cluster is not in the `HEALTH_OK` state. When the cluster is healthy, it throttles its changes so that it keeps the number of PGs that need to be moved under a 5% threshold. Configure the `target_max_misplaced_ratio` MGR setting to adjust this threshold:

```
[ceph: root@node /]# ceph config set mgr.* target_max_misplaced_ratio .10
```

The `balancer` module is enabled by default. Use the `ceph balancer on` and `ceph balancer off` commands to enable or disable the balancer.

Use the `ceph balancer status` command to display the balancer status.

```
[ceph: root@node /]# ceph balancer status
```

Automated Balancing

Automated balancing uses one of the following modes:

crush-compat

This mode uses the `compat weight-set` feature to calculate and manage an alternative set of weights for devices in the CRUSH hierarchy. The balancer optimizes these weight-set values, adjusting them up or down in small increments to achieve a distribution that matches the target distribution as closely as possible.

This mode is fully backwards compatible with older clients.

upmap

The PG upmap mode enables storing explicit PG mappings for individual OSDs in the OSD map as exceptions to the normal CRUSH placement calculation. The upmap mode analyzes PG placement, and then runs the required `pg-upmap-items` commands to optimize PG placement and achieve a balanced distribution.

Because these upmap entries provide fine-grained control over the PG mapping, the upmap mode is usually able to distribute PGs evenly among OSDs, or +/-1 PG if there is an odd number of PGs.

Setting the mode to upmap requires that all clients be Luminous or newer. Use the `ceph osd set-require-min-compat-client luminous` command to set the required minimum client version.

Use the `ceph balancer mode upmap` command to set the balancer mode to upmap.

```
[ceph: root@node /]# ceph balancer mode upmap
```

Use the `ceph balancer mode crush-compat` command to set the balancer mode to crush-compat.

```
[ceph: root@node /]# ceph balancer mode crush-compat
```

Manual Balancing

You can run the balancer manually to control when balancing occurs and to evaluate the balancer plan before executing it. To run the balancer manually, use the following commands to disable automatic balancing, and then generate and execute a plan.

- Evaluate and score the current distribution for the cluster.

```
[ceph: root@node /]# ceph balancer eval
```

Evaluate and score the current distribution for a specific pool.

```
[ceph: root@node /]# ceph balancer eval POOL_NAME
```

- Generate a PG optimization plan and give it a name.

```
[ceph: root@node /]# ceph balancer optimize PLAN_NAME
```

- Display the contents of the plan.

```
[ceph: root@node /]# ceph balancer show PLAN_NAME
```

- Analyze the predicted results of executing the plan.

```
[ceph: root@node /]# ceph balancer eval PLAN_NAME
```

- If you approve of the predicted results, then execute the plan.

```
[ceph: root@node /]# ceph balancer execute PLAN_NAME
```



Note

Only execute the plan if you expect it to improve the distribution. The plan is discarded after execution.

Use the `ceph balancer ls` command to show currently recorded plans.

```
[ceph: root@node /]# ceph balancer ls
```

Use the `ceph balancer rm` command to remove a plan.

```
[ceph: root@node /]# ceph balancer rm PLAN_NAME
```



References

For more information, refer to the *Red Hat Ceph Storage 5 Administration Guide* at https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/administration_guide/index

► Guided Exercise

Performing Cluster Administration and Monitoring

In this exercise, you will perform common administration operations on a Red Hat Ceph Storage cluster.

Outcomes

You should be able to administer and monitor the cluster, including starting and stopping specific services, analyzing placement groups, setting OSD primary affinity, verifying daemon versions, and querying cluster health and utilization.

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 cluster-admin
```

This command confirms that the hosts required for this exercise are accessible.

Instructions

- 1. Log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]#
```

- 2. View the enabled MGR modules. Verify that the dashboard module is enabled.

```
[ceph: root@clienta /]# ceph mgr module ls | more
{
    "always_on_modules": [
        "balancer",
        "crash",
        "devicehealth",
        "orchestrator",
        "pg_autoscaler",
        "progress",
        "rbd_support",
        "status",
        "telemetry",
        "volumes"
    ],
    "enabled_modules": [
        "cephadm",
        "mon",
        "osd",
        "mgr",
        "fs",
        "mds",
        "rbd",
        "iscsi",
        "http",
        "https",
        "metrics",
        "telemetry",
        "log",
        "crash",
        "devicehealth",
        "osd",
        "pg",
        "balance",
        "orchestrator",
        "status",
        "volume"
    ]
}
```

```
"dashboard",
"insights",
"iostat",
"prometheus",
"restful"
],
"disabled_modules": [
{
    "name": "alerts",
    "can_run": true,
    "error_string": "",
    "module_options": {
...output omitted...
```

► 3. Obtain the dashboard URL for the active MGR node.

```
[ceph: root@clienta /]# ceph mgr services
{
    "dashboard": "https://172.25.250.12:8443/",
    "prometheus": "http://172.25.250.12:9283/"
}
```

► 4. View the status of the Monitors on the Ceph Dashboard page.

- 4.1. Using a web browser, navigate to the dashboard URL obtained in the previous step. Log in as the **admin** user with the **redhat** password.
- 4.2. On the **Dashboard** page, click **Monitors** to view the status of the Monitor nodes and quorum.

► 5. View the status of all OSDs in the cluster.

```
[ceph: root@clienta /]# ceph osd stat
9 osds: 9 up (since 38m), 9 in (since 38m); epoch: e294
```

► 6. Find the location of the OSD 2 daemon, stop the OSD, and view the cluster OSD status.

- 6.1. Find the location of the OSD 2 daemon.

```
[ceph: root@clienta /]# ceph osd find 2
{
    "osd": 2,
    "addrs": [
        "addrvec": [
            {
                "type": "v2",
                "addr": "172.25.250.12:6808",
                "nonce": 2361545815
            },
            {
                "type": "v1",
                "addr": "172.25.250.12:6809",
                "nonce": 2361545815
            }
        ]
}
```

```

        }
    ],
},
"osd_fsid": "1163a19e-e580-40e0-918f-25fd94e97b86",
"host": "serverc.lab.example.com",
"crush_location": {
    "host": "serverc",
    "root": "default"
}
}
}

```

6.2. Log in to the serverc node. Stop the OSD 2 daemon.

```

[ceph: root@clienta /]# ssh admin@serverc
admin@serverc's password: redhat
[admin@serverc ~]$ sudo systemctl list-units "ceph*"
UNIT                                     LOAD  ACTIVE SUB
DESCRIPTION
...output omitted...
ceph-ff97a876-1fd2-11ec-8258-52540000fa0c@osd.0.service      loaded active running
  Ceph osd.0 for ff97a876-1fd2-11ec-8258-52540000fa0c
ceph-ff97a876-1fd2-11ec-8258-52540000fa0c@osd.1.service      loaded active running
  Ceph osd.1 for ff97a876-1fd2-11ec-8258-52540000fa0c
ceph-ff97a876-1fd2-11ec-8258-52540000fa0c@osd.2.service      loaded active running
  Ceph osd.2 for ff97a876-1fd2-11ec-8258-52540000fa0c
...output omitted...
[admin@serverc ~]$ sudo systemctl stop \
ceph-ff97a876-1fd2-11ec-8258-52540000fa0c@osd.2.service

```

6.3. Exit the serverc node. View the cluster OSD status.

```

[admin@serverc ~]$ exit
[ceph: root@clienta /]# ceph osd stat
9 osds: 8 up (since 24s), 9 in (since 45m); epoch: e296

```

► 7. Start osd.2 on the serverc node, and then view the cluster OSD status.

```

[ceph: root@clienta /]# ssh admin@serverc sudo systemctl start \
ceph-ff97a876-1fd2-11ec-8258-52540000fa0c@osd.2.service
admin@serverc's password: redhat
[ceph: root@clienta /]# ceph osd stat
9 osds: 9 up (since 6s), 9 in (since 47m); epoch: e298

```

► 8. View the log files for the osd.2 daemon. Filter the output to view only systemd events.

```
[ceph: root@clienta /]# ssh admin@serverc sudo journalctl \
-u ceph-ff97a876-1fd2-11ec-8258-52540000fa0c@osd.2.service | grep systemd
admin@serverc's password: redhat
...output omitted...
Sep 30 01:57:36 serverc.lab.example.com systemd[1]: Stopping Ceph osd.2 for
  ff97a876-1fd2-11ec-8258-52540000fa0c...
Sep 30 01:57:37 serverc.lab.example.com systemd[1]: ceph-
ff97a876-1fd2-11ec-8258-52540000fa0c@osd.2.service: Succeeded.
Sep 30 01:57:37 serverc.lab.example.com systemd[1]: Stopped Ceph osd.2 for
  ff97a876-1fd2-11ec-8258-52540000fa0c.
Sep 30 02:00:12 serverc.lab.example.com systemd[1]: Starting Ceph osd.2 for
  ff97a876-1fd2-11ec-8258-52540000fa0c...
Sep 30 02:00:13 serverc.lab.example.com systemd[1]: Started Ceph osd.2 for
  ff97a876-1fd2-11ec-8258-52540000fa0c.
```

- ▶ 9. Mark the osd.4 daemon as being **out** of the cluster and observe how it affects the cluster status. Then, mark the osd.4 daemon as being **in** the cluster again.

- 9.1. Mark the osd.4 daemon as being **out** of the cluster. Verify that the osd.4 daemon is marked **out** of the cluster and notice that the OSD's weight is now 0.

```
[ceph: root@clienta /]# ceph osd out 4
marked out osd.4.

[ceph: root@clienta /]# ceph osd stat
9 osds: 9 up (since 2m), 8 in (since 3s); epoch: e312
[ceph: root@clienta /]# ceph osd tree
ID CLASS WEIGHT TYPE NAME STATUS REWEIGHT PRI-AFF
-1          0.08817  root default
-3          0.02939  host serverc
 0  hdd  0.00980    osd.0      up   1.00000  1.00000
 1  hdd  0.00980    osd.1      up   1.00000  1.00000
 2  hdd  0.00980    osd.2      up   1.00000  1.00000
-7          0.02939  host serverd
 3  hdd  0.00980    osd.3      up   1.00000  1.00000
 5  hdd  0.00980    osd.5      up   1.00000  1.00000
 7  hdd  0.00980    osd.7      up   1.00000  1.00000
-5          0.02939  host servere
 4  hdd  0.00980    osd.4      up       0  1.00000
 6  hdd  0.00980    osd.6      up   1.00000  1.00000
 8  hdd  0.00980    osd.8      up   1.00000  1.00000
```



Note

Ceph recreates the missing object replicas previously available on the osd.4 daemon on different OSDs. You can trace the recovery of the objects using the `ceph status` or the `ceph -w` commands.

- 9.2. Mark the osd.4 daemon as being **in** again.

```
[ceph: root@clienta /]# ceph osd in 4
marked in osd.4.
```

**Note**

You can mark an OSD as `out` even though it is still running (`up`). The `in` or `out` status does not correlate to an OSD's running state.

- 10. Analyze the current utilization and number of PGs on the OSD 2 daemon.

```
[ceph: root@clienta /]# ceph osd df tree
ID CLASS WEIGHT  REWEIGHT SIZE    RAW USE   DATA    OMAP    META    AVAIL
%USE  VAR   PGS  STATUS   TYPE NAME
-1      0.08817      -  90 GiB  256 MiB   36 MiB  56 KiB  220 MiB  90 GiB
  0.28  1.00   -       root default
-3      0.02939      -  30 GiB   71 MiB   12 MiB  20 KiB  59 MiB  30 GiB
  0.23  0.83   -       host serverc
  0  hdd   0.00980  1.00000  10 GiB   26 MiB   4.0 MiB  11 KiB  22 MiB  10 GiB
  0.25  0.91   68     up       osd.0
  1  hdd   0.00980  1.00000  10 GiB   29 MiB   4.0 MiB   6 KiB  25 MiB  10 GiB
  0.28  1.01   74     up       osd.1
  2  hdd   0.00980  1.00000  10 GiB   16 MiB   3.9 MiB   3 KiB  12 MiB  10 GiB
  0.16  0.57   59     up       osd.2
...output omitted...
                           TOTAL 90 GiB  256 MiB   36 MiB  61 KiB  220 MiB  90 GiB
  0.28
MIN/MAX VAR: 0.57/1.48  STDDEV: 0.06
```

- 11. View the placement group status for the cluster. Create a test pool and a test object. Find the placement group to which the test object belongs and analyze that placement group's status.

- 11.1. View the placement group status for the cluster. Examine the PG states. Your output may be different in your lab environment.

```
[ceph: root@clienta /]# ceph pg stat
201 pgs: 201 active+clean; 8.6 KiB data, 261 MiB used, 90 GiB / 90 GiB avail; 511
B/s rd, 0 op/s
```

- 11.2. Create a pool called `testpool` and an object called `testobject` containing the `/etc/ceph/ceph.conf` file.

```
[ceph: root@clienta /]# ceph osd pool create testpool 32 32
pool 'testpool' created
[ceph: root@clienta /]# rados -p testpool put testobject /etc/ceph/ceph.conf
```

- 11.3. Find the placement group of the `testobject` object in the `testpool` pool and analyze its status. Use the placement group information from your lab environment in the query.

```
[ceph: root@clienta /]# ceph osd map testpool testobject
osdmap e332 pool 'testpool' (9) object 'testobject' -> pg 9.98824931 (9.11) -> up
([8,2,5], p8) acting ([8,2,5], p8)
[ceph: root@clienta /]# ceph pg 9.11 query
```

```
{
  "snap_trimq": "[ ]",
  "snap_trimq_len": 0,
  "state": "active+clean",
  "epoch": 334,
  "up": [
    8,
    2,
    5
  ],
  "acting": [
    8,
    2,
    5
  ],
  "acting_recovery_backfill": [
    "2",
    "5",
    "8"
  ],
  "info": {
    "pgid": "9.11",
    ...output omitted...
}
```

- ▶ 12. List the OSD and cluster daemon versions. This is a useful command to run after cluster upgrades.

12.1. List all cluster daemon versions.

```
[ceph: root@clienta /]# ceph versions
{
  "mon": {
    "ceph version 16.2.0-117.el8cp (0e34bb74700060ebfaa22d99b7d2cdc037b28a57)
    pacific (stable)": 4
  },
  "mgr": {
    "ceph version 16.2.0-117.el8cp (0e34bb74700060ebfaa22d99b7d2cdc037b28a57)
    pacific (stable)": 4
  },
  "osd": {
    "ceph version 16.2.0-117.el8cp (0e34bb74700060ebfaa22d99b7d2cdc037b28a57)
    pacific (stable)": 9
  },
  "mds": {
    "ceph version 16.2.0-117.el8cp (0e34bb74700060ebfaa22d99b7d2cdc037b28a57)
    pacific (stable)": 3
  },
  "rgw": {
    "ceph version 16.2.0-117.el8cp (0e34bb74700060ebfaa22d99b7d2cdc037b28a57)
    pacific (stable)": 2
  },
  "overall": {
```

```
"ceph version 16.2.0-117.el8cp (0e34bb74700060ebfaa22d99b7d2cdc037b28a57)
pacific (stable)": 22
}
}
```

12.2. List all OSD versions.

```
[ceph: root@clienta /]# ceph tell osd.* version
osd.0: {
    "version": "16.2.0-117.el8cp",
    "release": "pacific",
    "release_type": "stable"
}
osd.1: {
    "version": "16.2.0-117.el8cp",
    "release": "pacific",
    "release_type": "stable"
}
osd.2: {
    "version": "16.2.0-117.el8cp",
    "release": "pacific",
    "release_type": "stable"
}
...output omitted...
```

▶ 13. View the balancer status.

```
[ceph: root@clienta /]# ceph balancer status
{
    "active": true,
    "last_optimize_duration": "0:00:00.001072",
    "last_optimize_started": "Thu Sep 30 06:07:53 2021",
    "mode": "upmap",
    "optimize_result": "Unable to find further optimization, or pool(s) pg_num is
decreasing, or distribution is already perfect",
    "plans": []
}
```

▶ 14. Return to workstation as the student user.

```
[ceph: root@clienta /]# exit
[admin@clienta ~]$ exit
[student@workstation ~]$
```

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 cluster-admin
```

This concludes the guided exercise.

Performing Cluster Maintenance Operations

Objectives

After completing this section, you should be able to perform common cluster maintenance tasks, such as adding or removing MONs and OSDs, and recovering from various component failures.

Adding or Removing OSD Nodes

The cluster can operate and serve clients in a degraded state during cluster maintenance activities. However, adding or removing OSDs can affect cluster performance. Backfilling operations can generate large data transfers between OSDs, causing cluster performance to degrade.

Evaluate the potential performance impact before performing cluster maintenance activities. The following factors typically affect cluster performance when adding or removing OSD nodes:

- Client load

If an OSD node has a pool that is experiencing high client loads, then performance and recovery time could be negatively affected. Because write operations require data replication for resiliency, write-intensive client loads increase cluster recovery time.

- Node capacity

The capacity of the node being added or removed affects the cluster recovery time. The node's storage density also affects recovery times. For example, a node with 36 OSDs takes longer to recover than a node with 12 OSDs.

- Spare cluster capacity

When removing nodes, verify that you have sufficient spare capacity to avoid reaching the full or near full ratios. When a cluster reaches the full ratio, Ceph suspends write operations to prevent data loss.

- CRUSH rules

A Ceph OSD node maps to at least one CRUSH hierarchy, and that hierarchy maps to at least one pool via a CRUSH rule. Each pool using a specific CRUSH hierarchy experiences a performance impact when adding and removing OSDs.

- Pool types

Replication pools use more network bandwidth to replicate data copies, while erasure-coded pools use more CPU to calculate data and coding chunks.

The more data copies that exist, the longer it takes for the cluster to recover. For example, an erasure-coded pool with many chunks takes longer to recover than a replicated pool with fewer copies of the same data.

- Node hardware

Nodes with higher throughput characteristics, such as 10 Gbps network interfaces and SSDs, recover more quickly than nodes with lower throughput characteristics, such as 1 Gbps network interfaces and SATA drives.

Replacing a Failed OSD

Red Hat Ceph Storage is designed to be self-healing. When a storage device fails, extra data copies on other OSDs backfill automatically to recover the cluster to a healthy state.

When a storage device fails, the OSD status changes to down. Other cluster issues, such as a network error, can also mark an OSD as down. When an OSD is down, first verify if the physical device has failed.

Replacing a failed OSD requires replacing both the physical storage device and the software-defined OSD. When an OSD fails, you can replace the physical storage device and either reuse the same OSD ID or create a new one. Reusing the same OSD ID avoids having to reconfigure the CRUSH map.

If an OSD has failed, use the Dashboard GUI or the following CLI commands to replace the OSD.

To verify that the OSD has failed, perform the following steps.

- View the cluster status and verify that an OSD has failed.

```
[ceph: root@node /]# ceph health detail
```

- Identify the failed OSD.

```
[ceph: root@node /]# ceph osd tree | grep -i down
```

- Locate the OSD node where the OSD is running.

```
[ceph: root@node /]# ceph osd find osd.OSD_ID
```

- Attempt to start the failed OSD.

```
[ceph: root@node /]# ceph orch daemon start OSD_ID
```

If the OSD does not start, then the physical storage device might have failed. Use the `journalctl` command to view the OSD logs or use the utilities available in your production environment to verify that the physical device has failed.

If you have verified that the physical device needs replacement, perform the following steps.

- Temporarily disable scrubbing.

```
[ceph: root@node /]# ceph osd set noscrub ; ceph osd set nodeep-scrub
```

- Remove the OSD from the cluster.

```
[ceph: root@node /]# ceph osd out OSD_ID
```

- Watch cluster events and verify that a backfill operation has started.

```
[ceph: root@node /]# ceph -w
```

- Verify that the backfill process has moved all PGs off the OSD and it is now safe to remove.

```
[ceph: root@node /]# while ! ceph osd safe-to-destroy osd.OSD_ID ; \  
do sleep 10 ; done
```

- When the OSD is safe to remove, replace the physical storage device and destroy the OSD. Optionally, remove all data, file systems, and partitions from the device.

```
[ceph: root@node /]# ceph orch device zap HOST_NAME _OSD_ID --force
```

**Note**

Find the current device ID using the Dashboard GUI, or the `ceph-volume lvm list` or `ceph osd metadata` CLI commands.

- Replace the OSD using the same ID as the one that failed. Verify that the operation has completed before continuing.

```
[ceph: root@node /]# ceph orch osd rm OSD_ID --replace  
[ceph: root@node /]# ceph orch osd rm status
```

- Replace the physical device and recreate the OSD. The new OSD uses the same OSD ID as the one that failed.

**Note**

The device path of the new storage device might be different than the failed device. Use the `ceph orch device ls` command to find the new device path.

```
[ceph: root@node /]# ceph orch daemon add osd HOST_NAME:_DEVICE_PATH_
```

- Start the OSD and verify that the OSD is up.

```
[ceph: root@node /]# ceph orch daemon start OSD_ID  
[ceph: root@node /]# ceph osd tree
```

- Re-enable scrubbing.

```
[ceph: root@node /]# ceph osd unset noscrub ; ceph osd unset nodeep-scrub
```

Adding a MON

Add a MON to your cluster by performing the following steps.

- Verify the current MON count and placement.

```
[ceph: root@node /]# ceph orch ls --service_type=mon
```

- Add a new host to the cluster.

```
[ceph: root@node /]# ceph cephadm get-pub-key > ~/ceph.pub
[ceph: root@node /]# ssh-copy-id -f -i ~/ceph.pub root@HOST_NAME
[ceph: root@node /]# ceph orch host add HOST_NAME
```

- Specify the hosts where the MON nodes should run.



Note

Specify **all** MON nodes when running this command. If you only specify the new MON node, then the command removes all other MONs, leaving the cluster with only one MON node.

```
[ceph: root@node /]# ceph orch apply mon --placement="NODE1 NODE2 NODE3 NODE4 ..."
```

Removing a MON

Use the `ceph orch apply mon` command to remove a MON from the cluster. Specify all MONs except the one that you want to remove.

```
[ceph: root@node /]# ceph orch apply mon --placement="NODE1 NODE2 NODE3 ..."
```

Placing Hosts Into Maintenance Mode

Use the `ceph orch host maintenance` command to place hosts in and out of maintenance mode. Maintenance mode stops all Ceph daemons on the host. Use the optional `--force` option to bypass warnings.

```
[ceph: root@node /]# ceph orch host maintenance enter HOST_NAME [--force]
```

When finished with maintenance, exit maintenance mode.

```
[ceph: root@node /]# ceph orch host maintenance exit HOST_NAME
```



References

For more information, refer to the *Red Hat Ceph Storage 5 Operations Guide* at https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/operations_guide/index

► Guided Exercise

Performing Cluster Maintenance Operations

In this exercise, you will perform maintenance activities on an operational Red Hat Ceph Storage cluster.

Outcomes

You should be able to add, replace, and remove components in an operational Red Hat Ceph Storage cluster.

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 cluster-maint
```

This command confirms that the hosts required for this exercise are accessible and stops the `osd.3` daemon to simulate an OSD failure.

Instructions

- ▶ 1. Log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]#
```

- ▶ 2. Set the `noscrub` and `nodeep-scrub` flags to prevent the cluster from starting scrubbing operations temporarily.

```
[ceph: root@clienta /]# ceph osd set noscrub
noscrub is set
[ceph: root@clienta /]# ceph osd set nodeep-scrub
nodeep-scrub is set
```

- ▶ 3. Verify the Ceph cluster status. The cluster will transition to the `HEALTH_WARN` status after some time.

```
[ceph: root@clienta /]# ceph health detail
HEALTH_WARN noscrub,nodeep-scrub flag(s) set; 1 osds down; Degraded data
  redundancy: 82/663 objects degraded (12.368%), 14 pgs degraded
[WRN] OSDMAP_FLAGS: noscrub,nodeep-scrub flag(s) set
[WRN] OSD_DOWN: 1 osds down
  osd.3 (root=default,host=serverd) is down
```

```
[WRN] PG_DEGRADED: Degraded data redundancy: 82/663 objects degraded (12.368%), 14
pgs degraded
    pg 2.f is active+undersized+degraded, acting [8,0]
    pg 2.19 is active+undersized+degraded, acting [0,8]
    pg 3.0 is active+undersized+degraded, acting [8,1]
...output omitted...
```

▶ 4. Identity the failed OSD device for replacement.

4.1. Identify which OSD is down.

```
[ceph: root@clienta /]# ceph osd tree | grep -i down
3    hdd 0.00980          osd.3      down  1.00000 1.0000
```

4.2. Identify which host the OSD is on.

```
[ceph: root@clienta /]# ceph osd find osd.3
{
    "osd": 3,
...output omitted...
    "host": "serverd.lab.example.com",
    "crush_location": {
        "host": "serverd",
        "root": "default"
    }
}
```

4.3. Log in to the serverd node and use sudo to run the cephadm shell. Identify the device name for the failed OSD.

```
[ceph: root@clienta /]# ssh admin@serverd
admin@serverd's password: redhat
[admin@serverd ~]$ sudo cephadm shell
[ceph: root@serverd /]# ceph-volume lvm list

===== osd.3 =====
...output omitted...
    devices                  /dev/vdb
...output omitted...
```



Note

You can also identify the device name of an OSD by using the `ceph osd metadata OSD_ID` command from the admin node.

▶ 5. Exit the cephadm shell. Identify the service name of the `osd.3` daemon running on the `serverd` node. The service name will be different in your lab environment.

```
[ceph: root@serverd /]# exit
exit
[admin@serverd ~]$ sudo systemctl list-units --all "ceph*"
```

UNIT	LOAD	ACTIVE	SUB
DESCRIPTION			
ceph-2ae6d05a-229a-11ec-925e-52540000fa0c@crash.serverd.service	loaded	active	
running Ceph crash.serverd for 2a>			
ceph-2ae6d05a-229a-11ec-925e-52540000fa0c@mgr.serverd.klrkci.service	loaded	active	
running Ceph mgr.serverd.klr>			
ceph-2ae6d05a-229a-11ec-925e-52540000fa0c@mon.serverd.service	loaded	active	
running Ceph mon.serverd for 2ae6>			
ceph-2ae6d05a-229a-11ec-925e-52540000fa0c@node-exporter.serverd.service	loaded	active	
running Ceph node-exporter>			
ceph-2ae6d05a-229a-11ec-925e-52540000fa0c@osd.3.service	loaded	inactive	
dead Ceph osd.3 for 2ae6d05a-2>			
ceph-2ae6d05a-229a-11ec-925e-52540000fa0c@osd.5.service	loaded	active	
running Ceph osd.5 for 2ae6d05a-2>			
...output omitted...			

- ▶ 6. Check the recent log entries for the osd.3 service.

```
[admin@serverd ~]$ sudo journalctl -ru \
ceph-2ae6d05a-229a-11ec-925e-52540000fa0c@osd.3.service
...output omitted...
Oct 06 22:25:49 serverd.lab.example.com systemd[1]: Stopped Ceph osd.3 for
2ae6d05a-229a-11ec-925e-52540000fa0c.
...output omitted...
```

- ▶ 7. On the serverd node, start the osd.3 service. On the admin node, verify that the OSD has started.

```
[admin@serverd ~]$ sudo systemctl start \
ceph-2ae6d05a-229a-11ec-925e-52540000fa0c@osd.3.service
[admin@serverd ~]$ exit
logout
Connection to serverd closed.
[ceph: root@clienta /]# ceph osd tree
ID CLASS WEIGHT TYPE NAME STATUS REWEIGHT PRI-AFF
-1          0.09796  root default
-3          0.03918  host serverc
 0  hdd  0.00980      osd.0    up   1.00000  1.00000
 1  hdd  0.00980      osd.1    up   1.00000  1.00000
 2  hdd  0.00980      osd.2    up   1.00000  1.00000
-5          0.02939  host serverd
 3  hdd  0.00980      osd.3    up   1.00000  1.00000
 5  hdd  0.00980      osd.5    up   1.00000  1.00000
 7  hdd  0.00980      osd.7    up   1.00000  1.00000
-7          0.02939  host servere
 4  hdd  0.00980      osd.4    up   1.00000  1.00000
 6  hdd  0.00980      osd.6    up   1.00000  1.00000
 8  hdd  0.00980      osd.8    up   1.00000  1.00000
```

- 8. Clear the noscrub and nodeep-scrub flags. Verify that the cluster health status returns to HEALTH_OK. Press CTL+C to exit the ceph -w command.

```
[ceph: root@clienta /]# ceph osd unset noscrub
noscrub is unset
[ceph: root@clienta /]# ceph osd unset nodeep-scrub
nodeep-scrub is unset
[ceph: root@clienta /]# ceph -w
...output omitted...
health: HEALTH_OK
...output omitted...
```

- 9. Adjust the number of MONs and their placement in the cluster.

- 9.1. View the number of running MONs and their placement.

```
[ceph: root@clienta /]# ceph orch ls --service_type=mon
NAME    RUNNING    REFRESHED    AGE    PLACEMENT
mon      4/4       6m ago      5d
clienta.lab.example.com;serverc.lab.example.com;serverd.lab.example.com;servere.
lab.example.com
```

- 9.2. Add the serverg node to the cluster.

```
[ceph: root@clienta /]# ceph cephadm get-pub-key > ~/ceph.pub
[ceph: root@clienta /]# ssh-copy-id -f -i ~/ceph.pub root@serverg
root@serverg's password: redhat
...output omitted...
[ceph: root@clienta /]# ceph orch host add serverg.lab.example.com
Added host 'serverg.lab.example.com' with addr '172.25.250.16'
```

- 9.3. Add a MON and place it on the serverg node.

```
[ceph: root@clienta /]# ceph orch apply mon \
--placement="clienta.lab.example.com serverc.lab.example.c om \
serverd.lab.example.com servere.lab.example.com \
serverg.lab.example.com"
Scheduled mon update...
```

- 9.4. Verify that the MONs are active and correctly placed.

```
[ceph: root@clienta /]# ceph orch ls --service-type=mon
NAME    RUNNING    REFRESHED    AGE    PLACEMENT
mon      5/5       -           58s
clienta.lab.example.com;serverc.lab.example.com;serverd.lab.example.com;servere.
lab.example.com;serverg.lab.example.com
```

- 10. Remove the MON service from serverg node, remove its OSDs, and then remove serverg from the cluster. Verify that the serverg node is removed.

- 10.1. Remove the MON node from the serverg node.

```
[ceph: root@clienta /]# ceph orch apply mon \
--placement="clienta.lab.example.com serverc.lab.example.com \
serverd.lab.example.com servere.lab.example.com"
Scheduled mon update...
[ceph: root@clienta /]# ceph mon stat
e6: 4 mons at {clienta=[v2:172.25.250.10:3300/0,v1:172.25.250.10:6789/0],
serverc.lab.example.com=[v2:172.25.250.12:3300/0,v1:172.25.250.12:6789/0],
serverd=[v2:172.25.250.13:3300/0,v1:172.25.250.13:6789/0],
servere=[v2:172.25.250.14:3300/0,v1:172.25.250.14:6789/0]},
election epoch 46, leader 0 serverc.lab.example.com, quorum 0,1,2,3
serverc.lab.example.com,clienta,serverd,servere
```

**Important**

Always keep at least three MONs running in a production cluster.

10.2. Remove the `serverg` node's OSDs.

```
[ceph: root@clienta /]# ceph orch ps serverg.lab.example.com
NAME          HOST           STATUS      REFRESHED   AGE
PORTS    VERSION      IMAGE ID      CONTAINER ID
crash.serverg  serverg.lab.example.com  running (3m)  35s ago   3m  -
          16.2.0-117.el8cp 2142b60d7974  db0eb4d442b2
node-exporter.serverg  serverg.lab.example.com  running (3m)  35s ago   3m
          *:9100  0.18.1     68b1be7484d4  982fc365dc88
osd.10        serverg.lab.example.com  running (2m)  35s ago   2m  -
          16.2.0-117.el8cp 2142b60d7974  c503c770f6ef
osd.11        serverg.lab.example.com  running (2m)  35s ago   2m  -
          16.2.0-117.el8cp 2142b60d7974  3e4f85ad8384
osd.9         serverg.lab.example.com  running (2m)  35s ago   2m  -
          16.2.0-117.el8cp 2142b60d7974  ab9563910c19
[ceph: root@clienta /]# ceph osd stop 9 10 11
stop down osd.9. stop down osd.10. stop down osd.11.
[ceph: root@clienta /]# ceph osd out 9 10 11
marked out osd.9. marked out osd.10. marked out osd.11.
[ceph: root@clienta /]# ceph osd crush remove osd.9
removed item id 9 name 'osd.9' from crush map
[ceph: root@clienta /]# ceph osd crush remove osd.10
removed item id 10 name 'osd.10' from crush map
[ceph: root@clienta /]# ceph osd crush remove osd.11
removed item id 11 name 'osd.11' from crush map
[ceph: root@clienta /]# ceph osd rm 9 10 11
removed osd.9, osd.10, osd.11
```

10.3. Remove the `serverg` node from the cluster. Verify that the `serverg` node has been removed.

```
[ceph: root@clienta /]# ceph orch host rm serverg.lab.example.com
Removed host 'serverg.lab.example.com'
[ceph: root@clienta /]# ceph orch host ls
HOST           ADDR      LABELS  STATUS
clienta.lab.example.com  172.25.250.10 _admin
serverc.lab.example.com  172.25.250.12
serverd.lab.example.com  172.25.250.13
servere.lab.example.com  172.25.250.14
```

- ▶ 11. You receive an alert that there is an issue on the `servere` node. Put the `servere` node into maintenance mode, reboot the host, and then exit maintenance mode.

- 11.1. Put the `servere` node into maintenance mode, and then verify that it has a maintenance status.

```
[ceph: root@clienta /]# ceph orch host maintenance enter servere.lab.example.com
Ceph cluster 2ae6d05a-229a-11ec-925e-52540000fa0c on servere.lab.example.com moved
to maintenance
[ceph: root@clienta /]# ceph orch host ls
HOST           ADDR      LABELS  STATUS
clienta.lab.example.com  172.25.250.10 _admin
serverc.lab.example.com  172.25.250.12
serverd.lab.example.com  172.25.250.13
servere.lab.example.com  172.25.250.14          Maintenance
```

- 11.2. Reboot the `servere` node.

```
[ceph: root@clienta /]# ssh admin@servere sudo reboot
admin@servere's password: redhat
Connection to servere closed by remote host.
```

- 11.3. After the `servere` node reboots, exit maintenance mode.

```
[ceph: root@clienta /]# ceph orch host maintenance exit servere.lab.example.com
Ceph cluster 2ae6d05a-229a-11ec-925e-52540000fa0c on servere.lab.example.com has
exited maintenance mode
```

- ▶ 12. Return to workstation as the student user.

```
[ceph: root@servera /]# exit
[admin@clienta ~]$ exit
[student@workstation ~]$
```

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 cluster-maint
```

This concludes the guided exercise.

► Lab

Managing a Red Hat Ceph Storage Cluster

In this lab, you will perform common administration and maintenance operations on a Red Hat Ceph Storage cluster.

Outcomes

You should be able to locate the Ceph Dashboard URL, set an OSD out and in, watch cluster events, find and start a down OSD, find an object's PG location and state, and view the balancer status.

Before You Begin

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

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

This command confirms that the hosts required for this exercise are accessible.

Instructions

1. Log in to `clienta` as the `admin` user. Verify that the `dashboard` module is enabled. Find the dashboard URL of the active MGR.
2. You receive an alert that an OSD is down. Identify which OSD is down. Identify on which node the down OSD runs, and start the OSD.
3. Set the OSD 5 daemon to the `out` state and verify that all data has been migrated off of the OSD.
4. Set the OSD 5 daemon to the `in` state and verify that PGs have been placed onto it.
5. Display the balancer status.
6. Identify the PG for object `data1` in the `pool1` pool. Query the PG and find its state.
7. Return to `workstation` as the `student` user.

Evaluation

Grade your work by running the `lab grade cluster-review` command from your `workstation` machine. Correct any reported failures and rerun the script until successful.

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

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 cluster-review
```

This concludes the lab.

► Solution

Managing a Red Hat Ceph Storage Cluster

In this lab, you will perform common administration and maintenance operations on a Red Hat Ceph Storage cluster.

Outcomes

You should be able to locate the Ceph Dashboard URL, set an OSD out and in, watch cluster events, find and start a down OSD, find an object's PG location and state, and view the balancer status.

Before You Begin

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

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

This command confirms that the hosts required for this exercise are accessible.

Instructions

1. Log in to `clienta` as the `admin` user. Verify that the `dashboard` module is enabled. Find the dashboard URL of the active MGR.

- 1.1. Log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]#
```

- 1.2. Verify that the `dashboard` module is enabled.

```
[ceph: root@clienta /]# ceph mgr module ls | more
{
  ...output omitted...
  "enabled_modules": [
    "cephadm",
    "dashboard",
    "iostat",
    "prometheus",
    "restful"
  ],
  ...output omitted...
```

- 1.3. Find the dashboard URL of the active MGR.

```
[ceph: root@clienta /]# ceph mgr services
{
    "dashboard": "https://172.25.250.12:8443/",
    "prometheus": "http://172.25.250.12:9283/"
}
```

**Note**

Your output might be different depending on which MGR node is active in your lab environment.

2. You receive an alert that an OSD is down. Identify which OSD is down. Identify on which node the down OSD runs, and start the OSD.

- 2.1. Verify cluster health.

```
[ceph: root@clienta /]# ceph health detail
HEALTH_WARN 1 osds down; Degraded data redundancy: 72/666 objects degraded
(10.811%), 14 pgs degraded, 50 pgs undersized
[WRN] OSD_DOWN: 1 osds down
    osd.6 (root=default,host=servere) is down
[WRN] PG_DEGRADED: Degraded data redundancy: 72/666 objects degraded (10.811%), 14
pgs degraded, 50 pgs undersized
    pg 2.0 is stuck undersized for 61s, current state active+undersized, last
acting [3,0]
    pg 2.1 is stuck undersized for 61s, current state active+undersized, last
acting [2,3]
    pg 2.6 is stuck undersized for 61s, current state active+undersized, last
acting [1,3]
    pg 2.7 is stuck undersized for 61s, current state active+undersized, last
acting [3,2]
...output omitted...
```

- 2.2. Identify which OSD is down.

```
[ceph: root@clienta /]# ceph osd tree | grep -i down
6      hdd  0.00980          osd.6        down   1.00000  1.00000
```

- 2.3. Identify on which host the down OSD runs.

```
[ceph: root@clienta /]# ceph osd find osd.6 | grep host
    "host": "servere.lab.example.com",
    "host": "servere",
```

- 2.4. Start the OSD.

```
[ceph: root@clienta /]# ceph orch daemon start osd.6
Scheduled to start osd.6 on host 'servere.lab.example.com'
```

- 2.5. Verify that the OSD is up.

```
[ceph: root@clienta /]# ceph osd tree | grep osd.6
 6  hdd  0.00980          osd.6        up   1.00000  1.00000
```

3. Set the OSD 5 daemon to the `out` state and verify that all data has been migrated off of the OSD.

- 3.1. Set the OSD 5 daemon to the `out` state.

```
[ceph: root@clienta /]# ceph osd out 5
marked out osd.5.
```

- 3.2. Verify that all PGs have been migrated off of the OSD 5 daemon. It will take some time for the data migration to finish. Press `CTL+C` to exit the command.

```
[ceph: root@clienta /]# ceph -w
cluster:
  id:      2ae6d05a-229a-11ec-925e-52540000fa0c
  health:  HEALTH_WARN
            Reduced data availability: 5 pgs peering
            Degraded data redundancy: 1/663 objects degraded (0.151%), 1 pg
degraded

  services:
    mon: 4 daemons, quorum serverc.lab.example.com,clienta,serverd,servere (age
9h)
    mgr: serverc.lab.example.com.aiqepd(active, since 9h), standbys:
serverd.klrkci, servere.kjwyko, clienta.nncugs
    osd: 9 osds: 9 up (since 46s), 8 in (since 7s); 4 remapped pgs
    rgw: 2 daemons active (2 hosts, 1 zones)

  data:
    pools: 5 pools, 105 pgs
    objects: 221 objects, 4.9 KiB
    usage: 235 MiB used, 80 GiB / 80 GiB avail
    pgs: 12.381% pgs not active
          1/663 objects degraded (0.151%)
          92 active+clean
          10 remapped+peering
          2 activating
          1 activating+degraded

  io:
    recovery: 199 B/s, 0 objects/s

  progress:
    Global Recovery Event (2s)
    [.....]
2021-03-28 21:23:25.557849 mon.serverc [WRN] Health check failed: Reduced data
availability: 1 pg inactive, 1 pg peering (PG_AVAILABILITY)
2021-03-28 21:23:25.557884 mon.serverc [INF] Health check cleared: PG_DEGRADED
(was: Degraded data redundancy: 36/2163 objects degraded (1.664%), 5 pgs
degraded)
```

```
2021-03-28 21:23:31.741476 mon.serverc [INF] Health check cleared: PG_AVAILABILITY
(was: Reduced data availability: 1 pg inactive, 1 pg peering)
2021-03-28 21:23:31.741495 mon.serverc [INF] Cluster is now healthy
...output omitted...

[ceph: root@clienta /]# ceph osd df
ID CLASS ...output omitted... AVAIL %USE VAR PGS STATUS
0 hdd ...output omitted... 10 GiB 0.38 1.29 34 up
1 hdd ...output omitted... 10 GiB 0.33 1.13 42 up
2 hdd ...output omitted... 10 GiB 0.30 1.02 29 up
3 hdd ...output omitted... 10 GiB 0.28 0.97 58 up
5 hdd ...output omitted... 0 B 0 0 0 up
7 hdd ...output omitted... 10 GiB 0.29 0.99 47 up
4 hdd ...output omitted... 10 GiB 0.33 1.13 34 up
6 hdd ...output omitted... 10 GiB 0.10 0.36 39 up
8 hdd ...output omitted... 10 GiB 0.32 1.12 32 up
TOTAL ...output omitted... 80 GiB 0.29

MIN/MAX VAR: 0.36/1.29 STDDEV: 0.08
```

4. Set the OSD 5 daemon to the `in` state and verify that PGs have been placed onto it.

- 4.1. Set the OSD 5 daemon to the `in` state.

```
[ceph: root@clienta /]# ceph osd in 5
marked in osd.5.
```

- 4.2. Verify that PGs have been placed onto the OSD 5 daemon.

```
[ceph: root@clienta /]# ceph osd df
ID CLASS ...output omitted... AVAIL %USE VAR PGS STATUS
0 hdd ...output omitted... 10 GiB 0.23 0.76 34 up
1 hdd ...output omitted... 10 GiB 0.37 1.26 42 up
2 hdd ...output omitted... 10 GiB 0.34 1.15 29 up
3 hdd ...output omitted... 10 GiB 0.29 0.99 39 up
5 hdd ...output omitted... 10 GiB 0.37 1.24 31 up
7 hdd ...output omitted... 10 GiB 0.30 1.00 35 up
4 hdd ...output omitted... 10 GiB 0.33 1.12 34 up
6 hdd ...output omitted... 10 GiB 0.11 0.37 39 up
8 hdd ...output omitted... 10 GiB 0.33 1.11 32 up
TOTAL 90 GiB 0.30

MIN/MAX VAR: 0.37/1.26 STDDEV: 0.08
```

5. Display the balancer status.

```
[ceph: root@clienta /]# ceph balancer status
{
    "active": true,
    "last_optimize_duration": "0:00:00.000647",
    "last_optimize_started": "Thu Oct 14 01:38:13 2021",
    "mode": "upmap",
```

```
"optimize_result": "Unable to find further optimization, or pool(s) pg_num is
decreasing, or distribution is already perfect",
"plans": []
}
```

6. Identify the PG for object `data1` in the `pool1` pool. Query the PG and find its state.

- 6.1. Identify the PG for object `data1` in the `pool1` pool.

```
[ceph: root@clienta /]# ceph osd map pool1 data1
osdmap e218 pool 'pool1' (6) object 'data1' -> pg 6.d4f4553c (6.1c)` -> up
([8,2,3], p8) acting ([8,2,3], p8)
```



Note

In this example, the PG is **6.1c**. Use the PG value in the output displayed in your lab environment.

- 6.2. Query the PG and view its state and primary OSD.

```
[ceph: root@clienta /]# ceph pg 6.1c query
{
    "snap_trimq": "[]",
    "snap_trimq_len": 0,
    "state": "active+clean",
    "epoch": 218,
    "up": [
        8,
        2,
        3
    ],
    "acting": [
        8,
        2,
        3
    ],
    "acting_recovery_backfill": [
        "2",
        "3",
        "8"
    ],
    "info": {
        "pgid": "6.1c",
        ...
    }
}
```

7. Return to `workstation` as the student user.

```
[ceph: root@clienta /]# exit
[admin@clienta ~]$ exit
[student@workstation ~]$
```

Evaluation

Grade your work by running the `lab grade cluster-review` command from your `workstation` machine. Correct any reported failures and rerun the script until successful.

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

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 cluster-review
```

This concludes the lab.

Summary

In this chapter, you learned how to:

- Enable or disable Ceph Manager (MGR) modules, and more about the role of the Ceph Manager (MGR).
- Use the CLI to find the URL of the Dashboard GUI on the active MGR.
- View the status of cluster MONs by using the CLI or the Dashboard GUI.
- Monitor cluster health and interpret the cluster health status.
- Power down the entire cluster by setting cluster flags to stop background operations, then stopping daemons and nodes in a specific order by function.
- Power up the entire cluster by starting nodes and daemons in a specific order by function, then setting cluster flags to enable background operations.
- Start, stop, or restart individual cluster daemons and view daemon logs.
- Monitor cluster storage by viewing OSD and PG states and capacity.
- Identify the PG details for a specific object.
- Find and replace a failed OSD.
- Add or remove a cluster MON node.
- Use the balancer module to optimize the placement of PGs across OSDs.

Chapter 12

Tuning and Troubleshooting Red Hat Ceph Storage

Goal

Identify the key Ceph cluster performance metrics, and use them to tune and troubleshoot Ceph operations for optimal performance.

Objectives

- Choose Red Hat Ceph Storage architecture scenarios and operate Red Hat Ceph Storage-specific performance analysis tools to optimize cluster deployments.
- Protect OSD and cluster hardware resources from over-utilization by controlling scrubbing, deep scrubbing, backfill, and recovery processes to balance CPU, RAM, and I/O requirements.
- Identify key tuning parameters and troubleshoot performance for Ceph clients, including RADOS Gateway, RADOS Block Devices, and CephFS.

Sections

- Optimizing Red Hat Ceph Storage Performance (and Guided Exercise)
- Tuning Object Storage Cluster Performance (and Guided Exercise)
- Troubleshooting Clusters and Clients (and Guided Exercise)

Lab

Tuning and Troubleshooting Red Hat Ceph Storage

Optimizing Red Hat Ceph Storage Performance

Objectives

After completing this section, you should be able to choose Red Hat Ceph Storage architecture scenarios and operate Red Hat Ceph Storage-specific performance analysis tools to optimize cluster deployments.

Defining Performance Tuning

Performance tuning is the process of tailoring system configurations, so that specific, critical applications have the best possible response time or throughput. Performance tuning for a Ceph cluster has three metrics: latency, IOPS (input/output operations per second), and throughput.

Latency

It is a common misconception that disk latency and response time are the same thing. Disk latency is a function of the device, but response time is measured as a function of the entire server.

For hard drives using spinning platters, disk latency has two components:

- Seek time: The time it takes to position the drive heads on the correct track on the platter, typically 0.2 to 0.8 ms.
- Rotational latency: The additional time it takes for the correct starting sector on that track to pass under the drive heads, typically a few milliseconds.

After the drive has positioned the heads, it can start transferring data from the platter. At that point, the sequential data transfer rate is important.

For solid-state drives (SSDs), the equivalent metric is the random access latency of the storage device, which is typically less than a millisecond. For non-volatile memory express drives (NVMes), the random access latency of the storage drive is typically in microseconds.

I/O Operations Per Second (IOPS)

The number of read and write requests that the system can process per second depends on the storage device capabilities and the application. When an application issues an I/O request, the operating system transfers the request to the device and waits until the request completes. As a reference, hard drives using spinning platters achieve between 50 and 200 IOPS, SSDs are on the order of thousands to hundreds of thousands IOPS, and NVMes achieve some hundreds of thousands IOPS.

Throughput

Throughput refers to the actual number of bytes per second the system can read or write. The size of the block and the data transfer rate affect the throughput. The higher the disk block size, the more you attenuate the latency factor. The higher the data transfer rate, the faster a disk can transfer data from its surface to a buffer.

As a reference value, hard drives using spinning platters have a throughput around 150 Mb/s, SSDs are around 500 Mbps, and NVMes are in the order of 2,000 Mb/s.

You can measure throughput for networks and the whole system, from a remote client to a server.

Tuning Objectives

The hardware you use determines the performance limits of your system and your Ceph cluster. The objective of tuning performance is to use your hardware as efficiently as possible.

It is a common observation that tuning a specific subsystem can adversely affect the performance of another. For example, you can tune your system for low latency at the expense of high throughput. Therefore, before starting to tune, establish your goals to align with the expected workload of your Ceph cluster:

IOPS optimized

Workloads on block devices are often IOPS intensive, for example, databases running on virtual machines in OpenStack. Typical deployments require high-performance SAS drives for storage and journals placed on SSDs or NVMe devices.

Throughput optimized

Workloads on a RADOS Gateway are often throughput intensive. Objects can store significant amounts of data, such as audio and video content.

Capacity optimized

Workloads that require the ability to store a large quantity of data as inexpensively as possible usually trade performance for price. Selecting less-expensive and slower SATA drives is the solution for this kind of workload.

Depending on your workload, tuning objectives should include:

- Reduce latency
- Increase IOPS at the device
- Increase block size

Optimizing Ceph Performance

The following section describes recommended practices for tuning Ceph.

Ceph Deployment

It is important to plan a Ceph cluster deployment correctly. The MONs performance is critical for overall cluster performance. MONs should be on dedicated nodes for large deployments. To ensure a correct quorum, an odd number of MONs is required.

Designed to handle large quantities of data, Ceph can achieve improved performance if the correct hardware is used and the cluster is tuned correctly.

After the cluster installation, begin continuous monitoring of the cluster to troubleshoot failures and schedule maintenance activities. Although Ceph has significant self-healing abilities, many types of failure events require rapid notification and human intervention. Should performance issues occur, begin troubleshooting at the disk, network, and hardware level. Then, continue with diagnosing RADOS block devices and the Ceph RADOS Gateways.

Recommendations for OSDs

Use SSDs or NVMes to maximize efficiency when writing to BlueStore block database and write-ahead log (WAL). An OSD might have its data, block database, and WAL collocated on the same storage device, or *non-collocated* by using separate devices for each of these components.

In a typical deployment, OSDs use traditional spinning disks with high latency because they provide satisfactory metrics that meet defined goals at a lower cost per megabyte. By default, BlueStore OSDs place the data, block database, and WAL on the same block device. However,

you can maximize the efficiency by using separate low latency SSDs or NVMe devices for the block database and WAL. Multiple block databases and WALs can share the same SSD or NVMe device, reducing the cost of the storage infrastructure.

Consider the impact of the following SSD specifications against the expected workload:

- Mean Time Between Failures (MTBF) for the number of supported writes
- IOPS capabilities
- Data transfer rate
- Bus/SSD couple capabilities



Warning

When an SSD or NVMe device that hosts journals fails, every OSD using it to host its journal also becomes unavailable. Consider this when deciding how many block databases or WALs to place on the same storage device.

Recommendations for Ceph RADOS Gateways

Workloads on a RADOS Gateway are often throughput-intensive. Audio and video materials being stored as objects can be large. However, the bucket index pool typically displays a more I/O-intensive workload pattern. Store the index pools on SSD devices.

The RADOS Gateway maintains one index per bucket. By default, Ceph stores this index in one RADOS object. When a bucket stores more than 100,000 objects, the index performance degrades because the single index object becomes a bottleneck.

Ceph can keep large indexes in multiple RADOS objects, or *shards*. Enable this feature by setting the `rgw_override_bucket_index_max_shards` parameter. The recommended value is the number of objects expected in a bucket divided by 100,000.

As the index grows, Ceph must regularly reshuffle the bucket. Red Hat Ceph Storage provides a bucket index automatic reshuffling feature. The `rgw_dynamic_resharding` parameter, set to true by default, controls this feature.

Recommendations for CephFS

The metadata pool, which holds the directory structure and other indexes, can become a CephFS bottleneck. To minimize this limitation, use SSD devices for the metadata pool.

Each MDS maintains a cache in memory for different kinds of items, such as inodes. Ceph limits the size of this cache with the `mds_cache_memory_limit` parameter. Its default value, expressed in absolute bytes, is equal to 4 GB.

Placement Group Algebra

The total number of PGs in a cluster can impact overall performance due to unnecessary CPU and RAM activity on some OSD nodes. Red Hat recommends validating PG allocation for each pool before putting a cluster into production. Also consider specific testing of the backfill and recovery impact on client I/O requests.

There are two important values:

- The overall number of PGs in the cluster
- The number of PGs for a specific pool

Use this formula to estimate how many PGs should be available for a single, specific pool:

$$\text{Total Placement Groups} = (\text{OSDs} * 100) / \text{Number of replicas}$$

Apply the formula for each pool to get the total number of PGs for the cluster. Red Hat recommends between 100 and 200 PGs per OSD.



Note

Red Hat provides the **Ceph Placement Groups (PGs) per Pool Calculator** to recommend the number of PGs per pool, at <https://access.redhat.com/labs/cephpgc/>.

Splitting PGs

Ceph supports increasing or decreasing the number of PGs in a pool. If you do not specify a value when creating a pool, it is created with a default value of 8 PGs, which is very low.

The `pg_autoscale_mode` property allows Ceph to make recommendations and automatically adjust the `pg_num` and `pgp_num` parameters. This option is enabled by default when creating a new pool. The `pg_num` parameter defines the number of PGs for a specific pool. The `pgp_num` parameter defines the number of PGs that the CRUSH algorithm considers for placement.

Red Hat recommends that you make incremental increases in the number of placement groups until you reach the desired number of PGs. Increasing the number of PGs by a significant amount can cause cluster performance degradation, because the expected data relocation and rebalancing is intensive.

Use the `ceph osd pool set` command to manually increase or decrease the number of PGs by setting the `pg_num` parameter. You should only increase the number of PGs in a pool by small increments when doing it manually with the `pg_autoscale_mode` option disabled. Setting the total number of placement groups to a number that is a power of 2 provides better distribution of the PGs across the OSDs. Increasing the `pg_num` parameter automatically increases the `pgp_num` parameter, but at a gradual rate to minimize the impact on cluster performance.

Merging PGs

Red Hat Ceph Storage can merge two PGs into a larger PG, reducing the total number of PGs. Merging can be useful when the number of PGs in a pool is too large and performance is degraded. Because merging is a complex process, merge only one PG at a time to minimize the impact on cluster performance.

PG Auto-scaling

As discussed, the PG autoscale feature allows Ceph to make recommendations and automatically adjust the number of PGs. This feature is enabled by default when creating a pool. For existing pools, configure autoscaling with this command:

```
[admin@node ~]$ ceph osd pool set pool-name pg_autoscale_mode mode
```

Set the `mode` parameter to `off` to disable it, `on` to enable it and allow Ceph to automatically make adjustments in the number of PGs, or `warn` to raise health alerts when the number of PGs must be adjusted.

View the information provided by the autoscale module:

```
[admin@node ~]$ ceph osd pool autoscale-status
POOL      SIZE  TARGET SIZE  RATE  RAW CAPACITY  RATIO
pool1        0          3.0    92124M  0.0000
pool2     1323          3.0    92124M  0.0000
```

pool3	3702	3.0	92124M	0.0000	
TARGET RATIO	EFFECTIVE RATIO	BIAS	PG_NUM	NEW PG_NUM	AUTOSCALE
		1.0	1		on
		1.0	32		on
		1.0	32	64	warn

The previous table is split in two parts to make it easier to read on this page. The first two pools have the AUTOSCALE feature set to on, with Ceph automatically adjusting the number of PGs. The third pool is configured to provide a health alert if the number of PGs needs adjusting. The PG_NUM parameter is the current number of PGs in each pool or the number of PGs that the pool is working towards. The NEW PG_NUM parameter is the number of PGs that Ceph recommends to set in the pool.

Designing the Cluster Architecture

When designing a Ceph cluster, consider scaling choices to match your future data requirements and to facilitate sufficient throughput with the correct network size and architecture.

Scalability

You can scale clustered storage in two ways:

- **Scale out** by adding more nodes to a cluster.
- **Scale up** by adding more resources to existing nodes.

Scaling up requires that nodes can accept more CPU and RAM resources to handle an increase in the number of disks and disk size. Scaling out requires adding nodes with similar resources and capacity to match the cluster's existing nodes for balanced operations.

Networking Best Practices

The network interconnecting the nodes in a Ceph cluster is critical to good performance, because all client and cluster I/O operations use it. Red Hat recommends the following practices:

- To increase performance and provide better isolation for troubleshooting, use separate networks for OSD traffic and for client traffic.
- At a minimum, use 10 GB networks or larger for the storage cluster. 1 GB networks are not suitable for production environments.
- Evaluate network sizing based on both cluster and client traffic, and the amount of data stored.
- Network monitoring is highly recommended.
- Use separate NICs to connect to the networks where possible, or else use separate ports.

Figure 12.1 is a representation of such a network architecture.

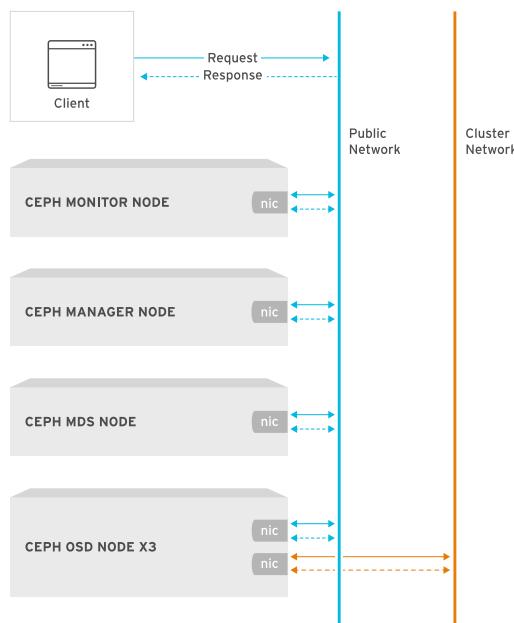


Figure 12.1: Separate networks for OSD and client traffic

The Ceph daemons automatically bind to the correct interfaces, such as binding MONs to the public network, and binding OSDs to both public and cluster networks.

Manually Controlling the Primary OSD for a PG

Use the primary affinity setting to influence Ceph's selection of a specific OSD as the primary OSD for a placement group. A higher setting makes an OSD more likely to be selected as a primary OSD. You can mitigate issues or bottlenecks by configuring the cluster to avoid using slow disks or controllers for a primary OSD. Use the `ceph osd primary-affinity` command to modify the primary affinity for an OSD. Affinity is a real number between 0 and 1.

```
[admin@node ~]$ ceph osd primary-affinity osd-number affinity
```

Recovery and Backfilling for OSDs

When Ceph adds or removes an OSD on a cluster, Ceph rebalances the PGs to use the new OSD or recreate replicas stored in the removed OSD. These backfilling and recovery operations can generate a high load of cluster network traffic, which can impact performance.

To avoid degraded cluster performance, adjust the backfilling and recovery operations to create a balance between rebalancing and normal cluster operations. Ceph provides parameters to limit the backfilling and recovery operations' I/O and network activity.

The following list includes some of those parameters:

Parameter	Definition
<code>osd_recovery_op_priority</code>	Priority for recovery operations
<code>osd_recovery_max_active</code>	Maximum number of active recovery requests per OSD in parallel
<code>osd_recovery_threads</code>	Number of threads for data recovery

Parameter	Definition
<code>osd_max_backfills</code>	Maximum number of back fills for an OSD
<code>osd_backfill_scan_min</code>	Minimum number of objects per backfill scan
<code>osd_backfill_scan_max</code>	Maximum number of objects per backfill scan
<code>osd_backfill_full_ratio</code>	Threshold for backfill requests to an OSD
<code>osd_backfill_retry_interval</code>	Seconds to wait before retrying backfill requests

Configuring Hardware

Using realistic metrics for your cluster's expected workload, build the cluster's hardware configuration to provide sufficient performance, but keep the cost as low as possible. Red Hat suggests these hardware configurations for the three performance priorities:

IOPS optimized

- Use two OSDs per NVMe device.
- NVMe drives have data, the block database, and WAL collocated on the same storage device.
- Assuming a 2 GHz CPU, use 10 cores per NVMe or 2 cores per SSD.
- Allocate 16 GB RAM as a baseline, plus 5 GB per OSD.
- Use 10 GbE NICs per 2 OSDs.

Throughput optimized

- Use one OSD per HDD.
- Place the block database and WAL on SSDs or NVMe.
- Use at least 7,200 RPM HDD drives.
- Assuming a 2 GHz CPU, use one-half core per HDD.
- Allocate 16 GB RAM as a baseline, plus 5 GB per OSD.
- Use 10 GbE NICs per 12 OSDs.

Capacity optimized

- Use one OSD per HDD.
- HDDs have data, the block database, and WAL collocated on the same storage device.
- Use at least 7,200 RPM HDD drives.
- Assuming a 2 GHz CPU, use one-half core per HDD.
- Allocate 16 GB RAM as a baseline, plus 5 GB per OSD.
- Use 10 GbE NICs per 12 OSDs.

Tuning with Ceph Performance Tools

Performance tools provide benchmarking metrics to examine clusters for performance issues.

Performance Counters and Gauges

Each Ceph daemon maintains a set of internal counters and gauges. Several tools are available to access these counters:

The Dashboard plug-in

The Dashboard plug-in exposes a web interface accessible on port 8443. The Cluster → OSDs menu provides basic real-time OSD statistics, for example, the number of read bytes, write bytes, read operations, and write operations. Enable the Dashboard plug-in by using the `ceph mgr module enable dashboard` command. If you bootstrap your cluster with the `cephadm bootstrap` command, then the dashboard is enabled by default.

The Manager (MGR) Prometheus plug-in

This plug-in exposes the performance metrics on port 9283, for an external Prometheus server to collect. Prometheus is an open source system monitoring and alerting utility.

The `ceph` command-line tool

The `ceph` command has options to view metrics and change daemon parameters.

Performance Stress Tools

Red Hat Ceph Storage provides tools to stress test and benchmark a Ceph cluster.

The RADOS bench command

RADOS bench is a simple tool for testing the RADOS Object Store. It executes write and read tests on your cluster and provides statistics. The general syntax of the command is:

```
[admin@node ~]$ rados -p pool-name bench seconds write|seq|rand \
-b objsize -t concurrency
```

These are the common parameters for the tool:

- The `seq` and `rand` tests are sequential and random read benchmarks. These tests require that a `write` benchmark is run first with the `--no-cleanup` option. By default, RADOS bench removes the objects created for the writing test. The `--no-cleanup` option keeps the objects, which can be useful for performing multiple tests on the same objects.
- The default object size, `objsize`, is 4 MB.
- The default number of concurrent operations, `concurrency`, is 16.

With the `--no-cleanup` option, you must manually remove data that remains in the pool after running the `rados bench` command.

For example, the following information is provided by the `rados bench` command, including throughput, IOPS, and latency:

```
[ceph: root@server /]# rados bench -p testbench 10 write --no-cleanup
hints = 1
Maintaining 16 concurrent writes of 4194304 bytes to objects of size 4194304 for
up to 10 seconds or 0 objects
Object prefix: benchmark_data_server.example.com_265
  sec Cur ops  started  finished  avg MB/s  cur MB/s last lat(s)  avg lat(s)
    0     0       0        0      0          0        0      -           0
    1    16      72       56   223.964      224  0.157623  0.241175
...output omitted...
  10    16     715      699   279.551      328  0.120089  0.226616
```

```
Total time run:          10.1406
Total writes made:       715
Write size:              4194304
Object size:             4194304
Bandwidth (MB/sec):    282.035
Stddev Bandwidth:        32.1911
Max bandwidth (MB/sec): 328
Min bandwidth (MB/sec): 224
Average IOPS:           70
Stddev IOPS:              8.04777
Max IOPS:                 82
Min IOPS:                 56
Average Latency(s):     0.225574
Stddev Latency(s):        0.105867
Max latency(s):           0.746961
Min latency(s):           0.0425166
```

The RBD bench command

The RBD bench measures I/O throughput and latency on an existing image, which you created for the test.

These are the default values:

- If you do not give a suffix to the size arguments, the command assumes bytes as the unit.
- The default pool name is rbd.
- The default for --io-size is 4096 bytes.
- The default for --io-threads is 16.
- The default for --io-total is 1GB.
- The default for --io-pattern is seq for sequential.

For example, information is provided by the `rbd bench` command, including throughput and latency:

```
[ceph: root@server /]# rbd bench --io-type write testimage --pool=testbench
bench type write io_size 4096 io_threads 16 bytes 1073741824 pattern sequential
  SEC      OPS      OPS/SEC      BYTES/SEC
    1      47360    46952.6    183 MiB/s
    2      88864    44461.5    174 MiB/s
    3     138016    46025.2    180 MiB/s
    4     178128    44546.4    174 MiB/s
    5     225264    45064.3    176 MiB/s
elapsed: 5  ops: 262144  ops/sec: 45274.6  bytes/sec: 177 MiB/s
```



References

`sysctl(8)`, `ceph(8)`, `rados(8)`, and `rbd(8)` man pages

For more information, refer to the *Ceph performance benchmark* chapter in the *Red Hat Ceph Administration Guide* at

https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/administration_guide/index#ceph-performance-benchmarking

► Guided Exercise

Optimizing Red Hat Ceph Storage Performance

In this exercise, you will run performance analysis tools and configure the Red Hat Ceph Storage cluster using the results.

Outcomes

You should be able to run performance analysis tools and configure the Red Hat Ceph Storage cluster using the results.

Before You Begin



Important

Do you need to reset your environment before performing this exercise?

If you performed the practice exercises in the *Managing a Red Hat Ceph Storage Cluster* chapter, but have not reset your environment to the default classroom cluster since that chapter, then you must reset your environment before executing the `lab start` command. All remaining chapters use the default Ceph cluster provided in the initial classroom 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 lab environment is available for the exercise.

```
[student@workstation ~]$ lab start tuning-optimize
```

Instructions

- Create a new pool called `testpool` and change the PG autoscale mode to `off`. Reduce the number of PGs, and then check the recommended number of PGs. Change the PG autoscale mode to `warn` and check the health warning message.
- Modify the primary affinity settings on an OSD so it is more likely to be set as primary for placement groups.
- Using the Ceph built in benchmarking tool known as the `rados bench`, measure the performance of a Ceph cluster at a pool level.
- The `clienta` node is set up as your admin node server.
- The `admin` user has SSH key-based access from the `clienta` node to the `admin` account on all cluster nodes, and has passwordless sudo access to the `root` and `ceph` accounts on all cluster nodes.

- The `serverc`, `serverd`, and `servere` nodes comprise an operational 3-node Ceph cluster. All three nodes operate as a MON, a MGR, and an OSD host with three 10 GB collocated OSDs.

**Warning**

The parameters used in this exercise are appropriate for this lab environment. In production, these parameters should only be modified by qualified Ceph administrators, or as directed by Red Hat Support.

- 1. Log in to `clienta` as the `admin` user. Create a new pool called `testpool`, set the PG autoscale mode to `warn`, reduce the number of PGs, and view the health warning messages. Set the PG autoscale mode to `on` again, and then verify the number of PGs and that cluster health is `OK` again.

- 1.1. Connect to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]#
```

- 1.2. Create a new pool called `testpool` with the default number of PGs.

```
[ceph: root@clienta /]# ceph osd pool create testpool
pool 'testpool' created
```

- 1.3. Verify the cluster health status and the information from the PG autoscaler. The autoscaler mode for the created pool `testpool` should be `on` and the number of PGs is 32.

```
[ceph: root@clienta /]# ceph health detail
HEALTH_OK
[ceph: root@clienta /]# ceph osd pool autoscale-status
POOL          SIZE TARGET_SIZE RATE  RAW_CAPACITY  RATIO  TARGET
  RATIO  EFFECTIVE RATIO  BIAS  PG_NUM  NEW PG_NUM  AUTOSCALE
device_health_metrics      0           3.0    92124M  0.0000
                           1.0       1           on
.rgw.root            1323          3.0    92124M  0.0000
                           1.0       32          on
default.rgw.log        3702          3.0    92124M  0.0000
                           1.0       32          on
default.rgw.control     0           3.0    92124M  0.0000
                           1.0       32          on
default.rgw.meta         0           3.0    92124M  0.0000
                           4.0       8           on
testpool              0           3.0    92124M  0.0000
                           1.0      32          on
```

- 1.4. Set the PG autoscale option to `off` for the pool `testpool`. Reduce the number of PGs to 8. Verify the autoscale recommended number of PGs, which should be 32. Verify that the cluster health is `OK`.

```
[ceph: root@clienta /]# ceph osd pool set testpool pg_autoscale_mode off
set pool 6 pg_autoscale_mode to off
[ceph: root@clienta /]# ceph osd pool set testpool pg_num 8
set pool 6 pg_num to 8
[ceph: root@clienta /]# ceph osd pool autoscale-status
POOL          SIZE  TARGET SIZE  RATE  RAW CAPACITY  RATIO  TARGET
  RATIO  EFFECTIVE RATIO  BIAS  PG_NUM  NEW PG_NUM  AUTOSCALE
device_health_metrics      0           3.0        92124M  0.0000
                           1.0       1           on
.rgw.root                1323        3.0        92124M  0.0000
                           1.0       32          on
default.rgw.log            3702        3.0        92124M  0.0000
                           1.0       32          on
default.rgw.control         0           3.0        92124M  0.0000
                           1.0       32          on
default.rgw.meta             0           3.0        92124M  0.0000
                           4.0       8           on
testpool                  0           3.0        92124M  0.0000
                           1.0       8           32  off
[ceph: root@clienta /]# ceph health detail
HEALTH_OK
```

- Set the PG autoscale option to warn for the pool `testpool`. Verify that cluster health status is now `WARN`, because the recommended number of PGs is higher than the current number of PGs. It might take several minutes before the cluster shows the health warning message.

```
[ceph: root@clienta /]# ceph osd pool set testpool pg_autoscale_mode warn
set pool 6 pg_autoscale_mode to warn
[ceph: root@clienta /]# ceph health detail
HEALTH_WARN 1 pools have too few placement groups
[WRN] POOL_TOO_FEW_PGS: 1 pools have too few placement groups
  Pool testpool has 8 placement groups, should have 32
```

- Enable the PG autoscale option and verify that the number of PGs has been increased automatically to 32, the recommended value. This increase might take a few minutes to display.

```
[ceph: root@clienta /]# ceph osd pool set testpool pg_autoscale_mode on
set pool 6 pg_autoscale_mode to on
[ceph: root@clienta /]# ceph osd pool autoscale-status
POOL          SIZE  TARGET SIZE  RATE  RAW CAPACITY  RATIO  TARGET
  RATIO  EFFECTIVE RATIO  BIAS  PG_NUM  NEW PG_NUM  AUTOSCALE
device_health_metrics      0           3.0        92124M  0.0000
                           1.0       1           on
.rgw.root                1323        3.0        92124M  0.0000
                           1.0       32          on
default.rgw.log            3702        3.0        92124M  0.0000
                           1.0       32          on
```

default.rgw.control	0	3.0	92124M	0.0000
	1.0	32	on	
default.rgw.meta	0	3.0	92124M	0.0000
	4.0	8	on	
testpool	0	3.0	92124M	0.0000
	1.0	32	on	

- ▶ 2. Modify the primary affinity settings on an OSD so that it is more likely to be selected as primary for placement groups. Set the primary affinity for OSD 7 to 0.

2.1. Modify the primary affinity settings for OSD 7.

```
[ceph: root@clienta /]# ceph osd primary-affinity 7 0
set osd.7 primary-affinity to 0 (802)
```

2.2. Verify the primary affinity settings for each OSD.

ID	CLASS	WEIGHT	TYPE	NAME	STATUS	REWEIGHT	PRI-AFF
-1		0.08817	root	default			
-3		0.02939	host	serverc			
0	hdd	0.00980		osd.0	up	1.00000	1.00000
1	hdd	0.00980		osd.1	up	1.00000	1.00000
2	hdd	0.00980		osd.2	up	1.00000	1.00000
-5		0.02939	host	serverd			
3	hdd	0.00980		osd.3	up	1.00000	1.00000
5	hdd	0.00980		osd.5	up	1.00000	1.00000
7	hdd	0.00980		osd.7	up	1.00000	0
-7		0.02939	host	servere			
4	hdd	0.00980		osd.4	up	1.00000	1.00000
6	hdd	0.00980		osd.6	up	1.00000	1.00000
8	hdd	0.00980		osd.8	up	1.00000	1.00000

2.3. Verify the primary affinity settings for OSDs in the cluster.

```
[ceph: root@clienta /]# ceph osd dump | grep affinity
osd.7 up in weight 1 primary_affinity 0 up_from
45 up_thru 92 down_at 0 last_clean_interval [0,0)
[v2:172.25.250.13:6816/3402621793,v1:172.25.250.13:6817/3402621793]
[v2:172.25.249.13:6818/3402621793,v1:172.25.249.13:6819/3402621793] exists,up
ebc2280d-1321-458d-a161-2250d2b4f32e
```

- ▶ 3. Create a pool called benchpool with the object clean-up feature turned off.

3.1. Create an OSD pool called benchpool.

```
[ceph: root@clienta /]# ceph osd pool create benchpool 100 100
pool 'benchpool' created
```

3.2. Use the rbd pool init command to initialize a custom pool to store RBD images. This step could take several minutes to complete.

```
[ceph: root@clienta /]# rbd pool init benchpool
```

- ▶ 4. Open a second terminal and log in to the `clienta` node as the `admin` user. Use the first terminal to generate a workload and use the second terminal to collect metrics. Run a write test to the RBD pool `benchpool`. This might take several minutes to complete.

**Note**

This step requires sufficient time to complete the write OPS for the test. Be prepared to run the `osd pref` command in the second terminal immediately after starting the `benchpool` command in the first terminal.

- 4.1. Open a second terminal. Log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]#
```

- 4.2. In the first terminal, generate the workload.

```
[ceph: root@clienta /]# rados -p benchpool bench 30 write
hints = 1
Maintaining 16 concurrent writes of 4194304 bytes to objects of size 4194304 for
up to 30 seconds or 0 objects
Object prefix: benchmark_data_clienta.lab.example.com_50
  sec Cur ops    started   finished   avg MB/s  cur MB/s last lat(s)  avg lat(s)
    0     0        0          0        0           0        0          -            0
    1     16       58         42      167.988      168    0.211943    0.322053
    2     16      112         96      191.982      216    0.122236    0.288171
    3     16      162        146      194.643      200    0.279456    0.300593
    4     16      217        201      200.975      220    0.385703    0.292009
...output omitted...
```

- 4.3. In the second terminal, collect performance metrics. The `commit_latency` data is the time for the OSD to write and commit the operation to its journal. The `apply_latency` data is the time to apply the write operation to the OSD file system back end. Note the OSD ID where the heavy load is occurring. Your OSD output might be different in your lab environment.

```
[ceph: root@clienta /]# ceph osd perf
osd  commit_latency(ms)  apply_latency(ms)
osd  commit_latency(ms)  apply_latency(ms)
  7              94          94
  8             117          117
  6              195          195
  1              73          73
  0              72          72
  2              80          80
```

3	72	72
4	135	135
5	59	59

**Note**

If no data displays, then use the first terminal to generate the workload again. The metric collection must run while the bench tool is generating workload.

- 4.4. In the second terminal, locate the system by using the OSD ID from the previous step, where the OSD has high latency. Determine the name of the system.

```
[ceph: root@clienta /]# ceph osd tree
ID CLASS WEIGHT TYPE NAME STATUS REWEIGHT PRI-AFF
-1          0.08817 root default
-3          0.02939 host serverc
 0  hdd  0.00980    osd.0   up   1.00000  1.00000
 1  hdd  0.00980    osd.1   up   1.00000  1.00000
 2  hdd  0.00980    osd.2   up   1.00000  1.00000
-5          0.02939 host serverd
 3  hdd  0.00980    osd.3   up   1.00000  1.00000
 5  hdd  0.00980    osd.5   up   1.00000  1.00000
 7  hdd  0.00980    osd.7   up   1.00000      0
-7          0.02939 host servere
 4  hdd  0.00980    osd.4   up   1.00000  1.00000
6  hdd  0.00980    osd.6  up   1.00000  1.00000
 8  hdd  0.00980    osd.8   up   1.00000  1.00000
```

- 5. Evaluate the OSD performance counters.

- 5.1. Verify the performance counters for the OSD. Redirect the output of the command to a file called `perfdump.txt`

```
[ceph: root@clienta /]# ceph tell osd.6 perf dump > perfdump.txt
```

- 5.2. In the `perfdump.txt` file, locate the section starting with `osd:`. Note the `op_latency` and `subop_latency` counters, which are the read and write operations and suboperations latency. Note the `op_r_latency` and `op_w_latency` parameters.

Each counter includes `avgcount` and `sum` fields that are required to calculate the exact counter value. Calculate the value of the `op_latency` and `subop_latency` counters by using the formula `counter = counter.sum / counter.avgcount`.

```
[ceph: root@clienta /]# cat perfdump.txt | grep -A88 'osd'
"osd": {
  "op_wip": 0,
  "op": 3664,
  "op_in_bytes": 994050158,
  "op_out_bytes": 985,
  "op_latency": {
    "avgcount": 3664,
    "sum": 73.819483299,
```

```

        "avgtime": 0.020147238
    },
...output omitted...
    "op_r_latency": {
        "avgcount": 3059,
        "sum": 1.395967825,
        "avgtime": 0.000456347
    },
...output omitted...
    "op_w_latency": {
        "avgcount": 480,
        "sum": 71.668254827,
        "avgtime": 0.149308864
    },
...output omitted...
    "op_rw_latency": {
        "avgcount": 125,
        "sum": 0.755260647,
        "avgtime": 0.006042085
    },
...output omitted...
    "subop_latency": {
        "avgcount": 1587,
        "sum": 59.679174303,
        "avgtime": 0.037605024
    },
...output omitted...

```

- 5.3. In the first terminal, repeat the capture using the `rados bench write` command.

```
[ceph: root@clienta /]# rados -p benchpool bench 30 write
...output omitted...
```

- 5.4. In the second terminal, view the variation of the value using the following formulas:

- `op_latency_sum_t2 - op_latency_sum_t1 = diff_sum`
- `op_latency_avgcount_t2 - op_latency_avgcount_t1 = diff_avgcount`
- `op_latency = diff_sum / diff_avgcount`

```
[ceph: root@clienta /]# ceph tell osd.6 perf dump > perfdump.txt
[ceph: root@clienta /]# cat perfdump.txt | grep -A88 '"osd"'
...output omitted...
```



Note

The values are cumulative and are returned when the command is executed.

- 6. View information about the last operations processed by an OSD.

- 6.1. In the second terminal, dump the information maintained in memory for the most recently processed operations. Redirect the dump to the `historicdump.txt`

file. By default, each OSD records information on the last 20 operations over 600 seconds. View the `historicdump.txt` file contents.

```
[ceph: root@clienta /]# ceph tell osd.6 dump_historic_ops > historicdump.txt
[ceph: root@clienta /]# head historicdump.txt
{
    "size": 20,
    "duration": 600,
    "ops": [
        {
            "description": "osd_op(client.44472.0:479 7.14
7:2a671f00:::benchmark_data_clienta.lab.example.com_92_object478:head [set-alloc-
hint object_size 4194304 write_size 4194304,write 0-4194304] snapc 0=[] ondisk
+write+known_if_redirected e642)",
            ...
        }
    ]
}
```

- 6.2. Update the values for the `osd_op_history_size` and `osd_op_history_duration` parameters. Set the size to 30 and the duration to 900. Verify that the change was successful.

```
[ceph: root@clienta /]# ceph tell osd.6 config set osd_op_history_size 30
{
    "success": "osd_op_history_size = '30' "
}
[ceph: root@clienta /]# ceph tell osd.6 config set osd_op_history_duration 900
{
    "success": "osd_op_history_duration = '900' "
}
[ceph: root@clienta /]# ceph tell osd.6 dump_historic_ops > historicops.txt
[ceph: root@clienta /]# head -n 3 historicops.txt
{
    "size": 30,
    "duration": 900,
```

- 6.3. Update the runtime value of the `osd_op_history_size` and `osd_op_history_duration` parameters. Verify that the change was successful.

```
[ceph: root@clienta /]# ceph tell osd.* config set osd_op_history_size 20
osd.0: {
    "success": "osd_op_history_size = '20' "
}
...
osd.8: {
    "success": "osd_op_history_size = '20' "
}
[ceph: root@clienta /]# ceph tell osd.* config set osd_op_history_duration 600
osd.0: {
    "success": "osd_op_history_duration = '600' "
}
...
osd.8: {
    "success": "osd_op_history_duration = '600' "
}
```

- 7. Exit the second terminal. Return to workstation as the student user.

```
[ceph: root@clienta /]# exit  
[admin@clienta ~]$ exit  
[student@workstation ~]$ exit
```

```
[ceph: root@clienta /]# exit  
[admin@clienta ~]$ exit  
[student@workstation ~]$
```

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 tuning-optimize
```

This concludes the guided exercise.

Tuning Object Storage Cluster Performance

Objectives

After completing this section, you should be able to protect OSD and cluster hardware resources from over-utilization by controlling scrubbing, deep scrubbing, backfill, and recovery processes to balance CPU, RAM, and I/O requirements.

Maintaining OSD Performance

Good client performance requires utilizing your OSDs within their physical limits. To maintain OSD performance, evaluate these tuning opportunities:

- Tune the BlueStore back end used by OSDs to store objects on physical devices.
- Adjust the schedule for automatic data scrubbing and deep scrubbing.
- Adjust the schedule of asynchronous snapshot trimming (deleting removed snapshots).
- Control how quickly backfill and recovery operations occur when OSDs fail or are added or replaced.

Storing Data on Ceph BlueStore

The default back-end object store for OSD daemons is BlueStore. The following list describes some of the main features of using BlueStore:

Direct management of storage devices

BlueStore consumes raw block devices or partitions. This simplifies the management of storage devices because no other abstraction layers, such as local file systems, are required.

Efficient copy-on-write

The Ceph Block Device and Ceph File System snapshots rely on a copy-on-write clone mechanism that is implemented efficiently in BlueStore. This results in efficient I/O for regular snapshots and for erasure-coded pools that rely on cloning to implement efficient two-phase commits.

No large double writes

BlueStore first writes any new data to unallocated space on a block device, and then commits a RocksDB transaction that updates the object metadata to reference the new region of the disk.

Multidevice support

BlueStore can use multiple block devices for storing the data, metadata, and write-ahead log.

In BlueStore, the raw partition is managed in chunks of the size specified by the `bluestore_min_alloc_size` variable. The `bluestore_min_alloc_size` is set by default to 4,096, which is equivalent to 4 KB, for HDDs and SSDs. If the data to write in the raw partition is smaller than the chunk size, then it is filled with zeroes. This can lead to a waste of the unused space if the chunk size is not properly sized for your workload, such as for writing many small objects.

Red Hat recommends setting the `bluestore_min_alloc_size` variable to match the smallest common write to avoid wasting unused space. For example, if your client writes 4 KB objects frequently, then configure the settings on OSD nodes such as `bluestore_min_alloc_size`

= 4096. Setting the `bluestore_min_alloc_size` variable overrides specific settings for HDD or SSD if previously set with the `bluestore_min_alloc_size_ssd` or `bluestore_min_alloc_size_hdd` variables.



Important

Red Hat does not recommend changing the `bluestore_min_alloc_size` value in your production environment before first contacting Red Hat Support.

Set the value for the `bluestore_min_alloc_size` variable by using the `ceph config` command:

```
[root@node ~]# ceph config set osd.ID bluestore_min_alloc_size_device-type value
```

The BlueStore Fragmentation Tool

An OSD's free space becomes fragmented over time. Fragmentation is normal, but excess fragmentation degrades OSD performance. When using BlueStore, review fragmentation levels using the BlueStore fragmentation tool. The BlueStore fragmentation tool generates a fragmentation level score for the BlueStore OSD. The fragmentation score is between 0 and 1, with 0 indicating no fragmentation, and 1 indicating severe fragmentation.

For reference, a value between 0 and 0.7 is considered small and acceptable fragmentation, a score between 0.7 and 0.9 is considerable but still safe fragmentation, and scores higher than 0.9 indicates severe fragmentation that is causing performance issues.

View the fragmentation score using the BlueStore fragmentation tool:

```
[root@node ~]# ceph daemon osd.ID bluestore allocator score block
```

Maintaining Data Coherence with Scrubbing

OSDs are responsible for validating data coherence, using light scrubbing and deep scrubbing. Light scrubbing verifies an object's presence, checksum, and size. Deep scrubbing reads the data and recalculates and verifies the object's checksum.

By default, Red Hat Ceph Storage performs light scrubbing every day and deep scrubbing every week. However, Ceph can begin the scrubbing operation at any time, which can impact cluster performance. You can enable or disable cluster level light scrubbing by using the `ceph osd set noscrub` and `ceph osd unset noscrub` commands. Although scrubbing has a performance impact, Red Hat recommends keeping the feature enabled because it maintains data integrity. Red Hat recommends setting the scrubbing parameters to restrict scrubbing to known periods with the lowest workloads.



Note

The default configuration allows light scrubbing at any time during the day.

Light Scrubbing

Tune the light scrubbing process by adding parameters in the [osd] section of the `ceph.conf` file. For example, use the `osd_scrub_begin_hour` parameter to set the time of day that light scrubbing begins, thereby avoiding light scrubbing during peak workloads.

The light scrubbing feature has the following tuning parameters: `osd_scrub_begin_hour = begin_hour`: The `begin_hour` parameter specifies the time to start scrubbing. Valid values are from 0 to 23. If the value is set to 0 and `osd_scrub_end_hour` is also 0, then scrubbing is allowed the entire day.

`osd_scrub_end_hour = end_hour`

The `end_hour` parameter specifies the time to stop scrubbing. Valid values are from 0 to 23. If the value is set to 0 and `osd_scrub_begin_hour` is also 0, then scrubbing is allowed the entire day.

`osd_scrub_load_threshold`

Perform a scrub if the system load is below the threshold, defined by the `getloadavg() / number_online CPUs` parameter. The default value is 0.5.

`osd_scrub_min_interval`

Perform a scrub no more often than the number of seconds defined in this parameter if the load is below the threshold set in the `osd_scrub_load_threshold` parameter. The default value is 1 day.

`osd_scrub_interval_randomize_ratio`

Add a random delay to the value defined in the `osd_scrub_min_interval` parameter. The default value is 0.5.

`osd_scrub_max_interval`

Do not wait more than this period before performing a scrub, regardless of load. The default value is 7 days.

`osd_scrub_priority`

Set the priority for scrub operations by using this parameter. The default value is 5. This value is relative to the value of the `osd_client_op_priority`, which has a higher default priority of 63.

Deep Scrubbing

You can enable and disable deep scrubbing at the cluster level by using the `ceph osd set nodeep-scrub` and `ceph osd unset nodeep-scrub` commands. You can configure deep scrubbing parameters by adding them to the [osd] section of the `ceph.conf` configuration file. As with the light scrubbing parameters, any changes made to the deep scrub configuration can impact cluster performance.

The following parameters are the most critical to tuning deep scrubbing:

`osd_deep_scrub_interval`

The interval for deep scrubbing. The default value is 7 days.

`osd_scrub_sleep`

Introduces a pause between deep scrub disk reads. Increase this value to slow down scrub operations and to have a lower impact on client operations. The default value is 0.

You can use an external scheduler to implement light and deep scrubbing by using the following commands:

- The `ceph pg dump` command displays the last light and deep scrubbing occurrences in the `LAST_SCRUB` and `LAST_DEEP_SCRUB` columns.
- The `ceph pg scrub pg-id` command schedules a deep scrub on a particular PG.
- The `ceph pg deep-scrub pg-id` command schedules a deep scrub on a particular PG.

Use the `ceph osd pool set pool-name parameter value` command to set these parameters for a specific pool.

Pool Parameters for Scrubbing

You can also control light scrubbing and deep scrubbing at the pool level with these pool parameters:

`noscrub`

If set to `true`, Ceph does not light scrub the pool. The default value is `false`.

`nodeep-scrub`

If set to `true`, Ceph does not deep scrub the pool. The default value is `false`.

`scrub_min_interval`

Scrub no more often than the number of seconds defined in this parameter. If set to the default 0, then Ceph uses the `osd_scrub_min_interval` global configuration parameter.

`scrub_max_interval`

Do not wait more than the period defined in this parameter before scrubbing the pool. If set to the default 0, Ceph uses the `osd_scrub_max_interval` global configuration parameter.

`deep_scrub_interval`

The interval for deep scrubbing. If set to the default 0, Ceph uses the `osd_deep_scrub_interval` global configuration parameter.

Trimming Snapshots and OSDs

Snapshots are available at the pool and RBD levels. When a snapshot is removed, Ceph schedules the removal of the snapshot data as an asynchronous operation known as *snapshot trimming*.

To reduce the impact of the snapshot trimming process on the cluster, you can configure a pause after the deletion of each snapshot object. Configure this pause by using the `osd_snap_trim_sleep` parameter, which is the time in seconds to wait before allowing the next snapshot trimming operation. The default value for this parameter is 0. Contact Red Hat Support for further advice on how to set this parameter based on your environment settings.

Control the snapshot trimming process using the `osd_snap_trim_priority` parameter, which has a default value of 5.

Controlling Backfill and Recovery

Controlling backfill and recovery operations is necessary to limit the impact of these operations and to preserve cluster performance.

Backfill occurs when a new OSD joins the cluster or when an OSD dies and Ceph reassigns its PGs to other OSDs. When such events occur, Ceph creates object replicas across the available OSDs.

Recovery occurs when a Ceph OSD becomes inaccessible and comes back online, for example due to a short outage. The OSD goes into recovery mode to obtain the latest copy of the data.

Use the following parameters to manage the backfill and recovery operations:

osd_max_backfills

Control the maximum number of concurrent backfill operations per OSD. The default value is 1.

osd_recovery_max_active

Control the maximum number of concurrent recovery operations per OSD. The default value is 3.

osd_recovery_op_priority

Set the recovery priority. The value can range from 1 - 63. The higher the number, the higher the priority. The default value is 3.



References

For more information, refer to the OSD Configuration Reference chapter of the *Configuration Guide for Red Hat Ceph Storage* at

https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/configuration_guide/index#ceph-monitor-and-osd-configuration-options_conf

For more information on scrubbing and backfilling, refer to

https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/configuration_guide/index#ceph-object-storage-daemon-configuration

For more information on tuning Red Hat Ceph Storage 5 BlueStore, refer to

https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/administration_guide/index#osd-bluestore

► Guided Exercise

Tuning Object Storage Cluster Performance

In this exercise, you will tune the recovery and backfill processes to preserve cluster performance.

Outcomes

You should be able to:

- Inspect your OSDs for BlueStore fragmentation.
- Preserve cluster performance under heavy loads.
- Adapt how backfill requests are processed for placement group migration to deal with added or removed OSDs.
- Adjust recovery requests to synchronize a recovered OSD with the OSD's peers following an OSD crash.

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 tuning-perf
```

Instructions

- The `clienta` node is the admin node and is a client of the Ceph cluster.
- The `serverc`, `serverd`, and `servere` nodes are an operational 3-node Ceph cluster. All three nodes operate as a MON, a MGR, and an OSD host with three 10 GB collocated OSDs.



Warning

The parameters used in this exercise are appropriate for this lab environment. In production, these parameters should only be modified by qualified Ceph administrators, or as directed by Red Hat Support.

- 1. Log in to `clienta` as the `admin` user. Inspect OSD 0 for BlueStore fragmentation.

- 1.1. Connect to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]#
```

- 1.2. Retrieve information about OSD 0 fragmentation. The value should be low because the number of operations in the cluster is low and the cluster is new.

```
[ceph: root@clienta /]# ceph tell osd.0 bluestore allocator score block
{
    "fragmentation_rating": 0.0016764709285897418
}
```

- ▶ 2. By default, Red Hat Ceph Storage allows one PG backfill at a time, to or from an OSD. Modify this parameter to 2 on a per-OSD basis. Configure PG backfilling on an OSD.
- 2.1. Select one OSD running on the `serverc` node and obtain its IDs. In the following example, the options are the `osd.0`, `osd.1` and `osd.2` OSDs. Yours might be different.

```
[ceph: root@clienta /]# ceph osd tree
ID CLASS WEIGHT TYPE NAME STATUS REWEIGHT PRI-AFF
-1          0.08817  root default
-3          0.02939  host serverc
  0  hdd  0.00980  osd.0        up   1.00000  1.00000
  1  hdd  0.00980  osd.1        up   1.00000  1.00000
  2  hdd  0.00980  osd.2        up   1.00000  1.00000
-7          0.02939  host serverd
  3  hdd  0.00980  osd.3        up   1.00000  1.00000
  5  hdd  0.00980  osd.5        up   1.00000  1.00000
  7  hdd  0.00980  osd.7        up   1.00000  1.00000
-5          0.02939  host servere
  4  hdd  0.00980  osd.4        up   1.00000  1.00000
  6  hdd  0.00980  osd.6        up   1.00000  1.00000
  8  hdd  0.00980  osd.8        up   1.00000  1.00000
```

- 2.2. On your selected OSD on host `serverc`, retrieve the value for the `osd_max_backfills` parameter. In this example, the selected OSD is `osd.0`.

```
[ceph: root@clienta /]# ceph tell osd.0 config get osd_max_backfills
{
    "osd_max_backfills": "1"
}
```

- 2.3. Modify the current runtime value for the `osd_max_backfills` parameter to 2.

```
[ceph: root@clienta /]# ceph tell osd.0 config set osd_max_backfills 2
{
    "success": "osd_max_backfills = '2' "
}
```

- ▶ 3. By default, Red Hat Ceph Storage allows three simultaneous recovery operations for HDDs and ten for SSDs. Modify the maximum number of data recovery operations to 1 per OSD.
- 3.1. Verify the value of the `osd_recovery_max_active` parameter on the OSD of your choice. The default value for the `osd_recovery_max_active` is 0, meaning that the values in `osd_recovery_max_active_hdd` and `osd_recovery_max_active_ssd` are used instead.

```
[ceph: root@clienta /]# ceph tell osd.0 config get osd_recovery_max_active
{
    "osd_recovery_max_active": "0"
}
[ceph: root@clienta /]# ceph tell osd.0 config get osd_recovery_max_active_hdd
{
    "osd_recovery_max_active_hdd": "3"
}
[ceph: root@clienta /]# ceph tell osd.0 config get osd_recovery_max_active_ssd
{
    "osd_recovery_max_active(ssd)": "10"
}
```

- 3.2. Set the current runtime for the `osd_recovery_max_active` parameter to 1 on the OSD of your choice. Verify that the changes are applied.

```
[ceph: root@clienta /]# ceph tell osd.0 config set osd_recovery_max_active 1
{
    "success": "osd_recovery_max_active = '1' "
}
[ceph: root@clienta /]# ceph tell osd.0 config get osd_recovery_max_active
{
    "osd_recovery_max_active": "1"
}
```

- 4. Return to workstation as the student user.

```
[ceph: root@clienta /]# exit
[admin@clienta ~]$ exit
[student@workstation ~]$
```

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 tuning-perf
```

This concludes the guided exercise.

Troubleshooting Clusters and Clients

Objectives

After completing this section, you should be able to identify key tuning parameters and troubleshoot performance for Ceph clients, including RADOS Gateway, RADOS Block Devices, and CephFS.

Beginning Troubleshooting

The hardware backing a Ceph cluster is subject to failure over time. The data in your cluster becomes fragmented and requires maintenance. You should perform consistent monitoring and troubleshooting in your cluster to keep it in a healthy state. This section presents some practices that enable troubleshooting for various issues on a Ceph cluster. You can perform this initial troubleshooting of your cluster before contacting Red Hat Support.

Identifying Problems

When troubleshooting issues with Ceph, the first step is to determine which Ceph component is causing the problem. Sometimes, you can find this component in the information provided by the `ceph health detail` or `ceph health status` commands. Other times, you must investigate further to discover the issue. Verify a cluster's status to help determine if there is a single failure or an entire node failure.

The following troubleshooting checklist suggests next steps:

- Identify the Ceph component causing the problem.
- Set debug logging for the identified component and view the logs.
- Verify that you have a supported configuration.
- Determine if there are slow or stuck operations.

Troubleshooting Cluster Health

Red Hat Ceph Storage continually runs various health checks to monitor the health of the cluster. When a health check fails, the cluster health state changes to either `HEALTH_WARN` or `HEALTH_ERR`, depending on the severity and impact of the failed health checks. Red Hat Ceph Storage also logs the health check warnings and errors to the cluster logs.

The `ceph status` and `ceph health` commands show the cluster health status. When the cluster health status is `HEALTH_WARN` or `HEALTH_ERR`, use the `ceph health detail` command to view the health check message so that you can begin troubleshooting the issue.

```
[ceph: root@node /]# ceph health detail
```

Some health status messages indicate a specific issue; others provide a more general indication. For example, if the cluster health status changes to `HEALTH_WARN` and you see the health message `HEALTH_WARN 1 osds down; Degraded data redundancy`, then that is a clear indication of the problem.

Other health status messages might require further troubleshooting because they might indicate several possible root causes. For example, the following message indicates an issue that has multiple possible solutions:

```
[ceph: root@node /]# ceph health detail
HEALTH_WARN 1 pools have too few placement groups
[WRN] POOL_TO0_FEW_PGS: 1 pools have too few placement groups
    Pool testpool has 8 placement groups, should have 32
```

You can resolve this issue by changing the `pg_num` setting on the specified pool, or by reconfiguring the `pg_autoscaler` mode setting from `warn` to `on` so that Ceph automatically adjusts the number of PGs.

Ceph sends health messages regarding performance when a cluster performance health check fails. For example, OSDs send heartbeat ping messages to each other to monitor OSD daemon availability. Ceph also uses the OSD ping response times to monitor network performance. A single failed OSD ping message could mean a delay from a specific OSD, indicating a potential problem with that OSD. Multiple failed OSD ping messages might indicate a failure of a network component, such as a network switch between OSD hosts.



Note

View the list of health check messages of a Ceph cluster at https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/troubleshooting_guide/index#health-messages-of-a-ceph-cluster_diag.

You can find a list of health check messages specific to CephFS at https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/file_system_guide/index#health-messages-for-the-ceph-file-system_fs.

Muting Ceph Health Alerts

You might want to temporarily mute some of the cluster warnings because you already know of them and do not need to fix them yet. For example, if you bring down an OSD for maintenance, then the cluster reports a `HEALTH_WARN` status. You can mute this warning message so that the health check does not affect the overall reported status.

Ceph specifies the health check alert by using health check codes. For example, the previous `HEALTH_WARN` message shows the `POOL_TO0_FEW_PGS` health code.

To mute a health alert message, use the `ceph health` command.

```
[ceph: root@node /]# ceph health mute health-code [duration]
```

The `health-code` is the code provided by the `ceph health detail` command. The optional parameter `duration` is the time that the health message is muted, specified in seconds, minutes, or hours. You can unmute a health message with the `ceph health unmute health-code` command.

When you mute a health message, Ceph automatically unmutes the alert if the health status further degrades. For example, if your cluster reports one OSD down and you mute that alert, Ceph automatically removes the mute if another OSD goes down. Any health alerts that can be measured unmute.

Configuring Logging

If there is a problem in a specific area of your cluster, then you can enable logging for that area. For example, if your OSDs are running adequately but your metadata servers are not, enable debug logging for the specific metadata server instances. Enable logging for each subsystem as needed.

Adding debugging to your Ceph configuration is typically done temporarily during runtime. You can add Ceph debug logging to your Ceph configuration database if you encounter issues when starting your cluster. View Ceph log files under the default location `/var/log/ceph`. Ceph stores logs in a memory-based cache.



Warning

Logging is resource-intensive. Verbose logging can generate over 1 GB of data per hour. If your OS disk reaches its capacity, then the node stops working. When you fix your cluster issues, revert the logging configuration to default values. Consider setting up log file rotation.

Understanding Ceph Logs

Configure Ceph logging by using the `ceph` command at runtime. If you encounter errors when starting up the cluster, then you can update the Ceph configuration database so that it logs during startup.

You can set different logging levels for each subsystem in your cluster. Debug levels are on a scale of 1 to 20, where 1 is terse and 20 is verbose.

Ceph does not send memory-based logs to the output logs except in the following circumstances:

- A fatal signal is raised.
- An assert in code is triggered.
- You request it.

To use different debug levels for the output log level and the memory level, use a slash (/) character. For example, `debug_mon = 1/5` sets the output log level of the `ceph-mon` daemon to 1 and its memory log level to 5.

Configure Logging at Runtime

To activate debugging output at runtime, use the `ceph tell` command.

```
[ceph: root@node /]# ceph tell type.id config set debug_subsystem debug-level
```

The `type` and `id` arguments are the type of the Ceph daemon and its ID. The `subsystem` is the specific subsystem whose debug level you want to modify.



Note

You can find a list of the subsystems at https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/troubleshooting_guide/index#ceph-subsystems_diag.

This example modifies the OSD 0 debug level for the messaging system between Ceph components:

```
[ceph: root@node /]# ceph tell osd.0 config set debug_ms 5
```

View the configuration settings at runtime as follows:

```
[ceph: root@node /]# ceph tell osd.0 config show
```

Configure Logging in the Configuration Database

Configure the subsystem debug levels so that they log to the default log file at boot time. Add the debugging settings to the Ceph configuration database by using the `ceph config set` command.

For example, add debug levels for specific Ceph daemons by setting these parameters in your Ceph configuration database:

```
[ceph: root@node /]# ceph config set global debug_ms 1/5
[ceph: root@node /]# ceph config set osd debug_ms 1
[ceph: root@node /]# ceph config set osd debug_osd 1/5
[ceph: root@node /]# ceph config set mon debug_mon 20
```

Setting Log File Rotation

Debug logging for Ceph components is resource-intensive and can generate a huge amount of data. If you have almost full disks, then accelerate log rotation by modifying the log rotation configuration at `/etc/logrotate.d/ceph`. The Cron job scheduler uses this file to schedule log rotation.

You can add a size setting after the rotation frequency, so that the log file is rotated when it reaches the specified size:

```
rotate 7
weekly
size size
compress
sharedscripts
```

Use the `crontab` command to add an entry to inspect the `/etc/logrotate.d/ceph` file.

```
[ceph: root@node /]# crontab -e
```

For example, you can instruct Cron to check `/etc/logrotate.d/ceph` every 30 minutes.

```
30 * * * * /usr/sbin/logrotate /etc/logrotate.d/ceph >/dev/null 2>&1
```

Troubleshooting Network Issues

Ceph nodes use the network for communicating with each other. Network issues can be the cause when OSDs are reported as down. Monitors with `clock skew` errors are a common cause of networking issues. A clock skew, or timing skew, is a phenomenon in synchronous digital circuit systems in which the same sourced clock signal arrives at different components at different times. If the difference between the readings is too far apart from what is configured in the cluster, then

you get the clock skew error. This error can cause packet loss, high latency, or limited bandwidth, impacting cluster performance and stability.

A following network troubleshooting checklist suggests next steps:

- Ensure that the `cluster_network` and `public_network` parameters in the cluster include correct values. You can retrieve their values by using the `ceph config get mon cluster_network` or `ceph config get mon public_network` commands, or by checking the `ceph.conf` file.
- Verify that all network interfaces are functional.
- Verify the Ceph nodes and verify they are able to reach each other using their host names.
- Ensure that Ceph nodes are able to reach each other on their appropriate ports, if firewalls are used. Open the appropriate firewall ports if necessary.
- Validate that network connectivity between hosts has the expected latency and no packet loss, for example, by using the `ping` command.
- Slower connected nodes could slow down the faster ones. Verify that the inter-switch links can handle the accumulated bandwidth of the connected nodes.
- Verify that NTP is working correctly in your cluster nodes. For example, you can check the information provided by the `chronyc tracking` command.

Troubleshooting Ceph Clients

The following list includes the most common problems that clients experience when accessing a Red Hat Ceph Storage cluster:

- Monitors (MONs) are not available to the client.
- Incorrect or missing command line arguments that result from using the CLI.
- The `/etc/ceph/ceph.conf` file is incorrect, missing, or inaccessible.
- Key-ring files are incorrect, missing, or inaccessible.

The `ceph-common` package provides bash tab completion for the `rados`, `ceph`, `rbd`, and `radosgw-admin` commands. You can access option and attribute completions by pressing the Tab key when you enter the command at the shell prompt.

Enabling and Changing Log Files

Increase the logging level when troubleshooting a client.

On the client system, you can add the `debug_ms = 1` parameter to the configuration database by using the `ceph config set client debug_ms 1` command. The Ceph client stores debug messages in the `/var/log/ceph/ceph-client.id.log` log file.

Most of the Ceph client commands, such as `rados`, `ceph`, or `rbd`, also accept the `--debug-ms=1` option to execute only that command with an increased logging level.

Enabling the Client Admin Socket

By default, Ceph clients create a UNIX domain socket on start up. You can use this socket to communicate with the client to retrieve real-time performance data or to dynamically get or set a configuration parameter.

In the `/var/run/ceph/fsid` directory, there is a list of admin sockets for that host. Allow one admin socket per OSD, one for each MON, and one for each MGR. Administrators can use the

ceph command with the `--admin-daemon socket-patch` option to query the client through the socket.

```
[ceph: root@node /]# sudo ls -al /var/run/ceph/fsid
total 0
drwxrwx--- 2 167 167 180 Oct 19 04:43 .
drwxr-xr-x 3 root root 60 Oct 19 03:51 ..
srwxr-xr-x 1 167 167 0 Oct 19 03:52 ceph-
client.rgw.realm.zone.serverc.agsgpq.6.93951066330432.asok
srwxr-xr-x 1 167 167 0 Oct 19 03:51 ceph-
mgr.serverc.lab.example.com.aiqepd.asok
srwxr-xr-x 1 167 167 0 Oct 19 03:51 ceph-mon.serverc.lab.example.com.asok
srwxr-xr-x 1 167 167 0 Oct 19 03:51 ceph-osd.0.asok
srwxr-xr-x 1 167 167 0 Oct 19 03:51 ceph-osd.1.asok
srwxr-xr-x 1 167 167 0 Oct 19 03:51 ceph-osd.2.asok
```

The following example mounts a CephFS file system with the FUSE client, gets the performance counters, and sets the `debug_ms` configuration parameter to 1:

```
[root@host ~]# ceph-fuse -n client.admin /mnt/mountpoint
2021-10-19T09:23:57.914-0400 7f1e7b914200 -1 init, newargv = 0x55d703b17a00
newargc=15
ceph-fuse[54240]: starting ceph client
ceph-fuse[54240]: starting fuse
[root@host ~]# ls /var/run/ceph/
2ae6d05a-229a-11ec-925e-52540000fa0c  ceph-client.admin.54240.94381967377904.asok
[root@host ~]# ceph --admin-daemon \
/var/run/ceph/ceph-client.admin.54240.94381967377904.asok perf dump
{
    "AsyncMessenger::Worker-0": {
        "msgr_recv_messages": 4,
        "msgr_send_messages": 3,
        "msgr_recv_bytes": 10112,
        "msgr_send_bytes": 480,
        "msgr_created_connections": 2,
        "msgr_active_connections": 1,
        "msgr_running_total_time": 0.002775454,
        "msgr_running_send_time": 0.001042138,
        "msgr_running_recv_time": 0.000868150,
        ...output omitted...
    }
}
[root@host ~]# ceph --admin-daemon \
/var/run/ceph/ceph-client.admin.54240.94381967377904.asok config show
...output omitted...
    "debug_ms": "0/0",
...output omitted...
[root@host ~]# ceph --admin-daemon \
/var/run/ceph/ceph-client.admin.54240.94381967377904.asok config set debug_ms 5
{
    "success": ""
}
[root@host ~]# ceph --admin-daemon \
/var/run/ceph/ceph-client.admin.54240.94381967377904.asok config show
```

```
...output omitted...
"debug_ms": "5/5",
...output omitted...
```

Comparing Ceph Versions and Features

Earlier versions of Ceph clients might not benefit from features provided by the installed version of the Ceph cluster. For example, an earlier client might fail to retrieve data from an erasure-coded pool. Therefore, when upgrading a Ceph cluster, you should also update the clients. The RADOS Gateway, the FUSE client for CephFS, the librbd, or the command-line tools, such as RADOS or RBD, are examples of Ceph clients.

From a client, you can find the version of the running Ceph cluster with the `ceph versions` command:

```
[ceph: root@node /]# ceph versions
{
    "mon": {
        "ceph version 16.2.0-117.el8cp (0e34bb74700060ebfaa22d99b7d2cdc037b28a57)
pacific (stable)": 4
    },
    "mgr": {
        "ceph version 16.2.0-117.el8cp (0e34bb74700060ebfaa22d99b7d2cdc037b28a57)
pacific (stable)": 4
    },
    "osd": {
        "ceph version 16.2.0-117.el8cp (0e34bb74700060ebfaa22d99b7d2cdc037b28a57)
pacific (stable)": 9
    },
    "mds": {
        "ceph version 16.2.0-117.el8cp (0e34bb74700060ebfaa22d99b7d2cdc037b28a57)
pacific (stable)": 1
    },
    "rgw": {
        "ceph version 16.2.0-117.el8cp (0e34bb74700060ebfaa22d99b7d2cdc037b28a57)
pacific (stable)": 2
    },
    "overall": {
        "ceph version 16.2.0-117.el8cp (0e34bb74700060ebfaa22d99b7d2cdc037b28a57)
pacific (stable)": 20
    }
}
```

You can also list the supported level of features with the `ceph features` command.

If you cannot upgrade the clients, the `ceph osd set-require-min-compat-client version-name` command specifies the minimum client version that the Ceph cluster must support.

Using this minimum client setting, Ceph denies the use of features that are not compatible with the current client version. Historically, the main exception has been changes to CRUSH. For example, if you run the `ceph osd set-require-min-compat-client jewel` command, then you cannot use the `ceph osd pg-upmap` command. This fails because "Jewel" version

clients do not support the PG upmap feature. Verify the minimum version required by your cluster with the `ceph osd` command:

```
[ceph: root@node /]# ceph osd get-require-min-compat-client  
luminous
```

Working with Cephx

Red Hat Ceph Storage provides the Cephx protocol for cryptographic authentication. If Cephx is enabled, then Ceph looks for the key ring in the default `/etc/ceph/` path.

Either enable Cephx for all components, or disable it completely. Ceph does not support a mixed setting, such as enabling Cephx for clients but disabling it for communication between the Ceph services. By default, Cephx is enabled and a client trying to access the Ceph cluster without Cephx receives an error message.



Important

Red Hat recommends using authentication in your production environment.

All Ceph commands authenticate as the `client.admin` user by default, although you can specify the user name or the user ID by using the `--name` and `--id` options.

Problems with Cephx are usually related to:

- Incorrect permissions on the key ring or `ceph.conf` files.
- Missing key ring and `ceph.conf` files.
- Incorrect or invalid `cephx` permissions for a given user. Use the `ceph auth list` command to identify the issue.
- Incorrect or misspelled user names, which you can also verify by using the `ceph auth list` command.

Troubleshooting Ceph Monitors

You can identify error messages by the `ceph health detail` command, or by reviewing the information provided by the Ceph logs.

The following is a list of the most common Ceph MON error messages:

mon.X is down (out of quorum)

If the Ceph MON daemon is not running, then an error is preventing the daemon from starting. For example, it is possible that the daemon has a corrupted store, or the `/var` partition might be full.

If the Ceph MON daemon is running but it is reported as down, then the cause depends on the MON state. If the Ceph MON is in the probing state longer than expected, then it cannot find the other Ceph Monitors. This problem can be caused by networking issues, or the Ceph Monitor can have an outdated Ceph Monitor map (`monmap`) is trying to reach the other Ceph Monitors on incorrect IP addresses.

If the Ceph MON is in the `electing` state longer than expected, then its clock might not be synchronized. If the state changes from `synchronizing` to `electing`, then it means that the Ceph MON is generating maps faster than the synchronization process can handle. If the state is either `leader` or `peon`, then the Ceph Mon has reached a quorum, but the rest

of the cluster does not recognize a quorum. This problem is mainly caused by a failed clock synchronization, an improperly working network, or the NTP synchronization is not correct.

clock skew

This error message indicates that the clocks for the MON might not be synchronized. The `mon_clock_drift_allowed` parameter controls the maximum difference between clocks that your cluster allows before showing the warning message. This problem is mainly caused by a failed clock synchronization, an improperly working network, or the NTP synchronization is not correct.

mon.X store is getting too big!

Ceph MON shows this warning message when the store is too big and it delays the response to client queries.

Troubleshooting Ceph OSDs

Use the `ceph status` command to review your monitor's quorum. If the cluster shows a health status, then your cluster can form a quorum. If you do not have a monitor quorum, or if there are errors with the monitor status, address the monitor issues first, and then proceed to verify the network.

The following is a list of the most common Ceph OSD error messages:

full osds

Ceph returns the `HEALTH_ERR full osds` message when the cluster reaches the capacity set by the `mon_osd_full_ratio` parameter. By default, this parameter is set to 0.95 which means 95% of the cluster capacity.

Use the `ceph df` command to determine the percentage of used raw storage, given by the `%RAW USED` column. If the percentage of raw storage is above 70%, then you can delete unnecessary data or scale the cluster by adding new OSD nodes to reduce it.

nearfull osds

Ceph returns the `nearfull osds` message when the cluster reaches the capacity set by the `mon_osd_nearfull_ratio` default parameter. By default, this parameter is set to 0.85 which means 85% of the cluster capacity.

The main causes for this warning message are:

- The OSDs are not balanced among the OSD nodes in the cluster.
- The placement group count is not correct based on number of OSDs, use case, target PGs per OSD, and OSD utilization.
- The cluster uses disproportionate CRUSH tunables.
- The back-end storage for OSDs is almost full.

To troubleshoot this issue:

- Verify that the PG count is sufficient.
- Verify that you use CRUSH tunables optimal to the cluster version and adjust them if not.
- Change the weight of OSDs by utilization.
- Determine how much space is left on the disks used by OSDs.

osds are down

Ceph returns the `osds are down` message when OSDs are down or flapping. The main cause for this message is that one of the `ceph-osd` processes is unavailable due to a possible failure, or problems networking with other OSDs.

Troubleshooting the RADOS Gateway

You can troubleshoot the Ceph RESTful interface and some common RADOS Gateway issues.

Debugging the Ceph RESTful Interface

The radosgw daemon is a Ceph client that sits between the Ceph cluster and HTTP clients. It includes its own web server, Beast, which supports HTTP and HTTPS.

In case of errors, you should consult the log file in the `/var/log/ceph/` folder.

To log to a file, set the `log_to_file` parameter to `true`. You can update the location of the log file and the log level by using the `log_file` and `debug` parameters, respectively. You can also enable the `rgw_enable_ops_log` and `rgw_enable_usage_log` parameters in the Ceph configuration database to log each successful RADOS Gateway operation and the usage, respectively.

```
[ceph: root@node /]# ceph config set client.rgw \
log_file /var/log/ceph/ceph-rgw-node.log
[ceph: root@node /]# ceph config set client.rgw log_to_file true
[ceph: root@node /]# ceph config set client.rgw debug_rgw 20
[ceph: root@node /]# ceph config set client.rgw rgw_enable_ops_log true
[ceph: root@node /]# ceph config set global rgw_enable_usage_log true
```

Verify the debugging logs using the `radosgw-admin log list` command. This command provides a list of the log objects that are available. View log file information using the `radosgw-admin log show` command. To retrieve the information directly from the log object, add the `--object` parameter with the object ID. To retrieve the information on the bucket at the timestamp, add the `--bucket`, `--date`, and `--bucket-id` parameters, which refer to the bucket name, the timestamp, and the bucket ID.

Common RADOS Gateway Issues

The most common error in RADOS Gateway is time skew between the client and the RADOS Gateway because the S3 protocol uses date and time for signing each request. To avoid this problem, use NTP on both Ceph and client nodes.

You can verify issues on RADOS Gateway request completion by looking for HTTP status lines in the RADOS Gateway log file.

The RADOS Gateway is a Ceph client that stores all of its configuration in RADOS objects. The RADOS PGs holding this configuration data must be in the `active+clean` state. If the state is not `active+clean`, then Ceph I/O requests will hang if the primary OSD becomes unable to serve data, and HTTP clients will eventually time out. Identify the inactive PGs with the `ceph health detail` command.

Troubleshooting CephFS

A CephFS Metadata Server (MDS) maintains a cache shared with its clients, FUSE, or the kernel so that an MDS can delegate part of its cache to clients. For example, a client accessing an inode can locally manage and cache changes to that object. If another client also requests access to the same inode, the MDS can request that the first client update the server with the new metadata.

To maintain cache consistency, an MDS requires a reliable network connection with its clients. Ceph can automatically disconnect, or evict, unresponsive clients. When this occurs, unflushed client data is lost.

When a client tries to gain access to CephFS, the MDS requests the client that has the current capabilities to release them. If the client is unresponsive, then CephFS shows an error message

after a timeout. You can configure the timeout by using the `session_timeout` attribute with the `ceph fs set` command. The default value is 60 seconds.

The `session_autoclose` attribute controls eviction. If a client fails to communicate with the MDS for more than the default 300 seconds, then the MDS evicts it.

Ceph temporarily bans evicted clients so that they cannot reconnect. If this ban occurs, you must reboot the client system or unmount and remount the file system to reconnect.



References

For more information, refer to the Configuring Logging chapter in the *Troubleshooting Guide for Red Hat Ceph Storage* at
https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/troubleshooting_guide/index#configuring-logging

For more information, refer to the *Troubleshooting Guide of the Red Hat Customer Portal Ceph Storage Guide* at
https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/5/html-single/troubleshooting_guide/index

► Guided Exercise

Troubleshooting Clusters and Clients

In this exercise, you will configure tuning parameters and diagnose common problems for various Red Hat Ceph Storage services.

Outcomes

You should be able to identify the error code for each Ceph component and resolve the issues.

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 tuning-troubleshoot
```

Instructions

- ▶ 1. Log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell. Verify the health of the Ceph storage cluster.

Two separate issues need troubleshooting. The first issue is a clock skew error, and the second issue is a down OSD which is degrading the PGs.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]# ceph health detail
HEALTH_WARN clock skew detected on mon.serverd; 1 osds down; Degraded data
redundancy: 63/567 objects degraded (11.111%), 14 pgs degraded, 10 pgs
undersized; 278 slow ops, oldest one blocked for 170 sec, mon.serverd has slow
ops
[WRN] MON_CLOCK_SKEW: clock skew detected on mon.serverd
    mon.serverd clock skew 299.103s > max 0.05s (latency 0.0204872s)
[WRN] OSD_DOWN: 1 osds down
    osd.0 (root=default,host=serverc) is down
[WRN] PG_DEGRADED: Degraded data redundancy: 63/567 objects degraded (11.111%), 14
pgs degraded, 10 pgs undersized
    pg 2.5 is stuck undersized for 2m, current state active+undersized, last
    acting [8,7]
    pg 2.c is stuck undersized for 2m, current state active+undersized, last
    acting [6,5]
...output omitted...
```



Note

The lab uses `chronyd` for time synchronization with the classroom server.

- 2. First, troubleshoot the clock skew issue.

Log in to `serverd` as the `admin` user. The previous health detail output stated that the time on the `serverd` system is 300 seconds different than on the other servers. Viewing the `chronyd` service status on the `serverd` system should identify the problem.

2.1. Exit the `cephadm` shell. On the `serverd` system, view the `chronyd` service status.

The `chronyd` service is inactive on the `serverd` system.

```
[ceph: root@clienta /]# exit
[admin@clienta ~]$ ssh admin@serverd
admin@serverd's password: redhat
[admin@serverd ~]$ systemctl status chronyd
● chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor
  preset: enabled)
    Active: inactive (dead) since Wed 2021-10-20 08:49:21 EDT; 13min ago
      Docs: man:chronyd(8)
             man:chrony.conf(5)
  Main PID: 876 (code=exited, status=0/SUCCESS)
```

2.2. Start the `chronyd` service.

```
[admin@serverd ~]$ sudo systemctl start chronyd
```

2.3. Verify that the `chronyd` service is active.

```
[admin@serverd ~]$ systemctl status chronyd
● chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor
  preset: enabled)
    Active: active (running) since Wed 2021-10-20 09:04:01 EDT; 3min 10s left
      Docs: man:chronyd(8)
             man:chrony.conf(5)
  Process: 15221 ExecStartPost=/usr/libexec/chrony-helper update-daemon
  (code=exited, status=0/SUCCESS)
  Process: 15218 ExecStart=/usr/sbin/chronyd $OPTIONS (code=exited, status=0/
  SUCCESS)
  Main PID: 15220 (chronyd)
    Tasks: 1 (limit: 36236)
   Memory: 360.0K
      CGroup: /system.slice/chronyd.service
              └─15220 /usr/sbin/chronyd
```

2.4. Return to the `clienta` system and use `sudo` to run the `cephadm` shell.

```
[admin@serverd ~]$ exit
Connection to serverd closed.
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]#
```

2.5. Verify the health of the storage cluster.

The time skew message might still display if the monitoring service has not yet updated the time. Allow the cluster sufficient time for services to obtain the corrected time. Continue with these exercise steps, but verify that the skew issue is resolved before finishing the exercise.

```
[ceph: root@clienta /]# ceph health detail
HEALTH_WARN Degraded data redundancy: 40/567 objects degraded (7.055%), 8 pgs
degraded, 20 pgs undersized; 1099 slow ops, oldest one blocked for 580 sec,
mon.serverd has slow ops
[WRN] PG_DEGRADED: Degraded data redundancy: 40/567 objects degraded (7.055%), 8
pgs degraded, 20 pgs undersized
    pg 2.0 is stuck undersized for 8m, current state active+undersized, last
    acting [3,6]
    pg 2.8 is stuck undersized for 8m, current state active+undersized, last
    acting [3,8]
...output omitted...
```



Note

The health detail output might show the cluster state as HEALTH_OK. When an OSD is down, the cluster migrates its PGs to other OSDs to return the cluster to a healthy state. However, the down OSD still requires troubleshooting.

▶ 3. Locate the down OSD.

The `osd.0` service on the `serverc` system is reported as down.

```
[ceph: root@clienta /]# ceph osd tree
ID CLASS WEIGHT TYPE NAME STATUS REWEIGHT PRI-AFF
-1          0.08817  root default
-3          0.02939  host serverc
  0  hdd  0.00980      osd.0    down      0  1.00000
  1  hdd  0.00980      osd.1    up   1.00000  1.00000
  2  hdd  0.00980      osd.2    up   1.00000  1.00000
-5          0.02939  host serverd
  3  hdd  0.00980      osd.3    up   1.00000  1.00000
  5  hdd  0.00980      osd.5    up   1.00000  1.00000
  7  hdd  0.00980      osd.7    up   1.00000  1.00000
-7          0.02939  host servere
  4  hdd  0.00980      osd.4    up   1.00000  1.00000
  6  hdd  0.00980      osd.6    up   1.00000  1.00000
  8  hdd  0.00980      osd.8    up   1.00000  1.00000
```

3.1. Exit the `cephadm` shell. On the `serverc` system, list the Ceph service units.

```
[ceph: root@clienta /]# exit
exit
[admin@clienta ~]$ ssh admin@serverc
admin@serverc's password: redhat
[admin@serverc ~]$ systemctl list-units --all 'ceph*'
UNIT
LOAD ACTIVE SUB     DESCRIPTION
...output omitted...
```

```
ceph-2ae6...fa0c@node-exporter.serverc.service           loaded active
  running Ceph node-exporter.serverc for 2ae6...fa0c
ceph-2ae6...fa0c@osd.0.service                         loaded inactive dead
  Ceph osd.0 for 2ae6...fa0c
ceph-2ae6...fa0c@osd.1.service                         loaded active
  running Ceph osd.1 for 2ae6...fa0c
ceph-2ae6...fa0c@osd.2.service                         loaded active
  running Ceph osd.2 for 2ae6...fa0c
ceph-2ae6...fa0c@prometheus.serverc.service          loaded active
  running Ceph prometheus.serverc for 2ae6...fa0c
...output omitted...
```

- 3.2. Restart the OSD 0 service. The fsid service and the OSD 0 service name are different in your lab environment.

```
[admin@serverc ~]$ sudo systemctl start ceph-2ae6...fa0c@osd.0.service
```

- 3.3. Verify the status of the OSD 0 service.

```
[admin@serverc ~]$ systemctl status ceph-2ae6...fa0c@osd.0.service
● ceph-2ae6...fa0c@osd.0.service - Ceph osd.0 for 2ae6...fa0c
   Loaded: loaded (/etc/systemd/system/ceph-2ae6...fa0c@.service; enabled; vendor
   preset: disabled)
     Active: active (running) since Wed 2021-10-20 09:16:45 EDT; 1min 7s ago
       Process: 14368 ExecStopPost=/bin/rm -f //run/ceph-2ae6...fa0c@osd.0.service-
   pid //run/ceph-2ae6...fa0c@osd.0.service-cid (code=exited, st>
       Process: 14175 ExecStopPost=/bin/bash /var/lib/ceph/2ae6...fa0c/osd.0/
   unit.poststop (code=exited, status=0/SUCCESS)
...output omitted...
```

- 3.4. Return to the client system and use sudo to run the cephadm shell.

```
[admin@serverc ~]$ exit
Connection to serverc closed.
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]#
```

- 3.5. Verify the OSD health. The OSD is now up.

```
[ceph: root@clienta /]# ceph osd tree
ID CLASS WEIGHT  TYPE NAME      STATUS REWEIGHT PRI-AFF
-1          0.08817  root default
-3          0.02939  host serverc
  0  hdd  0.00980    osd.0      up   1.00000  1.00000
  1  hdd  0.00980    osd.1      up   1.00000  1.00000
  2  hdd  0.00980    osd.2      up   1.00000  1.00000
-5          0.02939  host serverd
  3  hdd  0.00980    osd.3      up   1.00000  1.00000
  5  hdd  0.00980    osd.5      up   1.00000  1.00000
  7  hdd  0.00980    osd.7      up   1.00000  1.00000
-7          0.02939  host servere
```

```
4    hdd  0.00980          osd.4      up   1.00000  1.00000
6    hdd  0.00980          osd.6      up   1.00000  1.00000
8    hdd  0.00980          osd.8      up   1.00000  1.00000
```

- 3.6. Verify the health of the storage cluster. If the status is HEALTH_WARN and you have resolved the time skew and OSD issues, then wait until the cluster status is HEALTH_OK before continuing.

```
[ceph: root@clienta /]# ceph health
HEALTH_OK
```

- 4. Return to workstation as the student user.

```
[ceph: root@clienta /]# exit
[admin@clienta ~]$ exit
[student@workstation ~]$
```

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 tuning-troubleshoot
```

This concludes the guided exercise.

► Lab

Tuning and Troubleshooting Red Hat Ceph Storage

In this lab, you will tune and troubleshoot Red Hat Ceph Storage.

Outcomes

You should be able to:

- Troubleshoot Ceph clients and Ceph performance issues.
- Modify the logging options for Ceph clients.
- Tune the recovery and backfill processes to preserve cluster performance.

Before You Begin

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

This command ensures that the lab environment is available for the exercise.

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

Instructions

1. Log in to `clienta` as the `admin` user. Verify the health of the Ceph storage cluster. Hypothesize possible causes for the displayed issues.
2. First, troubleshoot the clock skew issue.
3. Troubleshoot the down OSD issue. Use diagnostic logging to find and correct a non-working configuration.
4. For the OSD 5 service, set the operations history size to track 40 completed operations and the operations history duration to 700 seconds.
5. For all OSDs, modify the current runtime value for the maximum concurrent backfills to 3 and for the maximum active recovery operations to 1.
6. Return to `workstation` as the student user.

Evaluation

Grade your work by running the `lab grade tuning-review` command from your `workstation` machine. Correct any reported failures and rerun the script until successful.

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

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 tuning-review
```

This concludes the lab.

► Solution

Tuning and Troubleshooting Red Hat Ceph Storage

In this lab, you will tune and troubleshoot Red Hat Ceph Storage.

Outcomes

You should be able to:

- Troubleshoot Ceph clients and Ceph performance issues.
- Modify the logging options for Ceph clients.
- Tune the recovery and backfill processes to preserve cluster performance.

Before You Begin

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

This command ensures that the lab environment is available for the exercise.

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

Instructions

1. Log in to `clienta` as the `admin` user. Verify the health of the Ceph storage cluster. Hypothesize possible causes for the displayed issues.

- 1.1. Log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]#
```

- 1.2. Verify the status of the storage cluster.

```
[ceph: root@clienta /]# ceph status
cluster:
  id: 2ae6d05a-229a-11ec-925e-52540000fa0c
  health: HEALTH_WARN
    1 failed cephadm daemon(s)
    clock skew detected on mon.serverd
    1 osds down
    Reduced data availability: 4 pgs inactive, 19 pgs peering
    Degraded data redundancy: 35/567 objects degraded (6.173%), 7 pgs
    degraded

  services:
```

```
mon: 4 daemons, quorum serverc.lab.example.com,clienta,serverd,servere (age 33s)
mgr: serverc.lab.example.com.aiqepd(active, since 9m), standbys: clienta.nncugs, servere.kjwyko, serverd.klrkci
osd: 9 osds: 8 up (since 32s), 9 in (since 8m)
rgw: 2 daemons active (2 hosts, 1 zones)

data:
pools: 5 pools, 105 pgs
objects: 189 objects, 4.9 KiB
usage: 105 MiB used, 90 GiB / 90 GiB avail
pgs: 18.095% pgs not active
      35/567 objects degraded (6.173%)
      68 active+clean
      19 peering
      11 active+undersized
      7 active+undersized+degraded
```

1.3. Verify the health details of the storage cluster.

Two separate issues need troubleshooting. The first issue is a clock skew error, and the second issue is a down OSD that is degrading the PGs.

```
[ceph: root@clienta /]# ceph health detail
HEALTH_WARN 1 failed cephadm daemon(s); clock skew detected on mon.serverd; 1
osds down; Degraded data redundancy: 63/567 objects degraded (11.111%), 15 pgs
degraded; 44 slow ops, oldest one blocked for 79 sec, mon.serverd has slow ops
[WRN] CEPHADM_FAILED_DAEMON: 1 failed cephadm daemon(s)
      daemon osd.4 on servere.lab.example.com is in unknown state
[WRN] MON_CLOCK_SKEW: clock skew detected on mon.serverd
      mon.serverd clock skew 299.62s > max 0.05s (latency 0.0155988s)
[WRN] OSD_DOWN: 1 osds down
      osd.4 (root=default,host=servere) is down
[WRN] PG_DEGRADED: Degraded data redundancy: 63/567 objects degraded (11.111%), 15
pgs degraded
      pg 3.1 is active+undersized+degraded, acting [0,7]
...output omitted...
      pg 4.15 is active+undersized+degraded, acting [0,7]
```

2. First, troubleshoot the clock skew issue.

2.1. Open a second terminal and log in to `serverd` as the `admin` user. The previous health detail output stated that the time on the `serverd` system is 300 seconds different from the other servers. View the `chronyd` service status on the `serverd` system to identify the problem.

The `chronyd` service is inactive on the `serverd` system.

```
[student@workstation ~]$ ssh admin@serverd
[admin@serverd ~]$ systemctl status chronyd
● chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor
  preset: enabled)
    Active: inactive (dead) since Mon 2021-10-25 03:51:44 EDT; 8min ago
      Docs: man:chronyd(8)
             man:chrony.conf(5)
  Main PID: 867 (code=exited, status=0/SUCCESS)
```

2.2. Restart the chronyd service.

```
[admin@serverd ~]$ sudo systemctl start chronyd
```

2.3. Verify that the chronyd service is active.

```
[admin@serverd ~]$ systemctl status chronyd
● chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor
  preset: enabled)
    Active: active (running) since Mon 2021-10-25 04:00:54 EDT; 4min 53s left
      Docs: man:chronyd(8)
             man:chrony.conf(5)
  Process: 14604 ExecStartPost=/usr/libexec/chrony-helper update-daemon
  (code=exited, status=0/SUCCESS)
  Process: 14601 ExecStart=/usr/sbin/chronyd $OPTIONS (code=exited, status=0/
  SUCCESS)
  Main PID: 14603 (chronyd)
    Tasks: 1 (limit: 36236)
   Memory: 360.0K
      CGroup: /system.slice/chronyd.service
              └─14603 /usr/sbin/chronyd
```

2.4. Return to workstation as the student user and close the second terminal.

```
[admin@serverd ~]$ exit
Connection to serverd closed.
[student@workstation ~]$ exit
```

2.5. In the first terminal, verify the health of the storage cluster

The time skew message might still display if the monitoring service has not yet updated the time. Allow the cluster sufficient time for services to obtain the corrected time. Continue with these exercise steps, but verify that the skew issue is resolved before finishing the exercise.

```
[ceph: root@clienta /]# ceph health detail
HEALTH_WARN 1 failed cephadm daemon(s); 1 osds down; Degraded data redundancy:
 63/567 objects degraded (11.11%), 15 pgs degraded, 16 pgs undersized; 45 slow
 ops, oldest one blocked for 215 sec, mon.serverd has slow ops
[WRN] CEPHADM_FAILED_DAEMON: 1 failed cephadm daemon(s)
  daemon osd.4 on servere.lab.example.com is in error state
```

```
[WRN] OSD_DOWN: 1 osds down
    osd.4 (root=default,host=servere) is down
[WRN] PG_DEGRADED: Degraded data redundancy: 63/567 objects degraded (11.111%), 15
    pgs degraded, 16 pgs undersized
        pg 2.4 is stuck undersized for 4m, current state active+undersized, last
        acting [1,3]
...output omitted...
        pg 5.6 is stuck undersized for 4m, current state active+undersized, last
        acting [2,5]
```

3. Troubleshoot the down OSD issue. Use diagnostic logging to find and correct a non-working configuration.

3.1. Locate the down OSD.

```
[ceph: root@clienta /]# ceph osd tree
ID  CLASS  WEIGHT  TYPE NAME      STATUS  REWEIGHT PRI-AFF
-1          0.08817  root default
-3          0.02939  host serverc
  0  hdd   0.00980      osd.0     up    1.00000  1.00000
  1  hdd   0.00980      osd.1     up    1.00000  1.00000
  2  hdd   0.00980      osd.2     up    1.00000  1.00000
-5          0.02939  host serverd
  3  hdd   0.00980      osd.3     up    1.00000  1.00000
  5  hdd   0.00980      osd.5     up    1.00000  1.00000
  7  hdd   0.00980      osd.7     up    1.00000  1.00000
-7          0.02939  host servere
  4  hdd   0.00980      osd.4     down   1.00000  1.00000
  6  hdd   0.00980      osd.6     up    1.00000  1.00000
  8  hdd   0.00980      osd.8     up    1.00000  1.00000
```

3.2. Attempt to restart the OSD 4 service with the `ceph orch` command.

OSD 4 remains down after waiting a sufficient time.

```
[ceph: root@clienta /]# ceph orch daemon restart osd.4
Scheduled to restart osd.4 on host 'servere.lab.example.com'
```

3.3. Attempt to retrieve the configuration from the OSD 4 service.

An error message indicates a communication problem with the OSD 4 service.

```
[ceph: root@clienta /]# ceph tell osd.4 config show
Error ENXIO: problem getting command descriptions from osd.4
```

3.4. Open a second terminal and log in to `servere` as the `admin` user. On the `servere` system, list the Ceph units.

```
[student@workstation ~]$ ssh admin@servere
[admin@servere ~]$ systemctl list-units --all 'ceph*'
UNIT                                     LOAD
ACTIVE SUB      DESCRIPTION
ceph-2ae6...fa0c@crash.servere.service      loaded active running Ceph
crash.servere for 2ae6...fa0c
```

```

ceph-2ae6...fa0c@mgr.servere.kjwyko.service    loaded active running Ceph
mgr.servere.kjwyko for 2ae6...fa0c
ceph-2ae6...fa0c@mon.servere.service           loaded active running Ceph
mon.servere for 2ae6...fa0c
ceph-2ae6...fa0c@node-exporter.servere.service loaded active running Ceph node-
exporter.servere for 2ae6...fa0c
● ceph-2ae6...fa0c@osd.4.service              loaded failed failed Ceph osd.4
for 2ae6...fa0c
ceph-2ae6...fa0c@osd.6.service                loaded active running Ceph osd.6
for 2ae6...fa0c
ceph-2ae6...fa0c@osd.8.service                loaded active running Ceph osd.8
for 2ae6...fa0c
...output omitted...

```

The OSD 4 services might not yet list as **failed** if the orchestrator is still attempting to restart the service. Wait until the service lists as **failed** before continuing this exercise.

- 3.5. Restart the OSD 4 service. The **fsid** service and the OSD 0 service name are different in your lab environment.

The OSD 4 service still fails to start.

```

[admin@servere ~]$ sudo systemctl restart \
ceph-2ae6d05a-229a-11ec-925e-52540000fa0c@osd.4.service
Job for ceph-2ae6...fa0c@osd.4.service failed because the control process exited
with error code.
See "systemctl status ceph-2ae6...fa0c@osd.4.service" and "journalctl -xe" for
details.

```

- 3.6. In the first terminal, modify the OSD 4 logging configuration to write to the `/var/log/ceph/myosd4.log` file and increase the logging level for OSD 4. Attempt to restart the OSD 4 service with the `ceph orch` command.

```

[ceph: root@clienta ~]# ceph config set osd.4 log_file /var/log/ceph/myosd4.log
[ceph: root@clienta ~]# ceph config set osd.4 log_to_file true
[ceph: root@clienta ~]# ceph config set osd.4 debug_ms 1
[ceph: root@clienta /]# ceph orch daemon restart osd.4
Scheduled to restart osd.4 on host 'servere.lab.example.com'

```

- 3.7. In the second terminal window, view the `myosd4.log` file to discover the issue.

Error messages indicate an incorrect cluster network address configuration.

```
[admin@servere ~]$ sudo cat \
/var/log/ceph/2ae6d05a-229a-11ec-925e-52540000fa0c/myosd4.log
...output omitted...
2021-10-25T08:26:22.617+0000 7f86c1f9e080 1 bdev(0x561a77661000 /var/lib/ceph/
osd/ceph-4/block) close
2021-10-25T08:26:22.877+0000 7f86c1f9e080 1 bdev(0x561a77660c00 /var/lib/ceph/
osd/ceph-4/block) close
2021-10-25T08:26:23.125+0000 7f86c1f9e080 0 starting osd.4 osd_data /var/lib/
ceph/osd/ceph-4 /var/lib/ceph/osd/ceph-4/journal
2021-10-25T08:26:23.125+0000 7f86c1f9e080 -1 unable to find any IPv4 address in
networks '192.168.0.0/24' interfaces ''
2021-10-25T08:26:23.125+0000 7f86c1f9e080 -1 Failed to pick cluster address.
```

- 3.8. Return to workstation as the student user and close the second terminal.

```
[admin@servere ~]$ exit
[student@workstation ~]$ exit
```

- 3.9. In the first terminal, compare the cluster network addresses for the OSD 0 and OSD 4 services.

```
[ceph: root@clienta /]# ceph config get osd.0 cluster_network
172.25.249.0/24
[ceph: root@clienta /]# ceph config get osd.4 cluster_network
192.168.0.0/24
```

- 3.10. Modify the cluster network value for the OSD 4 service. Attempt to restart the OSD 4 service.

```
[ceph: root@clienta /]# ceph config set osd.4 cluster_network 172.25.249.0/24
[ceph: root@clienta /]# ceph orch daemon restart osd.4
```

- 3.11. Verify that the OSD 4 service is now up. Verify the health of the storage cluster.

```
[ceph: root@clienta /]# ceph osd tree
ID CLASS WEIGHT TYPE NAME STATUS REWEIGHT PRI-AFF
-1          0.08817  root default
-3          0.02939  host serverc
 0  hdd  0.00980    osd.0      up   1.00000  1.00000
 1  hdd  0.00980    osd.1      up   1.00000  1.00000
 2  hdd  0.00980    osd.2      up   1.00000  1.00000
-7          0.02939  host serverd
 3  hdd  0.00980    osd.3      up   1.00000  1.00000
 5  hdd  0.00980    osd.5      up   1.00000  1.00000
 7  hdd  0.00980    osd.7      up   1.00000  1.00000
-5          0.02939  host servere
 4  hdd  0.00980    osd.4      up   1.00000  1.00000
 6  hdd  0.00980    osd.6      up   1.00000  1.00000
 8  hdd  0.00980    osd.8      up   1.00000  1.00000
[ceph: root@clienta /]# ceph health detail
HEALTH_OK
```

4. For the OSD 5 service, set the operations history size to track 40 completed operations and the operations history duration to 700 seconds.

- 4.1. Modify the `osd_op_history_size` and `osd_op_history_duration` parameters. Set the history size parameter to 40 and the history duration parameter to 700. Use the `ceph daemon` command to verify the changed values.

```
[ceph: root@clienta /]# ceph tell osd.5 config set osd_op_history_size 40
{
    "success": "osd_op_history_size = '40' "
}
[ceph: root@clienta /]# ceph tell osd.5 config set osd_op_history_duration 700
{
    "success": "osd_op_history_duration = '700' "
}
[ceph: root@clienta /]# ceph tell osd.5 dump_historic_ops | head -n 3
{
    "size": 40,
    "duration": 700,
```

5. For all OSDs, modify the current runtime value for the maximum concurrent backfills to 3 and for the maximum active recovery operations to 1.

- 5.1. Set the value of the `osd_max_backfills` parameter to 3.

```
[ceph: root@clienta /]# ceph tell osd.* config set osd_max_backfills 3
osd.0: {
    "success": "osd_max_backfills = '3' "
}
...output omitted...
osd.8: {
    "success": "osd_max_backfills = '3' "
}
```

- 5.2. Set the value of the `osd_recovery_max_active` parameter to 1.

```
[ceph: root@clienta /]# ceph tell osd.* config set osd_recovery_max_active 1
osd.0: {
    "success": "osd_recovery_max_active = '1' "
}
...output omitted...
osd.8: {
    "success": "osd_recovery_max_active = '1' "
}
```

6. Return to `workstation` as the `student` user.

```
[ceph: root@clienta /]# exit
[admin@clienta ~]$ exit
[student@workstation ~]$
```

Evaluation

Grade your work by running the `lab grade tuning-review` command from your workstation machine. Correct any reported failures and rerun the script until successful.

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

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 tuning-review
```

This concludes the lab.

Summary

In this chapter, you learned:

- Red Hat Ceph Storage 5 performance depends on the performance of the underlying storage, network, and operating system file system components.
- Performance is improved by reducing latency, increasing IOPS, and increasing throughput. Tuning for one metric often adversely affects the performance of another. Your primary tuning metric must consider the expected workload behavior of your storage cluster.
- Ceph implements a scale-out model architecture. Increasing the number of OSD nodes increases the overall performance. The greater the parallel access, the greater the load capacity.
- The RADOS and RBD bench commands are used to stress and benchmark a Ceph cluster.
- Controlling scrubbing, deep scrubbing, backfill, and recovery processes helps avoid cluster over-utilization.
- Troubleshooting Ceph issues starts with determining which Ceph component is causing the issue.
- Enabling logging for a failing Ceph subsystem provides diagnostic information about the issue.
- The log debug level is used to increase the logging verbosity.

Chapter 13

Managing Cloud Platforms with Red Hat Ceph Storage

Goal

Manage Red Hat cloud infrastructure to use Red Hat Ceph Storage to provide image, block, volume, object, and shared file storage.

Objectives

- Describe Red Hat OpenStack Platform storage requirements, and compare the architecture choices for using Red Hat Ceph Storage as an RHOSP storage back end.
- Describe how OpenStack implements Ceph storage for each storage-related OpenStack component.
- Describe Red Hat OpenShift Container Platform storage requirements, and compare the architecture choices for using Red Hat Ceph Storage as an RHOCOP storage back end.
- Describe how OpenShift implements Ceph storage for each storage-related OpenShift feature.

Sections

- Introducing OpenStack Storage Architecture (and Quiz)
- Implementing Storage in OpenStack Components (and Quiz)
- Introducing OpenShift Storage Architecture (and Quiz)
- Implementing Storage in OpenShift Components (and Quiz)

Introducing OpenStack Storage Architecture

Objectives

After completing this section, you should be able to describe Red Hat OpenStack Platform storage requirements, and compare the architecture choices for using Red Hat Ceph Storage as an RHOSP storage back end.

Red Hat OpenStack Platform Overview

Red Hat OpenStack Platform (RHOSP) is implemented as a collection of interacting services that control compute, storage, and networking resources. Cloud users deploy virtual machines by using resources through a self-service interface. Cloud operators, in cooperation with storage operators, ensure that sufficient storage space is created, configured, and available for each of the OpenStack components that consume or provide storage to cloud users.

Figure 13.1 presents a high-level overview of the core service relationships of a simple RHOSP installation. All services interact with the **Identity** service (Keystone) to authenticate users, services, and privileges before any operation is allowed. Cloud users can choose to use the command-line interface or the graphical **Dashboard** service to access existing resources and to create and deploy virtual machines.

The **Orchestration** service is the primary component for installing and modifying an RHOSP cloud. This section introduces the OpenStack services for integrating Ceph into an OpenStack infrastructure.

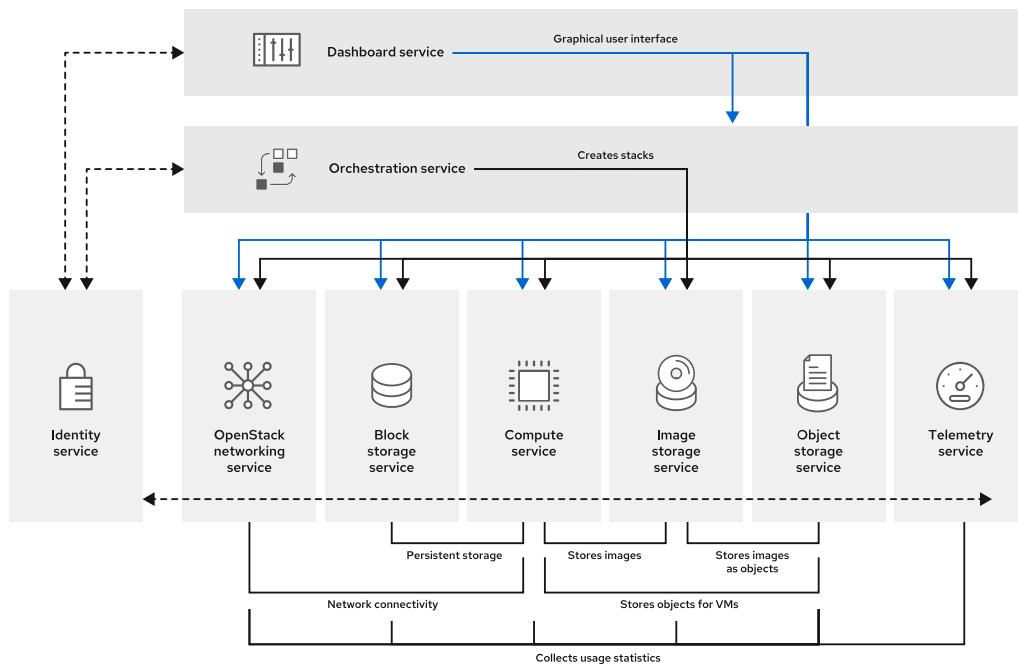


Figure 13.1: A simple set of OpenStack services

Introducing the Storage Services

These core OpenStack services provide storage resources in various formats and by multiple access methods. Cloud users deploy application VMs that consume these storage resources.

Compute Service (Nova)

The Compute service manages VM instances that run on hypervisor nodes. It uses storage to provide system disks, swap volumes, and other ephemeral disks for launching and running instances. This service interacts with the Identity service for authentication, the Image service to obtain images, and other storage services to access additional forms of persistent storage for running instances to use. The Compute service uses libvirt, qemu, and kvm for the hypervisor.

Block Storage Service (Cinder)

The Block Storage service manages storage volumes for virtual machines, including both ephemeral and persistent block storage for instances that the Compute service manages. The service implements snapshots for backing up and creating new block storage volumes.

Image Service (Glance)

The Image service acts as a registry for images that build instance system disks when they are launched. Live instances can be saved as images for later use to build new instances.

Shared File Systems Service (Manila)

The Shared File System service uses the network infrastructure to implement file sharing as a service. Because cloud users normally do not have connection privileges to the file share server, this service brokers connections to configured back ends. The service uses NFS and CIFS protocols to access file share servers. Administrators can configure this service to access multiple file share servers.

Object Store Service (Swift)

The Object Store provides storage for users to upload and retrieve objects as files. The Object Store architecture is distributed across disk devices and servers for horizontal scaling and to provide redundancy. It is common practice to configure the Image service to use the Object Store service as its storage back end, to support image and snapshot replication across the Object Store infrastructure. This service also provides a backup solution for other services by storing backup results as retrievable objects.

Red Hat Ceph Storage (Ceph)

Red Hat Ceph Storage is a distributed data object store that is used as the back end for all the other storage services. Ceph is the most common back end that is used with OpenStack. Ceph integrates with OpenStack services such as Compute, Block Storage, Shared File Systems, Image, and Object Store to provide easier storage management and cloud scalability.

Introducing Services for Integrating Storage

These additional core services provide overcloud installation, service container deployment, and the authentication support that are necessary to implement storage integration.

Identity Service (Keystone)

The Identity service authenticates and authorizes all OpenStack services. This service creates and manages users and roles in domains and projects. This service provides a central catalog of services and their associated endpoints that are available in an OpenStack cloud. The Identity service acts as a single sign-on (SSO) authentication service for both users and service components.

Deployment Service (TripleO)

The Deployment service installs, upgrades, and operates OpenStack clouds by using the Director node, which is an OpenStack cloud.

Orchestration Service (Heat)

The Orchestration service can provision both infrastructure and application workloads, by using resources that are defined in Heat orchestration Template (HOT) files. HOT template and environment files are the primary configuration method for deploying overclouds. In later Red Hat OpenStack Platform versions, the Orchestration template and environment files define the services, resources, and architecture to deploy, while Ansible Playbooks implement the software provisioning.

Container Deployment Service (Kolla)

In later RHOSP versions, the OpenStack services are containerized. The Container Deployment service provides production-ready containers and configuration management for operation of OpenStack services.

Bare Metal Service (Ironic)

The Bare Metal provisioning service prepares and provisions both physical hardware and KVM virtual machines. The service works with standard and vendor-specific drivers, such as PXE and IPMI, to communicate with a wide range of hardware.

Selecting a Ceph Integration Architecture

A storage operator, who works closely with infrastructure architects and network engineers, chooses the Ceph integration architecture and server node roles that are needed to support the organization's application use cases and sizing forecasts. Ceph can be integrated into an OpenStack infrastructure by using either of two implementation designs. Both Ceph designs are implemented by TripleO, which uses Ansible Playbooks for the bulk of the software deployment and configuration.

**Note**

RHOSP 16.1 and 16.2 support RHCS 5 only as an external cluster. RHOSP 17 supports dedicated RHCS 5 deployment with cephadm to replace ceph-ansible.

Dedicated

An organization without an existing, stand-alone Ceph cluster installs a dedicated Ceph cluster that is composed of Ceph services and storage nodes during an RHOSP overcloud installation. Only services and workloads that are deployed for, or on, an OpenStack overcloud can use an OpenStack-dedicated Ceph implementation. External applications cannot access or use OpenStack-dedicated Ceph cluster storage.

External

An organization can use an existing, stand-alone Ceph cluster for storage when creating a new OpenStack overcloud. The TripleO deployment is configured to access that external cluster to create the necessary pools, accounts, and other resources during overcloud installation. Instead of creating internal Ceph services, the deployment configures the OpenStack overcloud to access the existing Ceph cluster as a Ceph client.

A dedicated Ceph cluster supports a maximum of 750 OSDs when running the Ceph control plane services on the RHOSP controllers. An external Ceph cluster can scale significantly larger, depending on the hardware configuration. Updates and general maintenance are easier on an external cluster because they can occur independently of RHOSP operations.

To maintain Red Hat support, RHOSP installations must be built and configured with the TripleO Orchestration service. For a dedicated storage configuration, RHOSP 16 TripleO uses the same RHCS 4 ceph-ansible playbooks that are used to install stand-alone Ceph clusters. However, because TripleO dynamically organizes the playbooks and environment files to include in the deployment, direct use of Ansible without TripleO is not supported.

Node Roles Available in a Dedicated Ceph Implementation

A dedicated Ceph implementation is the TripleO default, and is sufficient for most small, medium, and moderately large OpenStack installations. A storage operator has significant choices for service distribution across overcloud nodes by using composable node roles. Except where stated otherwise, these node roles are included by default in later RHOSP versions.

Figure 13.2 presents an example of overcloud nodes to implement different service roles in a simple overcloud.

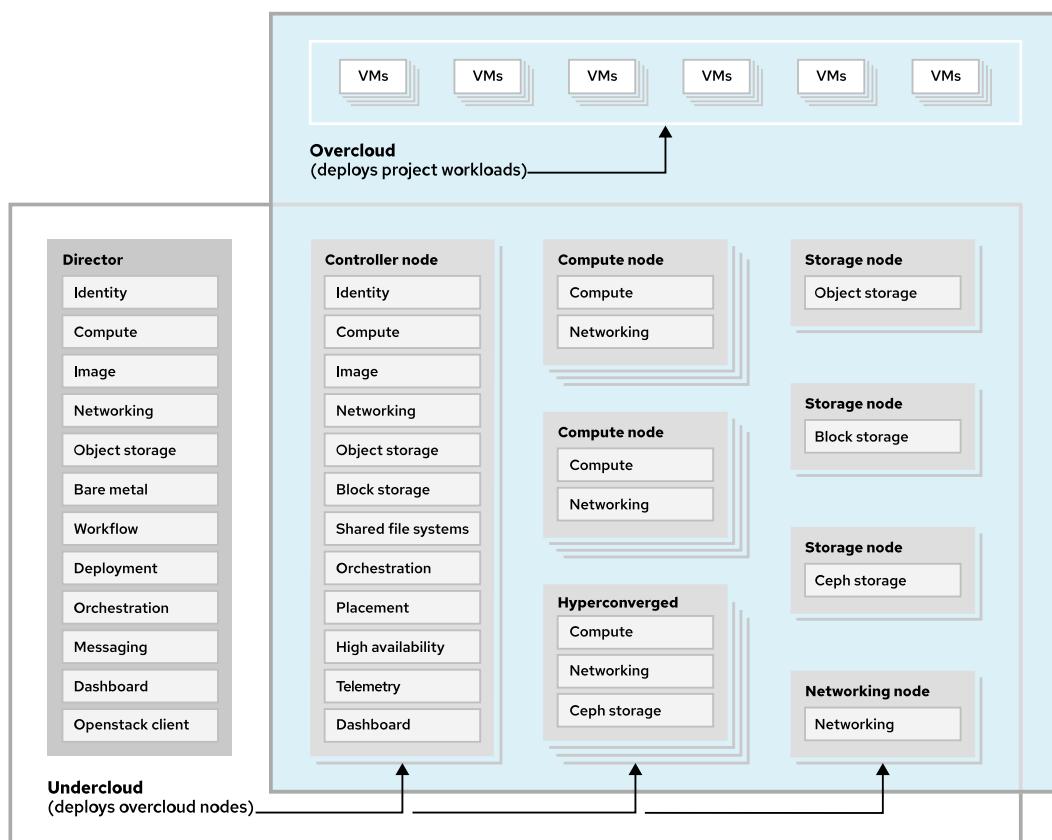


Figure 13.2: An example overcloud with multiple node roles

The following node roles determine the services that are placed on storage nodes that handle data plane traffic and on the physical storage devices.

The CephStorage role is the default, and control plane services are expected to be installed on controller nodes.

- **CephStorage** - The most common dedicated Ceph storage node configuration. Contains OSDs only, without control plane services.
- **CephAll** - A stand-alone full storage node with OSDs and all control plane services. This configuration might be used with the ControllerNoCeph node role.

- **CephFile** - A node to scale out file sharing. Contains OSDs and MDS services.
- **CephObject** - A node to scale out object gateway access. Contains OSDs and RGW services.

When storage management traffic increases, controller nodes can become overloaded. The following node roles support various configurations and distributions of Ceph control plane services across multiple nodes. Coordinate controller node roles with role choices for storage nodes to ensure that all wanted control plane services are deployed.

- **Controller** - The most common controller node configuration. Contains all normal control plane services, including Ceph MGR, MDS, MON, RBD, and RGW services.
- **ControllerStorageDashboard** - A normal controller node plus a Grafana dashboard service. This node role adds a further network to isolate storage monitoring traffic from the storage back end.
- **ControllerStorageNFS** - A normal controller node plus a Ganesha service as a CephFS to NFS gateway.
- **ControllerNoCeph** - A normal controller, but without Ceph control plane services. This node role is selected when Ceph control plane services are moved to segregated nodes for increased performance and scaling.

The following node roles are not included by default in the RHOSP distribution, but are described in Red Hat online documentation. Use these roles to alleviate overloaded controller nodes by moving primary Ceph services to separate, dedicated nodes. These roles are commonly found in larger OpenStack installations with increased storage traffic requirements.

- **CephMon** - A custom-created node role that moves only the MON service from the controllers to a separate node.
- **CephMDS** - A custom-created node role that moves only the MDS service from the controllers to a separate node.

A Hyperconverged Infrastructure (HCI) node is a configuration with both compute and storage services and devices on the same node. This configuration can result in increased performance for heavy storage throughput applications. The default is the **ComputeHCI** role, which adds only OSDs to a compute node, effectively enlarging your dedicated Ceph cluster. Ceph control plane services remain on the controller nodes. The other node roles add various choices of control plane services to the hyperconverged node.

- **ComputeHCI** - A compute node plus OSDs. These nodes have no Ceph control plane services.
- **HciCephAll** - A compute node plus OSDs and all Ceph control plane services.
- **HciCephFile** - A compute node plus OSDs and the MDS service. Used for scaling out file sharing storage capacity.
- **HciCephMon** - A compute node plus OSDs and the MON and MGR services. Used for scaling out block storage capacity.
- **HciCephObject** - A compute node plus OSDs and the RGW service. Used for scaling out object gateway access.

A Distributed Compute Node (DCN) is another form of hyperconverged node that is designed for use in remote data centers or branch offices that are part of the same OpenStack overcloud. For DCN, the overcloud deployment creates a dedicated Ceph cluster, with a minimum of three nodes, per remote site in addition to the dedicated Ceph cluster at the primary site. This architecture is not a stretch cluster configuration. Later DCN versions support installing the Glance in the remote location for faster local image access.

- **DistributedComputeHCI** - A DCN node with Ceph, Cinder, and Glance.
- **DistributedComputeHCIScaleOut** - A DCN node with Ceph, Cinder, and HAProxy for Glance.

Implementing an External Red Hat Ceph Storage Cluster

RHOSP overcloud installations have an undercloud node, which is referred to as the **Director** node in *Figure 13.1*. **Triple0** installs overcloud from the Director node. The default orchestration templates for **Triple0** services are in the `/usr/share/openstack-tripleo-heat-templates` directory on the undercloud. When deploying OpenStack integrated with Ceph, the undercloud node becomes the Ansible controller and cluster administration host.



Note

The following narrative provides a limited view of **Triple0** cloud deployment resources. Your organization's deployment will require further design effort, because every production overcloud has unique storage needs.

Because the default orchestration files are continuously being enhanced, you must not modify default template files in their original location. Instead, create a directory to store your custom environment files and parameter overrides. The following `ceph-ansible-external.yaml` environment file instructs **Triple0** to use the `ceph-ansible` client role to access a preexisting, external Ceph cluster. To override the default settings in this file, use a custom parameter file.

```
[stack@director ceph-ansible]$ cat ceph-ansible-external.yaml
resource_registry:
  OS::TripleO::Services::CephExternal: ../../deployment/ceph-ansible/ceph-external.yaml

parameter_defaults:
  # NOTE: These example parameters are required when using CephExternal
  #CephClusterFSID: '4b5c8c0a-ff60-454b-a1b4-9747aa737d19'
  #CephClientKey: 'AQDLOh1VgEp6FRAAFzT7Zw+Y9V6JJExQAsRnRQ=='
  #CephExternalMonHost: '172.16.1.7, 172.16.1.8'

  # the following parameters enable Ceph backends for Cinder, Glance, Gnocchi and Nova
  NovaEnableRbdBackend: true
  CinderEnableRbdBackend: true
  CinderBackupBackend: ceph
  GlanceBackend: rbd
  # Uncomment below if enabling legacy telemetry
  # GnocchiBackend: rbd
  # If the Ceph pools which host VMs, Volumes and Images do not match these
  # names OR the client keyring to use is not called 'openstack', edit the
  # following as needed.
  NovaRbdPoolName: vms
  CinderRbdPoolName: volumes
  CinderBackupRbdPoolName: backups
  GlanceRbdPoolName: images
  # Uncomment below if enabling legacy telemetry
  # GnocchiRbdPoolName: metrics
  CephClientUserName: openstack
```

```
# finally we disable the Cinder LVM backend
CinderEnableIscsiBackend: false
```

A **TripleO** deployment specifies a list of environment files for all of the overcloud services to be deployed, with an `openstack overcloud deploy` command. Before deployment, the `openstack tripleo container image prepare` command is used to determine all of the services that are referenced in the configuration, and prepare a list of the corrector containers to download and provide for the overcloud deployment. During the installation, **Kolla** is used to configure and start each service container on the correct nodes, as defined by the node roles.

For this external Ceph cluster example, **TripleO** needs a parameter file that specifies the real cluster parameters, to override the parameter defaults in the `ceph-ansible-external.yaml` file. This example `parameter-overrides.yaml` file is placed in your custom deployment files directory. You can obtain the key from the result of an appropriate `ceph auth add client.openstack` command.

```
parameter_defaults:
  # The cluster FSID
  CephClusterFSID: '4b5c8c0a-ff60-454b-a1b4-9747aa737d19'
  # The CephX user auth key
  CephClientKey: 'AQDL0h1VgEp6FRAAFzT7Zw+Y9V6JJExQAsRnRQ=='
  # The list of Ceph monitors
  CephExternalMonHost: '172.16.1.7, 172.16.1.8, 172.16.1.9'
```

TripleO relies on the **Bare Metal** service to prepare nodes before installing them as Ceph servers. Disk devices, both physical and virtual, must be cleaned of all partition tables and other artifacts. Otherwise, Ceph refuses to overwrite the device, after determining that the device is in use. To delete all metadata from disks, and create GPT labels, set the following parameter in the `/home/stack/undercloud.conf` file on the undercloud. The **Bare Metal** service boots the nodes and cleans the disks each time the node status is set to `available` for provisioning.

```
clean_nodes=true
```



References

For more information, refer to the *Storage Guide* at

https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/16.2/html-single/storage_guide/index

For more information, refer to the *Integrating an Overcloud with an Existing Red Hat Ceph Cluster* at

https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/16.1/html-single/integrating_an_overcloud_with_an_existing_red_hat_ceph_cluster/index

For more information, refer to the *Hyperconverged Infrastructure Guide* at

https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/16.1/html-single/hyperconverged_infrastructure_guide/index

For more information, refer to the *Deploying an Overcloud with Containerized Red Hat Ceph* guide at

https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/16.1/html-single/deploying_an_overcloud_with_containerized_red_hat_ceph/index

► Quiz

Introducing OpenStack Storage Architecture

Choose the correct answers to the following questions:

- ▶ 1. **The default object storage service in OpenStack is known by which name?**
 - a. Nova
 - b. Glance
 - c. RGW
 - d. Swift

- ▶ 2. **Which two of the following options describe the implementation choices for Ceph integration designs? (Choose two.)**
 - a. Stand-alone
 - b. External
 - c. Dedicated
 - d. Containerized

- ▶ 3. **Which of the following options is the most common node role that is used when TripleO builds a Ceph server?**
 - a. CephStorage
 - b. CephAll
 - c. ControllerStorage
 - d. StorageNode

- ▶ 4. **Which of the following options are benefits of a dedicated Ceph integration with OSP (Choose two.)?**
 - a. The number of OSDs in the cluster is limited only by hardware configuration.
 - b. Integrated installation and update strategies.
 - c. Hyperconverged infrastructure with compute resources.
 - d. Storage resources are available to external clients.
 - e. The Ceph cluster can support multiple OSP environments.

► Solution

Introducing OpenStack Storage Architecture

Choose the correct answers to the following questions:

- ▶ 1. **The default object storage service in OpenStack is known by which name?**
 - a. Nova
 - b. Glance
 - c. RGW
 - d. Swift

- ▶ 2. **Which two of the following options describe the implementation choices for Ceph integration designs? (Choose two.)**
 - a. Stand-alone
 - b. External
 - c. Dedicated
 - d. Containerized

- ▶ 3. **Which of the following options is the most common node role that is used when TripleO builds a Ceph server?**
 - a. CephStorage
 - b. CephAll
 - c. ControllerStorage
 - d. StorageNode

- ▶ 4. **Which of the following options are benefits of a dedicated Ceph integration with OSP (Choose two.)?**
 - a. The number of OSDs in the cluster is limited only by hardware configuration.
 - b. Integrated installation and update strategies.
 - c. Hyperconverged infrastructure with compute resources.
 - d. Storage resources are available to external clients.
 - e. The Ceph cluster can support multiple OSP environments.

Implementing Storage in OpenStack Components

Objectives

After completing this section, you should be able to describe how OpenStack implements Ceph storage for each storage-related OpenStack component.

OpenStack Storage Implementation Overview

Before Ceph, each of the storage components used local storage, such as direct attached physical or virtual disks, or network-attached storage (NAS), or storage area network (SAN) hardware. Use of NAS and SAN configurations supports larger storage that the control plane nodes can share, but that requires additional physical NICs or host adapters, limiting the ability of the control plane to scale easily.

The Network File System (NFS) is also a valid method for shared storage access across compute and controller nodes. Although mature and capable of significant performance and resilience when configured for redundancy, NFS has scaling limitations and was not designed for cloud application requirements. OpenStack needs a scalable storage solution design for use in the cloud.

Reviewing Ceph Capabilities in OSP

Ceph is a scalable storage solution that replicates data across commodity storage nodes. Ceph uses an object storage architecture for data storage, and provides multiple storage interfaces for object storage, block storage, and file systems.

Ceph integration with OpenStack features:

- Supports the same API that the Swift Object Store uses.
- Supports thin provisioning by using copy-on-write, making volume-based provisioning fast.
- Supports Keystone identity authentication, for transparent integration with or replacement of the Swift Object Store.
- Consolidates object storage and block storage.
- Supports the CephFS distributed file system interface.

Storage Implementation by Type

Each OpenStack service is an API abstraction that hides the back-end implementation. Many services can configure multiple back ends for use simultaneously. A service might configure multiple pools in Ceph, and use tiering or tagging criteria for transparently selecting an appropriate storage pool. Tiers can also accommodate workloads with different performance requirements. If tiers are implemented in an existing cluster, then CRUSH rule changes can result in significant pool data movement.

OpenStack services use unique service accounts, which are called after the service. The service account runs service actions on behalf of the requesting user or of another service. Similar accounts are created in Ceph for each OpenStack service that requires storage access. For example, the Image service is configured for Ceph access by using this command:

```
[admin@node ~]# ceph auth get-or-create client.glance mon 'profile rbd' \
osd 'profile rbd pool= images' mgr 'profile rbd pool=images'
```

Image Storage

In OpenStack, the default back end for the Image service is a file store that is located with the Glance API node on the controller node. The location is configurable, with a default of /var/lib/glance. To improve scalability, the Image service implemented an image cache at the default /var/lib/glance/image-cache/ location on the controller node. When the Compute service loads images that are stored in the default QCOW2 format to convert to RAW for use on the compute nodes, then the converted image is cached.

When Red Hat OpenStack Platform is installed with the Swift Object Store, TripleO places the image service back end on Swift by default. The Swift service creates a container called glance for storing Glance images.

When Ceph storage is integrated into RHOSP, TripleO places the image service back end on Ceph RADOS Block Devices (RBD) by default. Glance images are stored in a Ceph pool called images. RHOSP works with images as immutable blobs and handles them accordingly. The pool name is configurable with the glance_pool_name property. The images pool is configured as a replicated pool by default, which means that all images are replicated across storage devices for transparent resilience.

An image pool can be configured as erasure-coded to conserve disk space with a slight increase in CPU utilization.

When using Ceph as the storage back end, it is important to disable the image cache, as it is not needed because Ceph expects Glance images to be stored in the RAW format. When using RAW images, all image interactions occur within Ceph, including image clone and snapshot creation. Disabling the image cache eliminates significant CPU and network activity on controller nodes.

When using a distributed architecture with Distributed Compute Nodes (DCN), TripleO can configure the Image service with an image pool at each remote site. You can copy images between the central (hub) site and the remote sites. The DCN Ceph cluster uses RBD technologies, such as copy-on-write and snapshot layering, for fast instance launching. The Image, Block Storage, and Compute services must all be configured to use Ceph RBD as their back-end storage.

Object Storage

Object storage is implemented in OpenStack by the Object Store service (Swift). The Object Store service implements both the Swift API and the Amazon S3 API. The default storage back end is file-based, and uses an XFS-formatted partition mount in subdirectories of /srv/node on the designated storage node. You can also configure the Object Store service to use an existing, external Swift cluster as a back end.

When Ceph storage is integrated into RHOSP, TripleO configures the Object Store service to use the RADOS Gateway (RGW) as the back end. Similarly, the Image service is configured for RGW because Swift would not be available as a back end.

The Ceph Object Gateway can be integrated with the Keystone identity service. This integration configures RGW to use the Identity service as the user authority. If Keystone authorizes a user to access the gateway, then the user is also initially created on the Ceph Object Gateway. Identity tokens that Keystone validates are considered valid by the Ceph Object Gateway. The Ceph Object Gateway is also configured as an object-storage endpoint in Keystone.

Block Storage

Block storage is implemented in OpenStack by the **Block Storage** service (Cinder). The **Block Storage** service provides persistent volumes that remain in storage and are stable when not attached to any instance. It is common to configure the **Block Storage** service with multiple back ends. The default storage back end is the Logical Volume Manager (LVM), which is configured to use a volume group called `cinder-volumes`. **TripleO** can create the volume group during an installation, or use an existing `cinder-volumes` volume group.

When Ceph storage is integrated into RHOSP, **TripleO** configures the **Block Storage** service to use RADOS **Block Devices** (RBD) as the back end. **Block Storage** volumes are stored in a Ceph pool called `volumes`. Volume backups are stored in a Ceph pool called `backups`. Ceph block device images attach to an OpenStack instance by using `libvirt`, which configures the QEMU interface to the `librbd` Ceph module. Ceph stripes block volumes across multiple OSDs within the cluster, providing increased performance for large volumes when compared to local drives.

OpenStack volumes, snapshots, and clones are implemented as block devices. OpenStack uses volumes to boot VMs, or to attach to running VMs as further application storage.

File Storage

File storage is implemented in OpenStack by the **Shared File Systems** service (Manila). The **Shared File Systems** service supports multiple back ends and can provision shares from one or more back ends. Share servers export file shares by using various file system protocols such as NFS, CIFS, GlusterFS, or HDFS.

The **Shared File Systems** service is persistent storage and can be mounted to any number of client machines. You can detach file shares from one instance and attach them to another instance without data loss. The **Shared File Systems** service manages share attributes, access rules, quotas, and rate limits. Because unprivileged users are not allowed to use the `mount` command, the **Shared File Systems** service acts as a broker to mount and unmount shares that the storage operator configured.

When Ceph storage is integrated into RHOSP, **TripleO** configures the **Shared File Systems** service to use CephFS as the back end. CephFS uses the NFS protocol with the **Shared File Systems** service. **TripleO** can use the `ControllerStorageNFS` server role to configure an NFS Ganesh cluster as the scalable interface to the `libcephfs` back end.

Compute Storage

Ephemeral storage is implemented in OpenStack by the **Compute** service (Nova). The **Compute** service uses the KVM hypervisor with `libvirt` to launch compute workloads as VMs. The **Compute** service requires two types of storage for `libvirt` operations:

- **Base image:** A cached and formatted copy of the image from the Image service.
- **Instance overlay:** A layered volume to be overlaid on the base image to become the VM's instance disk.

When Ceph storage is integrated into RHOSP, **TripleO** configures the **Compute** service to use RADOS **Block Devices** (RBD) as the back end. With RBD, instance operating system disks can be managed either as ephemeral, to be deleted when the instance is shut down, or as a persistent volume. An ephemeral disk behaves like a normal disk, to be listed, formatted, mounted, and used as a block device. However, the disk and its data cannot be preserved or accessed beyond the instance that it is attached to.

In earlier OpenStack releases, a disk of a running VM appeared in the `/var/lib/nova/instances/<uuid>` directory of the hypervisor file system. Earlier Ceph versions could boot VMs only with the Block Storage services `boot-from-volume` function.

In recent versions, you can boot every VM inside Ceph directly without using the `Block Storage` service. This feature enables hypervisors to use the `live-migration` and `evacuate` operations to restore VMs in another hypervisor during a maintenance operation or on a hardware failure.



References

Cinder Administration: Configure multiple-storage back ends

<https://docs.openstack.org/cinder/latest/admin/blockstorage-multi-backend.html>

For more information, refer to the *Creating and Managing Instances Guide* at

https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/16.1/html-single/creating_and_managing_instances/index

For more information, refer to the *Distributed compute node and storage deployment* at

https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/16.1/html-single/distributed_compute_node_and_storage_deployment/index

For more information, refer to the *CephFS Back End Guide for the Shared File System Service* at

https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/16.1/html-single/cephfs_back_end_guide_for_the_shared_file_system_service/index

For more information, refer to the *Deploying the Shared File Systems service with CephFS through NFS* at

https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/16.1/html-single/deploying_the_shared_file_systems_service_with_cephfs_through_nfs/index

► Quiz

Implementing Storage in OpenStack Components

Choose the correct answers to the following questions:

- ▶ 1. Which of the following is the only image format supported by Red Hat OpenStack Platform for use with an integrated Ceph cluster?
 - a. QCOW2
 - b. VMDK
 - c. RAW
 - d. VHDX
- ▶ 2. Which four of the following are the default Ceph pool names used by OpenStack services? (Choose four.)
 - a. vms
 - b. volumes
 - c. backups
 - d. glance
 - e. shares
 - f. images
 - g. compute
- ▶ 3. Which three of the following are the OpenStack services backed by Ceph RADOS Block Devices? (Choose three.)
 - a. Deployment
 - b. Images
 - c. Shared File Systems
 - d. Block Storage
 - e. Object Storage
 - f. Compute
- ▶ 4. Which three of the following are required parameters to integrate an external Ceph Storage cluster with OpenStack? (Choose three.)
 - a. FSID
 - b. Manager node list
 - c. Monitor node list
 - d. client.openstack key-ring
 - e. admin.openstack key-ring
 - f. Monitor map

► Solution

Implementing Storage in OpenStack Components

Choose the correct answers to the following questions:

- ▶ 1. **Which of the following is the only image format supported by Red Hat OpenStack Platform for use with an integrated Ceph cluster?**
 - a. QCOW2
 - b. VMDK
 - c. RAW
 - d. VHDX

- ▶ 2. **Which four of the following are the default Ceph pool names used by OpenStack services? (Choose four.)**
 - a. vms
 - b. volumes
 - c. backups
 - d. glance
 - e. shares
 - f. images
 - g. compute

- ▶ 3. **Which three of the following are the OpenStack services backed by Ceph RADOS Block Devices? (Choose three.)**
 - a. Deployment
 - b. Images
 - c. Shared File Systems
 - d. Block Storage
 - e. Object Storage
 - f. Compute

- ▶ 4. **Which three of the following are required parameters to integrate an external Ceph Storage cluster with OpenStack? (Choose three.)**
 - a. FSID
 - b. Manager node list
 - c. Monitor node list
 - d. client.openstack key-ring
 - e. admin.openstack key-ring
 - f. Monitor map

Introducing OpenShift Storage Architecture

Objectives

After completing this section, you should be able to describe Red Hat OpenShift Container Platform storage requirements, and compare the architecture choices for using Red Hat Ceph Storage as an RHOCP storage back end.

Red Hat OpenShift Container Platform overview

Kubernetes is an orchestration service that deploys, manages, and scales containerized applications. Developers can use Kubernetes to iterate on application building and automate administrative tasks. Kubernetes wraps containers and other resources into pods, and enables abstracting an application into a single deployment unit.

Red Hat OpenShift Container Platform (RHOCP) is a collection of modular components and services that are built on top of a Kubernetes container infrastructure. OpenShift Container Platform provides remote management, multitenancy, monitoring, auditing, and application lifecycle management. It features enhanced security capabilities and self-service interfaces. It also integrates with major Red Hat products, that extend the capabilities of the platform.

OpenShift Container Platform is available in most clouds, whether as a managed cloud service in public clouds or as self-managed software in your data center. These implementations offer different levels of platform automation, update strategies, and operation customization. This training material references RHOCP 4.8.

OpenShift Container Platform assigns the responsibilities of each node within the cluster by using different roles. The *machine config pools (MCP)* are sets of hosts where a role is assigned. Each MCP manages the hosts and their configuration. The *control plane* and the *compute* MCPs are created by default.

Compute nodes are responsible for running the scheduled workloads of the control plane. Compute nodes contain services such as *CRI-O* (Container Runtime Interface with Open Container Initiative compatibility), to run, stop or restart containers, and *kubelet*, which acts as an agent to accept a request to operate the containers.

Control plane nodes are responsible for running the main OpenShift services, such as the following ones:

- OpenShift API server. It validates and configures the data for OpenShift resources, such as projects, routes, and templates.
- OpenShift controller manager. It watches the *etcd* service for changes to resources and uses the API to enforce the specified state.
- OpenShift OAuth API server. It validates and configures the data to authenticate to the OpenShift Container Platform, such as users, groups, and OAuth tokens.

Describing Operators and Custom Resource Definitions

Operators are applications that invoke the OpenShift controller API to manage resources. Operators provide a repeatable way to package, deploy, and manage a containerized application.

The operator container image defines the requirements for deployment, such as dependent services and hardware resources. Because operators require resource access, they typically use custom security settings. Operators provide an API for resource management and service configuration, and deliver levels of automated management and upgrade strategies.

OpenShift Container Platform uses the *Operator Lifecycle Manager (OLM)* to manage operators. OLM orchestrates the deployment, update, resource utilization, and deletion of other operators from the operator catalog. Every operator has a *Cluster Service Version (CSV)* that describes the required technical information to run the operator, such as the RBAC rules that it requires and the resources that it manages or depends on. OLM is itself an operator.

Custom Resource Definition (CRD) objects define unique object types in the cluster. *Custom Resource (CR)* objects are created from CRDs. Only cluster administrators can create CRDs. Developers with CRD read permission can add defined CR object types into their project.

Operators use CRDs by packaging them with any required RBAC policy and other software-specific logic. Cluster administrators can add CRDs manually to a cluster independently from an Operator lifecycle, to be available to all users.

Introducing Red Hat OpenShift Data Foundation

Red Hat OpenShift Data Foundation (formerly Red Hat OpenShift Container Storage) is a highly integrated collection of cloud storage and data services that acts as the storage control plane for OpenShift Container Platform. OpenShift Data Foundation is available as an operator in the *Red Hat OpenShift Container Platform Service Catalog*. OpenShift Data Foundation 4.8 uses *Red Hat Ceph Storage* 4.2.

Introducing OpenShift Container Storage Operator

The OpenShift Container Storage operator integrates three operators as an operator bundle to initialize and manage OpenShift Data Foundation services. The three operators are OpenShift Container Storage (`ocs-operator`), Rook-Ceph, and Multicloud Object Gateway (NooBaa). The operator bundle includes a converged CSV and all the needed CRDs to deploy `ocs-operator`, Rook-Ceph, and NooBaa operators.

Describing the `ocs-operator`

The operator `ocs-operator` initializes tasks for the OpenShift Data Foundation service and acts as a configuration gateway for Rook-Ceph and NooBaa. The operator `ocs-operator` depends on the configuration that is defined in the `OCSInitialization` and `StorageCluster` CRDs in the CSV bundle.

When the operator bundle is installed, the `ocs-operator` starts and creates an `OCSInitialization` resource if it does not already exist. The `OCSInitialization` resource performs basic setup and initializes services. It creates the `openshift-storage` namespace in which other bundle operators will create resources. You can edit this resource to adjust the tools that are included in the OpenShift Data Foundation operator. If the `OCSInitialization` resource is in a failed state, further start requests are ignored until the resource is deleted.

The `StorageCluster` resource manages the creation and reconciliation of CRDs for the Rook-Ceph and NooBaa operators. These CRDs are defined by known best practices and policies that Red Hat supports. You can create a `StorageCluster` resource with an installation wizard in OpenShift Container Platform.

Unlike the `OCSInitialization` resource, the `StorageCluster` resource creation or deletion operates outside `ocs-operator` control. Only one `StorageCluster` resource per OpenShift Container Platform cluster is supported.

Describing the Rook-Ceph Operator

Rook is a cloud-native storage orchestrator that provides the platform to abstract the complexity of Ceph layout and configuration. Rook-Ceph is the primary component for the `ocs-operator`. It incorporates the Ceph cluster into the operator bundle.

Rook-Ceph is responsible for the initial storage cluster bootstrap, administrative tasks, and the creation of the pods and other dependent resources in the `openshift-storage` namespace. Many advanced Ceph features, such as Placement Groups and CRUSH maps, are reserved for Rook management. Rook-Ceph facilitates a seamless storage consumption experience and minimizes the required cluster administration.

Monitoring is an important Rook-Ceph duty. Rook-Ceph watches the storage cluster state to ensure that it is available and healthy. Rook-Ceph monitors Ceph Placement Groups and automatically adjusts their configuration based on pool sizing, and monitors Ceph daemons. Rook-Ceph communicates with OpenShift APIs to request the necessary resources when the cluster scales.

Rook-Ceph provides two *Container Storage Interface (CSI)* drivers to create volumes, the RBD driver and the CephFS driver. These drivers provide the channel for OpenShift Container Platform to consume storage.



Note
The OpenShift Container Storage operator does not create *Persistent Volume* resources, but tracks resources that Ceph-CSI drivers created.

Figure 13.3 visualizes the Rook-Ceph operator interaction with OpenShift Container Platform.

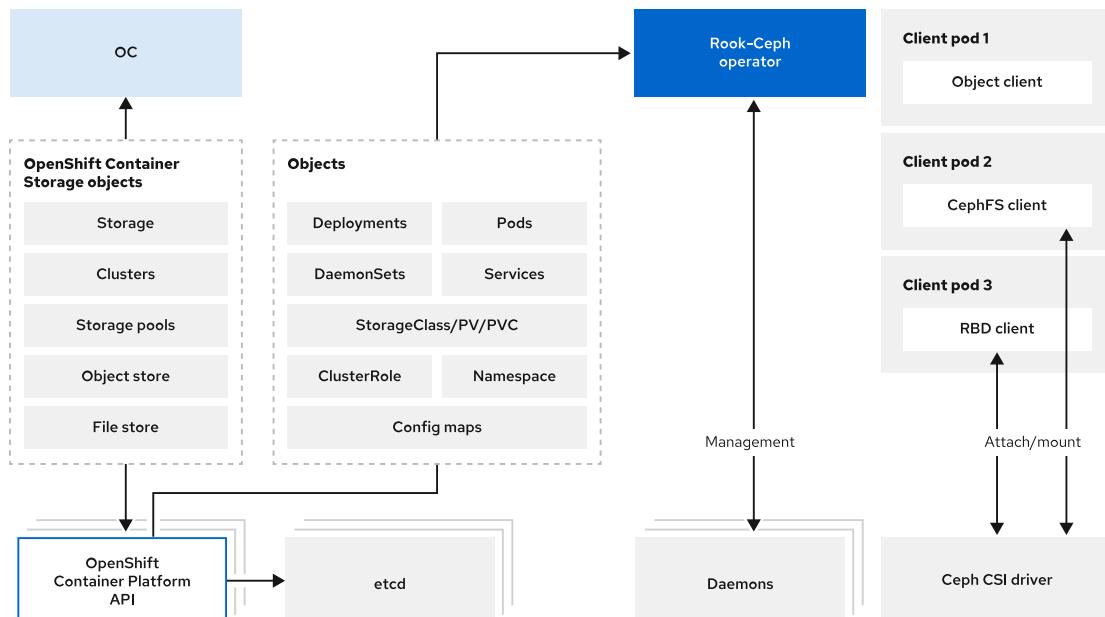


Figure 13.3: Rook Architecture

You can update the configuration of Rook-Ceph components via CRD updates. Rook-Ceph looks for configuration changes that the service API requested and applies them to the cluster. The cluster state reflects whether the cluster is in the desired state or is approaching it. Important CRDs for the cluster configuration are `CephCluster`, `CephObjectStore`, `CephFilesystem`, and `CephBlockPool`.

Describing the NooBaa operator

NooBaa is a multicloud object storage service that delivers an S3-compatible API in the OpenShift container storage operator bundle. NooBaa provides data placement policies that enable hybrid and multicloud interconnectivity, allowing active/active reads and writes across different clouds.

The NooBaa operator creates and reconciles changes for the NooBaa service and creates the following resources:

- Backing store
- Namespace store
- Bucket class
- Object bucket claims (OBCs)
- Prometheus rules and service monitoring
- Horizontal pod autoscaler (HPA)

NooBaa requires a backing store resource to save objects. A default backing store is created in an OpenShift Data Foundation deployment, but depends on the platform that OpenShift Container Platform is running on. For example, when OpenShift Container Platform or OpenShift Data Foundation is deployed on Amazon Web Services (AWS), it creates the backing store as an AWS:S3 bucket. For Microsoft Azure, the default backing store is a blob container. NooBaa can define multiple, concurrent backing stores.

Red Hat OpenShift Data Foundation installation

Red Hat OpenShift Data Foundation can be deployed in a cloud (AWS, Azure, IBM), or as a virtualized (RHV, VMware vSphere) or bare-metal environment. OpenShift Data Foundation uses the underlying provider's infrastructure to obtain the required resources for the storage cluster. When installing OpenShift Data Foundation, either install the storage cluster in *Internal mode* or use an existing Ceph Storage cluster for *External mode*. Both modes require the installation of the OpenShift Container Storage operator. Differences exist between installing the operator in *Internal mode* and *External mode*.

Internal Installation Mode

The OpenShift Container Storage operator *Internal mode* installation provisions the base services and makes available additional storage classes to applications.

OpenShift Data Foundation pods can be scheduled on the same nodes as application pods or on separate nodes. When using the same nodes, compute and storage resources must be scaled together. When using separate nodes, compute and storage resources scale independently. One *Internal* installation mode benefit is that the OpenShift Container Platform dashboard integrates cluster lifecycle management and monitoring.

The *Internal* installation mode uses the back-end infrastructure to provide storage resources by default. An alternative configuration is to choose the *Internal - Attached Devices* option, which uses the available local storage.

Use of the *Internal - Attached Devices* option has the following requirements:

- The *Local Storage* operator must be installed in the OpenShift cluster.

- The target nodes are scanned for disks to match specific criteria.
- The default namespace for the Local Storage operator is *openshift-local-storage*.

The **Internal** installation mode has the following advantages:

- Automated deployment of the Ceph cluster.
- Automated administrative tasks and monitoring.
- Integrated Dashboard with OpenShift Container Platform.
- Straightforward deployment with the OpenShift Container Platform back-end infrastructure.
- Hyperconverged infrastructure (with worker nodes).
- Lifecycle management and auto-update strategies.



Important

In an **Internal** storage installation, the integrated Ceph Storage cluster can be used only by the OpenShift Container Platform cluster where it is installed.

External installation mode

The **External** installation mode uses an existing Ceph Storage cluster that is expected to be in a healthy state. The OpenShift Container Platform cluster operates the resources that are created inside the storage cluster, but it does not monitor, update, or manage the cluster daemons or configuration.

The OpenShift Container Platform and the Ceph Storage cluster must meet certain conditions to correctly integrate the storage cluster.

- The OpenShift Container Platform version is 4.8 or above.
- OpenShift Container Storage 4.8 is installed.
- Ceph Storage 4.2z1 or later is installed. Ceph Storage 5 is not yet supported.
- The Ceph Storage dashboard is installed and configured.
- The external Ceph Storage cluster enables the PG Autoscaler with target_size_ratio of 0.49, as Red Hat recommends.
- The external Ceph cluster has an existing preconfigured RBD pool for use.

The **External** installation mode has the following advantages:

- The Ceph Storage cluster can use advanced customization.
- Storage can scale independently from the OpenShift Container Platform infrastructure.
- The external Ceph Storage cluster can support multiple OpenShift Container Platform clusters by creating separate pools.
- Object devices can be accessed and isolated between different OpenShift Container Platform clusters.
- OpenShift Container Platform Compute resource consumption is less than in **Internal** mode.



References

For more information, refer to the [Red Hat OpenShift Container Storage 4.8 Documentation](#).

Red Hat OpenShift Container Storage

https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.8

For more information, refer to the [Red Hat OpenShift Container Platform 4.8 Documentation](#).

Red Hat OpenShift Container Platform

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.8

► Quiz

Introducing OpenShift Storage Architecture

Choose the correct answers to the following questions:

- ▶ 1. **Which component in OpenShift Data Foundation provides the interface for OpenShift Container Platform to consume storage from Ceph? (Choose one.)**
 - a. CSI drivers
 - b. NooBaa
 - c. OCSSInitialization
 - d. CustomResourceDefinitions

- ▶ 2. **What are the advantages of installing OpenShift Data Foundation in internal mode? (Choose three.)**
 - a. Support for several OpenShift Container Platform clusters.
 - b. Storage back end can use the same infrastructure as the OpenShift Container Platform cluster.
 - c. Advanced feature and configuration customization.
 - d. Automated Ceph storage installation and configuration.
 - e. Seamless auto-update and lifecycle management.

- ▶ 3. **What are the main capabilities of the Rook-Ceph operator? (Choose three.)**
 - a. Provides a bundle with operators and resources to deploy the storage cluster.
 - b. Monitors Ceph daemons and ensures that the cluster is in a healthy state.
 - c. Deploys the storage cluster according to best practices and recommendations.
 - d. Interacts with multiple clouds to provide object services.
 - e. Looks for and applies configuration changes to the storage cluster.

- ▶ 4. **Which CustomerResourceDefinitions are provided by ocs-operator? (Choose two.)**
 - a. StorageCluster
 - b. CSI Drivers
 - c. OpenShiftStorage
 - d. OCSSInitialization

► Solution

Introducing OpenShift Storage Architecture

Choose the correct answers to the following questions:

- ▶ 1. **Which component in OpenShift Data Foundation provides the interface for OpenShift Container Platform to consume storage from Ceph? (Choose one.)**
 - a. CSI drivers
 - b. NooBaa
 - c. OCSSInitialization
 - d. CustomResourceDefinitions

- ▶ 2. **What are the advantages of installing OpenShift Data Foundation in internal mode? (Choose three.)**
 - a. Support for several OpenShift Container Platform clusters.
 - b. Storage back end can use the same infrastructure as the OpenShift Container Platform cluster.
 - c. Advanced feature and configuration customization.
 - d. Automated Ceph storage installation and configuration.
 - e. Seamless auto-update and lifecycle management.

- ▶ 3. **What are the main capabilities of the Rook-Ceph operator? (Choose three.)**
 - a. Provides a bundle with operators and resources to deploy the storage cluster.
 - b. Monitors Ceph daemons and ensures that the cluster is in a healthy state.
 - c. Deploys the storage cluster according to best practices and recommendations.
 - d. Interacts with multiple clouds to provide object services.
 - e. Looks for and applies configuration changes to the storage cluster.

- ▶ 4. **Which CustomerResourceDefinitions are provided by ocs-operator? (Choose two.)**
 - a. StorageCluster
 - b. CSI Drivers
 - c. OpenShiftStorage
 - d. OCSSInitialization

Implementing Storage in OpenShift Components

Objectives

After completing this section, you should be able to describe how OpenShift implements Ceph storage for each storage-related OpenShift feature.

Implementing Storage in Red Hat OpenShift Container Platform

Red Hat Data Foundation provides the storage infrastructure for Red Hat OpenShift Container Platform. To provide persistent storage resources to developers, OpenShift Container Platform uses Kubernetes object models.

Administrators can use a `StorageClass` resource to describe the storage types and characteristics of the cluster. Administrators can use classes to define storage needs such as QoS levels or provisioner types.

A `PersistentVolume (PV)` or `volume` resource type is a storage element in an OpenShift Container Platform cluster. `PersistentVolume` resources specify the type of disk, level of performance, and storage implementation type. A cluster administrator can manually create these objects, or a `StorageClass` resource can provide them dynamically. Resources, such as pods, can use `PersistentVolume` resources while maintaining lifecycle independence.

A `PersistentVolumeClaim (PVC)` or `claim` is a cluster user storage request from inside a project. `PersistentVolumeClaim` resources contain the requested storage and the required access mode.



Note

The `StorageClass` and `PersistentVolume` resources are cluster resources that are independent of any projects.

The following operations are the most common interactions between a `PersistentVolume` and `PersistentVolumeClaim` resources.

- **Provisioning storage.** In advance, administrators can create `PersistentVolume` resources with different types and sizes for future storage requests. By using a `StorageClass` resource, you can create `PersistentVolumes` resources dynamically. PVs have a *reclaim policy*, which is specified in the `reclaimPolicy` field of the class with a value of `Delete` or `Retain`. The default is `Delete`.

When installing OpenShift Data Foundation, the following storage classes are created:

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

**Note**

Red Hat recommends changing the default `StorageClass` to `ocs-storagecluster-ceph-rbd` backed by OpenShift Data Foundation.

- **Binding to a PersistentVolumeClaim.** The PVC request specifies the storage amount, access mode, and an optional storage class. If an existing unbound PV's attributes match the PVC, then the PV binds to the PVC. If no PV matches the PVC request, then a new PV is created. PVCs can remain unbound indefinitely if a matching PV does not exist or cannot be created. Claims are bound as matching volumes become available.
- **Using volumes.** A pod sees a `PersistentVolume` resource as a volume plug-in. When scheduling a pod, define the `PersistentVolumeClaim` in the `volumes` block. The cluster then looks for the `PersistentVolume` that is bound to that claim and mounts that volume. It is not recommended to use a `PersistentVolume` directly, because a different `PersistentVolumeClaim` volume might be bound at a later time.
- **Releasing a PersistentVolume.** To release a volume, delete the associated `PersistentVolumeClaim` object. Depending on the release policy of the `PersistentVolume` resource, the volume can be deleted or retained. The reclaim policy can be changed at any time.

Describing PersistentVolume Access Modes

A `PersistentVolume` can have different read-write access options depending on the provider capabilities. Storage providers can support different access modes for a volume, but a volume can have only one access mode at a time. Access modes are listed in this table.

Access Mode	Short Name	Description
ReadWriteOnce	RWO	The volume can be mounted as read-write by a single node.
ReadOnlyMany	ROX	The volume can be mounted as read-only by many nodes.
ReadWriteMany	RWX	The volume can be mounted as read-write by many nodes.

Volumes are matched to `PersistentVolumeClaims` resources with similar access modes. An exact match with access modes is preferred and is attempted first; however, the volume can have a wider access mode than the PVC requests. Similarly, a volume can be of the exact requested size or larger. In any case, the provided volume will have at least the required characteristics, but never less.

**Important**

Access modes are a description of the volume's access capabilities. The cluster does not enforce the claim's requested access, but permits access according to the volume's capabilities.

Introducing Rook-Ceph Toolbox

The Rook-Ceph Toolbox is a container that provides an interface to connect to the underlying Ceph Storage cluster of the OpenShift Container Storage operator. The toolbox is useful to run Ceph commands to view the cluster status, maps, and the devices that the cluster uses. The toolbox requires an existing, running Rook-Ceph cluster.

To install the toolbox, run the following command:

```
[cloud-user@ocp ~]$ oc patch OCSInitialization ocsinit -n openshift-storage \
--type json --patch '[{"op": "replace", "path": "/spec/enableCephTools", \
"value": true}]'
```

Verify that the container is running with the following command:

```
[cloud-user@ocp ~]$ oc get pods -n openshift-storage
```

You can run a remote shell to access the container:

```
[cloud-user@ocp ~]$ TOOLS_POD=$(oc get pods -n openshift-storage -l \
app=rook-ceph-tools -o name)
[cloud-user@ocp ~]$ oc rsh -n openshift-storage $TOOLS_POD
sh-4.4$ ceph status
cluster:
  id:      0f05478d-359b-4009-942f-a099f79a490b
  health: HEALTH_OK

  services:
    mon: 3 daemons, quorum a,b,c (age 23m)
    mgr: a(active, since 23m)
    mds: ocs-storagecluster-cephfilesystem:1 {0=ocs-storagecluster-cephfilesystem-
b=up:active} 1 up:standby-replay
    osd: 3 osds: 3 up (since 22m), 3 in (since 22m)
    rgw: 1 daemon active (ocs.storagecluster.cephobjectstore.a)
```

You can list the pools that Rook-Ceph created during the cluster creation:

```
sh-4.4$ ceph osd lspools
1 ocs-storagecluster-cephblockpool
2 ocs-storagecluster-cephobjectstore.rgw.control
3 ocs-storagecluster-cephfilesystem-metadata
4 ocs-storagecluster-cephfilesystem-data0
5 ocs-storagecluster-cephobjectstore.rgw.meta
6 ocs-storagecluster-cephobjectstore.rgw.log
7 ocs-storagecluster-cephobjectstore.rgw.buckets.index
8 ocs-storagecluster-cephobjectstore.rgw.buckets.non-ec
9 .rgw.root
10 ocs-storagecluster-cephobjectstore.rgw.buckets.data
```

Reviewing PersistentVolume Backed by Ceph RBD

The `ocs-storagecluster-ceph-rbd` storage class is used to create `ReadWriteOnce` (RWO) persistent storage in Red Hat Data Foundation. You can request an RBD volume by creating a `PersistentVolumeClaim`. This example and further examples run in a Red Hat Container Platform cluster with Red Hat Data Foundation installed.

To change the default `StorageClass` resource to `ocs-storagecluster-ceph-rbd`, find the current default `StorageClass`. Notice the `default` label in the name.

```
[cloud-user@ocp ~]$ oc get storageclass
NAME           PROVISIONER
RECLAIMPOLICY VOLUMEBINDINGMODE ALLOWVOLUMEEXPANSION AGE
ocs-storagecluster-ceph-rbd   openshift-storage.rbd.csi.ceph.com   Delete
  Immediate      true          100m
ocs-storagecluster-ceph-rgw   openshift-storage.ceph.rook.io/bucket Delete
  Immediate      false         100m
ocs-storagecluster-cephfs    openshift-storage.cephfs.csi.ceph.com Delete
  Immediate      true          100m
openshift-storage.noobaa.io  openshift-storage.noobaa.io/obc     Delete
  Immediate      false         96m
standard (default)          kubernetes.io/cinder               Delete
  WaitForFirstConsumer true        162m
standard-csi                cinder.csi.openstack.org          Delete
  WaitForFirstConsumer true        162m
[cloud-user@ocp ~]$ oc describe sc/standard
Name:           standard
IsDefaultClass: Yes
Annotations:    storageclass.kubernetes.io/is-default-class=true
...output omitted...
```

Set the value for `storageclass.kubernetes.io/is-default-class` to `false` to remove the property.

```
[cloud-user@ocp ~]$ oc patch storageclass standard -p '{"metadata": \
{"annotations": {"storageclass.kubernetes.io/is-default-class": "false"}}}'
storageclass.storage.k8s.io/standard patched
[cloud-user@ocp ~]$ oc describe sc/standard
Name:           standard
IsDefaultClass: No
Annotations:    storageclass.kubernetes.io/is-default-class=false
...output omitted...
```

Then, set the value for `storageclass.kubernetes.io/is-default-class` to `true` to make it the default StorageClass.

```
[cloud-user@ocp ~]$ oc patch storageclass ocs-storagecluster-ceph-rbd -p \
'{"metadata": {"annotations": {"storageclass.kubernetes.io/is-default-class": \
"true"}}}'
storageclass.storage.k8s.io/ocs-storagecluster-ceph-rbd patched
[cloud-user@ocp ~]$ oc describe storageclass/ocs-storagecluster-ceph-rbd
Name:           ocs-storagecluster-ceph-rbd
IsDefaultClass: Yes
...output omitted...
```

To request a volume, a YAML file is needed with the `PersistentVolumeClaim` resource definition. Notice the `accessModes` and `storage` fields. Then, create the resource.

```
[cloud-user@ocp ~]$ cat cl260-pvc-01.yml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
```

```

name: cl260-pvc-01
spec:
  storageClassName: ocs-storagecluster-ceph-rbd
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
[cloud-user@ocp ~]$ oc create -f cl260-pvc-01.yml
persistentvolumeclaim/cl260-pvc-01 created

```

View the `PersistentVolumeClaim` resource details to check whether it is bound. When the status is `Bound`, then view the volume resource.

```

[cloud-user@ocp ~]$ oc describe pvc/cl260-pvc-01
Name:           cl260-pvc-01
Namespace:      ceph-rbd-backend
StorageClass:   ocs-storagecluster-ceph-rbd
Status:         Bound
Volume:         pvc-0bf8894b-45db-4b5e-9d49-c03a1ea391fd
...output omitted...

```

To match the volume resource with the rbd device, inspect the `VolumeHandle` attribute in the `PersistentVolume` description.

```

[cloud-user@ocp ~]$ oc describe pv/pvc-0bf8894b-45db-4b5e-9d49-c03a1ea391fd
Name:           pvc-0bf8894b-45db-4b5e-9d49-c03a1ea391fd
...output omitted...
StorageClass:   ocs-storagecluster-ceph-rbd
Status:         Bound
Claim:          ceph-rbd-backend/cl260-pvc-01
...output omitted...
Capacity:       5Gi
...output omitted...
VolumeHandle:   0001-0011-openshift-storage-0000000000000001-
e39e9ebc-1032-11ec-8f56-0a580a800230
...output omitted...

```

To find the device in the Ceph Storage cluster, log in to the Rook-Ceph Toolbox shell and list the `ocs-storagecluster-cephblockpool` pool. Observe that the device name matches the second part of the `VolumeHandle` property of the volume resource.

```

[cloud-user@ocp ~]$ TOOLS_POD=$(oc get pods -n openshift-storage -l \
app=rook-ceph-tools -o name)
[cloud-user@ocp ~]$ oc rsh -n openshift-storage $TOOLS_POD
sh-4.4$ rbd ls ocs-storagecluster-cephblockpool
csi-vol-e39e9ebc-1032-11ec-8f56-0a580a800230
...output omitted...

```

To resize the volume, edit the YAML file and edit the `storage` field to the new wanted capacity. Then, apply the changes.

```
[cloud-user@ocp ~]$ cat cl260-pvc-01.yml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cl260-pvc-01
spec:
  storageClassName: ocs-storagecluster-ceph-rbd
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
[cloud-user@ocp ~]$ oc apply -f cl260-pvc-01.yml
persistentvolumeclaim/cl260-pvc-01 created
```

Verify the new capacity in the volume.

```
[cloud-user@ocp ~]$ oc describe pv/pvc-0bf8894b-45db-4b5e-9d49-c03a1ea391fd
Name:          pvc-0bf8894b-45db-4b5e-9d49-c03a1ea391fd
...output omitted...
StorageClass:  ocs-storagecluster-ceph-rbd
Status:        Bound
Claim:         ceph-rbd-backend/cl260-pvc-01
...output omitted...
Capacity:     10Gi
...output omitted...
```

Reviewing PersistentVolume Backed by CephFS

You can use the `ocs-storagecluster-cephfs` storage class to create file devices that are backed by the CephFS that Rook-Ceph manages. It is typical to create volume resources with RWX (ReadWriteMany) access mode. This access mode is used when presenting a volume to several pods.

Define `ocs-storagecluster-cephfs` in the `storage-class` field.

```
[cloud-user@ocp ~]$ cat cl260-pvc-02.yml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cl260-pvc-02
spec:
  storageClassName: ocs-storagecluster-cephfs
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
[cloud-user@ocp ~]$ oc create -f cl260-pvc-02.yml
persistentvolumeclaim/cl260-pvc-02 created
[cloud-user@ocp ~]$ oc describe pvc/cl260-pvc-02
Name:          cl260-pvc-02
Namespace:     cl260-cephfs
```

```
StorageClass: ocs-storagecluster-cephfs
Status: Bound
Volume: pvc-793c06bc-4514-4c11-9272-cf6ce51996e8
...output omitted...
Capacity: 10Gi
Access Modes: RWX
VolumeMode: Filesystem
...output omitted...
```

To test the volume, create a demo application and scale the deployment to three nodes.

```
[cloud-user@ocp ~]$ kubectl create deployment hello-node \
--image=k8s.gcr.io/serve_hostname
deployment.apps/hello-node created

[cloud-user@ocp ~]$ oc get deployment
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
hello-node 1/1       1           1          2m54s
[cloud-user@ocp ~]$ oc scale deployment hello-node --replicas 3
deployment.apps/hello-node scaled
```

Then, mount the volume and verify that all nodes can see the volume.

```
[cloud-user@ocp ~]$ oc set volume deployment/hello-node --add \
--mount-path=/cl260-data \
--name=pv c-793c06bc-4514-4c11-9272-cf6ce51996e8
deployment.apps/hello-node volume updated
[cloud-user@ocp ~]$ oc get pods
NAME                  READY   STATUS    RESTARTS   AGE
hello-node-67754bb7bf-25xv8 1/1     Running   0          27s
hello-node-67754bb7bf-9cdk4 1/1     Running   0          31s
hello-node-67754bb7bf-lzssb 1/1     Running   0          35s
[cloud-user@ocp ~]$ oc describe pod/hello-node-67754bb7bf-25xv8
[cloud-user@ocp ~]$ oc describe pod/hello-node-67754bb7bf-25xv8 | grep -A1 Mounts
Mounts:
/cl260-data from pvc-793c06bc-4514-4c11-9272-cf6ce51996e8 (rw)
[cloud-user@bastion ~]$ oc describe pod/hello-node-67754bb7bf-9cdk4 | grep -A1
Mounts
Mounts:
/cl260-data from pvc-793c06bc-4514-4c11-9272-cf6ce51996e8 (rw)
[cloud-user@bastion ~]$ oc describe pod/hello-node-67754bb7bf-lzssb | grep -A1
Mounts
Mounts:
/cl260-data from pvc-793c06bc-4514-4c11-9272-cf6ce51996e8 (rw)
```

Reviewing Object Resources Managed by NooBaa

The Multicloud Object Gateway (MCG) or NooBaa service simplifies the interaction with object devices across cloud providers and clusters. The NooBaa dashboard provides an overview of the operator status. NooBaa provides a CLI for management tasks and for ease of use of operations. The NooBaa CLI must be installed and configured to access the backing stores.

You can view status, access, and secret keys with the noobaa backingstore status command:

```
[cloud-user@ocp ~]$ noobaa backingstore status noobaa-default-backing-store
INFO[0001] Exists: BackingStore "noobaa-default-backing-store"
INFO[0001] Exists: Secret "rook-ceph-object-user-ocs-storagecluster-
cephobjectstore-noobaa-ceph-objectstore-user"
INFO[0001] BackingStore "noobaa-default-backing-store" Phase is Ready

# BackingStore spec:
s3Compatible:
  endpoint: http://rook-ceph-rgw-ocs-storagecluster-cephobjectstore.openshift-
storage.svc:80
  secret:
    name: rook-ceph-object-user-ocs-storagecluster-cephobjectstore-noobaa-ceph-
objectstore-user
    namespace: openshift-storage
...output omitted...
type: s3-compatible

# Secret data:
AccessKey: D7VHJ1I32B0LVJ0EEL9W
Endpoint: http://rook-ceph-rgw-ocs-storagecluster-cephobjectstore.openshift-
storage.svc:80
SecretKey: wY8ww5Dbd0qwre8gj1HTiA0fADY61zNcX1w8z
```

To create an ObjectBucketClaim with NooBaa, run the following command:

```
[cloud-user@ocp ~]$ noobaa obc create cl260-obc-01
INFO[0001] Exists: StorageClass "openshift-storage.noobaa.io"
INFO[0001] Created: ObjectBucketClaim "cl260-obc-01"
...output omitted...
INFO[0040] OBC "cl260-obc-01" Phase is "Pending"
INFO[0043] OBC "cl260-obc-01" Phase is Bound
...output omitted...
Connection info:
  BUCKET_HOST      : s3.openshift-storage.svc
  BUCKET_NAME     : cl260-obc-01-0d1ccb90-9caa-4515-969b-0b80a3ce8cd0
  BUCKET_PORT      : 443
  AWS_ACCESS_KEY_ID : TFV8sT9aKaxW3xfRHkwo
  AWS_SECRET_ACCESS_KEY : XQ6TBED4LoFq5Fj1/Le+m0cGzGEaa2wmByYPbTqz
...output omitted...
```

When NooBaa finishes the creation, it communicates with the OpenShift Container Platform cluster and delegates the characteristics of the `ObjectBucketClaim` that is created.

```
[cloud-user@ocp ~]$ oc get obc
NAME      STORAGE-CLASS      PHASE   AGE
cl260-obc-01  openshift-storage.noobaa.io  Bound  2m24s
```

You can review the attributes of a ObjectBucketClaim resource with the -o yaml option to query the resource definition. You can use this option to view the S3 access credentials of the resource.

```
[cloud-user@ocp ~]$ oc get obc cl260-objc-01 -o yaml
...output omitted...
spec:
  bucketName: cl260-objc-01-0d1ccb90-9caa-4515-969b-0b80a3ce8cd0
  generateBucketName: cl260-objc-01
  objectBucketName: objc-openshift-storage-cl260-objc-01
  storageClassName: openshift-storage.noobaa.io
status:
  phase: Bound
```

You can view the buckets in Rook-Ceph with the toolbox:

```
[cloud-user@ocp ~]$ oc rsh -n openshift-storage $TOOLS_POD
sh-4.4$ radosgw-admin bucket list
[
    "nb.1631071958739.apps.cluster-ea50.dynamic.opentlc.com",
    "rook-ceph-bucket-checker-12a9cf6b-502b-4aff-b5a3-65e5b0467437"
]
sh-4.4$ radosgw-admin bucket stats
[
...
...output omitted...
{
    "bucket": "rook-ceph-bucket-checker-12a9cf6b-502b-4aff-b5a3-65e5b0467437",
    "num_shards": 11,
    "tenant": "",
    "zonegroup": "68a58f7b-d282-467f-b28d-d862b4c98e1d",
    "placement_rule": "default-placement",
...
...output omitted...
    "bucket_quota": {
        "enabled": false,
        "check_on_raw": false,
        "max_size": -1,
        "max_size_kb": 0,
        "max_objects": -1
    }
}
]
```

OpenShift Container Platform manages Rook-Ceph resources, provided by OpenShift Data Foundation, by using storage classes and persistent volumes frameworks. Developers can request persistent volumes by defining persistent volume claims with the desired size and access mode. Once a persistent volume attaches to a persistent volume claim, it can be mounted and used by one or more pods as a regular storage device.



References

For more information, refer to the *Red Hat OpenShift Container Storage 4.8 Documentation*.

Red Hat OpenShift Container Storage

https://access.redhat.com/documentation/en-us/red_hat_openshift_container_storage/4.8

For more information, refer to the *Red Hat OpenShift Container Platform 4.8 Documentation*.

Red Hat OpenShift Container Platform

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.8

► Quiz

Implementing Storage in OpenShift Components

Choose the correct answers to the following questions:

- ▶ 1. **What is the relationship between PersistentVolume and PersistentVolumeClaim? (Choose one.)**
 - a. A PersistentVolumeClaim can define several PersistentVolumes to be attached.
 - b. A PersistentVolume requests a claim and the PersistentVolumeClaim is attached.
 - c. A PersistentVolumeClaim requests a volume and the PersistentVolume is attached to that claim.
 - d. A PersistentVolumeClaim is created with the definition of a PersistentVolume.

- ▶ 2. **What type of resource can NooBaa interact with? (Choose one.)**
 - a. CustomResourceDefinitions
 - b. StorageClass
 - c. PersistentVolumeClaim
 - d. ObjectBucketClaim

- ▶ 3. **In what scenario, where an application mounts a volume, is a volume with RWX access mode required? (Choose one.)**
 - a. Mounted to many pods that all have read and write permissions.
 - b. Mounted to many pods that have read permission only.
 - c. Mounted to one pod that has read and write permissions.
 - d. Mounted to one pod that has read permission only.

- ▶ 4. **What OpenShift Container Platform resource declares the storage back-end characteristics such as QoS and provisioner type? (Choose two.)**
 - a. CustomResourceDefinitions
 - b. StorageClass
 - c. PersistentVolumeClaim
 - d. ObjectBucketClaim

► Solution

Implementing Storage in OpenShift Components

Choose the correct answers to the following questions:

- ▶ 1. **What is the relationship between PersistentVolume and PersistentVolumeClaim? (Choose one.)**
 - a. A PersistentVolumeClaim can define several PersistentVolumes to be attached.
 - b. A PersistentVolume requests a claim and the PersistentVolumeClaim is attached.
 - c. A PersistentVolumeClaim requests a volume and the PersistentVolume is attached to that claim.
 - d. A PersistentVolumeClaim is created with the definition of a PersistentVolume.
- ▶ 2. **What type of resource can NooBaa interact with? (Choose one.)**
 - a. CustomResourceDefinitions
 - b. StorageClass
 - c. PersistentVolumeClaim
 - d. ObjectBucketClaim
- ▶ 3. **In what scenario, where an application mounts a volume, is a volume with RWX access mode required? (Choose one.)**
 - a. Mounted to many pods that all have read and write permissions.
 - b. Mounted to many pods that have read permission only.
 - c. Mounted to one pod that has read and write permissions.
 - d. Mounted to one pod that has read permission only.
- ▶ 4. **What OpenShift Container Platform resource declares the storage back-end characteristics such as QoS and provisioner type? (Choose two.)**
 - a. CustomResourceDefinitions
 - b. StorageClass
 - c. PersistentVolumeClaim
 - d. ObjectBucketClaim

Summary

In this chapter, you learned:

- Red Hat Ceph Storage can provide a unified storage back end for OpenStack services that consume block, image, object, and file-based storage.
- OpenStack Glance can use Ceph RBD images to store the operating system images that it manages.
- OpenStack Cinder can also use RADOS block devices to provide block-based storage for virtual machines that run as cloud instances.
- The RADOS Gateway can replace the native OpenStack Swift storage by providing object storage for applications that use the OpenStack Swift API, and integrates its user authentication with OpenStack Keystone.
- Red Hat OpenShift Data Foundation is an operator bundle that provides cloud storage and data services to Red Hat OpenShift Container Platform; it is composed of the ocs-storage, NooBaa, and Rook-Ceph operators.
- Rook-Ceph is a cloud storage orchestrator that installs, monitors, and manages the underlying Ceph cluster in the OpenShift Data Foundation bundle operator. Rook-Ceph provides the required drivers to request storage to the cluster.
- PersistentVolumeClaims are an OpenShift resource type that represent a request for a storage object. They contain the StorageClass which describes the PersistentVolume that should bind to it.
- Access modes describe the mount capabilities of a PersistentVolume on pods.

Chapter 14

Comprehensive Review

Goal

Review tasks from *Cloud Storage with Red Hat Ceph Storage*

Objectives

- Review tasks from *Cloud Storage with Red Hat Ceph Storage*

Sections

- Comprehensive Review

Labs

- Deploying Red Hat Ceph Storage
- Configuring Red Hat Ceph Storage
- Deploying CephFS
- Deploying and Configuring Block Storage with RBD
- Deploying and Configuring RADOS Gateway

Comprehensive Review

Objectives

After completing this section, you should be able to demonstrate knowledge and skills learned in *Cloud Storage with Red Hat Ceph Storage*.

Reviewing Cloud Storage with Red Hat Ceph Storage

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, Introducing Red Hat Ceph Storage Architecture

Describe Red Hat Ceph Storage architecture, including data organization, distribution, and client access methods.

- Describe the personas in the cloud storage ecosystem that characterize the use cases and tasks taught in this course.
- Describe the Red Hat Ceph Storage architecture, introduce the Object Storage Cluster, and describe the choices in data access methods.
- Describe and compare the use cases for the various management interfaces provided for Red Hat Ceph Storage.

Chapter 2, Deploying Red Hat Ceph Storage

Deploy a new Red Hat Ceph Storage cluster and expand the cluster capacity.

- Prepare for and perform a Red Hat Ceph Storage cluster deployment using cephadm command-line tools.
- Expand capacity to meet application storage requirements by adding OSDs to an existing cluster.

Chapter 3, Configuring a Red Hat Ceph Storage Cluster

Manage the Red Hat Ceph Storage configuration, including the primary settings, the use of monitors, and the cluster network layout.

- Identify and configure the primary settings for the overall Red Hat Ceph Storage cluster.
- Describe the purpose of cluster monitors and the quorum procedures, query the monitor map, manage the configuration database, and describe Cephx.
- Describe the purpose for each of the cluster networks, and view and modify the network configuration.

Chapter 4, Creating Object Storage Cluster Components

Create and manage the components that comprise the object storage cluster, including OSDs, pools, and the cluster authorization method.

- Describe OSD configuration scenarios and create BlueStore OSDs using ceph-volume.
- Describe and compare replicated and erasure coded pools, and create and configure each pool type.
- Describe Cephx and configure user authentication and authorization for Ceph clients.

Chapter 5, Creating and Customizing Storage Maps

Manage and adjust the CRUSH and OSD maps to optimize data placement to meet the performance and redundancy requirements of cloud applications.

- Administer and update the cluster CRUSH map used by the Ceph cluster.
- Describe the purpose and modification of the OSD maps.

Chapter 6, Providing Block Storage Using RADOS Block Devices

Configure Red Hat Ceph Storage to provide block storage for clients using RADOS block devices (RBDs).

- Provide block storage to Ceph clients using RADOS block devices (RBDs), and manage RBDs from the command line.
- Create and configure RADOS block devices snapshots and clones.
- Export an RBD image from the cluster to an external file and import it into another cluster.

Chapter 7, Expanding Block Storage Operations

Expand block storage operations by implementing remote mirroring and the iSCSI Gateway.

- Configure an RBD mirror to replicate an RBD block device between two Ceph clusters for disaster recovery purposes.
- Configure the Ceph iSCSI Gateway to export RADOS Block Devices using the iSCSI protocol, and configure clients to use the iSCSI Gateway.

Chapter 8, Providing Object Storage Using a RADOS Gateway

Configure Red Hat Ceph Storage to provide object storage for clients using a RADOS Gateway (RGW).

- Deploy a RADOS Gateway to provide clients with access to Ceph object storage.
- Configure the RADOS Gateway with multisite support to allow objects to be stored in two or more geographically diverse Ceph storage clusters.

Chapter 9, Accessing Object Storage Using a REST API

Configure the RADOS Gateway to provide access to object storage using REST APIs.

- Configure the RADOS Gateway to provide access to object storage compatible with the Amazon S3 API, and manage objects stored using that API.

- Configure the RADOS Gateway to provide access to object storage compatible with the Swift API, and manage objects stored using that API.

Chapter 10, Providing File Storage with CephFS

Configure Red Hat Ceph Storage to provide file storage for clients using the Ceph File System (CephFS).

- Provide file storage on the Ceph cluster by deploying the Ceph File System (CephFS).
- Configure CephFS, including snapshots, replication, memory management, and client access.

Chapter 11, Managing a Red Hat Ceph Storage Cluster

Manage an operational Ceph cluster using tools to check status, monitor services, and properly start and stop all or part of the cluster. Perform cluster maintenance by replacing or repairing cluster components, including MONs, OSDs, and PGs.

- Administer and monitor a Red Hat Ceph Storage cluster, including starting and stopping specific services or the full cluster, and querying cluster health and utilization.
- Perform common cluster maintenance tasks, such as adding or removing MONs and OSDs, and recovering from various component failures.

Chapter 12, Tuning and Troubleshooting Red Hat Ceph Storage

Identify the key Ceph cluster performance metrics, and use them to tune and troubleshoot Ceph operations for optimal performance.

- Choose Red Hat Ceph Storage architecture scenarios and operate Red Hat Ceph Storage-specific performance analysis tools to optimize cluster deployments.
- Protect OSD and cluster hardware resources from over-utilization by controlling scrubbing, deep scrubbing, backfill, and recovery processes to balance CPU, RAM, and I/O requirements.
- Identify key tuning parameters and troubleshoot performance for Ceph clients, including RADOS Gateway, RADOS Block Devices, and CephFS.

Chapter 13, Managing Cloud Platforms with Red Hat Ceph Storage

Manage Red Hat cloud infrastructure to use Red Hat Ceph Storage to provide image, block, volume, object, and shared file storage.

- Describe Red Hat OpenStack Platform storage requirements, and compare the architecture choices for using Red Hat Ceph Storage as an RHOSP storage back end.
- Describe how OpenStack implements Ceph storage for each storage-related OpenStack component.
- Describe Red Hat OpenShift Container Platform storage requirements, and compare the architecture choices for using Red Hat Ceph Storage as an RHOCP storage back end.
- Describe how OpenShift implements Ceph storage for each storage-related OpenShift feature.

▶ Lab

Deploying Red Hat Ceph Storage

In this review, you deploy a Red Hat Ceph Storage cluster using a service specification file.

Outcomes

You should be able to deploy a Red Hat Ceph Storage cluster using a service specification file.

Before You Begin

If you did not reset your classroom virtual machines at the end of the last chapter, save any work you want to keep from earlier exercises on those machines and reset the classroom environment now.



Important

Reset your environment before performing this exercise. All comprehensive review labs start with a clean, initial classroom environment that includes a pre-built, fully operational Ceph cluster. This first comprehensive review will remove that cluster, but still requires the rest of the clean classroom environment.

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

```
[student@workstation ~]$ lab start comprehensive-review1
```

This command confirms that the local container registry for the classroom is running and deletes the prebuilt Ceph cluster so it can be redeployed with the steps in this exercise.



Important

This lab start script immediately deletes the prebuilt Ceph cluster and takes a few minutes to complete. Wait for the command to finish before continuing.

Specifications

- Deploy a four node Red Hat Ceph Storage cluster using a service specification file with these parameters:
 - Use the registry at `registry.lab.example.com` with the username `registry` and the password `redhat`.
 - Deploy MONs on the `clienta`, `serverc`, `serverd`, and `servere` nodes.

- Deploy RGWs on the `serverc` and `serverd` nodes, with the `service_id` set to `realm.zone`.
- Deploy MGRs on the `clienta`, `serverc`, `serverd`, and `servere` nodes.
- Deploy OSDs on the `serverc`, `serverd`, and `servere` nodes, with the `service_id` set to `default_drive_group`. On all OSD nodes, use the `/dev/vdb`, `/dev/vdc`, and `/dev/vdd` drives as data devices.

Hostname	IP Address
<code>clienta.lab.example.com</code>	172.25.250.10
<code>serverc.lab.example.com</code>	172.25.250.12
<code>serverd.lab.example.com</code>	172.25.250.13
<code>servere.lab.example.com</code>	172.25.250.14

- After the cluster is installed, manually add the `/dev/vde` and `/dev/vdf` drives as data devices on the `servere` node.
 - Set the OSD journal size to 1024 MiB.
 - Use `172.25.250.0/24` for the OSD public network, and `172.25.249.0/24` for the OSD cluster network.

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-review1
```

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-review1
```

This concludes the lab.

► Solution

Deploying Red Hat Ceph Storage

In this review, you deploy a Red Hat Ceph Storage cluster using a service specification file.

Outcomes

You should be able to deploy a Red Hat Ceph Storage cluster using a service specification file.

Before You Begin

If you did not reset your classroom virtual machines at the end of the last chapter, save any work you want to keep from earlier exercises on those machines and reset the classroom environment now.



Important

Reset your environment before performing this exercise. All comprehensive review labs start with a clean, initial classroom environment that includes a pre-built, fully operational Ceph cluster. This first comprehensive review will remove that cluster, but still requires the rest of the clean classroom environment.

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

```
[student@workstation ~]$ lab start comprehensive-review1
```

This command confirms that the local container registry for the classroom is running and deletes the prebuilt Ceph cluster so it can be redeployed with the steps in this exercise.



Important

This lab start script immediately deletes the prebuilt Ceph cluster and takes a few minutes to complete. Wait for the command to finish before continuing.

1. Using the **serverc** host as the bootstrap host, install the **cephadm-ansible** package, create the inventory file, and run the pre-flight playbook to prepare cluster hosts. `.
 1. On the **serverc** host, install the **cephadm-ansible** package.

```
[student@workstation ansible]$ ssh admin@serverc
[admin@serverc ~]$ sudo -i
[root@serverc ~]# yum install cephadm-ansible
...output omitted...
Complete!
```

- 1.2. Create the hosts inventory file in the /usr/share/cephadm-ansible directory.

```
[root@serverc ~]# cd /usr/share/cephadm-ansible
[root@serverc cephadm-ansible]# cat hosts
clienta.lab.example.com
serverc.lab.example.com
serverd.lab.example.com
servere.lab.example.com
```

- 1.3. Run the cephadm-preflight.yml playbook.

```
[root@serverc cephadm-ansible]# ansible-playbook -i hosts \
cephadm-preflight.yml --extra-vars "ceph_origin="
...output omitted...
```



Note

The `ceph_origin` variable is set to empty, which causes some playbooks tasks to be skipped because, in this classroom, the Ceph packages are installed from a local classroom repository. In a production environment, set `ceph_origin` to `rhcs` to enable the Red Hat Storage Tools repository for your supported deployment.

2. On the serverc host, create the `initial-config-primary-cluster.yaml` cluster service specification file in the `/root/ceph` directory. Include four hosts with the following specifications:

- Deploy MONs on `clienta`, `serverc`, `serverd`, and `servere`.
- Deploy RGWs on `serverc` and `serverd`, with the `service_id` set to `realm.zone`.
- Deploy MGRs on `clienta`, `serverc`, `serverd`, and `servere`.
- Deploy OSDs on the `serverc`, `serverd`, and `servere` nodes, with the `service_id` set to `default_drive_group`. On all OSD nodes, use the `/dev/vdb`, `/dev/vdc`, and `/dev/vdd` drives as data devices.

Hostname	IP Address
clienta.lab.example.com	172.25.250.10
serverc.lab.example.com	172.25.250.12
serverd.lab.example.com	172.25.250.13
servere.lab.example.com	172.25.250.14

- 2.1. Create the `initial-config-primary-cluster.yaml` cluster service specification file in the `/root/ceph` directory.

```
[root@serverc cephadm-ansible]# cd /root/ceph
[root@serverc ceph]# cat initial-config-primary-cluster.yaml
service_type: host
addr: 172.25.250.10
hostname: clienta.lab.example.com
---
service_type: host
addr: 172.25.250.12
hostname: serverc.lab.example.com
---
service_type: host
addr: 172.25.250.13
hostname: serverd.lab.example.com
---
service_type: host
addr: 172.25.250.14
hostname: servere.lab.example.com
---
service_type: mon
placement:
hosts:
- clienta.lab.example.com
- serverc.lab.example.com
- serverd.lab.example.com
- servere.lab.example.com
---
service_type: rgw
service_id: realm.zone
placement:
hosts:
- serverc.lab.example.com
- serverd.lab.example.com
---
service_type: mgr
placement:
hosts:
- clienta.lab.example.com
- serverc.lab.example.com
- serverd.lab.example.com
- servere.lab.example.com
---
service_type: osd
service_id: default_drive_group
placement:
host_pattern: 'server*'
data_devices:
paths:
- /dev/vdb
- /dev/vdc
- /dev/vdd
```

3. As the `root` user on the `serverc` host, bootstrap the Ceph cluster using the created service specification file.

- 3.1. Set the Ceph dashboard password to redhat and use the `--dashboard-password-noupdate` option. Use the `--allow-fqdn-hostname` to use fully qualified domain names for the hosts. The registry URL is `registry.lab.example.com`, the username is `registry`, and the password is `redhat`.
- 3.2. As the `root` user on the `serverc` host, run the `cephadm bootstrap` command with the provided parameters to bootstrap the Ceph cluster. Use the created service specification file.

```
[root@serverc ceph]# cephadm bootstrap --mon-ip=172.25.250.12 \
--apply-spec=initial-config-primary-c luster.yaml \
--initial-dashboard-password=redhat \
--dashboard-password-noupdate \
--allow-fqdn-hostname \
--registry-url=registry.lab.example.com \
--registry-username=registry \
--registry-password=redhat
...output omitted...
Ceph Dashboard is now available at:
```

URL: `https://serverc.lab.example.com:8443/`
User: admin
Password: redhat

```
Applying initial-config-primary-cluster.yaml to cluster
Adding ssh key to clienta.lab.example.com
Adding ssh key to serverd.lab.example.com
Adding ssh key to servere.lab.example.com
Added host 'clienta.lab.example.com' with addr '172.25.250.10'
Added host 'serverc.lab.example.com' with addr '172.25.250.12'
Added host 'serverd.lab.example.com' with addr '172.25.250.13'
Added host 'servere.lab.example.com' with addr '172.25.250.14'
Scheduled mon update...
Scheduled rgw.realm.zone update...
Scheduled mgr update...
Scheduled osd.default_drive_group update...
```

You can access the Ceph CLI with:

```
sudo /sbin/cephadm shell --fsid cd6a42ce-36f6-11ec-8c67-52540000fa0c -c /etc/
ceph/ceph.conf -k /etc/ceph/ceph.client.admin.keyring
```

Please consider enabling telemetry to help improve Ceph:

`ceph telemetry on`

For more information see:

<https://docs.ceph.com/docs/pacific/mgr/telemetry/>

Bootstrap complete.

- 3.3. As the `root` user on the `serverc` host, run the `cephadm shell`.

```
[root@serverc ceph]# cephadm shell
```

- 3.4. Verify that the cluster status is **HEALTH_OK**. Wait until the cluster reaches the **HEALTH_OK** status.

```
[ceph: root@serverc /]# ceph status
cluster:
  id:      cd6a42ce-36f6-11ec-8c67-52540000fa0c
  health: HEALTH_OK

  services:
    mon: 1 daemons, quorum serverc.lab.example.com (age 2m)
    mgr: serverc.lab.example.com.anabtp(active, since 91s), standbys:
    clienta.trffqp
    osd: 9 osds: 9 up (since 21s), 9 in (since 46s)

  data:
    pools:   1 pools, 1 pgs
    objects: 0 objects, 0 B
    usage:   47 MiB used, 90 GiB / 90 GiB avail
    pgs:     1 active+clean
```

4. Label the **clienta** host as the admin node. Manually copy the **ceph.conf** and **ceph.client.admin.keyring** files to the admin node. On the admin node, test the **cephadm** shell.

- 4.1. Label the **clienta** host as the admin node.

```
[ceph: root@serverc /]# ceph orch host label add clienta.lab.example.com _admin
Added label _admin to host clienta.lab.example.com
```

- 4.2. Copy the **ceph.conf** and **ceph.client.admin.keyring** files from the **serverc** host to the **clienta** host. Locate these files in **/etc/ceph** on both hosts.

```
[ceph: root@serverc /]# exit
exit
[root@serverc ceph]# cd /etc/ceph
[root@serverc ceph]# scp {ceph.client.admin.keyring,ceph.conf} \
root@clienta:/etc/ceph/
Warning: Permanently added 'clienta' (ECDSA) to the list of known hosts.
ceph.client.admin.keyring              100%   63   105.6KB/s  00:00
ceph.conf                            100%  177   528.3KB/s  00:00
```

- 4.3. On the admin node, test the **cephadm** shell.

```
[root@serverc ceph]# exit
logout
[admin@serverc ~]$ exit
Connection to serverc closed.
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
```

```
Inferring fsid cd6a42ce-36f6-11ec-8c67-52540000fa0c
Inferring config /var/lib/ceph/cd6a42ce-36f6-11ec-8c67-52540000fa0c/mon.clienta/
config
Using recent ceph image registry.redhat.io/rhceph/rhceph-5-
rhel8@sha256:6306...47ff
[ceph: root@clienta /]# ceph health
HEALTH_OK
```

5. Manually add OSDs to the **servere** node using devices **/dev/vde** and **/dev/vdf**. Set **172.25.250.0/24** for the OSD public network and **172.25.249.0/24** for the OSD cluster network.

- 5.1. Display the **servere** node storage device inventory on the Ceph cluster. Verify that the **/dev/vde** and **/dev/vdf** devices are available.

```
[ceph: root@clienta /]# ceph orch device ls --hostname=servere.lab.example.com
Hostname          Path      Type  Serial           Size   Health
Ident  Fault Available
servere.lab.example.com  /dev/vde  hdd   4d212d34-e5a0-4347-9  10.7G Unknown N/A
      N/A   Yes
servere.lab.example.com  /dev/vdf  hdd   d86b1a78-10b5-46af-9  10.7G Unknown N/A
      N/A   Yes
servere.lab.example.com  /dev/vdb  hdd   1880975e-c78f-4347-8  10.7G Unknown N/A
      N/A   No
servere.lab.example.com  /dev/vdc  hdd   2df15dd0-8eb6-4425-8  10.7G Unknown N/A
      N/A   No
servere.lab.example.com  /dev/vdd  hdd   527656ac-8c51-47b2-9  10.7G Unknown N/A
      N/A   No
```

- 5.2. Create the OSDs using the **/dev/vde** and **/dev/vdf** devices on the **servere** node.

```
[ceph: root@clienta /]# ceph orch daemon add osd servere.lab.example.com:/dev/vde
Created osd(s) 9 on host 'servere.lab.example.com'
[ceph: root@clienta /]# ceph orch daemon add osd servere.lab.example.com:/dev/vdf
Created osd(s) 10 on host 'servere.lab.example.com'
```

- 5.3. For the OSD options, set **public_network** to **172.25.250.0/24** and **cluster_network** to **172.25.249.0/24**.

```
[ceph: root@clienta /]# ceph config set osd public_network 172.25.250.0/24
[ceph: root@clienta /]# ceph config set osd cluster_network 172.25.249.0/24
```

- 5.4. Return to **workstation** as the **student** user.

```
[ceph: root@clienta /]# exit
[admin@clienta ~]$ exit
[student@workstation ~]$
```

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-review1
```

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-review1
```

This concludes the lab.

▶ Lab

Configuring Red Hat Ceph Storage

In this review, you configure a Red Hat Ceph Storage cluster using specified requirements.

Outcomes

You should be able to configure cluster settings and components, such as pools, users, OSDs, and the CRUSH map.

Before You Begin

If you did not reset your classroom virtual machines at the end of the last chapter, save any work you want to keep from earlier exercises on those machines and reset the classroom environment now.



Important

Reset your environment before performing this exercise. All comprehensive review labs start with a clean, initial classroom environment that includes a pre-built, fully operational Ceph cluster. All remaining comprehensive reviews use the default Ceph cluster provided in the initial classroom environment.

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

This script ensures that all cluster hosts are reachable.

```
[student@workstation ~]$ lab start comprehensive-review2
```

Specifications

- Set the value of `osd_pool_default_pg_num` to 250 in the configuration database.
- Create a CRUSH rule called `onhdd` to target HDD-based OSDs for replicated pools.
- Create a replicated pool called `rbd1` that uses the `onhdd` CRUSH map rule. Set the application type to `rbd` and the number of replicas for the objects in this pool to five.
- Create the following CRUSH hierarchy. Do not associate any OSD with this new tree.

```
default-4-lab      (root bucket)
    DC01          (datacenter bucket)
        firstfloor (room bucket)
            hostc   (host bucket)
        secondfloor (room bucket)
            hostd   (host bucket)
```

- Create a new erasure code profile called `cl260`. Pools using this profile must set two data chunks and one coding chunk per object.

- Create an erasure coded pool called `testec` that uses your new `cl260` profile. Set its application type to `rgw`.
- Create a user called `client.fortestec` that can store and retrieve objects under the `docs` namespace in the pool called `testec`. This user must not have access to any other pool or namespace. Save the associated key-ring file as `/etc/ceph/ceph.client.fortestec.keyring` on `clienta`.
- Upload the `/usr/share/doc/ceph/sample.ceph.conf` file as an object called `report` under the `docs` namespace in the pool called `testec`.
- Update the OSD near-capacity limit information for the `serverc`, `serverd`, and `servere` cluster. Set the `full` ratio to 90% and the `near-full` ratio to 86%.
- Locate the host on which the `ceph-osd-7` service is running. List the available storage devices on that host.

Evaluation

Grade your work by running the `lab grade comprehensive-review2` command from your workstation machine. Correct any reported failures and rerun the script until successful.

```
[student@workstation ~]$ lab grade comprehensive-review2
```

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-review2
```

This concludes the lab.

► Solution

Configuring Red Hat Ceph Storage

In this review, you configure a Red Hat Ceph Storage cluster using specified requirements.

Outcomes

You should be able to configure cluster settings and components, such as pools, users, OSDs, and the CRUSH map.

Before You Begin

If you did not reset your classroom virtual machines at the end of the last chapter, save any work you want to keep from earlier exercises on those machines and reset the classroom environment now.



Important

Reset your environment before performing this exercise. All comprehensive review labs start with a clean, initial classroom environment that includes a pre-built, fully operational Ceph cluster. All remaining comprehensive reviews use the default Ceph cluster provided in the initial classroom environment.

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

This script ensures that all cluster hosts are reachable.

```
[student@workstation ~]$ lab start comprehensive-review2
```

- Set the value of `osd_pool_default_pg_num` to 250 in the configuration database.

- Log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]#
```

- Set the value of `osd_pool_default_pg_num` to 250 in the configuration database.

```
[ceph: root@clienta /]# ceph config set mon osd_pool_default_pg_num 250
```

- Verify the setting.

```
[ceph: root@clienta /]# ceph config get mon osd_pool_default_pg_num
250
[ceph: root@clienta /]# ceph config dump | grep osd_pool_default_pg_num
mon                               advanced  osd_pool_default_pg_num
250
```

2. Create a CRUSH rule called onhdd to target HDD-based OSDs for replicated pools.

- 2.1. Create a new rule called onhdd to target HDD-based OSDs for replicated pools.

```
[ceph: root@clienta /]# ceph osd crush rule create-replicated onhdd default \
host hdd
```

- 2.2. Verify that the new rule exists.

```
[ceph: root@clienta /]# ceph osd crush rule ls
replicated_rule
onhdd
```

3. Create a replicated pool called rbd1 that uses the onhdd CRUSH map rule. Set the application type to rbd and the number of replicas for the objects in this pool to five.

- 3.1. Create a new replicated pool called rbd1 that uses the onhdd CRUSH map rule.

```
[ceph: root@clienta /]# ceph osd pool create rbd1 onhdd
pool 'rbd1' created
```

- 3.2. Set rbd as the application type for the pool.

```
[ceph: root@clienta /]# ceph osd pool application enable rbd1 rbd
enabled application 'rbd' on pool 'rbd1'
```

- 3.3. Increase the number of replicas for the pool to five and verify the new value.

```
[ceph: root@clienta /]# ceph osd pool set rbd1 size 5
set pool 6 size to 5
[ceph: root@clienta /]# ceph osd pool ls detail
...output omitted...
pool 6 'rbd1' replicated size 5 min_size 3 crush_rule 1 object_hash rjenkins
pg_num 250 pgp_num 250 autoscale_mode on last_change 235 flags hashpspool
stripe_width 0 application rbd
```

4. Create the following CRUSH hierarchy. Do not associate any OSD with this new tree.

4.1.

```
default-4-lab      (root bucket)
    DC01          (datacenter bucket)
        firstfloor   (room bucket)
            hostc     (host bucket)
        secondfloor   (room bucket)
            hostd     (host bucket)
```

4.2. Create the buckets.

```
[ceph: root@clienta /]# ceph osd crush add-bucket default-4-lab root
added bucket default-4-lab type root to crush map
[ceph: root@clienta /]# ceph osd crush add-bucket DC01 datacenter
added bucket DC01 type datacenter to crush map
[ceph: root@clienta /]# ceph osd crush add-bucket firstfloor room
added bucket firstfloor type room to crush map
[ceph: root@clienta /]# ceph osd crush add-bucket hostc host
added bucket hostc type host to crush map
[ceph: root@clienta /]# ceph osd crush add-bucket secondfloor room
added bucket secondfloor type room to crush map
[ceph: root@clienta /]# ceph osd crush add-bucket hostd host
added bucket hostd type host to crush map
```

4.3. Build the hierarchy.

```
[ceph: root@clienta /]# ceph osd crush move DC01 root=default-4-lab
moved item id -10 name 'DC01' to location {root=default-4-lab} in crush map
[ceph: root@clienta /]# ceph osd crush move firstfloor datacenter=DC01
moved item id -11 name 'firstfloor' to location {datacenter=DC01} in crush map
[ceph: root@clienta /]# ceph osd crush move hostc room=firstfloor
moved item id -12 name 'hostc' to location {room=firstfloor} in crush map
[ceph: root@clienta /]# ceph osd crush move secondfloor datacenter=DC01
moved item id -13 name 'secondfloor' to location {datacenter=DC01} in crush map
[ceph: root@clienta /]# ceph osd crush move hostd room=secondfloor
moved item id -14 name 'hostd' to location {room=secondfloor} in crush map
```

4.4. Display the CRUSH map tree to verify the new hierarchy.

```
[ceph: root@clienta /]# ceph osd crush tree
ID  CLASS  WEIGHT  TYPE NAME
-9       0  root default-4-lab
-10      0  datacenter DC01
-11      0      room firstfloor
-12      0      host hostc
-13      0      room secondfloor
-14      0      host hostd
-1      0.08817  root default
-3      0.02939  host serverc
  0  hdd  0.00980      osd.0
  1  hdd  0.00980      osd.1
  2  hdd  0.00980      osd.2
-7      0.02939  host serverd
```

```
3  hdd  0.00980      osd.3
5  hdd  0.00980      osd.5
7  hdd  0.00980      osd.7
-5          0.02939    host servere
4  hdd  0.00980      osd.4
6  hdd  0.00980      osd.6
8  hdd  0.00980      osd.8
```

5. Create a new erasure code profile called cl260. Pools that use this profile must set two data chunks and one coding chunk per object.

- 5.1. Create a new erasure code profile called cl260.

```
[ceph: root@clienta /]# ceph osd erasure-code-profile set cl260 k=2 m=1
```

- 5.2. Verify the new erasure code profile parameters.

```
[ceph: root@clienta /]# ceph osd erasure-code-profile get cl260
crush-device-class=
crush-failure-domain=host
crush-root=default
jerasure-per-chunk-alignment=false
k=2
m=1
plugin=jerasure
technique=reed_sol_van
w=8
```

6. Create an erasure coded pool called testec that uses your new cl260 profile. Set its application type to rgw.

- 6.1. Create an erasure coded pool called testec that uses the cl260 profile.

```
[ceph: root@clienta /]# ceph osd pool create testec erasure cl260
pool 'testec' created
```

- 6.2. Set rgw as the application type for the pool.

```
[ceph: root@clienta /]# ceph osd pool application enable testec rgw
enabled application 'rgw' on pool 'testec'
```

- 6.3. List the new pool parameters.

```
[ceph: root@clienta /]# ceph osd pool ls detail
...output omitted...
pool 7 'testec' erasure profile cl260 size 3 min_size 2 crush_rule 2 object_hash
jenkins pg_num 250 pgp_num 250 autoscale_mode on last_change 309 flags
hashpspool stripe_width 8192 application rgw
```

7. Create a user called client.fortestec that can store and retrieve objects under the docs namespace in the pool called testec. This user must not have access to any other pool or namespace. Save the associated key-ring file as /etc/ceph/ceph.client.fortestec.keyring on clienta.

- 7.1. Exit from the current cephadm shell. Start a new cephadm shell with the /etc/ceph directory as a bind mount.

```
[ceph: root@clienta /]# exit
exit
[admin@clienta ~]$ sudo cephadm shell --mount /etc/ceph:/etc/ceph
```

- 7.2. Create a user called `client.fortestec`, with read and write capabilities in the namespace `docs` within the pool `testec`. Save the associated key-ring file as `/etc/ceph/ceph.client.fortestec.keyring` in the mounted directory.

```
[ceph: root@clienta /]# ceph auth get-or-create client.fortestec mon 'allow r' \
osd 'allow rw pool=test ec namespace=docs' \
-o /etc/ceph/ceph.client.fortestec.keyring
```

- 7.3. To verify your work, attempt to store and retrieve an object. The `diff` command returns no output when the file contents are the same. When finished, remove the object.

```
[ceph: root@clienta /]# rados --id fortestec -p testec -N docs \
put testdoc /etc/services
[ceph: root@clienta /]# rados --id fortestec -p testec -N docs \
get testdoc /tmp/test
[ceph: root@clienta /]# diff /etc/services /tmp/test
[ceph: root@clienta /]# rados --id fortestec -p testec -N docs rm testdoc
```

8. Upload the `/usr/share/doc/ceph/sample.ceph.conf` file as an object called `report` under the `docs` namespace in the pool called `testec`.

- 8.1. Use the `rados` command to upload the `/usr/share/doc/ceph/sample.ceph.conf` file.

```
[ceph: root@clienta ~]# rados --id fortestec -p testec -N docs \
put report /usr/share/doc/ceph/sample.ceph.conf
```

- 8.2. Obtain `report` object details to confirm that the upload was successful.

```
[ceph: root@clienta ~]# rados --id fortestec -p testec -N docs stat report
testec/report mtime 2021-10-29T11:44:21.000000+0000, size 19216
```

9. Update the OSD near-capacity limit information for the cluster. Set the `full` ratio to 90% and the `nearfull` ratio to 86%.

- 9.1. Set the `full_ratio` parameter to 0.9 (90%) and the `nearfull_ratio` to 0.86 (86%) in the OSD map.

```
[ceph: root@clienta ~]# ceph osd set-full-ratio 0.9
osd set-full-ratio 0.9
[ceph: root@clienta ~]# ceph osd set-nearfull-ratio 0.86
osd set-nearfull-ratio 0.86
```

- 9.2. Dump the OSD map and verify the new value of the two parameters.

```
[ceph: root@clienta ~]# ceph osd dump | grep ratio
full_ratio 0.9
backfillfull_ratio 0.9
nearfull_ratio 0.86
```

10. Locate the host with the OSD 7 service. List that host's available storage devices.

- 10.1. Locate the OSD 7 service. The location of the OSD 7 service might be different in your lab environment.

```
[ceph: root@clienta ~]# ceph osd find osd.7
{
    "osd": 7,
    "addrs": [
        "addrvec": [
            {
                "type": "v2",
                "addr": "172.25.250.13:6816",
                "nonce": 1160376750
            },
            {
                "type": "v1",
                "addr": "172.25.250.13:6817",
                "nonce": 1160376750
            }
        ]
    },
    "osd_fsid": "53f9dd65-430a-4e5a-a2f6-536c5453f02a",
    "host": "serverd.lab.example.com",
    "crush_location": {
        "host": "serverd",
        "root": "default"
    }
}
```

- 10.2. Use the `ceph orch device ls` command to list the available storage devices on the located host. Use the host you located in your environment.

```
[ceph: root@clienta ~]# ceph orch device ls --hostname=serverd.lab.example.com
Hostname          Path   Type  Serial           Size  Health
Ident  Fault Available
serverd.lab.example.com /dev/vde hdd  65d5b32d-594c-4dbe-b 10.7G Unknown N/A
      N/A   Yes
serverd.lab.example.com /dev/vdf hdd  63124a05-de2b-434f-8 10.76 Unknown N/A
      N/A   Yes
serverd.lab.example.com /dev/vdb hdd  e20fad81-6237-409e-9 10.7G Unknown N/A
      N/A   No
serverd.lab.example.com /dev/vdc hdd  6dad9f98-2a5a-4aa8-b 10.76 Unknown N/A
      N/A   No
serverd.lab.example.com /dev/vdd hdd  880d431d-15f3-4c20-b 10.7G Unknown N/A
      N/A   No
```

10.3. Return to `workstation` as the `student` user.

```
[ceph: root@clienta /]# exit  
[admin@clienta ~]$ exit  
[student@workstation ~]$
```

Evaluation

Grade your work by running the `lab grade comprehensive-review2` command from your `workstation` machine. Correct any reported failures and rerun the script until successful.

```
[student@workstation ~]$ lab grade comprehensive-review2
```

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-review2
```

This concludes the lab.

▶ Lab

Deploying CephFS

In this review, you will deploy CephFS on an existing Red Hat Ceph Storage cluster using specified requirements.

Outcomes

You should be able to deploy a Metadata Server, provide storage with CephFS, and configure clients for its use.

Before You Begin

If you did not reset your classroom virtual machines at the end of the last chapter, save any work you want to keep from earlier exercises on those machines and reset the classroom environment now.



Important

Reset your environment before performing this exercise. All comprehensive review labs start with a clean, initial classroom environment that includes a pre-built, fully operational Ceph cluster. All remaining comprehensive reviews use the default Ceph cluster provided in the initial classroom environment.

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

```
[student@workstation ~]$ lab start comprehensive-review3
```

This command ensures that all cluster hosts are reachable.

Specifications

- Create a CephFS file system **c1260-fs**. Create an MDS service called **c1260-fs** with two MDS instances, one on the **serverc** node and another on the **serverd** node. Create a data pool called **cephfs.c1260-fs.data** and a metadata pool called **cephfs.c1260-fs.meta**. Use replicated as the type for both pools.
- Mount the CephFS file system to the **/mnt/cephfs** directory on the **clienta** host and owned by the **admin** user. Save the **client.admin** key-ring to the **/root/secretfile** and use the file to authenticate the mount operation.
- Create the **ceph01** and **ceph02** directories. Create an empty file called **firstfile** in the **ceph01** directory. Verify the directories and its contents are owned by the **admin** user.
- Modify the **ceph.dir.layout.stripe_count** layout attribute for the **/mnt/cephfs/dir1** directory. Verify that new files created with the directory inherit the attribute.
- Use the **ceph-fuse** client to mount a new directory called **/mnt/cephfuse**.

- Configure the CephFS file system to be mounted on each system startup. Verify that the /etc/fstab file is updated accordingly.

Evaluation

Grade your work by running the `lab grade comprehensive-review3` command from your **workstation** machine. Correct any reported failures and rerun the script until successful.

```
[student@workstation ~]$ lab grade comprehensive-review3
```

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-review3
```

This concludes the lab.

► Solution

Deploying CephFS

In this review, you will deploy CephFS on an existing Red Hat Ceph Storage cluster using specified requirements.

Outcomes

You should be able to deploy a Metadata Server, provide storage with CephFS, and configure clients for its use.

Before You Begin

If you did not reset your classroom virtual machines at the end of the last chapter, save any work you want to keep from earlier exercises on those machines and reset the classroom environment now.



Important

Reset your environment before performing this exercise. All comprehensive review labs start with a clean, initial classroom environment that includes a pre-built, fully operational Ceph cluster. All remaining comprehensive reviews use the default Ceph cluster provided in the initial classroom environment.

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

```
[student@workstation ~]$ lab start comprehensive-review3
```

This command ensures that all cluster hosts are reachable.

1. Create a CephFS file system **cl260-fs**. Create an MDS service called **cl260-fs** with an MDS instance on **serverc** and another on **serverd**. Create a data pool called **cephfs.cl260-fs.data** and a metadata pool called **cephfs.cl260-fs.meta**. Use **replicated** as the type for both pools. Verify that the MDS service is up and running.
 - 1.1. Log in to **clienta** and use **sudo** to run the **cephadm** shell.

```
[student@workstation ~]$ ssh admin@clienta
[admin@clienta ~]$ sudo cephadm shell
[ceph: root@clienta /]#
```

- 1.2. Create a data pool called **cephfs.cl260-fs.data** and a metadata pool called **cephfs.cl260-fs.meta** for the CephFS service.

```
[ceph: root@clienta /]# ceph osd pool create cephfs.cl260-fs.data
pool 'cephfs.cl260-fs.data' created
[ceph: root@clienta /]# ceph osd pool create cephfs.cl260-fs.meta
pool 'cephfs.cl260-fs.meta' created
[ceph: root@clienta /]# ceph osd pool ls
...output omitted...
cephfs.cl260-fs.data
cephfs.cl260-fs.meta
```

- 1.3. Create a CephFS file system called cl260-fs.

```
[ceph: root@clienta /]# ceph fs new cl260-fs cephfs.cl260-fs.data \
cephfs.cl260-fs.meta
new fs with metadata pool 14 and data pool 15
[ceph: root@clienta /]# ceph fs ls
name: cl260-fs, metadata pool: cephfs.cl260-fs.data, data pools: [cephfs.cl260-
fs.meta ]
```

- 1.4. Create an MDS service called cl260-fs with an MDS instance on serverc.

```
[ceph: root@clienta /]# ceph orch apply mds cl260-fs \
--placement="2 serverc.lab.example.com serverd.lab.example.com"
Scheduled mds.cl260-fs update...
[ceph: root@clienta /]# ceph orch ps --daemon-type mds
NAME HOST STATUS REFRESHED ...
mds.cl260-fs.serverc.iuwzrt serverc.lab.example.com running (53s) 46s ago ...
mds.cl260-fs.serverd.lapeyj serverd.lab.example.com running (50s) 47s ago ...
```

- 1.5. Verify that the MDS service is up and running.

```
[ceph: root@clienta /]# ceph mds stat
cl260-fs:1 {0=cl260-fs.serverd.lapeyj=up:active} 1 up:standby
[ceph: root@clienta /]# ceph status
cluster:
  id:      2ae6d05a-229a-11ec-925e-52540000fa0c
  health: HEALTH_OK

  services:
  ...output omitted...
  mds: 1/1 daemons up, 1 standby
  ...output omitted...
```

2. Install the `ceph-common` package. Mount the CephFS file system to the `/mnt/cephfs` directory on the `clienta` host. Save the key-ring associated with the `client.admin` user to the `/root/secretfile` file. Use this file to authenticate the mount operation. Verify that the `/mnt/cephfs` directory is owned by the `admin` user.

- 2.1. Exit the `cephadm` shell and switch to the `root` user. Install the `ceph-common` package.

```
[ceph: root@clienta /]# exit  
exit  
[admin@clienta ~]$ sudo -i  
[root@clienta ~]# yum install -y ceph-common  
...output omitted...
```

- 2.2. Extract the key-ring associated with the `client.admin` user, and save it in the `/root/secretfile` file.

```
[root@clienta ~]# ceph auth get-key client.admin | tee /root/secretfile  
...output omitted...
```

- 2.3. Create a new directory called `/mnt/cephfs` to use as a mount point for the CephFS file system. Mount your new CephFS file system on that directory.

```
[root@clienta ~]# mkdir /mnt/cephfs  
[root@clienta ~]# mount -t ceph serverc:/ /mnt/cephfs \  
-o name=admin,secretfile=/root/secretfile
```

- 2.4. Verify the mount.

```
[root@clienta ~]# df /mnt/cephfs  
Filesystem 1K-blocks Used Available Use% Mounted on  
172.25.250.12:/ 29790208 0 29790208 0% /mnt/cephfs
```

- 2.5. Change the ownership of the top-level directory of the mounted file system to user and group `admin`.

```
[root@clienta ~]# chown admin:admin /mnt/cephfs
```

3. As the `admin` user, create the `ceph01` and `ceph02` directories. Create an empty file called `firstfile` on the `ceph01` directory. Ensure the directories and its contents are owned by the `admin` user.
 - 3.1. Exit the `root` user session. Create two directories directly underneath the mount point, and name them `dir1` and `dir2`.

```
[root@clienta ~]# exit  
exit  
[admin@clienta ~]$ mkdir /mnt/cephfs/ceph01  
[admin@clienta ~]$ mkdir /mnt/cephfs/ceph02
```

- 3.2. Create an empty file called `firstfile` in the `ceph01` directory.

```
[admin@clienta ~]$ touch /mnt/cephfs/ceph01/firstfile
```

4. Set the `ceph.dir.layout.stripe_count` layout attribute to 2 for files created on the `/mnt/cephfs/ceph01` directory. Create a 10 MB file called `secondfile` on the `/mnt/cephfs/ceph01` directory with the new layout attribute.

- 4.1. Verify the current layout of the /mnt/cephfs/ceph01 directory.

```
[admin@clienta ~]$ getfattr -n ceph.dir.layout /mnt/cephfs/ceph01
/mnt/cephfs/ceph01: ceph.dir.layout: No such attribute
```

- 4.2. Set the ceph.dir.layout.stripe_count layout attribute to 2 for the /mnt/cephfs/ceph01 directory.

```
[admin@clienta ~]$ setfattr -n ceph.dir.layout.stripe_count -v 2 \
/mnt/cephfs/ceph01
```

- 4.3. Create a 10 MB file called secondfile in the /mnt/cephfs/ceph01 directory.

```
[admin@clienta ~]$ dd if=/dev/zero of=/mnt/cephfs/ceph01/secondfile \
bs=1024 count=10000
10000+0 records in
10000+0 records out
10240000 bytes (10 MB, 9.8 MiB) copied, 0.0391504 s, 262 MB/s
```

- 4.4. Verify that the secondfile file has the correct layout attribute.

```
[admin@clienta ~]$ getfattr -n ceph.file.layout /mnt/cephfs/ceph01/secondfile
getfattr: Removing leading '/' from absolute path names
# file: mnt/cephfs/ceph01/secondfile
ceph.file.layout="stripe_unit=4194304 stripe_count=2 object_size=4194304
pool=cephfs.cl260-fs.meta"
```

5. Switch to the root user. Install and use the ceph-fuse client to mount a new directory called cephfuse.

- 5.1. Switch to the root user. Install the ceph-fuse package.

```
[admin@clienta ~]$ sudo -i
[root@clienta ~]# yum install -y ceph-fuse
...output omitted...
```

- 5.2. Create a directory called /mnt/cephfuse to use as a mount point for the Fuse client. Mount your new Ceph-Fuse file system on that directory.

```
[root@clienta ~]# mkdir /mnt/cephfuse
[root@clienta ~]# ceph-fuse -m serverc: /mnt/cephfuse/
2021-11-01T20:18:33.004-0400 7f14907ea200 -1 init, newargv = 0x5621fed5f550
newargc=15
ceph-fuse[48516]: starting ceph client
ceph-fuse[48516]: starting fuse
```

- 5.3. View the contents of the /mnt directory.

```
[root@clienta ~]# tree /mnt
/mnt
|-- cephfs
```

```
|   |-- ceph01
|   |   |-- firstfile
|   |   `-- secondfile
|   '-- ceph02
`-- cephfuse
    |-- ceph01
    |   |-- firstfile
    |   `-- secondfile
    '-- ceph02

6 directories, 4 files
```

6. Configure the CephFS file system to be persistently mounted at startup. Use the contents of the /root/secretfile file to configure the mount operation in the /etc/fstab file. Verify that the configuration works as expected by using the `mount -a` command.
- 6.1. View the contents of the admin key-ring in the /root/secretfile file.

```
[root@clienta ~]# cat /root/secretfile
AQAA11VZhyq8VGRAA0us0I5xLwMSdAw/759e32A==
```

- 6.2. Configure the /etc/fstab file to mount the file system at startup. The /etc/fstab file should look like the following output.

```
[root@clienta ~]# cat /etc/fstab
...output omitted...
serverc:/ /mnt/cephfs ceph
    rw,seclabel,relatime,name=admin,secret=AQAA11VZhyq8VGRAA0us0I5xLwMSdAw/759e32A==,acl
    0 0
```

- 6.3. Unmount the CephFS file system, then test mount using the `mount -a` command. Verify the mount.

```
[root@clienta ~]# umount /mnt/cephfs
[root@clienta ~]# mount -a
[root@clienta ~]# df /mnt/cephfs/
Filesystem      1K-blocks  Used Available Use% Mounted on
172.25.250.12:/  29773824 12288  29761536   1% /mnt/cephfs
```

- 6.4. Return to workstation as the student user.

```
[root@clienta ~]# exit
[admin@clienta ~]$ exit
[student@workstation ~]$
```

Evaluation

Grade your work by running the `lab grade comprehensive-review3` command from your workstation machine. Correct any reported failures and rerun the script until successful.

```
[student@workstation ~]$ lab grade comprehensive-review3
```

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-review3
```

This concludes the lab.

▶ Lab

Deploying and Configuring Block Storage with RBD

In this review, you will configure a Red Hat Ceph Storage cluster for RBD using specified requirements.

Outcomes

You should be able to:

- Deploy and configure Red Hat Ceph Storage for RBD mirroring.
- Configure a client to access RBD images.
- Manage RBD images, RBD mirroring, and RBD snapshots and clones.

Before You Begin

If you did not reset your classroom virtual machines at the end of the last chapter, save any work you want to keep from earlier exercises on those machines and reset the classroom environment now.



Important

Reset your environment before performing this exercise. All comprehensive review labs start with a clean, initial classroom environment that includes a pre-built, fully operational Ceph cluster. All remaining comprehensive reviews use the default Ceph cluster provided in the initial classroom environment.

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

```
[student@workstation ~]$ lab start comprehensive-review4
```

This command ensures that production and backups clusters are running and have the RBD storage pools called `rbd`, `rbdpoolmode`, and `rbdimage mode` in both clusters, also creates the `data` image in the `rbd` pool in the production cluster.

Specifications

- Deploy and configure a Red Hat Ceph Storage cluster for RBD mirroring between two clusters:
 - In the production cluster, create an RBD image called `vm1` in the `rbdpoolmode` pool configured as one-way `pool-mode` and with a size of 128 MiB. Create an RBD image called `vm2` in the `rbdimage mode` pool configured as one-way `image-mode` and with a size of 128 MiB. Both images should be enabled for mirroring.
 - Production and backup clusters should be called `production` and `bck`, respectively.
 - Map the image called `rbd/data` using the kernel RBD client on `clienta` and format the device with an XFS file system. Store a copy of the `/usr/share/dict/words` at the root of

the file system. Create a snapshot called `beforeprod` of the RBD image data, and create a clone called `prod1` from the snapshot called `beforeprod`.

- Export the image called `data` to the `/home/admin/cr4/data.img` file. Import it as an image called `data` to the `rbdimage mode` pool. Create a snapshot called `beforeprod` of the new `data` image in the `rbdimage mode` pool.
- Map again the image called `rbd/data` using the kernel RBD client on `clienta`. Copy the `/etc/services` file to the root of the file system. Export changes to the `rbd/data` image to the `/home/admin/cr4/data-diff.img` file.
- Configure the `clienta` node so that it will persistently mount the `rbd/data` RBD image as `/mnt/data`.

Evaluation

Grade your work by running the `lab grade comprehensive-review4` command from your `workstation` machine. Correct any reported failures and rerun the script until successful.

```
[student@workstation ~]$ lab grade comprehensive-review4
```

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-review4
```

This concludes the lab.

► Solution

Deploying and Configuring Block Storage with RBD

In this review, you will configure a Red Hat Ceph Storage cluster for RBD using specified requirements.

Outcomes

You should be able to:

- Deploy and configure Red Hat Ceph Storage for RBD mirroring.
- Configure a client to access RBD images.
- Manage RBD images, RBD mirroring, and RBD snapshots and clones.

Before You Begin

If you did not reset your classroom virtual machines at the end of the last chapter, save any work you want to keep from earlier exercises on those machines and reset the classroom environment now.



Important

Reset your environment before performing this exercise. All comprehensive review labs start with a clean, initial classroom environment that includes a pre-built, fully operational Ceph cluster. All remaining comprehensive reviews use the default Ceph cluster provided in the initial classroom environment.

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

```
[student@workstation ~]$ lab start comprehensive-review4
```

This command ensures that production and backups clusters are running and have the RBD storage pools called `rbd`, `rbdpoolmode`, and `rbdimageemode` in both clusters, also creates the `data` image in the `rbd` pool in the production cluster.

1. Using two terminals, log in to `clienta` for the production cluster and `serverf` for the backup cluster as the `admin` user. Verify that each cluster is reachable and has a `HEALTH_OK` status.
 - 1.1. In the first terminal, log in to `clienta` as the `admin` user and use `sudo` to run the `cephadm` shell. Verify the health of the production cluster.

```
[student@workstation ~]$ ssh admin@clienta
...output omitted...
[admin@clienta ~]$ sudo cephadm shell
...output omitted...
[ceph: root@clienta /]# ceph health
HEALTH_OK
```

- 1.2. In the second terminal, log in to `serverf` as `admin` and use `sudo` to run the `cephadm` shell. Verify the health of the backup cluster. Exit from the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@serverf
...output omitted...
[admin@serverf ~]$ sudo cephadm shell
...output omitted...
[ceph: root@serverf /]# ceph health
HEALTH_OK
[ceph: root@serverf /]# exit
[admin@serverf ~]$
```

2. In the production cluster, create the `rbdpoolmode/vm1` RBD image, enable one-way pool-mode mirroring on the pool, and view the image information.
- 2.1. Create an RBD image called `vm1` in the `rbdpoolmode` pool in the production cluster. Specify a size of 128 megabytes, enable `exclusive-lock`, and `journaling` RBD image features.

```
[ceph: root@clienta /]# rbd create vm1 \
--size 128 \
--pool rbdpoolmode \
--image-feature=exclusive-lock,journaling
```

- 2.2. Enable pool-mode mirroring on the `rbdpoolmode` pool.

```
[ceph: root@clienta /]# rbd mirror pool enable rbdpoolmode pool
```

- 2.3. View the `vm1` image information. Exit from the `cephadm` shell.

```
[ceph: root@clienta /]# rbd info --pool rbdpoolmode vm1
rbd image 'vm1':
  size 128 MiB in 32 objects
  order 22 (4 MiB objects)
  snapshot_count: 0
  id: ad7c2dd2d3be
  block_name_prefix: rbd_data.ad7c2dd2d3be
  format: 2
  features: exclusive-lock, journaling
  op_features:
  flags:
  create_timestamp: Tue Oct 26 23:46:28 2021
  access_timestamp: Tue Oct 26 23:46:28 2021
  modify_timestamp: Tue Oct 26 23:46:28 2021
```

```
journal: ad7c2dd2d3be
mirroring state: enabled
mirroring mode: journal
mirroring global id: 6ea4b768-a53d-4195-a1f5-37733eb9af76
mirroring primary: true
[ceph: root@clienta /]# exit
exit
[admin@clienta ~]$
```

3. In the production cluster, run the `cephadm` shell with a bind mount of `/home/admin/cr4/`. Bootstrap the storage cluster peer and create Ceph user accounts, and save the token in the `/home/admin/cr4/pool_token_prod` file in the container. Name the production cluster `prod`. Copy the bootstrap token file to the backup storage cluster.
 - 3.1. In the production cluster, use `sudo` to run the `cephadm` shell with a bind mount of the `/home/admin/cr4/` directory.

```
[admin@clienta ~]$ sudo cephadm shell --mount /home/admin/cr4/
...output omitted...
[ceph: root@clienta /]#
```

- 3.2. Bootstrap the storage cluster peer, and create Ceph user accounts, save the output in the `/mnt/pool_token_prod` file. Name the production cluster `prod`.

```
[ceph: root@clienta /]# rbd mirror pool peer bootstrap create \
--site-name prod rbdpoolmode > /mnt/pool_token_prod
```

- 3.3. Exit the `cephadm` shell. Copy the bootstrap token file to the backup storage cluster in the `/home/admin/cr4/` directory.

```
[ceph: root@clienta /]# exit
exit
[admin@clienta ~]$ sudo rsync -avP /home/admin/cr4/ \
serverf:/home/admin/cr4/
...output omitted...
```

4. In the backup cluster, run the `cephadm` shell with a bind mount of `/home/admin/cr4/`. Deploy an `rbd-mirror` daemon in the `serverf` node. Import the bootstrap token located in the `/home/admin/cr4/` directory. Name the backup cluster `bck`. Verify that the RBD image is present.

- 4.1. In the backup cluster, use `sudo` to run the `cephadm` shell with a bind mount of the `/home/admin/cr4/` directory.

```
[admin@serverf ~]$ sudo cephadm shell --mount /home/admin/cr4/
...output omitted...
[ceph: root@serverf /]#
```

- 4.2. Deploy a `rbd-mirror` daemon, by using the `--placement` option to select the `serverf.lab.example.com` node. Verify the placement.

```
[ceph: root@serverf /]# ceph orch apply rbd-mirror \
--placement=serverf.lab.example.com
Scheduled rbd-mirror update...
```

```
[ceph: root@serverf /]# ceph orch ps --format=yaml --service-name=rbd-mirror
daemon_type: rbd-mirror
daemon_id: serverf.hhunqx
hostname: serverf.lab.example.com
...output omitted...
```

- 4.3. Import the bootstrap token located in /mnt/pool_token_prod. Name the backup cluster bck.

```
[ceph: root@serverf /]# rbd mirror pool peer bootstrap import \
--site-name bck --direction rx-only rbdp oolmode /mnt/pool_token_prod
```



Important

Ignore the known error containing the following text: auth: unable to find a keyring on ...

- 4.4. Verify that the RBD image is present. Wait until the RBD image is displayed.

```
[ceph: root@serverf /]# rbd --pool rbdpoolmode ls
vm1
```

5. In the production cluster, create the rbdimagemode/vm2 RBD image, enable one-way image-mode mirroring on the pool. Also, enable mirroring for the vm2 RBD image in the rbdimagemode pool

- 5.1. In the production cluster, use sudo to run the cephadm shell with a bind mount of the /home/admin/cr4/ directory.

```
[admin@clienta ~]$ sudo cephadm shell --mount /home/admin/cr4/
...output omitted...
[ceph: root@clienta /]#
```

- 5.2. Create an RBD image called vm2 in the rbdimagemode pool in the production cluster. Specify a size of 128 megabytes, enable exclusive-lock, and journaling RBD image features.

```
[ceph: root@clienta /]# rbd create vm2 \
--size 128 \
--pool rbdimagemode \
--image-feature=exclusive-lock,journaling
```

- 5.3. Enable image-mode mirroring on the rbdimagemode pool.

```
[ceph: root@clienta /]# rbd mirror pool enable rbdimagemode image
```

- 5.4. Enable mirroring for the `vm2` RBD image in the `rbdimage mode` pool.

```
[ceph: root@clienta /]# rbd mirror image enable rbdimage mode/vm2
Mirroring enabled
```

6. In the production cluster, bootstrap the storage cluster peer and create Ceph user accounts, and save the token in the `/home/admin/cr4/image_token_prod` file in the container. Copy the bootstrap token file to the backup storage cluster.
- 6.1. Bootstrap the storage cluster peer and create Ceph user accounts, and save the output in the `/mnt/image_token_prod` file.

```
[ceph: root@clienta /]# rbd mirror pool peer bootstrap create \
rbdimage mode > /mnt/image_token_prod
```

- 6.2. Exit from the `cephadm` shell. Copy the bootstrap token file to the backup storage cluster in the `/home/admin/cr4/` directory.

```
[ceph: root@clienta /]# exit
exit
[admin@clienta ~]$ sudo rsync -avP /home/admin/cr4/ \
serverf:/home/admin/cr4/
...output omitted...
```

7. In the backup cluster, import the bootstrap token. Verify that the RBD image is present.

- 7.1. Import the bootstrap token located in `/mnt/image_token_prod`. Name the backup cluster `bck`.

```
[ceph: root@serverf /]# rbd mirror pool peer bootstrap import \
--direction rx-only rbdimage mode /mnt/image_token_prod
```



Important

Ignore the known error containing the following text: auth: unable to find a keyring on ...

- 7.2. Verify that the RBD image is present. Wait until the RBD image appears.

```
[ceph: root@serverf /]# rbd --pool rbdimage mode ls
vm2
```

- 7.3. Return to `workstation` as the `student` user and Exit the second terminal.

```
[ceph: root@serverf /]# exit
exit
[admin@serverf ~]$ exit
[student@workstation ~]$ exit
```

8. In the production cluster, map the image called `rbd/data` using the kernel RBD client on `clienta`. Format the device with an XFS file system. Temporarily mount the file system and

Chapter 14 | Comprehensive Review

store a copy of the /usr/share/dict/words file at the root of the file system. Unmount and unmap the device when done.

- 8.1. Map the data image in the rbd pool using the kernel RBD client.

```
[admin@clienta ~]$ sudo rbd map --pool rbd data  
/dev/rbd0
```

- 8.2. Format the /dev/rbd0 device with an XFS file system and mount the file system on the /mnt/data directory.

```
[admin@clienta ~]$ sudo mkfs.xfs /dev/rbd0  
meta-data=/dev/rbd0          isize=512    agcount=8, agsize=4096 blks  
                      =           sectsz=512  attr=2, projid32bit=1  
                      =           crc=1      finobt=1, sparse=1, rmapbt=0  
                      =           reflink=1  
data     =           bsize=4096   blocks=32768, imaxpct=25  
          =           sunit=16    swidth=16 blks  
naming   =version 2        bsize=4096   ascii-ci=0, ftype=1  
log      =internal log     bsize=4096   blocks=1872, version=2  
          =           sectsz=512  sunit=16 blks, lazy-count=1  
realtime =none            extsz=4096   blocks=0, rtextents=0  
Discarding blocks...Done.  
[admin@clienta ~]$ sudo mount /dev/rbd0 /mnt/data
```

- 8.3. Copy the /usr/share/dict/words file to the root of the file system, /mnt/data. List the content to verify the copy.

```
[admin@clienta ~]$ sudo cp /usr/share/dict/words /mnt/data/  
[admin@clienta ~]$ ls /mnt/data/  
words
```

- 8.4. Unmount and unmap the /dev/rbd0 device.

```
[admin@clienta ~]$ sudo umount /dev/rbd0  
[admin@clienta ~]$ sudo rbd unmap --pool rbd data
```

9. In the production cluster, create a snapshot called beforeprod of the RBD image data. Create a clone called prod1 from the snapshot called beforeprod.

- 9.1. In the production cluster, use sudo to run the cephadm shell. Create a snapshot called beforeprod of the RBD image data in the rbd pool.

```
[admin@clienta ~]$ sudo cephadm shell  
...output omitted...  
[ceph: root@clienta /]# rbd snap create rbd/data@beforeprod  
Creating snap: 100% complete...done.
```

- 9.2. Verify the snapshot by listing the snapshots of the data RBD image in the rbd pool.

```
[ceph: root@clienta /]# rbd snap list --pool rbd data
SNAPID  NAME      SIZE    PROTECTED   TIMESTAMP
4  beforeprod  128 MiB           Thu Oct 28 00:03:08 2021
```

- 9.3. Protect the `beforeprod` snapshot and create the clone. Exit from the `cephadm` shell.

```
[ceph: root@clienta /]# rbd snap protect rbd/data@beforeprod
[ceph: root@clienta /]# rbd clone rbd/data@beforeprod rbd/prod1
[ceph: root@clienta /]# exit
```

- 9.4. Verify that the clone also contains the `words` file by mapping and mounting the clone image. Unmount the file system and unmap the device after verification.

```
[admin@clienta ~]$ sudo rbd map --pool rbd prod1
/dev/rbd0
[admin@clienta ~]$ sudo mount /dev/rbd0 /mnt/data
[admin@clienta ~]$ ls /mnt/data
words
[admin@clienta ~]$ sudo umount /mnt/data
[admin@clienta ~]$ sudo rbd unmap --pool rbd prod1
```

10. In the production cluster, export the image called `data` to the `/home/admin/cr4/data.img` file. Import it as an image called `data` to the `rbdimage mode` pool. Create a snapshot called `beforeprod` of the new `data` image in the `rbdimage mode` pool.
- 10.1. In the production cluster, use `sudo` to run the `cephadm` shell with a bind mount of the `/home/admin/cr4/` directory. Export the image called `data` to the `/mnt/data.img` file.

```
[admin@clienta ~]$ sudo cephadm shell --mount /home/admin/cr4/
...output omitted...
[ceph: root@clienta /]# rbd export --pool rbd data /mnt/data.img
Exporting image: 100% complete...done.
```

- 10.2. Import the `/mnt/data.img` file as an image called `data` to the pool called `rbdimage mode`. Verify the import by listing the images in the `rbdimage mode` pool.

```
[ceph: root@clienta /]# rbd import /mnt/data.img rbdimage mode/data
Importing image: 100% complete...done.
[ceph: root@clienta /]# rbd --pool rbdimage mode ls
data
vm2
```

- 10.3. Create a snapshot called `beforeprod` of the image called `data` in the pool called `rbdimage mode`. Exit from the `cephadm` shell.

```
[ceph: root@clienta /]# rbd snap create rbdimage mode/data@beforeprod  
Creating snap: 100% complete...done.  
[ceph: root@clienta /]# exit  
exit
```

11. On the clienta host, use the kernel RBD client to remap and remount the RBD image called data in the pool called rbd. Copy the /etc/services file to the root of the file system. Unmount the file system and unmap the device when done.
 - 11.1. Map the data image in the rbd pool using the kernel RBD client. Mount the file system on /mnt/data.

```
[admin@clienta ~]$ sudo rbd map --pool rbd data  
/dev/rbd0  
[admin@clienta ~]$ sudo mount /dev/rbd0 /mnt/data
```

- 11.2. Copy the /etc/services file to the root of the file system, /mnt/data. List the contents of /mnt/data for verification.

```
[admin@clienta ~]$ sudo cp /etc/services /mnt/data/  
[admin@clienta ~]$ ls /mnt/data/  
services words
```

- 11.3. Unmount the file system and unmap the data image in the rbd pool.

```
[admin@clienta ~]$ sudo umount /mnt/data  
[admin@clienta ~]$ sudo rbd unmap --pool rbd data
```

12. In the production cluster, export changes to the rbd/data image, after the creation of the beforeprod snapshot, to a file called /home/admin/cr4/data-diff.img. Import the changes from the /mnt/data-diff.img file to the image called data in the rbdimage mode pool.
 - 12.1. In the production cluster, use sudo to run the cephadm shell with a bind mount of the /home/admin/cr4/ directory. Export changes to the data image in the rbd pool, after the creation of the beforeprod snapshot, to a file called /mnt/token/data-diff.img.

```
[admin@clienta ~]$ sudo cephadm shell --mount /home/admin/cr4/  
...output omitted...  
[ceph: root@clienta /]# rbd export-diff \  
--from-snap beforeprod rbd/data \  
/mnt/data-diff.img  
Exporting image: 100% complete...done.
```

- 12.2. Import changes from the /mnt/data-diff.img file to the image called data in the pool called rbdimage mode. Exit from the cephadm shell.

```
[ceph: root@clienta /]# rbd import-diff \
/mnt/data-diff.img \
rbdimagemode/data
Importing image diff: 100% complete...done.
[ceph: root@clienta /]# exit
exit
```

- 12.3. Verify that the image called `data` in the pool called `rbdimagemode` also contains the `services` file by mapping and mounting the image. When done, unmount the file system and unmap the image.

```
[admin@clienta ~]$ sudo rbd map rbdimagemode/data
/dev/rbd0
[admin@clienta ~]$ sudo mount /dev/rbd0 /mnt/data
[admin@clienta ~]$ ls /mnt/data
services words
[admin@clienta ~]$ sudo umount /mnt/data
[admin@clienta ~]$ sudo rbd unmap --pool rbdimagemode data
```

13. Configure the `clienta` host so that it will persistently mount the `rbd/data` RBD image as `/mnt/data`. Authenticate as the `admin` Ceph user by using existing keys found in the `/etc/ceph/ceph.client.admin.keyring` file.

- 13.1. Create an entry for `rbd/data` in the `/etc/ceph/rbdmap` RBD map file. The resulting file should have the following contents:

```
[admin@clienta ~]$ cat /etc/ceph/rbdmap
# RbdDevice Parameters
#poolname/imagename id=client,keyring=/etc/ceph/ceph.client.keyring
rbd/data id=admin,keyring=/etc/ceph/ceph.client.admin.keyring
```

- 13.2. Create an entry for `/dev/rbd/rbd/data` in the `/etc/fstab` file. The resulting file should have the following contents:

```
[admin@clienta ~]$ cat /etc/fstab
UUID=d47ead13-ec24-428e-9175-46aefa764b26 / xfs defaults 0 0
UUID=7B77-95E7 /boot/efi vfat defaults,uid=0,gid=0,umask=077,shortname=winnt 0 2
/dev/rbd/rbd/data /mnt/data xfs noauto 0 0
```

- 13.3. Use the `rbdmap` command to verify your RBD map configuration.

```
[admin@clienta ~]$ sudo rbdmap map
[admin@clienta ~]$ rbd showmapped
id pool namespace image snap device
0 rbd data - /dev/rbd0
[admin@clienta ~]$ sudo rbdmap unmap
[admin@clienta ~]$ rbd showmapped
```

- 13.4. After you have verified that the RBD mapped devices work, enable the `rbdmap` service. Reboot the `clienta` host to verify that the RBD device mounts persistently.

```
[admin@clienta ~]$ sudo systemctl enable rbdmap
Created symlink /etc/systemd/system/multi-user.target.wants/rbdmap.service → /usr/
lib/systemd/system/rbdmap.service.
[admin@clienta ~]$ sudo reboot
Connection to clienta closed by remote host.
Connection to clienta closed.
```

- 13.5. When `clienta` finishes rebooting, log in to `clienta` as the `admin` user, and verify that it has mounted the RBD device.

```
[student@workstation ~]$ ssh admin@clienta
...output omitted...
[admin@clienta ~]$ df /mnt/data
Filesystem      1K-blocks   Used Available Use% Mounted on
/dev/rbd0        123584    13460     110124  11% /mnt/data
```

- 13.6. Return to `workstation` as the `student` user.

```
[admin@clienta ~]$ exit
[student@workstation ~]$
```

Evaluation

Grade your work by running the `lab grade comprehensive-review4` command from your `workstation` machine. Correct any reported failures and rerun the script until successful.

```
[student@workstation ~]$ lab grade comprehensive-review4
```

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-review4
```

This concludes the lab.

▶ Lab

Deploying and Configuring RADOS Gateway

In this review, you will deploy and configure RADOS Gateways using specified requirements.

Outcomes

You should be able to:

- Deploy RADOS Gateway services.
- Configure multisite replication.
- Create and manage users to access the RADOS Gateway.
- Create buckets and store objects by using the Amazon S3 and Swift APIs.

Before You Begin

If you did not reset your classroom virtual machines at the end of the last chapter, save any work you want to keep from earlier exercises on those machines and reset the classroom environment now.



Important

Reset your environment before performing this exercise. All comprehensive review labs start with a clean, initial classroom environment that includes a pre-built, fully operational Ceph cluster. All remaining comprehensive reviews use the default Ceph cluster provided in the initial classroom environment.

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

```
[student@workstation ~]$ lab start comprehensive-review5
```

This command ensures that all cluster hosts are reachable. It also installs the AWS and Swift clients on the `serverc` and `serverf` nodes.

The primary Ceph cluster contains the `serverc`, `serverd`, and `servere` nodes. The secondary Ceph cluster contains the `serverf` node.

Specifications

- Create a realm, zonegroup, zone, and a system user called `Replication User` on the primary Ceph cluster. Configure each resource as the default. Treat this zone as the primary zone. Use the names provided in this table:

Resource	Name
Realm	cl260

Resource	Name
Zonegroup	classroom
Zone	main
System User	repl.user
Access Key	replication
Secret Key	secret
Endpoint	http://serverc:80

- Deploy a RADOS Gateway service in the primary cluster called `cl260-1` with one RGW instance on `serverc`. Configure the primary zone name and disable dynamic bucket index resharding.
- On the secondary Ceph cluster, configure a secondary zone called `fallback` for the `classroom` zonegroup. Object resources created in the primary zone must replicate to the secondary zone. Configure the endpoint of the secondary zone as `http://serverf:80`
- Deploy a RADOS Gateway service in the secondary cluster called `cl260-2` with one RGW instance. Configure the secondary zone name and disable dynamic bucket index resharding.
- Create an Amazon S3 API user called `S3 User` with a uid of `apiuser`, an access key of `review`, and a secret key of `securekey`. Create a Swift API subuser with secret key of `secureospkey`. Grant full access to both the user and subuser.
- Create a bucket called `images` by using the Amazon S3 API. Upload the `/etc/favicon.png` file to the `images` container by using the Swift API. The object must be available as `favicon-image`

Evaluation

Grade your work by running the `lab grade comprehensive-review5` command from your `workstation` machine. Correct any reported failures and rerun the script until successful.

```
[student@workstation ~]$ lab grade comprehensive-review5
```

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-review5
```

This concludes the lab.

► Solution

Deploying and Configuring RADOS Gateway

In this review, you will deploy and configure RADOS Gateways using specified requirements.

Outcomes

You should be able to:

- Deploy RADOS Gateway services.
- Configure multisite replication.
- Create and manage users to access the RADOS Gateway.
- Create buckets and store objects by using the Amazon S3 and Swift APIs.

Before You Begin

If you did not reset your classroom virtual machines at the end of the last chapter, save any work you want to keep from earlier exercises on those machines and reset the classroom environment now.



Important

Reset your environment before performing this exercise. All comprehensive review labs start with a clean, initial classroom environment that includes a pre-built, fully operational Ceph cluster. All remaining comprehensive reviews use the default Ceph cluster provided in the initial classroom environment.

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

```
[student@workstation ~]$ lab start comprehensive-review5
```

This command ensures that all cluster hosts are reachable. It also installs the AWS and Swift clients on the **serverc** and **serverf** nodes.

The primary Ceph cluster contains the **serverc**, **serverd**, and **servere** nodes. The secondary Ceph cluster contains the **serverf** node.

1. Log in to **serverc** as the **admin** user. Create a realm called **cl260**, a zonegroup called **classroom**, a zone called **main**, and a system user called **Replication User**. Use the UID of **repl.user**, access key of **replication**, and secret key of **secret** for the user. Set the zone endpoint as **http://serverc:80**.
 - 1.1. Log in to **serverc** as the **admin** user and use **sudo** to run the **cephadm** shell.

```
[student@workstation ~]$ ssh admin@serverc
[admin@serverc ~]$ sudo cephadm shell
[ceph: root@serverc /]#
```

- 1.2. Create a realm called cl260. Set the realm as default.

```
[ceph: root@serverc /]# radosgw-admin realm create --rgw-realm=cl260 --default
{
    "id": "8ea5596f-e2bb-4ac5-8fc8-9122de311e26",
    "name": "cl260",
    "current_period": "75c34edd-428f-4c7f-a150-6236bf6102db",
    "epoch": 1
}
```

- 1.3. Create a zonegroup called classroom. Configure the classroom zonegroup with an endpoint on the serverc node. Set the classroom zonegroup as the default.

```
[ceph: root@serverc /]# radosgw-admin zonegroup create --rgw-zonegroup=classroom \
--endpoints=http://serverc:80 --master --default
{
    "id": "2b1495f8-5ac3-4ec5-897e-ae5e0923d0b9",
    "name": "classroom",
    "api_name": "classroom",
    "is_master": "true",
    "endpoints": [
        "http://serverc:80"
    ],
    ...output omitted...
```

- 1.4. Create a master zone called main. Configure the zone with an endpoint pointing to http://serverc:80. Use replication as the access key and secret as the secret key. Set the main zone as the default.

```
[ceph: root@serverc /]# radosgw-admin zone create --rgw-zonegroup=classroom \
--rgw-zone=main --endpoints=http://serverc:80 --access-key=replication \
--secret=secret --master --default
{
    "id": "b50c6d11-6ab6-4a3e-9fb6-286798ba950d",
    "name": "main",
    ...output omitted...
    "system_key": {
        "access_key": "replication",
        "secret_key": "secret"
    },
    ...output omitted...
    "realm_id": "8ea5596f-e2bb-4ac5-8fc8-9122de311e26",
    "notif_pool": "main.rgw.log:notif"
}
```

- 1.5. Create a system user called repl.user to access the zone pools. The keys for the repl.user user must match the keys configured for the zone.

```
[ceph: root@serverc /]# radosgw-admin user create --uid="repl.user" \
--display-name="Replication User" --secret=secret --system \
--access-key=replication
{
    "user_id": "repl.user",
    "display_name": "Replication User",
    ...output omitted...
    {
        "user": "repl.user",
        "access_key": "replication",
        "secret_key": "secret"
    }
}
```

1.6. Commit the realm configuration structure changes to the period.

```
[ceph: root@serverc /]# radosgw-admin period update --commit
{
    "id": "93a7f406-0bbd-43a5-a32a-c217386d534b",
    "epoch": 1,
    "predecessor_uuid": "75c34edd-428f-4c7f-a150-6236bf6102db",
    ...output omitted...
        "id": "2b1495f8-5ac3-4ec5-897e-ae5e0923d0b9",
        "name": "classroom",
        "api_name": "classroom",
        "is_master": "true",
        "endpoints": [
            "http://serverc:80"
        ],
    ...output omitted...
        "id": "b50c6d11-6ab6-4a3e-9fb6-286798ba950d",
        "name": "main",
        "endpoints": [
            "http://serverc:80"
        ],
    ...output omitted...
    "realm_id": "8ea5596f-e2bb-4ac5-8fc8-9122de311e26",
    "realm_name": "cl260",
    "realm_epoch": 2
}
```

2. Create a RADOS Gateway service called cl260-1 with a single RGW daemon on serverc. Verify that the RGW daemon is up and running. Configure the zone name in the configuration database and disable dynamic bucket index resharding.

 - 2.1. Create a RADOS gateway service called cl260-1 with a single RGW daemon on the serverc node.

```
[ceph: root@serverc /]# ceph orch apply rgw cl260-1 --realm=cl260 --zone=main \
--placement="1 serverc.lab.example.com"
Scheduled rgw.cl260-1 update...
[ceph: root@serverc /]# ceph orch ps --daemon-type rgw
NAME                      HOST          STATUS      REFRESHED
AGE  PORTS ...
rgw.cl260-1.serverc.iwsaop    serverc.lab.example.com  running (70s)  65s
ago    70s  *:80 ...
```

2.2. Configure the zone name in the configuration database.

```
[ceph: root@serverc /]# ceph config set client.rgw rgw_zone main
[ceph: root@serverc /]# ceph config get client.rgw rgw_zone
main
```

2.3. Disable dynamic bucket index resharding

```
[ceph: root@serverc /]# ceph config set client.rgw rgw_dynamic_resharding false
[ceph: root@serverc /]# ceph config get client.rgw rgw_dynamic_resharding
false
```

3. Log in to `serverf` as the `admin` user. Pull the realm and period configuration from the `serverc` node. Use the credentials for `rep1.user` to authenticate. Verify that the *pulled* realm and zonegroup are set as default for the secondary cluster. Create a secondary zone called `fallback` for the `classroom` zonegroup.

3.1. In a second terminal, log in to `serverf` as the `admin` user and use `sudo` to run the `cephadm` shell.

```
[student@workstation ~]$ ssh admin@serverf
[admin@serverf ~]$ sudo cephadm shell
[ceph: root@serverf /]#
```

3.2. Pull the realm and period configuration from `serverc`.

```
[ceph: root@serverf /]# radosgw-admin realm pull --url=http://serverc:80 \
--access-key=replication --secret-key=secret
{
  "id": "8ea5596f-e2bb-4ac5-8fc8-9122de311e26",
  "name": "cl260",
  "current_period": "93a7f406-0bbd-43a5-a32a-c217386d534b",
  "epoch": 2
}
[ceph: root@serverf /]# radosgw-admin period pull --url=http://serverc:80 \
--access-key=replication --secret-key=secret
{
  "id": "93a7f406-0bbd-43a5-a32a-c217386d534b",
  "epoch": 1,
  "predecessor_uuid": "75c34edd-428f-4c7f-a150-6236bf6102db",
  "sync_status": [],
  "period_map": {
```

```

    "id": "93a7f406-0bbd-43a5-a32a-c217386d534b",
    "zonegroups": [
        {
            "id": "2b1495f8-5ac3-4ec5-897e-ae5e0923d0b9",
            "name": "classroom",
            "api_name": "classroom",
            "is_master": "true",
            "endpoints": [
                "http://serverc:80"
            ...
            output omitted...
            "zones": [
                {
                    "id": "b50c6d11-6ab6-4a3e-9fb6-286798ba950d",
                    "name": "main",
                    "endpoints": [
                        "http://serverc:80"
                    ...
                    output omitted...
                    "master_zonegroup": "2b1495f8-5ac3-4ec5-897e-ae5e0923d0b9",
                    "master_zone": "b50c6d11-6ab6-4a3e-9fb6-286798ba950d",
                    ...
                    output omitted...
                    "realm_id": "8ea5596f-e2bb-4ac5-8fc8-9122de311e26",
                    "realm_name": "cl260",
                    "realm_epoch": 2
                }
            ]
        }
    ]
}

```

- 3.3. Set the cl260 realm and classroom zone group as default.

```
[ceph: root@serverf /]# radosgw-admin realm default --rgw-realm=cl260
[ceph: root@serverf /]# radosgw-admin zonegroup default --rgw-zonegroup=classroom
```

- 3.4. Create a zone called fallback. Configure the fallback zone with the endpoint pointing to http://serverf:80.

```
[ceph: root@serverf /]# radosgw-admin zone create --rgw-zonegroup=classroom \
    --rgw-zone=fallback --endpoints=http://serverf:80 --access-key=replication \
    --secret-key=secret --default
{
    "id": "fe105db9-fd00-4674-9f73-0d8e4e93c98c",
    "name": "fallback",
    ...
    output omitted...
    "system_key": {
        "access_key": "replication",
        "secret_key": "secret"
    ...
    output omitted...
        "realm_id": "8ea5596f-e2bb-4ac5-8fc8-9122de311e26",
        "notif_pool": "fallback.rgw.log:notif"
    }
}
```

- 3.5. Commit the site configuration.

```
[ceph: root@serverf /]# radosgw-admin period update --commit --rgw-zone=fallback
{
    "id": "93a7f406-0bbd-43a5-a32a-c217386d534b",
```

```

"epoch": 2,
"predecessor_uuid": "75c34edd-428f-4c7f-a150-6236bf6102db",
"sync_status": [],
"period_map": {
    "id": "93a7f406-0bbd-43a5-a32a-c217386d534b",
    "zonegroups": [
        {
            "id": "2b1495f8-5ac3-4ec5-897e-ae5e0923d0b9",
            "name": "classroom",
            "api_name": "classroom",
            "is_master": "true",
            "endpoints": [
                "http://serverc:80"
            ],
            ...output omitted...
            "zones": [
                {
                    "id": "b50c6d11-6ab6-4a3e-9fb6-286798ba950d",
                    "name": "main",
                    "endpoints": [
                        "http://serverc:80"
                    ],
                    ...output omitted...
                },
                {
                    "id": "fe105db9-fd00-4674-9f73-0d8e4e93c98c",
                    "name": "fallback",
                    "endpoints": [
                        "http://serverf:80"
                    ],
                    ...output omitted...
                }
            ],
            "master_zonegroup": "2b1495f8-5ac3-4ec5-897e-ae5e0923d0b9",
            "master_zone": "b50c6d11-6ab6-4a3e-9fb6-286798ba950d",
            ...output omitted...
            "realm_id": "8ea5596f-e2bb-4ac5-8fc8-9122de311e26",
            "realm_name": "cl260",
            "realm_epoch": 2
        }
    ]
}

```

4. Create a RADOS Gateway service called cl260-2 with a single RGW daemon on the serverf node. Verify that the RGW daemon is up and running. Configure the zone name in the configuration database and disable dynamic bucket index resharding.

- 4.1. Create a RADOS gateway service called cl260-2 with a single RGW daemon on serverf.

```

[ceph: root@serverf /]# ceph orch apply rgw cl260-2 --zone=fallback \
--placement="1 serverf.lab.example.com" --realm=cl260
Scheduled rgw.cl260-2 update...
[ceph: root@serverf /]# ceph orch ps --daemon-type rgw
NAME                      HOST          STATUS      REFRESHED
AGE  PORTS ...
rgw.cl260-2.serverf.lqcjui      serverf.lab.example.com  running (20s)  14s
ago    20s  *:80 ...

```

- 4.2. Configure the zone name in the configuration database.

```
[ceph: root@serverf /]# ceph config set client.rgw rgw_zone fallback
[ceph: root@serverf /]# ceph config get client.rgw rgw_zone
fallback
```

- 4.3. Disable dynamic bucket index resharding

```
[ceph: root@serverf /]# ceph config set client.rgw rgw_dynamic_resharding false
[ceph: root@serverf /]# ceph config get client.rgw rgw_dynamic_resharding
false
```

- 4.4. Verify the synchronization status.

```
[ceph: root@serverf /]# radosgw-admin sync status
      realm 8ea5596f-e2bb-4ac5-8fc8-9122de311e26 (cl260)
      zonegroup 2b1495f8-5ac3-4ec5-897e-ae5e0923d0b9 (classroom)
          zone fe105db9-fd00-4674-9f73-0d8e4e93c98c (fallback)
      metadata sync syncing
          full sync: 0/64 shards
          incremental sync: 64/64 shards
              metadata is caught up with master
      data sync source: b50c6d11-6ab6-4a3e-9fb6-286798ba950d (main)
          syncing
          full sync: 0/128 shards
          incremental sync: 128/128 shards
              data is caught up with source
```

5. On serverc, use the radosgw-admin command to create a user called apiuser for the Amazon S3 API and a subuser called apiuser:swift for the Swift API. For the apiuser user, utilize the access key of review, secret key of securekey, and grant full access. For the apiuser:swift subuser, utilize the secret of secureospkey and grant the subuser full access.

- 5.1. Create an Amazon S3 API user called S3 user with the UID of apiuser. Assign an access key of review and a secret of securekey, and grant the user full access.

```
[ceph: root@serverc /]# radosgw-admin user create --display-name="S3 user" \
    --uid="apiuser" --access= "full" --access_key="review" --secret="securekey"
{
    "user_id": "apiuser",
    "display_name": "S3 user",
    "email": "",
    "suspended": 0,
    "max_buckets": 1000,
    "subusers": [],
    "keys": [
        {
            "user": "apiuser",
            "access_key": "review",
            "secret_key": "securekey"
        }
    ]
}
```

```
[],
"swift_keys": [],
...output omitted...
```

- 5.2. Create a Swift subuser called `apiuser:swift`, set `secureospkey` as the subuser secret and grant full access.

```
[ceph: root@serverc /]# radosgw-admin subuser create --uid="apiuser" \
--access="full" --subuser="apiuser:swift" --secret="secureospkey"
{
    "user_id": "apiuser",
    "display_name": "S3 user",
    "email": "",
    "suspended": 0,
    "max_buckets": 1000,
    "subusers": [
        {
            "id": "apiuser:swift",
            "permissions": "full-control"
        }
    ],
    "keys": [
        {
            "user": "apiuser",
            "access_key": "review",
            "secret_key": "securekey"
        }
    ],
    "swift_keys": [
        {
            "user": "apiuser:swift",
            "secret_key": "secureospkey"
        }
    ]
...output omitted...
```

6. On the `serverc` node, exit the `cephadm` shell. Create a bucket called `review`. Configure the AWS CLI tool to use the `apiuser` user credentials. Use the `swift upload` command to upload the `/etc/favicon.png` file to the `image` bucket.

- 6.1. Exit the `cephadm` shell. Configure the AWS CLI tool to use operator credentials. Enter `review` as the access key and `securekey` as the secret key.

```
[ceph: root@serverc /]# exit
exit
[admin@serverc ~]$ aws configure --profile=ceph
AWS Access Key ID [None]: review
AWS Secret Access Key [None]: securekey
Default region name [None]: Enter
Default output format [None]: Enter
```

- 6.2. Create a bucket called `images`.

```
[admin@serverc ~]$ aws --profile=ceph --endpoint=http://serverc:80 s3 \
  mb s3://images
make_bucket: images
```

- 6.3. Use the upload command of the swift API to upload the /etc/favicon.png file to the images bucket. The object must be available as favicon-image.

```
[admin@serverc ~]$ swift -V 1.0 -A http://serverc:80/auth/v1 -U apiuser:swift \
  -K secureoskey upload images /etc/favicon.png --object-name favicon-image
favicon-image
```

- 6.4. Exit and close the second terminal. Return to workstation as the student user.

```
[root@serverf ~]# exit
[admin@serverf ~]$ exit
[student@workstation ~]$ exit
```

```
[admin@serverc ~]$ exit
[student@workstation ~]$
```

Evaluation

Grade your work by running the lab grade comprehensive-review5 command from your workstation machine. Correct any reported failures and rerun the script until successful.

```
[student@workstation ~]$ lab grade comprehensive-review5
```

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-review5
```

This concludes the lab.

