



SUSE Enterprise Storage 7

Administration and Operations Guide

Administration and Operations Guide

SUSE Enterprise Storage 7

by Tomáš Bažant, Alexandra Settle, and Liam Proven

Publication Date: 12 Dec 2024

<https://documentation.suse.com> 

Copyright © 2020–2024 SUSE LLC and contributors. All rights reserved.

Except where otherwise noted, this document is licensed under Creative Commons Attribution-ShareAlike 4.0 International (CC-BY-SA 4.0): <https://creativecommons.org/licenses/by-sa/4.0/legalcode> .

For SUSE trademarks, see <http://www.suse.com/company/legal/> . All third-party trademarks are the property of their respective owners. Trademark symbols (®, ™ etc.) denote trademarks of SUSE and its affiliates. Asterisks (*) denote third-party trademarks.

All information found in this book has been compiled with utmost attention to detail. However, this does not guarantee complete accuracy. Neither SUSE LLC, its affiliates, the authors nor the translators shall be held liable for possible errors or the consequences thereof.

Contents

About this guide xvii

- 1 Available documentation xvii
- 2 Giving feedback xviii
- 3 Documentation conventions xix
- 4 Support xxi
 - Support statement for SUSE Enterprise Storage xxi • Technology previews xxii
- 5 Ceph contributors xxii
- 6 Commands and command prompts used in this guide xxiii
 - Salt-related commands xxiii • Ceph related commands xxiii • General Linux commands xxiv • Additional information xxiv

I CEPH DASHBOARD 1

1 About the Ceph Dashboard 2

2 Dashboard's Web user interface 3

- 2.1 Logging in 3
- 2.2 Utility menu 4
- 2.3 Main menu 5
- 2.4 Content pane 6
- 2.5 Common Web UI features 6
- 2.6 Dashboard widgets 6
 - Status widgets 7 • Capacity widgets 7 • Performance widgets 8

3 Manage Ceph Dashboard users and roles 10

- 3.1 Listing users 10
- 3.2 Adding new users 10
- 3.3 Editing users 11
- 3.4 Deleting users 11
- 3.5 Listing user roles 12
- 3.6 Adding custom roles 12
- 3.7 Editing custom roles 14
- 3.8 Deleting custom roles 14

4 View cluster internals 15

- 4.1 Viewing cluster nodes 15
- 4.2 Accessing cluster inventory 15
- 4.3 Viewing Ceph Monitors 16
- 4.4 Displaying services 17
- 4.5 Displaying Ceph OSDs 18
 - Adding OSDs 21
- 4.6 Viewing cluster configuration 24
- 4.7 Viewing the CRUSH Map 24
- 4.8 Viewing manager modules 25
- 4.9 Viewing logs 26
- 4.10 Viewing monitoring 26

5 Manage pools 27

- 5.1 Adding a new pool 27
- 5.2 Deleting pools 28
- 5.3 Editing a pool's options 28

6 Manage RADOS Block Device 29

- 6.1 Viewing details about RBDs 30
- 6.2 Viewing RBD's configuration 31
- 6.3 Creating RBDs 32
- 6.4 Deleting RBDs 33
- 6.5 Creating RADOS Block Device snapshots 33
- 6.6 RBD mirroring 34
 - Configuring primary and secondary clusters 35 • Enabling the rbd-mirror daemon 35 • Disabling mirroring 36 • Bootstrapping peers 37 • Removing cluster peer 38 • Configuring pool replication in the Ceph Dashboard 38 • Verifying that RBD image replication works 42
- 6.7 Managing iSCSI Gateways 45
 - Adding iSCSI targets 46 • Editing iSCSI targets 48 • Deleting iSCSI targets 48
- 6.8 RBD Quality of Service (QoS) 48
 - Configuring options globally 49 • Configuring options on a new pool 49 • Configuring options on an existing pool 50 • Configuration options 50 • Creating RBD QoS options with a new RBD image 51 • Editing RBD QoS options on existing images 51 • Changing configuration options when copying or cloning images 51

7 Manage NFS Ganesha 52

- 7.1 Creating NFS exports 53
- 7.2 Deleting NFS exports 55
- 7.3 Editing NFS exports 55

8 Manage CephFS 57

- 8.1 Viewing CephFS overview 57

9 Manage the Object Gateway 59

- 9.1 Viewing Object Gateways 59

- 9.2 **Managing Object Gateway users 60**
 - Adding a new gateway user 61 • Deleting gateway users 63 • Editing gateway user details 63
- 9.3 **Managing the Object Gateway buckets 63**
 - Adding a new bucket 63 • Viewing bucket details 64 • Editing the bucket 64 • Deleting a bucket 65
- 10 Manual configuration 66**
- 10.1 **Configuring TLS/SSL support 66**
 - Creating self-signed certificates 67 • Using certificates signed by CA 67
- 10.2 **Changing host name and port number 68**
- 10.3 **Adjusting user names and passwords 69**
- 10.4 **Enabling the Object Gateway management front-end 69**
- 10.5 **Enabling iSCSI management 70**
- 10.6 **Enabling Single Sign-On 71**
- 11 Manage users and roles on the command line 73**
- 11.1 **Managing the password policy 73**
- 11.2 **Managing user accounts 74**
- 11.3 **User roles and permissions 74**
 - Defining security scopes 75 • Specifying user roles 76
- 11.4 **Proxy configuration 78**
 - Accessing the dashboard with reverse proxies 79 • Disabling re-directions 79 • Configuring error status codes 79 • HAProxy example configuration 79
- 11.5 **Auditing API requests 80**
- 11.6 **Configuring NFS Ganesha in the Ceph Dashboard 81**
 - Configuring multiple NFS Ganesha clusters 82
- 11.7 **Debugging plugins 82**

II CLUSTER OPERATION 83

12 Determine the cluster state 84

- 12.1 Checking a cluster's status 84
- 12.2 Checking cluster health 86
- 12.3 Checking a cluster's usage stats 95
- 12.4 Checking OSD status 97
- 12.5 Checking for full OSDs 97
- 12.6 Checking the monitor status 98
- 12.7 Checking placement group states 99
- 12.8 Storage capacity 99
- 12.9 Monitoring OSDs and placement groups 102
 - Monitoring OSDs 102 • Assigning placement group sets 103 • Peering 105 • Monitoring placement group states 105 • Finding an object location 110

13 Operational tasks 112

- 13.1 Modifying the cluster configuration 112
- 13.2 Adding nodes 112
- 13.3 Removing nodes 113
- 13.4 OSD management 115
 - Listing disk devices 115 • Erasing disk devices 115 • Adding OSDs using DriveGroups specification 116 • Removing OSDs 126 • Replacing OSDs 127
- 13.5 Moving the Salt Master to a new node 128
- 13.6 Updating the cluster nodes 130
 - Software repositories 130 • Repository staging 130 • Downtime of Ceph services 131 • Running the update 131

- 13.7 Updating Ceph 131
 - Starting the update 131 • Monitoring the update 132 • Cancelling an update 132
- 13.8 Halting or rebooting cluster 132
- 13.9 Removing an entire Ceph cluster 133
- 14 Operation of Ceph services 135**
- 14.1 Operating individual services 135
- 14.2 Operating service types 136
- 14.3 Operating services on a single node 136
 - Identifying services and targets 136 • Operating all services on a node 137 • Operating an individual service on a node 137 • Querying the service status 138
- 14.4 Shutting down and restarting the whole Ceph cluster 138
- 15 Backup and restore 140**
- 15.1 Back Up Cluster Configuration and Data 140
 - Back up ceph-salt configuration 140 • Back up Ceph configuration 140 • Back up Salt configuration 140 • Back up custom configurations 141
- 15.2 Restoring a Ceph node 141
- 16 Monitoring and alerting 143**
- 16.1 Configuring custom or local images 144
- 16.2 Updating monitoring services 145
- 16.3 Disabling monitoring 146
- 16.4 Configuring Grafana 147
- 16.5 Configuring the Prometheus Manager Module 147
 - Configuring the network interface 148 • Configuring scrape_interval 148 • Configuring the cache 148 • Enabling RBD-image monitoring 149

- 16.6 Prometheus security model 149
- 16.7 Prometheus Alertmanager SNMP webhook 150
 - Configuring the prometheus-webhook-snmp service 151 • Configuring the Prometheus Alertmanager for SNMP 152

III STORING DATA IN A CLUSTER 153

17 Stored data management 154

- 17.1 OSD devices 155
 - Device classes 155
- 17.2 Buckets 162
- 17.3 Rule sets 165
 - Iterating the node tree 167 • `firstn` and `indep` 169
- 17.4 Placement groups 170
 - Using placement groups 170 • Determining the value of `PG_NUM` 172 • Setting the number of placement groups 173 • Finding the number of placement groups 174 • Finding a cluster's PG statistics 174 • Finding statistics for stuck PGs 174 • Searching a placement group map 175 • Retrieving a placement groups statistics 175 • Scrubbing a placement group 175 • Prioritizing backfill and recovery of placement groups 176 • Reverting lost objects 176 • Enabling the PG auto-scaler 177
- 17.5 CRUSH Map manipulation 177
 - Editing a CRUSH Map 178 • Adding or moving an OSD 179 • Difference between `ceph osd reweight` and `ceph osd crush reweight` 179 • Removing an OSD 180 • Adding a bucket 180 • Moving a bucket 181 • Removing a bucket 181
- 17.6 Scrubbing placement groups 181

18 Manage storage pools 184

- 18.1 Creating a pool 184
- 18.2 Listing pools 186

- 18.3 Renaming a pool 186
- 18.4 Deleting a pool 186
- 18.5 Other operations 187
 - Associating pools with an application 187 • Setting pool quotas 188 • Showing pool statistics 188 • Getting pool values 190 • Setting pool values 191 • Setting the number of object replicas 194
- 18.6 Pool migration 195
 - Limitations 196 • Migrating using cache tier 196 • Migrating RBD images 198
- 18.7 Pool snapshots 199
 - Making a snapshot of a pool 199 • Listing snapshots of a pool 200 • Removing a snapshot of a pool 200
- 18.8 Data compression 200
 - Enabling compression 200 • Pool compression options 201 • Global compression options 202
- 19 Erasure coded pools 204**
- 19.1 Prerequisite for erasure coded Pools 204
- 19.2 Creating a sample erasure coded pool 204
- 19.3 Erasure code profiles 205
 - Creating a new erasure code profile 207 • Removing an erasure code profile 208 • Displaying an erasure code profile's details 208 • Listing erasure code profiles 209
- 19.4 Marking erasure coded pools with RADOS Block Device 209
- 20 RADOS Block Device 210**
- 20.1 Block device commands 210
 - Creating a block device image in a replicated pool 210 • Creating a block device image in an erasure coded pool 211 • Listing block device images 211 • Retrieving image information 212 • Resizing a block device image 212 • Removing a block device image 212

- 20.2 Mounting and unmounting **212**
 - Creating a Ceph user account **213** • User authentication **213** • Preparing a RADOS Block Device for use **213** • **rbdmap** Map RBD devices at boot time **215** • Increasing the size of RBD devices **217**
- 20.3 Snapshots **217**
 - Enabling and configuring cephx **218** • Snapshot basics **218** • Snapshot layering **220**
- 20.4 RBD image mirrors **224**
 - Pool configuration **225** • RBD Image configuration **229** • Checking the mirror status **234**
- 20.5 Cache settings **234**
- 20.6 QoS settings **236**
- 20.7 Read-ahead settings **237**
- 20.8 Advanced features **237**
- 20.9 Mapping RBD using old kernel clients **239**
- 20.10 Enabling block devices and Kubernetes **240**
 - Using Ceph block devices in Kubernetes **243**

IV ACCESSING CLUSTER DATA **247**

21 Ceph Object Gateway **248**

- 21.1 Object Gateway restrictions and naming limitations **248**
 - Bucket limitations **248** • Stored object limitations **248** • HTTP header limitations **249**
- 21.2 Deploying the Object Gateway **249**
- 21.3 Operating the Object Gateway service **249**
- 21.4 Configuration options **249**
- 21.5 Managing Object Gateway access **249**
 - Accessing Object Gateway **250** • Manage S3 and Swift accounts **252**

- 21.6 HTTP front-ends 255
- 21.7 Enable HTTPS/SSL for Object Gateways 256
 - Creating a self-signed certificate 256
 - Configuring Object Gateway with SSL 257
- 21.8 Synchronization modules 257
 - Configuring synchronization modules 258
 - Synchronizing zones 259
 - ElasticSearch synchronization module 260
 - Cloud synchronization module 263
 - Archive synchronization module 267
- 21.9 LDAP authentication 268
 - Authentication mechanism 268
 - Requirements 269
 - Configuring Object Gateway to use LDAP authentication 269
 - Using a custom search filter to limit user access 270
 - Generating an access token for LDAP authentication 271
- 21.10 Bucket index sharding 272
 - Bucket index resharding 272
 - Bucket index sharding for new buckets 275
- 21.11 OpenStack Keystone integration 276
 - Configuring OpenStack 276
 - Configuring the Ceph Object Gateway 277
- 21.12 Pool placement and storage classes 279
 - Displaying placement targets 279
 - Storage classes 280
 - Configuring zonegroups and zones 280
 - Placement customization 282
 - Using storage classes 283
- 21.13 Multisite Object Gateways 284
 - Requirements and assumptions 285
 - Limitations 285
 - Configuring a master zone 285
 - Configure secondary zones 291
 - General Object Gateway maintenance 297
 - Performing failover and disaster recovery 300
- 22 Ceph iSCSI gateway 302**
- 22.1 ceph-iscsi managed targets 302
 - Connecting to open-iscsi 302
 - Connecting Microsoft Windows (Microsoft iSCSI initiator) 306
 - Connecting VMware 313
- 22.2 Conclusion 319

23 Clustered file system 320

- 23.1 Mounting CephFS 320
 - Preparing the client 320 • Creating a secret file 321 • Mounting CephFS 321
- 23.2 Unmounting CephFS 323
- 23.3 Mounting CephFS in /etc/fstab 323
- 23.4 Multiple active MDS daemons (active-active MDS) 323
 - Using active-active MDS 323 • Increasing the MDS active cluster size 324 • Decreasing the number of ranks 324 • Manually pinning directory trees to a rank 325
- 23.5 Managing failover 326
 - Configuring standby replay 326
- 23.6 Setting CephFS quotas 326
 - CephFS quota limitations 327 • Configuring CephFS quotas 327
- 23.7 Managing CephFS snapshots 328
 - Creating snapshots 329 • Deleting snapshots 329

24 Export Ceph data via Samba 330

- 24.1 Export CephFS via Samba share 330
 - Configuring and exporting Samba packages 330 • Single gateway example 330 • Configuring high availability 333
- 24.2 Joining Samba Gateway and Active Directory 339
 - Preparing Samba installation 339 • Verifying DNS 340 • Resolving SRV records 340 • Configuring Kerberos 341 • Resolving localhost name 341 • Configuring Samba 342 • Joining the Active Directory domain 345 • Configuring the name service switch 345 • Starting the services 346 • Test the winbindd connectivity 346

25 NFS Ganesha 347

- 25.1 Creating an NFS service 348
- 25.2 Starting or Restarting NFS Ganesha 349

- 25.3 Listing objects in the NFS recovery pool 349
- 25.4 Creating an NFS export 349
- 25.5 Verifying the NFS export 350
- 25.6 Mounting the NFS export 351
- 25.7 Multiple NFS Ganesha clusters 351

V INTEGRATION WITH VIRTUALIZATION TOOLS 352

26 libvirt and Ceph 353

- 26.1 Configuring Ceph with libvirt 353
- 26.2 Preparing the VM manager 354
- 26.3 Creating a VM 355
- 26.4 Configuring the VM 355
- 26.5 Summary 358

27 Ceph as a back-end for QEMU KVM instance 359

- 27.1 Installing qemu-block-rbd 359
- 27.2 Using QEMU 359
- 27.3 Creating images with QEMU 359
- 27.4 Resizing images with QEMU 360
- 27.5 Retrieving image info with QEMU 360
- 27.6 Running QEMU with RBD 361
- 27.7 Enabling discard and TRIM 361
- 27.8 Setting QEMU cache options 362

VI	CONFIGURING A CLUSTER	363
28	Ceph cluster configuration	364
28.1	Configure the <code>ceph.conf</code> file	364
	Accessing <code>ceph.conf</code> inside container images	364
28.2	Configuration database	365
	Configuring sections and masks	365
	Setting and reading configuration options	365
	Configuring daemons at runtime	366
28.3	<code>config-key</code> store	368
	iSCSI Gateway	369
28.4	Ceph OSD and BlueStore	370
	Configuring automatic cache sizing	370
28.5	Ceph Object Gateway	371
	General Settings	371
	Configuring HTTP front-ends	380
29	Ceph Manager modules	383
29.1	Balancer	383
	The 'crush-compat' mode	384
	Planning and executing of data balancing	384
29.2	Enabling the telemetry module	386
30	Authentication with cephx	388
30.1	Authentication architecture	388
30.2	Key management	390
	Background information	391
	Managing users	394
	Managing keyrings	398
	Commandline usage	401
A	Ceph maintenance updates based on upstream 'Octopus' point releases	402
	Glossary	407

About this guide

This guide focuses on routine tasks that you as an administrator need to take care of after the basic Ceph cluster has been deployed (day 2 operations). It also describes all the supported ways to access data stored in a Ceph cluster.

SUSE Enterprise Storage 7 is an extension to SUSE Linux Enterprise Server 15 SP2. It combines the capabilities of the Ceph (<http://ceph.com/>) storage project with the enterprise engineering and support of SUSE. SUSE Enterprise Storage 7 provides IT organizations with the ability to deploy a distributed storage architecture that can support a number of use cases using commodity hardware platforms.

1 Available documentation



Note: Online documentation and latest updates

Documentation for our products is available at <https://documentation.suse.com/>, where you can also find the latest updates, and browse or download the documentation in various formats. The latest documentation updates can be found in the English language version.

In addition, the product documentation is available in your installed system under `/usr/share/doc/manual`. It is included in an RPM package named `ses-manual_LANG_CODE`. Install it if it is not already on your system, for example:

```
# zypper install ses-manual_en
```

The following documentation is available for this product:

Deployment Guide (<https://documentation.suse.com/ses/html/ses-all/book-storage-deployment.html>)

This guide focuses on deploying a basic Ceph cluster, and how to deploy additional services. It also cover the steps for upgrading to SUSE Enterprise Storage 7 from the previous product version.

Administration and Operations Guide (<https://documentation.suse.com/ses/html/ses-all/book-storage-admin.html>) ↗

This guide focuses on routine tasks that you as an administrator need to take care of after the basic Ceph cluster has been deployed (day 2 operations). It also describes all the supported ways to access data stored in a Ceph cluster.

Security Hardening Guide (<https://documentation.suse.com/ses/html/ses-all/book-storage-security.html>) ↗

This guide focuses on how to ensure your cluster is secure.

Troubleshooting Guide (<https://documentation.suse.com/ses/html/ses-all/book-storage-troubleshooting.html>) ↗

This guide takes you through various common problems when running SUSE Enterprise Storage 7 and other related issues to relevant components such as Ceph or Object Gateway.

SUSE Enterprise Storage for Windows Guide (<https://documentation.suse.com/ses/html/ses-all/book-storage-windows.html>) ↗

This guide describes the integration, installation, and configuration of Microsoft Windows environments and SUSE Enterprise Storage using the Windows Driver.

2 Giving feedback

We welcome feedback on, and contributions to, this documentation. There are several channels for this:

Service requests and support

For services and support options available for your product, see <http://www.suse.com/support/> ↗.

To open a service request, you need a SUSE subscription registered at SUSE Customer Center. Go to <https://scc.suse.com/support/requests> ↗, log in, and click *Create New*.

Bug reports

Report issues with the documentation at <https://bugzilla.suse.com/> ↗. Reporting issues requires a Bugzilla account.

To simplify this process, you can use the *Report Documentation Bug* links next to headlines in the HTML version of this document. These preselect the right product and category in Bugzilla and add a link to the current section. You can start typing your bug report right away.

Contributions

To contribute to this documentation, use the *Edit Source* links next to headlines in the HTML version of this document. They take you to the source code on GitHub, where you can open a pull request. Contributing requires a GitHub account.

For more information about the documentation environment used for this documentation, see the repository's README at <https://github.com/SUSE/doc-ses>.

Mail

You can also report errors and send feedback concerning the documentation to doc-team@suse.com. Include the document title, the product version, and the publication date of the document. Additionally, include the relevant section number and title (or provide the URL) and provide a concise description of the problem.

3 Documentation conventions

The following notices and typographic conventions are used in this document:

- /etc/passwd: Directory names and file names
- PLACEHOLDER: Replace PLACEHOLDER with the actual value
- PATH: An environment variable
- ls, --help: Commands, options, and parameters
- user: The name of user or group
- package_name: The name of a software package
- **Alt**, **Alt - F1**: A key to press or a key combination. Keys are shown in uppercase as on a keyboard.
- *File*, *File* > *Save As*: menu items, buttons
- **AMD/Intel** This paragraph is only relevant for the AMD64/Intel 64 architectures. The arrows mark the beginning and the end of the text block. ◁
- **IBM Z, POWER** This paragraph is only relevant for the architectures IBM Z and POWER. The arrows mark the beginning and the end of the text block. ◁
- *Chapter 1, “Example chapter”*: A cross-reference to another chapter in this guide.

- Commands that must be run with `root` privileges. Often you can also prefix these commands with the `sudo` command to run them as non-privileged user.

```
# command  
> sudo command
```

- Commands that can be run by non-privileged users.

```
> command
```

- Notices



Warning: Warning notice

Vital information you must be aware of before proceeding. Warns you about security issues, potential loss of data, damage to hardware, or physical hazards.



Important: Important notice

Important information you should be aware of before proceeding.



Note: Note notice

Additional information, for example about differences in software versions.



Tip: Tip notice

Helpful information, like a guideline or a piece of practical advice.

- Compact Notices



Additional information, for example about differences in software versions.



Helpful information, like a guideline or a piece of practical advice.

4 Support

Find the support statement for SUSE Enterprise Storage and general information about technology previews below. For details about the product lifecycle, see <https://www.suse.com/lifecycle>. If you are entitled to support, find details on how to collect information for a support ticket at <https://documentation.suse.com/sles-15/html/SLES-all/cha-adm-support.html>.

4.1 Support statement for SUSE Enterprise Storage

To receive support, you need an appropriate subscription with SUSE. To view the specific support offerings available to you, go to <https://www.suse.com/support/> and select your product.

The support levels are defined as follows:

L1

Problem determination, which means technical support designed to provide compatibility information, usage support, ongoing maintenance, information gathering and basic troubleshooting using available documentation.

L2

Problem isolation, which means technical support designed to analyze data, reproduce customer problems, isolate problem area and provide a resolution for problems not resolved by Level 1 or prepare for Level 3.

L3

Problem resolution, which means technical support designed to resolve problems by engaging engineering to resolve product defects which have been identified by Level 2 Support.

For contracted customers and partners, SUSE Enterprise Storage is delivered with L3 support for all packages, except for the following:

- Technology previews.
- Sound, graphics, fonts, and artwork.
- Packages that require an additional customer contract.
- Some packages shipped as part of the module *Workstation Extension* are L2-supported only.
- Packages with names ending in `-devel` (containing header files and similar developer resources) will only be supported together with their main packages.

SUSE will only support the usage of original packages. That is, packages that are unchanged and not recompiled.

4.2 Technology previews

Technology previews are packages, stacks, or features delivered by SUSE to provide glimpses into upcoming innovations. Technology previews are included for your convenience to give you a chance to test new technologies within your environment. We would appreciate your feedback! If you test a technology preview, please contact your SUSE representative and let them know about your experience and use cases. Your input is helpful for future development.

Technology previews have the following limitations:

- Technology previews are still in development. Therefore, they may be functionally incomplete, unstable, or in other ways *not* suitable for production use.
- Technology previews are *not* supported.
- Technology previews may only be available for specific hardware architectures.
- Details and functionality of technology previews are subject to change. As a result, upgrading to subsequent releases of a technology preview may be impossible and require a fresh installation.
- SUSE may discover that a preview does not meet customer or market needs, or does not comply with enterprise standards. Technology previews can be removed from a product at any time. SUSE does not commit to providing a supported version of such technologies in the future.

For an overview of technology previews shipped with your product, see the release notes at https://www.suse.com/releasenotes/x86_64/SUSE-Enterprise-Storage/7.

5 Ceph contributors

The Ceph project and its documentation is a result of the work of hundreds of contributors and organizations. See <https://ceph.com/contributors/> for more details.

6 Commands and command prompts used in this guide

As a Ceph cluster administrator, you will be configuring and adjusting the cluster behavior by running specific commands. There are several types of commands you will need:

6.1 Salt-related commands

These commands help you to deploy Ceph cluster nodes, run commands on several (or all) cluster nodes at the same time, or assist you when adding or removing cluster nodes. The most frequently used commands are `ceph-salt` and `ceph-salt config`. You need to run Salt commands on the Salt Master node as `root`. These commands are introduced with the following prompt:

```
root@master #
```

For example:

```
root@master # ceph-salt config ls
```

6.2 Ceph related commands

These are lower-level commands to configure and fine tune all aspects of the cluster and its gateways on the command line, for example `ceph`, `cephadm`, `rbd`, or `radosgw-admin`.

To run Ceph related commands, you need to have read access to a Ceph key. The key's capabilities then define your privileges within the Ceph environment. One option is to run Ceph commands as `root` (or via `sudo`) and use the unrestricted default keyring 'ceph.client.admin.key'. The safer and recommended option is to create a more restrictive individual key for each administrator user and put it in a directory where the users can read it, for example:

```
~/ceph/ceph.client.USERNAME.keyring
```



Tip: Path to Ceph keys

To use a custom admin user and keyring, you need to specify the user name and path to the key each time you run the `ceph` command using the `-n client.USER_NAME` and `--keyring PATH/TO/KEYRING` options.

To avoid this, include these options in the `CEPH_ARGS` variable in the individual users' `~/.bashrc` files.

Although you can run Ceph-related commands on any cluster node, we recommend running them on the Admin Node. This documentation uses the `cephuser` user to run the commands, therefore they are introduced with the following prompt:

```
cephuser@adm >
```

For example:

```
cephuser@adm > ceph auth list
```



Tip: Commands for specific nodes

If the documentation instructs you to run a command on a cluster node with a specific role, it will be addressed by the prompt. For example:

```
cephuser@mon >
```

6.2.1 Running `ceph-volume`

Starting with SUSE Enterprise Storage 7, Ceph services are running containerized. If you need to run `ceph-volume` on an OSD node, you need to prepend it with the `cephadm` command, for example:

```
cephuser@adm > cephadm ceph-volume simple scan
```

6.3 General Linux commands

Linux commands not related to Ceph, such as `mount`, `cat`, or `openssl`, are introduced either with the `cephuser@adm >` or `#` prompts, depending on which privileges the related command requires.

6.4 Additional information

For more information on Ceph key management, refer to [Section 30.2, "Key management"](#).

I Ceph Dashboard

- 1 About the Ceph Dashboard 2
- 2 Dashboard's Web user interface 3
- 3 Manage Ceph Dashboard users and roles 10
- 4 View cluster internals 15
- 5 Manage pools 27
- 6 Manage RADOS Block Device 29
- 7 Manage NFS Ganesha 52
- 8 Manage CephFS 57
- 9 Manage the Object Gateway 59
- 10 Manual configuration 66
- 11 Manage users and roles on the command line 73

1 About the Ceph Dashboard

The Ceph Dashboard is a built-in Web-based Ceph management and monitoring application that administers various aspects and objects of the cluster. The dashboard is automatically enabled after the basic cluster is deployed in *Book "Deployment Guide", Chapter 7 "Deploying the bootstrap cluster using ceph-salt"*.

The Ceph Dashboard for SUSE Enterprise Storage 7 has added more Web-based management capabilities to make it easier to administer Ceph, including monitoring and application administration to the Ceph Manager. You no longer need to know complex Ceph-related commands to manage and monitor your Ceph cluster. You can either use the Ceph Dashboard's intuitive interface, or its built-in REST API.

The Ceph Dashboard module visualizes information and statistics about the Ceph cluster using a Web server hosted by `ceph-mgr`. See *Book "Deployment Guide", Chapter 1 "SES and Ceph", Section 1.2.3 "Ceph nodes and daemons"* for more details on Ceph Manager.

2 Dashboard's Web user interface

2.1 Logging in

To log in to the Ceph Dashboard, point your browser to its URL including the port number. Run the following command to find the address:

```
cephuser@adm > ceph mgr services | grep dashboard
"dashboard": "https://host:port/",
```

The command returns the URL where the Ceph Dashboard is located. If you are having issues with this command, see *Book “Troubleshooting Guide”, Chapter 10 “Troubleshooting the Ceph Dashboard”, Section 10.1 “Locating the Ceph Dashboard”*.

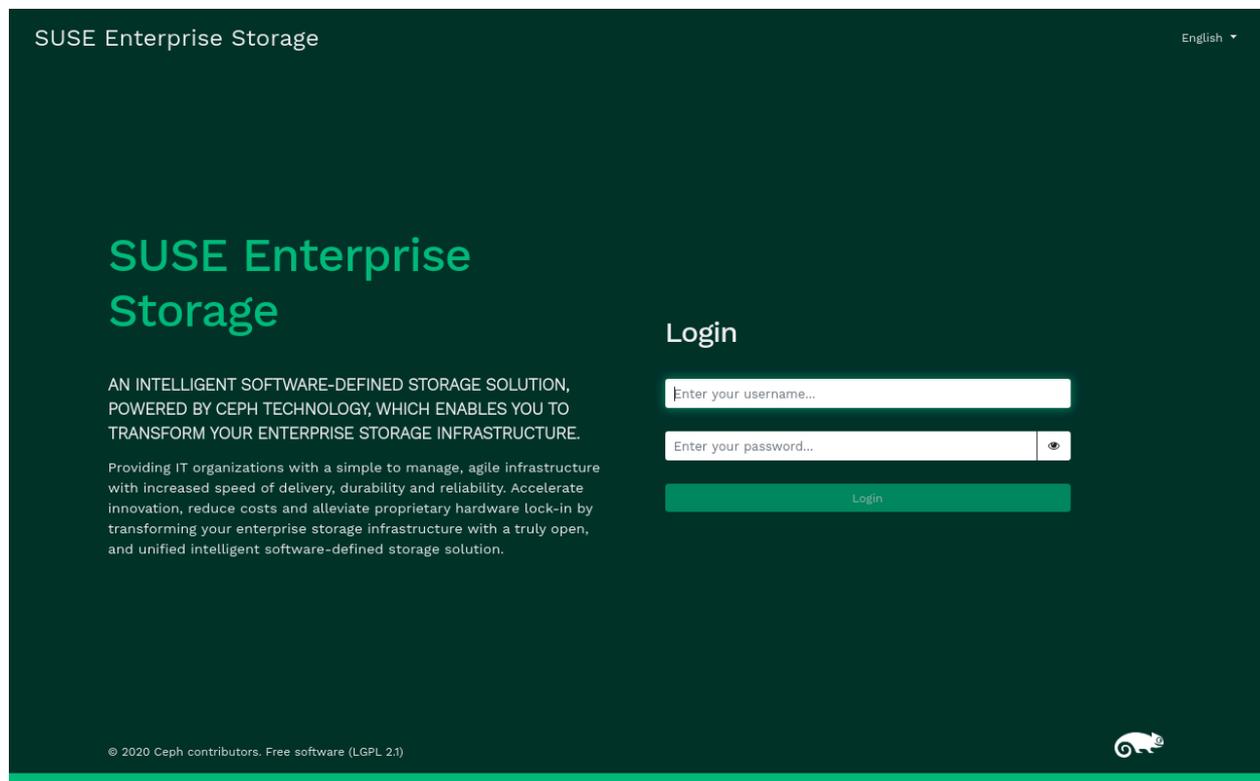


FIGURE 2.1: CEPH DASHBOARD LOGIN SCREEN

Log in by using the credentials that you created during cluster deployment (see *Book “Deployment Guide”, Chapter 7 “Deploying the bootstrap cluster using ceph - salt”, Section 7.2.9 “Configuring the Ceph Dashboard login credentials”*).



Tip: Custom user account

If you do not want to use the default *admin* account to access the Ceph Dashboard, create a custom user account with administrator privileges. Refer to [Chapter 11, Manage users and roles on the command line](#) for more details.

The dashboard user interface is graphically divided into several *blocks*: the *utility menu* in the top right-hand side of the screen, the *main menu* on the left-hand side, and the main *content pane*.

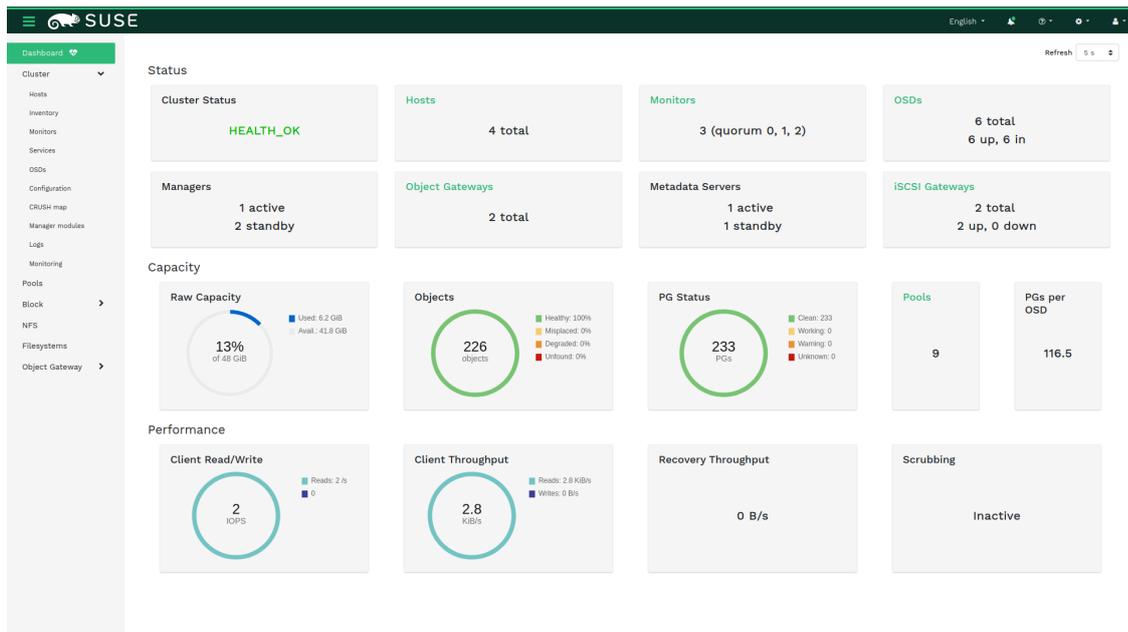


FIGURE 2.2: CEPH DASHBOARD HOME PAGE

2.2 Utility menu

The top right-hand side of the screen contains a utility menu. It includes general tasks related more to the dashboard than to the Ceph cluster. By clicking the options, you can access the following topics:

- Change the dashboard's language interface to: Czech, German, English, Spanish, French, Indonesian, Italian, Japanese, Korean, Polish, Portuguese (Brazilian), and Chinese.
- Tasks and notifications
- View the documentation, information about the REST API, or further information about the dashboard.

- User management and telemetry configuration.



Note

For more detailed command line descriptions for user roles, see [Chapter 11, Manage users and roles on the command line](#).

- Log in configuration; change the password or sign out.

2.3 Main menu

The dashboard's main menu occupies the left-hand side of the screen. It covers the following topics:

Dashboard

Return to Ceph Dashboard's home page.

Cluster

View detailed information about hosts, inventory, Ceph Monitors, services, Ceph OSDs, cluster configuration, CRUSH Map, Ceph Manager modules, logs, and monitoring.

Pools

View and manage cluster pools.

Block

View detailed information and manage RADOS Block Device images, mirroring, and iSCSI.

NFS

View and manage NFS Ganesha deployments.



Note

If NFS Ganesha is not deployed, an information notice appears. See [Section 11.6, "Configuring NFS Ganesha in the Ceph Dashboard"](#).

Filesystems

View and manage CephFSs.

Object Gateway

View and manage Object Gateway's daemons, users, and buckets.



Note

If Object Gateway is not deployed, an informative notice appears. See [Section 10.4](#), “Enabling the Object Gateway management front-end”.

2.4 Content pane

The content pane occupies the main part of the dashboard's screen. The dashboard home page shows plenty of helpful widgets to inform you briefly about the current status of the cluster, capacity, and performance information.

2.5 Common Web UI features

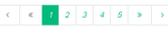
In Ceph Dashboard, you often work with *lists*—for example, lists of pools, OSD nodes, or RBD devices. All lists will automatically refresh themselves by default every five seconds. The following common widgets help you manage or adjust these list:

Click  to trigger a manual refresh of the list.

Click  to display or hide individual table columns.

Click  and enter (or select) how many rows to display on a single page.

Click inside  and filter the rows by typing the string to search for.

Use  to change the currently displayed page if the list spans across multiple pages.

2.6 Dashboard widgets

Each dashboard widget shows specific status information related to a specific aspect of a running Ceph cluster. Some widgets are active links and after clicking them, they will redirect you to a related detailed page of the topic they represent.



Tip: More details on mouse over

Some graphical widgets show you more detail when you move the mouse over them.

2.6.1 Status widgets

Status widgets give you a brief overview about the cluster's current status.



FIGURE 2.3: STATUS WIDGETS

Cluster Status

Presents basic information about the cluster's health.

Hosts

Shows the total number of cluster nodes.

Monitors

Shows the number of running monitors and their quorum.

OSDs

Shows the total number of OSDs, as well as the number of *up* and *in* OSDs.

Managers

Shows the number of active and standby Ceph Manager daemons.

Object Gateways

Shows the number of running Object Gateways.

Metadata Servers

Shows the number of Metadata Servers.

iSCSI Gateways

Shows the number of configured iSCSI gateways.

2.6.2 Capacity widgets

Capacity widgets show brief information about the storage capacity.

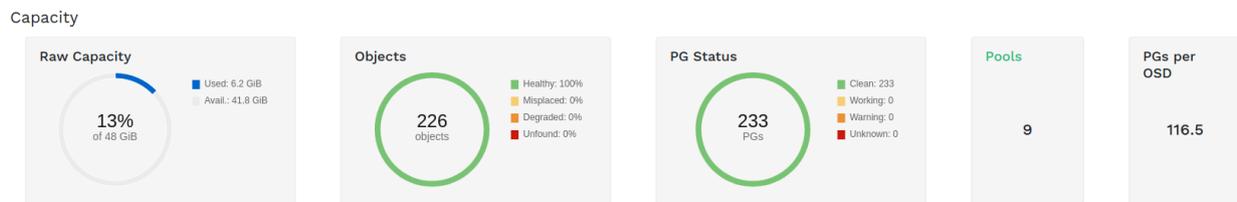


FIGURE 2.4: CAPACITY WIDGETS

Raw Capacity

Shows the ratio of used and available raw storage capacity.

Objects

Shows the number of data objects stored in the cluster.

PG Status

Displays a chart of the placement groups according to their status.

Pools

Shows the number of pools in the cluster.

PGs per OSD

Shows the average number of placement groups per OSD.

2.6.3 Performance widgets

Performance widgets refer to basic performance data of Ceph clients.

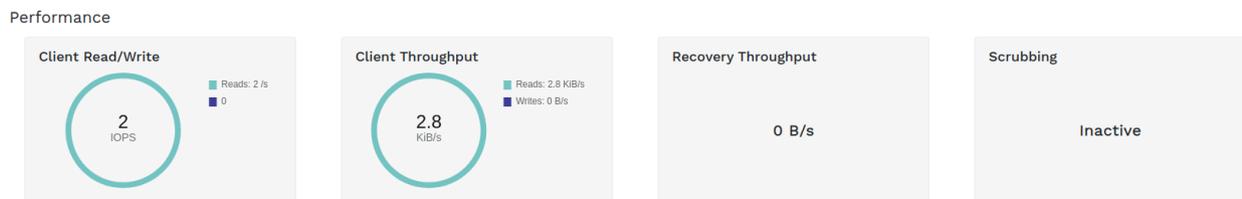


FIGURE 2.5: PERFORMANCE WIDGETS

Client Read/Write

The amount of clients' read and write operations per second.

Client Throughput

The amount of data transferred to and from Ceph clients in bytes per second.

Recovery Throughput

The throughput of data recovered per second.

Scrubbing

Shows the scrubbing (see [Section 17.4.9, "Scrubbing a placement group"](#)) status. It is either inactive, enabled, or active.

3 Manage Ceph Dashboard users and roles

Dashboard user management performed by Ceph commands on the command line was already introduced in [Chapter 11, Manage users and roles on the command line](#).

This section describes how to manage user accounts by using the Dashboard Web user interface.

3.1 Listing users

Click  in the utility menu and select *User Management*.

The list contains each user's user name, full name, e-mail, a list of assigned roles, whether the role is enabled, and the password expiration date.



The screenshot shows the 'Users' management interface. At the top, there are tabs for 'Users' and 'Roles'. Below the tabs is a '+ Create' button. The main area contains a table with the following columns: Username, Name, Email, Roles, Enabled, and Password expiration date. The table lists four users: 'admin', 'Alex', 'dashboard user 1', and 'rgw user'. Each user has a corresponding name, email address, and a list of roles. The 'Enabled' column shows a checkmark for all users. The 'Password expiration date' column is empty for all users. At the bottom left of the table, it says '0 selected / 4 total'.

Username	Name	Email	Roles	Enabled	Password expiration date
admin			administrator	✓	
Alex	Alexandra Settle	tux@example.com	cluster-manager, pool-manager	✓	
dashboard user 1	Dashboard User1	du1@example.com		✓	
rgw user	RGW User	rgw@example.com	pool-manager, rgw-manager	✓	

FIGURE 3.1: USER MANAGEMENT

3.2 Adding new users

Click *Create* in the top left of the table heading to add a new user. Enter their user name, password, and optionally a full name and an e-mail.

Create User

Username *	potato ✓
Password ⓘ ✓ 👁
Confirm password ✓ 👁
Password expiration date ⓘ	Password expiration date... ✕
Full name	Mr Potato ✓
Email	potato@example.com ✓
Roles	✎ There are no roles.
	<input checked="" type="checkbox"/> Enabled
	<input checked="" type="checkbox"/> User must change password at next logon

Create User
Cancel

FIGURE 3.2: ADDING A USER

Click the little pen icon to assign predefined roles to the user. Confirm with *Create User*.

3.3 Editing users

Click a user's table row to highlight the selection Select *Edit* to edit details about the user. Confirm with *Edit User*.

3.4 Deleting users

Click a user's table row to highlight the selection Select the drop-down button next to *Edit* and select *Delete* from the list to delete the user account. Activate the *Yes, I am sure* check box and confirm with *Delete User*.

3.5 Listing user roles

Click  in the utility menu and select *User Management*. Then click the *Roles* tab.

The list contains each role's name, description, and whether it is a system role.



The screenshot shows the 'Roles' tab in the Ceph User Management interface. It features a table with columns for Name, Description, and System Role. The table lists eight roles, all of which are system roles. The 'Create' button is visible in the top left of the table area.

Name	Description	System Role
administrator	Administrator	✓
block-manager	Block Manager	✓
cephfs-manager	CephFS Manager	✓
cluster-manager	Cluster Manager	✓
ganeshha-manager	NFS Ganeshha Manager	✓
pool-manager	Pool Manager	✓
read-only	Read-Only	✓
rgw-manager	RGW Manager	✓

0 selected / 8 total

FIGURE 3.3: USER ROLES

3.6 Adding custom roles

Click *Create* in the top left of the table heading to add a new custom role. Enter the *Name* and *Description* and next to *Permissions*, select the appropriate permissions.



Tip: Purging custom roles

If you create custom user roles and intend to remove the Ceph cluster with the `ceph-salt purge` command later on, you need to purge the custom roles first. Find more details in [Section 13.9, "Removing an entire Ceph cluster"](#).

Create Role

Name * ✓

Description ✓

<input type="checkbox"/> All	<input type="checkbox"/> Read	<input type="checkbox"/> Creat	<input type="checkbox"/> Upda	<input type="checkbox"/> Delet
<input type="checkbox"/> cephfs	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> config-opt	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> dashboard-settings	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> grafana	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> hosts	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> iscsi	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> log	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> manager	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> monitor	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/> nfs-ganesha	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/> osd	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> pool	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> prometheus	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> rbd-image	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> rbd-mirroring	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> rgw	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> user	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

FIGURE 3.4: ADDING A ROLE



Tip: Multiple activation

By activating the check box that precedes the topic name, you activate all permissions for that topic. By activating the *All* check box, you activate all permissions for all the topics.

Confirm with *Create Role*.

3.7 Editing custom roles

Click a user's table row to highlight the selection Select *Edit* in the top left of the table heading to edit a description and permissions of the custom role. Confirm with *Edit Role*.

3.8 Deleting custom roles

Click a role's table row to highlight the selection Select the drop-down button next to *Edit* and select *Delete* from the list to delete the role. Activate the *Yes, I am sure* check box and confirm with *Delete Role*.

4 View cluster internals

The *Cluster* menu item lets you view detailed information about Ceph cluster hosts, inventory, Ceph Monitors, services, OSDs, configuration, CRUSH Map, Ceph Manager, logs, and monitoring files.

4.1 Viewing cluster nodes

Click *Cluster* > *Hosts* to view a list of cluster nodes.



The screenshot shows the 'Hosts List' interface with a table of cluster nodes. The table has columns for Hostname, Services, Labels, and Version. There are four rows: 'master', 'node1', 'node2', and 'node3'. Each node row has a right-pointing arrow next to the hostname. The 'Services' column lists daemons for each node. The 'Version' column shows the Ceph version for each node.

Hostname	Services	Labels	Version
master			
node1	mgr.node1.wbnmqa, mon.node1, osd.0, osd.3		15.2.4-557-g4ac763f0b3
node2	mgr.node2.qcwalk, mon.node2, osd.1, osd.4		15.2.4-557-g4ac763f0b3
node3	mgr.node3.rhkzzy, mon.node3, osd.2, osd.5		15.2.4-557-g4ac763f0b3

FIGURE 4.1: HOSTS

Click the drop-down arrow next to a node name in the *Hostname* column to view the performance details of the node.

The *Services* column lists all daemons that are running on each related node. Click a daemon name to view its detailed configuration.

4.2 Accessing cluster inventory

Click *Cluster* > *Inventory* to view a list of devices. The list includes the device path, type, availability, vendor, model, size, and the OSDs.

Click to select a node name in the *Hostname* column. When selected, click *Identify* to identify the device the host is running on. This tells the device to blink its LEDs. Select the duration of this action between 1, 2, 5, 10, or 15 minutes. Click *Execute*.

Devices

Hostname	Device path	Type	Available	Vendor	Model	Size	OSDs
master	/dev/vda	HDD		0x1af4		42 GiB	
node1	/dev/vda	HDD		0x1af4		42 GiB	
node1	/dev/vdb	HDD		0x1af4		8 GiB	osd.0
node1	/dev/vdc	HDD		0x1af4		8 GiB	osd.3
node2	/dev/vda	HDD		0x1af4		42 GiB	
node2	/dev/vdb	HDD		0x1af4		8 GiB	osd.1
node2	/dev/vdc	HDD		0x1af4		8 GiB	osd.4
node3	/dev/vda	HDD		0x1af4		42 GiB	
node3	/dev/vdb	HDD		0x1af4		8 GiB	osd.2
node3	/dev/vdc	HDD		0x1af4		8 GiB	osd.5

0 selected / 10 total

FIGURE 4.2: SERVICES

4.3 Viewing Ceph Monitors

Click *Cluster > Monitors* to view a list of cluster nodes with running Ceph monitors. The content pane is split into two views: Status, and In Quorum or Not In Quorum.

The *Status* table shows general statistics about the running Ceph Monitors, including the following:

- Cluster ID
- monmap modified
- monmap epoch
- quorum con
- quorum mon
- required con
- required mon

The In Quorum and Not In Quorum panes include each monitor's name, rank number, public IP address, and number of open sessions.

Click a node name in the *Name* column to view the related Ceph Monitor configuration.

Status	
Cluster ID	42137a54-dd71-11ea-9caa-525400d84504
monmap modified	2020-08-14T08:43:26.524800Z
monmap epoch	3
quorum con	4540138292836696063
quorum mon	kraken,luminous,mimic,osdmap-prune,nautilus,octopus
required con	2449858747315978244
required mon	kraken,luminous,mimic,osdmap-prune,nautilus,octopus

In Quorum

Name	Rank	Public Address	Open Sessions
node1	0	10.20.156.201:6789/0
node2	2	10.20.156.202:6789/0
node3	1	10.20.156.203:6789/0
3 total			

Not In Quorum

Name	Rank	Public Address
No data to display		
0 total		

FIGURE 4.3: CEPH MONITORS

4.4 Displaying services

Click *Cluster > Services* to view details on each of the available services: `crash`, Ceph Manager, and Ceph Monitors. The list includes the container image name, container image ID, status of what is running, size, and when it was last refreshed.

Click the drop-down arrow next to a service name in the *Service* column to view details of the daemon. The detail list includes the host name, daemon type, daemon ID, container ID, container image name, container image ID, version number, status, and when it was last refreshed.

Cluster > Services

Service	Container image name	Container image ID	Running	Size	Last Refreshed
crash	registry.suse.de/devel/storage/7.0/containers/ses7/ceph/ceph:latest	6549871c3f67	4		2020-08-14T13:37:34.148847

Hostname	Daemon type	Daemon ID	Container ID	Container image name	Container image ID	Version	Status	Last Refreshed
master	crash	master	3acfc11b607e	registry.suse.de/devel/storage/7.0/containers/ses	6549871c3f67	15.2.4.557	running	2020-08-14T13:37:34.148847
node1	crash	node1	3d56e2a421eb	registry.suse.de/devel/storage/7.0/containers/ses	6549871c3f67	15.2.4.557	running	2020-08-14T13:37:35.371944
node2	crash	node2	8fa9790b9a51	registry.suse.de/devel/storage/7.0/containers/ses	6549871c3f67	15.2.4.557	running	2020-08-14T13:37:35.208871
node3	crash	node3	bd47531bbf2a	registry.suse.de/devel/storage/7.0/containers/ses	6549871c3f67	15.2.4.557	running	2020-08-14T13:37:35.965886
4 total								

>	mgr	registry.suse.de/devel/storage/7.0/containers/ses7/ceph/ceph:latest	6549871c3f67		3	2020-08-14T13:37:35.209899
>	mon	registry.suse.de/devel/storage/7.0/containers/ses7/ceph/ceph:latest	6549871c3f67		3	2020-08-14T13:37:35.210052
>	osd.sesdev_osd_deploym				3	

1 selected / 4 total

FIGURE 4.4: SERVICES

4.5 Displaying Ceph OSDs

Click *Cluster* > *OSDs* to view a list of nodes with running OSD daemons. The list includes each node's name, ID, status, device class, number of placement groups, size, usage, reads/writes chart in time, and the rate of read/write operations per second.



The screenshot shows the 'OSDs List' interface with a table of OSDs. The table has the following columns: Host, ID, Status, Device class, PGs, Size, Usage, Read bytes, Write bytes, Read ops, and Write ops. There are 6 rows of data, each representing an OSD on a different node. The status for all OSDs is 'in up'. The device class is 'hdd'. The size is 8 GiB. The usage is 13%. The read and write rates are 0 /s.

	Host	ID	Status	Device class	PGs	Size	Usage	Read bytes	Write bytes	Read ops	Write ops
<input type="checkbox"/>	> node1	0	in up	hdd	0	8 GiB	13%	0 /s	0 /s
<input type="checkbox"/>	> node2	1	in up	hdd	1	8 GiB	13%	0 /s	0 /s
<input type="checkbox"/>	> node3	2	in up	hdd	1	8 GiB	13%	0 /s	0 /s
<input type="checkbox"/>	> node1	3	in up	hdd	1	8 GiB	13%	0 /s	0 /s
<input type="checkbox"/>	> node2	4	in up	hdd	1	8 GiB	13%	0 /s	0 /s
<input type="checkbox"/>	> node3	5	in up	hdd	0	8 GiB	13%	0 /s	0 /s

0 selected / 6 total

FIGURE 4.5: CEPH OSDS

Select *Flags* from the *Cluster-wide configuration* drop-down menu in the table heading to open a pop-up window. This has a list of flags that apply to the whole cluster. You can activate or deactivate individual flags, and confirm with *Submit*.

×
Cluster-wide OSD Flags

No In
OSDs that were previously marked out will not be marked back in when they start

No Out
OSDs will not automatically be marked out after the configured interval

No Up
OSDs are not allowed to start

No Down
OSD failure reports are being ignored, such that the monitors will not mark OSDs down

Pause
Pauses reads and writes

No Scrub
Scrubbing is disabled

No Deep Scrub
Deep Scrubbing is disabled

No Backfill
Backfilling of PGs is suspended

No Rebalance
OSD will choose not to backfill unless PG is also degraded

No Recover
Recovery of PGs is suspended

Bitwise Sort
Use bitwise sort

Submit
Cancel

FIGURE 4.6: OSD FLAGS

Select *Recovery Priority* from the *Cluster-wide configuration* drop-down menu in the table heading to open a pop-up window. This has a list of OSD recovery priorities that apply to the whole cluster. You can activate the preferred priority profile and fine-tune the individual values below. Confirm with *Submit*.

OSD Recovery Priority ✕

Priority * Custom ⇅

Customize priority values

Max Backfills * ? 1

Recovery Max Active * ? 0

Recovery Max Single Start * 1

Recovery Sleep * ? 0

Submit
Cancel

FIGURE 4.7: OSD RECOVERY PRIORITY

Click the drop-down arrow next to a node name in the *Host* column to view an extended table with details about the device settings and performance. Browsing through several tabs, you can see lists of *Attributes*, *Metadata*, *Device health*, *Performance counter*, a graphical *Histogram* of reads and writes, and *Performance details*.

OSDs List
Overall Performance

Edit
Cluster-wide configuration

✓	Host	ID	Status	Device class	PGs	Size	Usage	Read bytes	Write bytes	Read ops	Write ops
<input checked="" type="checkbox"/>	node1	0	In Up	hdd	0	8 GiB	13%	0 / s	0 / s

Devices
Attributes (OSD map)
Metadata
Device health
Performance counter
Histogram
Performance Details

Name	Description	Value
bluefs.bytes_written_slow	Bytes written to WAL/SSTs at slow device	0
bluefs.bytes_written_sst	Bytes written to SSTs	0
bluefs.bytes_written_wal	Bytes written to WAL	0
bluefs.db_total_bytes	Total bytes (main db device)	1073741824
bluefs.db_used_bytes	Used bytes (main db device)	4980736
bluefs.log_bytes	Size of the metadata log	163840
bluefs.logged_bytes	Bytes written to the metadata log	0
bluefs.num_files	File count	10
bluefs.read_bytes	Bytes requested in buffered read mode	0
bluefs.read_prefetch_bytes	Bytes requested in prefetch read mode	0

112 total

1
2
3
4
5

FIGURE 4.8: OSD DETAILS



Tip: Performing specific tasks on OSDs

After you click an OSD node name, the table row is highlighted. This means that you can now perform a task on the node. You can choose to perform any of the following actions: *Edit*, *Create*, *Scrub*, *Deep Scrub*, *Reweight*, *Mark out*, *Mark In*, *Mark Down*, *Mark Lost*, *Purge*, *Destroy*, or *Delete*.

Click the down arrow in the top left of the table heading next to the *Create* button and select the task you want to perform.

4.5.1 Adding OSDs

To add new OSDs, follow these steps:

1. Verify that some cluster nodes have storage devices whose status is available. Then click the down arrow in the top left of the table heading and select *Create*. This opens the *Create OSDs* window.

The screenshot shows a dialog box titled "Create OSDs". It contains several sections for configuring storage devices. The "Primary devices" section has a "+ Add" button. The "Shared devices" section includes "WAL devices" and "DB devices", each with a "+ Add" button. The "Configuration" section has a "Features" section with an "Encryption" checkbox. At the bottom right, there are "Preview" and "Cancel" buttons.

FIGURE 4.9: CREATE OSDS

2. To add primary storage devices for OSDs, click *Add*. Before you can add storage devices, you need to specify filtering criteria in the top right of the *Primary devices* table—for example *Type hdd*. Confirm with *Add*.



FIGURE 4.10: ADDING PRIMARY DEVICES

3. In the updated *Create OSDs* window, optionally add shared WAL and BD devices, or enable device encryption.

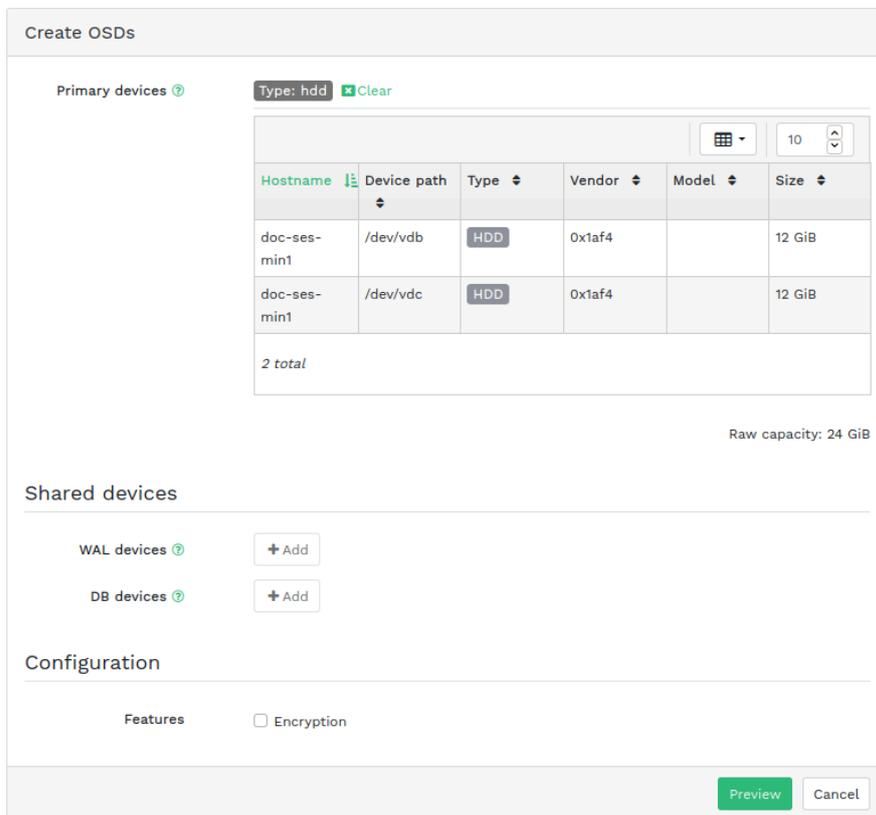


FIGURE 4.11: CREATE OSDS WITH PRIMARY DEVICES ADDED

4. Click *Preview* to view the preview of DriveGroups specification for the devices you previously added. Confirm with *Create*.

OSD creation preview
✕

DriveGroups

```

[
  {
    "service_type": "osd",
    "service_id": "dashboard-admin-1600784434446",
    "host_pattern": "*",
    "data_devices": {
      "rotational": true
    }
  }
]

```

Create
Cancel

FIGURE 4.12:

5. New devices will be added to the list of OSDs.

	Host	ID	Status	Device class	PGs	Size	Usage	Read bytes	Write bytes	Read ops	Write ops
<input type="checkbox"/>	> doc-ses-min2	0	in up	hdd	119	10 GiB	<div style="width: 11%; background-color: #4CAF50; height: 8px;"></div> 11%			0.7999105934891158 /s	0 /s
<input type="checkbox"/>	> doc-ses-min3	1	in up	hdd	108	10 GiB	<div style="width: 11%; background-color: #4CAF50; height: 8px;"></div> 11%			1.5998816768416986 /s	0 /s
<input type="checkbox"/>	> doc-ses-min4	2	in up	hdd	126	10 GiB	<div style="width: 11%; background-color: #4CAF50; height: 8px;"></div> 11%			0 /s	0 /s
<input type="checkbox"/>	> doc-ses-min1	3	in up	hdd	96	12 GiB	<div style="width: 9%; background-color: #4CAF50; height: 8px;"></div> 9%			0.3999455526088382 /s	0 /s
<input type="checkbox"/>	> doc-ses-min1	4	in up	hdd	76	12 GiB	<div style="width: 9%; background-color: #4CAF50; height: 8px;"></div> 9%			1.9995708432976873 /s	0 /s

0 selected / 5 total

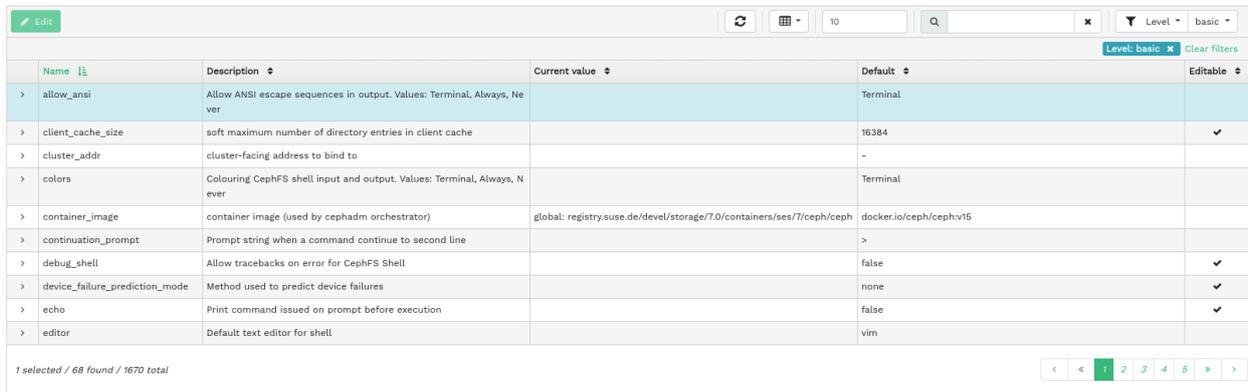
FIGURE 4.13: NEWLY ADDED OSDS

Note

There is no progress visualization of the OSD creation process. It takes some time before they are actually created. The OSDs will appear in the list when they have been deployed. If you want to check the deployment status, view the logs by clicking *Cluster > Logs*.

4.6 Viewing cluster configuration

Click *Cluster > Configuration* to view a complete list of Ceph cluster configuration options. The list contains the name of the option, its short description, and its current and default values, and whether the option is editable.



The screenshot shows a web interface for viewing Ceph cluster configuration. At the top left is an 'Edit' button. The main area is a table with columns: Name, Description, Current value, Default, and Editable. The table lists various configuration options like 'allow_ansi', 'client_cache_size', 'cluster_addr', etc. The first row is highlighted in light blue. At the bottom left, it says '1 selected / 68 found / 1670 total'. At the bottom right, there are pagination controls showing page 1 of 8.

Name	Description	Current value	Default	Editable
> allow_ansi	Allow ANSI escape sequences in output. Values: Terminal, Always, Never		Terminal	
> client_cache_size	soft maximum number of directory entries in client cache		16384	✓
> cluster_addr	cluster-facing address to bind to		-	
> colors	Colouring CephFS shell input and output. Values: Terminal, Always, Never		Terminal	
> container_image	container image (used by cephadm orchestrator)	global: registry.suse.de/develop/storage/7.0/containers/ces/7/ceph/ceph	docker.io/ceph/ceph:v15	
> continuation_prompt	Prompt string when a command continue to second line		>	
> debug_shell	Allow tracebacks on error for CephFS Shell		false	✓
> device_failure_prediction_mode	Method used to predict device failures		none	✓
> echo	Print command issued on prompt before execution		false	✓
> editor	Default text editor for shell		vim	

FIGURE 4.14: CLUSTER CONFIGURATION

Click the drop-down arrow next to a configuration option in the *Name* column to view an extended table with detailed information about the option, such as its type of value, minimum and maximum permitted values, whether it can be updated at runtime, and many more.

After highlighting a specific option, you can edit its value(s) by clicking the *Edit* button in the top left of the table heading. Confirm changes with *Save*.

4.7 Viewing the CRUSH Map

Click *Cluster > CRUSH map* to view a CRUSH Map of the cluster. For more general information on CRUSH Maps, refer to [Section 17.5, “CRUSH Map manipulation”](#).

Click the root, nodes, or individual OSDs to view more detailed information, such as crush weight, depth in the map tree, device class of the OSD, and many more.

CRUSH map viewer

- default (root)
 - node3 (host)
 - osd.2 (osd)
 - osd.5 (osd)
 - node1 (host)
 - osd.0 (osd)
 - osd.3 (osd)
 - node2 (host)
 - osd.1 (osd)
 - osd.4 (osd)

osd.2 (osd)

crush_weight	0.0077972412109375
depth	2
device_class	hdd
exists	1
id	2
primary_affinity	1
reweight	1
type_id	0

FIGURE 4.15: CRUSH MAP

4.8 Viewing manager modules

Click *Cluster > Manager modules* to view a list of available Ceph Manager modules. Each line consists of a module name and information on whether it is currently enabled or not.

Name	Enabled
ansible	
balancer	
crash	
dashboard	✓
deepsea	
devicehealth	
diskprediction_local	
influx	
insights	
iostat	✓

1 selected / 24 total

FIGURE 4.16: MANAGER MODULES

Click the drop-down arrow next to a module in the *Name* column to view an extended table with detailed settings in the *Details* table below. Edit them by clicking the *Edit* button in the top left of the table heading. Confirm changes with *Update*.

Click the drop-down arrow next to the *Edit* button in the top left of the table heading to *Enable* or *Disable* a module.

4.9 Viewing logs

Click *Cluster* > *Logs* to view a list of cluster's recent log entries. Each line consists of a time stamp, the type of the log entry, and the logged message itself.

Click the *Audit Logs* tab to view log entries of the auditing subsystem. Refer to [Section 11.5, "Auditing API requests"](#) for commands to enable or disable auditing.



```
Cluster Logs Audit Logs

2019-04-04 02:53:02.989912 [INF] from='mgr.3269933 10.100.24.61:0/27247' entity='mgr.doc-ses6min2'
cmd=[{"prefix": "osd in", "format": "json", "ids": ["8"]}]:
finished

2019-04-04 02:53:02.240046 [INF] from='mgr.3269933 10.100.24.61:0/27247' entity='mgr.doc-ses6min2'
cmd=[{"prefix": "osd in", "format": "json", "ids": ["8"]}]:
dispatch

2019-04-04 02:52:34.282994 [INF] from='mgr.3269933 10.100.24.61:0/27247' entity='mgr.doc-ses6min2'
cmd=[{"prefix": "osd purge-actual", "format": "json", "id": 4,
"yes_i_really_mean_it": true}]: finished

2019-04-04 02:52:33.946127 [INF] from='mgr.3269933 10.100.24.61:0/27247' entity='mgr.doc-ses6min2'
cmd=[{"prefix": "osd purge-actual", "format": "json", "id": 4,
"yes_i_really_mean_it": true}]: dispatch

2019-04-04 02:52:09.570264 [INF] from='mgr.3269933 10.100.24.61:0/27247' entity='mgr.doc-ses6min2'
cmd=[{"prefix": "osd in", "format": "json", "ids": ["4"]}]:
finished
```

FIGURE 4.17: LOGS

4.10 Viewing monitoring

Click *Cluster* > *Monitoring* to manage and view details on Prometheus alerts.

If you have Prometheus active, in this content pane you can view detailed information on *Active Alerts*, *All Alerts*, or *Silences*.



Note

If you do not have Prometheus deployed, an information banner will appear and link to relevant documentation.

5 Manage pools



Tip: More information on pools

For more general information about Ceph pools, refer to [Chapter 18, Manage storage pools](#).

For information specific to erasure code pools, refer to [Chapter 19, Erasure coded pools](#).

To list all available pools, click *Pools* from the main menu.

The list shows each pool's name, type, related application, placement group status, replica size, last change, erasure coded profile, crush ruleset, usage, and read/write statistics.

Name	Type	Applications	PG Status	Replic Size	Last Chan	Erasure Coded Profile	Crush Ruleset	Usage	Read bytes	Write bytes	Read ops	Write ops
.rgw.root	replicated	rgw	8 active+clean	3	22		replicated_rule	0%			0 /s	0 /s
cephfs_data	replicated	cephfs	256 active+clean	3	209		replicated_rule	0%			0 /s	0 /s
cephfs_metadata	replicated	cephfs	64 active+clean	3	210		replicated_rule	0%			0 /s	0 /s
default.rgw.buckets.index	replicated	rgw	8 active+clean	3	75		replicated_rule	0%			0 /s	0 /s
default.rgw.control	replicated	rgw	8 active+clean	3	25		replicated_rule	0%			0 /s	0 /s
default.rgw.log	replicated	rgw	8 active+clean	3	30		replicated_rule	0%			0 /s	0 /s
default.rgw.meta	replicated	rgw	8 active+clean	3	28		replicated_rule	0%			0 /s	0 /s
family_photos	replicated	cephfs	128 active+clean	3	226		replicated_rule	0%			0 /s	0 /s
testing_rbd_pool	replicated	cephfs,rbd	128 active+clean	3	76		replicated_rule	0%			0.8 /s	0 /s

FIGURE 5.1: LIST OF POOLS

Click the drop-down arrow next to a pool name in the *Name* column to view an extended table with detailed information on the pool, such as the general details, performance details, and configuration.

5.1 Adding a new pool

To add a new pool, click *Create* in the top left of the pools table. In the pool form you can enter the pool's name, type, its applications, compression mode, and quotas including maximum bytes and maximum objects. The pool form itself pre-calculates the number of placement groups that

best suited to this specific pool. The calculation is based on the amount of OSDs in the cluster and the selected pool type with its specific settings. As soon as a placement groups number is set manually, it will be replaced by a calculated number. Confirm with *Create Pool*.

The screenshot shows a 'Create Pool' form with the following fields and values:

- Name *: potato-pool
- Pool type *: replicated
- PG Autoscale: on
- Replicated size *: 3
- Applications: cephfs
- CRUSH ruleset: replicated_rule
- Compression Mode: none
- Max bytes: e.g., 10GiB
- Max objects: 0

Buttons: Create Pool, Cancel

FIGURE 5.2: ADDING A NEW POOL

5.2 Deleting pools

To delete a pool, select the pool in the table row. Click the drop-down arrow next to the *Create* button and click *Delete*.

5.3 Editing a pool's options

To edit a pool's options, select the pool in the table row and click *Edit* in the top left of the pools table.

You can change the name of the pool, increase the number of placement groups, change the list of the pool's applications and compression settings. Confirm with *Edit Pool*.

6 Manage RADOS Block Device

To list all available RADOS Block Devices (RBDs), click *Block > Images* from the main menu.

The list shows brief information about the device, such as the device's name, the related pool name, namespace, size of the device, number and size of objects on the device, details on the provisioning of the details, and the parent.



Name	Pool	Namespace	Size	Objects	Object size	Provisioned	Total provisioned	Parent
example_rbd_device	rbd		4 MIB	1	4 MIB	0 B	0 B	-
potato-rbd	rbd		10 MIB	3	4 MIB	0 B	0 B	-

0 selected / 2 total

FIGURE 6.1: LIST OF RBD IMAGES

6.1 Viewing details about RBDs

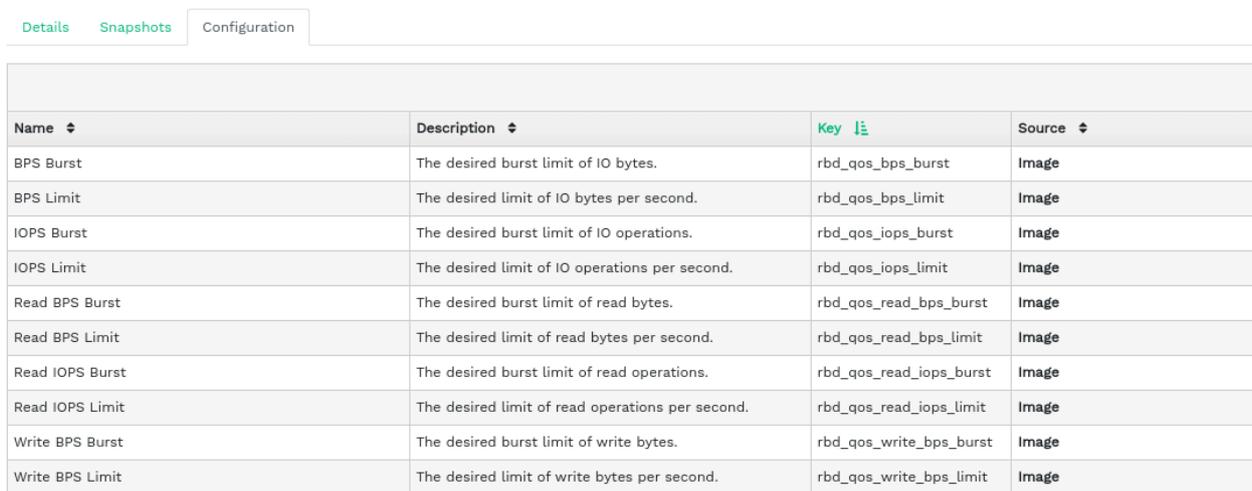
To view more detailed information about a device, click its row in the table:

Details Snapshots Configuration	
Name	rbd-potato
Pool	rbd
Data Pool	-
Created	8/21/20 1:01:59 PM
Size	10 GiB
Objects	2.6 k
Object size	4 MiB
Features	<code>deep-flatten</code> <code>exclusive-lock</code> <code>fast-diff</code> <code>layering</code> <code>object-map</code>
Provisioned	0 B
Total provisioned	0 B
Striping unit	4 MiB
Striping count	1
Parent	-
Block name prefix	rbd_data.38fa6b6fc53
Order	22
Format Version	2

FIGURE 6.2: RBD DETAILS

6.2 Viewing RBD's configuration

To view detailed configuration of a device, click its row in the table and then the *Configuration* tab in the lower table:



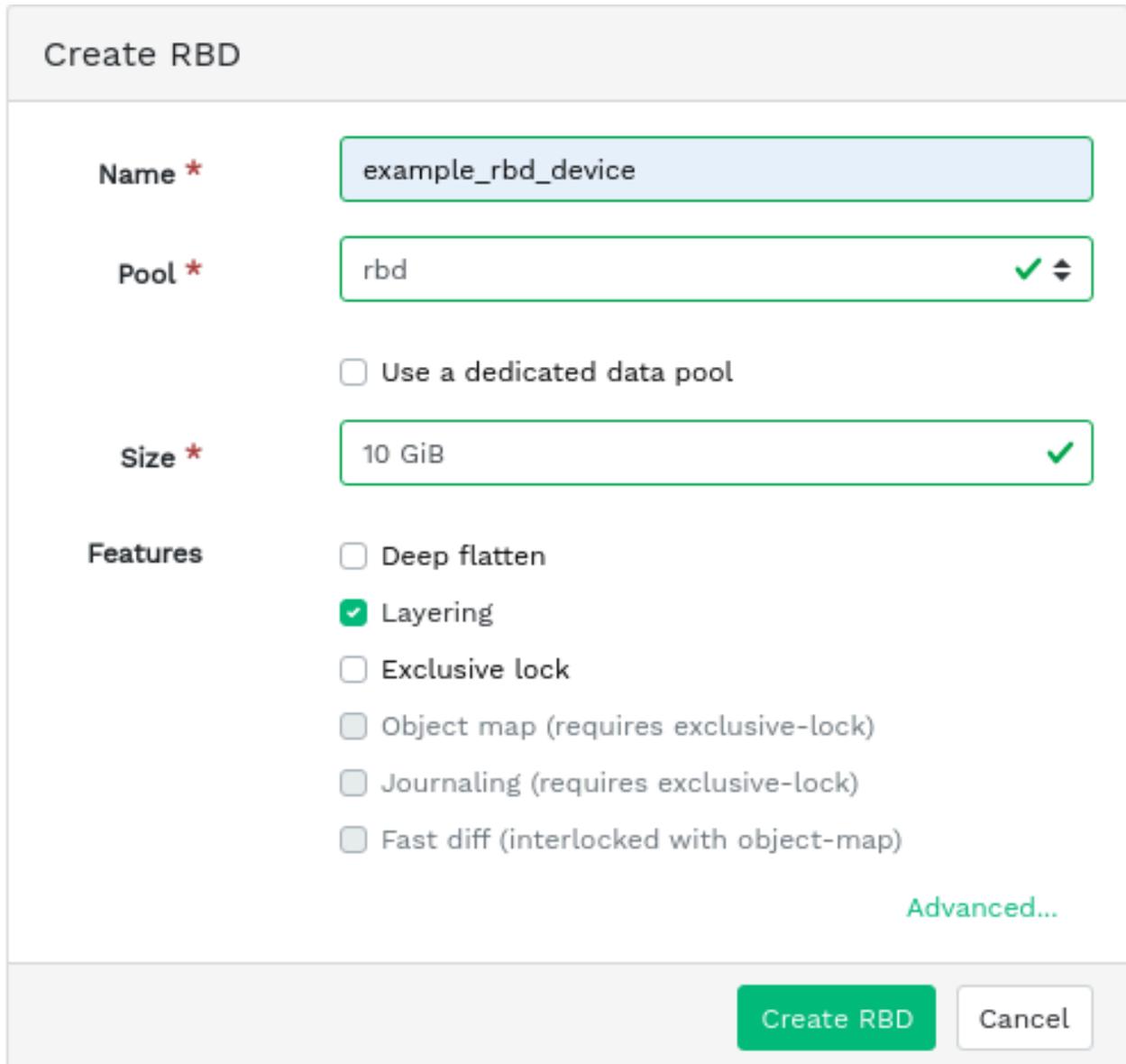
The screenshot shows a web interface with three tabs: 'Details', 'Snapshots', and 'Configuration'. The 'Configuration' tab is active, displaying a table with the following data:

Name ↕	Description ↕	Key ↗	Source ↕
BPS Burst	The desired burst limit of IO bytes.	rbd_qos_bps_burst	Image
BPS Limit	The desired limit of IO bytes per second.	rbd_qos_bps_limit	Image
IOPS Burst	The desired burst limit of IO operations.	rbd_qos_iops_burst	Image
IOPS Limit	The desired limit of IO operations per second.	rbd_qos_iops_limit	Image
Read BPS Burst	The desired burst limit of read bytes.	rbd_qos_read_bps_burst	Image
Read BPS Limit	The desired limit of read bytes per second.	rbd_qos_read_bps_limit	Image
Read IOPS Burst	The desired burst limit of read operations.	rbd_qos_read_iops_burst	Image
Read IOPS Limit	The desired limit of read operations per second.	rbd_qos_read_iops_limit	Image
Write BPS Burst	The desired burst limit of write bytes.	rbd_qos_write_bps_burst	Image
Write BPS Limit	The desired limit of write bytes per second.	rbd_qos_write_bps_limit	Image

FIGURE 6.3: RBD CONFIGURATION

6.3 Creating RBDs

To add a new device, click *Create* in the top left of the table heading and do the following on the *Create RBD* screen:



The screenshot shows a 'Create RBD' form with the following fields and options:

- Name ***: A text input field containing 'example_rbd_device'.
- Pool ***: A dropdown menu showing 'rbd' with a green checkmark and a double-headed arrow icon.
- Use a dedicated data pool
- Size ***: A text input field containing '10 GiB' with a green checkmark.
- Features**: A list of checkboxes:
 - Deep flatten
 - Layering
 - Exclusive lock
 - Object map (requires exclusive-lock)
 - Journaling (requires exclusive-lock)
 - Fast diff (interlocked with object-map)

At the bottom right of the form, there is a link labeled 'Advanced...'. At the bottom center, there are two buttons: 'Create RBD' (green) and 'Cancel' (white).

FIGURE 6.4: ADDING A NEW RBD

1. Enter the name of the new device. Refer to *Book "Deployment Guide", Chapter 2 "Hardware requirements and recommendations", Section 2.11 "Name limitations"* for naming limitations.
2. Select the pool with the `rbd` application assigned from which the new RBD device will be created.

3. Specify the size of the new device.
4. Specify additional options for the device. To fine-tune the device parameters, click *Advanced* and enter values for object size, stripe unit, or stripe count. To enter Quality of Service (QoS) limits, click *Quality of Service* and enter them.
5. Confirm with *Create RBD*.

6.4 Deleting RBDs

To delete a device, select the device in the table row. Click the drop-down arrow next to the *Create* button and click *Delete*. Confirm the deletion with *Delete RBD*.



Tip: Moving RBDs to trash

Deleting an RBD is an irreversible action. If you *Move to Trash* instead, you can restore the device later on by selecting it on the *Trash* tab of the main table and clicking *Restore* in the top left of the table heading.

6.5 Creating RADOS Block Device snapshots

To create a RADOS Block Device snapshot, select the device in the table row and the detailed configuration content pane appears. Select the *Snapshots* tab and click *Create* in the top left of the table heading. Enter the snapshot's name and confirm with *Create RBD Snapshot*.

After selecting a snapshot, you can perform additional actions on the device, such as rename, protect, clone, copy, or delete. *Rollback* restores the device's state from the current snapshot.

Details Snapshots Configuration				
<div style="display: flex; justify-content: space-between; align-items: center;"> + Create ▾ 10 Q ✕ </div>				
Name	Size	Provisioned	State	Created
testing_rbd-20190215T095402Z	10 MiB	0 B	UNPROTECTED	2/15/19 10:54:08 AM
testing_rbd-20190405T074138Z	10 MiB	0 B	UNPROTECTED	4/5/19 9:41:42 AM
0 selected / 2 total				

FIGURE 6.5: RBD SNAPSHOTS

6.6 RBD mirroring

RADOS Block Device images can be asynchronously mirrored between two Ceph clusters. You can use the Ceph Dashboard to configure replication of RBD images between two or more clusters. This capability is available in two modes:

Journal-based

This mode uses the RBD journaling image feature to ensure point-in-time, crash-consistent replication between clusters.

Snapshot-based

This mode uses periodically scheduled or manually created RBD image mirror-snapshots to replicate crash-consistent RBD images between clusters.

Mirroring is configured on a per-pool basis within peer clusters and can be configured on a specific subset of images within the pool or configured to automatically mirror all images within a pool when using journal-based mirroring only.

Mirroring is configured using the `rbd` command, which is installed by default in SUSE Enterprise Storage 7. The `rbd-mirror` daemon is responsible for pulling image updates from the remote, peer cluster and applying them to the image within the local cluster. See [Section 6.6.2, “Enabling the `rbd-mirror` daemon”](#) for more information on enabling the `rbd-mirror` daemon.

Depending on the need for replication, RADOS Block Device mirroring can be configured for either one- or two-way replication:

One-way Replication

When data is only mirrored from a primary cluster to a secondary cluster, the `rbd-mirror` daemon runs only on the secondary cluster.

Two-way Replication

When data is mirrored from primary images on one cluster to non-primary images on another cluster (and vice-versa), the `rbd-mirror` daemon runs on both clusters.



Important

Each instance of the `rbd-mirror` daemon must be able to connect to both the local and remote Ceph clusters simultaneously, for example all monitor and OSD hosts. Additionally, the network must have sufficient bandwidth between the two data centers to handle mirroring workload.



Tip: General information

For general information and the command line approach to RADOS Block Device mirroring, refer to [Section 20.4, “RBD image mirrors”](#).

6.6.1 Configuring primary and secondary clusters

A *primary* cluster is where the original pool with images is created. A *secondary* cluster is where the pool or images are replicated from the *primary* cluster.



Note: Relative naming

The *primary* and *secondary* terms can be relative in the context of replication because they relate more to individual pools than to clusters. For example, in two-way replication, one pool can be mirrored from the *primary* cluster to the *secondary* one, while another pool can be mirrored from the *secondary* cluster to the *primary* one.

6.6.2 Enabling the `rbd-mirror` daemon

The following procedures demonstrate how to perform the basic administrative tasks to configure mirroring using the `rbd` command. Mirroring is configured on a per-pool basis within the Ceph clusters.

The pool configuration steps should be performed on both peer clusters. These procedures assume two clusters, named “primary” and “secondary”, are accessible from a single host for clarity.

The `rbd-mirror` daemon performs the actual cluster data replication.

1. Rename `ceph.conf` and keyring files and copy them from the primary host to the secondary host:

```
cephuser@secondary > cp /etc/ceph/ceph.conf /etc/ceph/primary.conf
cephuser@secondary > cp /etc/ceph/ceph.admin.client.keyring \
/etc/ceph/primary.client.admin.keyring
cephuser@secondary > scp PRIMARY_HOST:/etc/ceph/ceph.conf \
/etc/ceph/secondary.conf
cephuser@secondary > scp PRIMARY_HOST:/etc/ceph/ceph.client.admin.keyring \
/etc/ceph/secondary.client.admin.keyring
```

- To enable mirroring on a pool with `rbd`, specify the `mirror pool enable`, the pool name, and the mirroring mode:

```
cephuser@adm > rbd mirror pool enable POOL_NAME MODE
```



Note

The mirroring mode can either be `image` or `pool`. For example:

```
cephuser@secondary > rbd --cluster primary mirror pool enable image-pool
image
cephuser@secondary > rbd --cluster secondary mirror pool enable image-pool
image
```

- On the Ceph Dashboard, navigate to *Block > Mirroring*. The *Daemons* table to the left shows actively running `rbd-mirror` daemons and their health.

Daemons

Instance	ID	Hostname	Version	Health
292255	test	doc-ses-min4	14.2.2-354-g8878cf2360	OK

1 total

FIGURE 6.6: RUNNING `rbd-mirror` DAEMONS

6.6.3 Disabling mirroring

To disable mirroring on a pool with `rbd`, specify the `mirror pool disable` command and the pool name:

```
cephuser@adm > rbd mirror pool disable POOL_NAME
```

When mirroring is disabled on a pool in this way, mirroring will also be disabled on any images (within the pool) for which mirroring was enabled explicitly.

6.6.4 Bootstrapping peers

In order for the `rd-mirror` to discover its peer cluster, the peer needs to be registered to the pool and a user account needs to be created. This process can be automated with `rd` by using the `mirror pool peer bootstrap create` and `mirror pool peer bootstrap import` commands.

To manually create a new bootstrap token with `rd`, specify the `mirror pool peer bootstrap create` command, a pool name, along with an optional site name to describe the local cluster:

```
cephuser@adm > rbd mirror pool peer bootstrap create [--site-name local-site-name] pool-name
```

The output of `mirror pool peer bootstrap create` will be a token that should be provided to the `mirror pool peer bootstrap import` command. For example, on the primary cluster:

```
cephuser@adm > rbd --cluster primary mirror pool peer bootstrap create --site-name primary
image-pool
eyJmc2lkIjoiOWY1MjgyZGItYjg5OS00NTk2LTgwOTgtMzIwYzFmYzM5NmYzIiw1Y2xpZW50X2lkIjoicmJkLW1pcnJvcilwZWVyIiwia2V5IjoiQVFBUnczOWQwdkhvQmhBQVlMM1I4RmR5dHNJQU50bkFTZ0l0TVE9PSIsIm1vbl9ob3N0I
\
joiW3YyOjE5Mi4xNjguMS4z0jY4MjAsdjE6MTkyLjE2OC4xLjM6NjgyMV0ifQ==
```

To manually import the bootstrap token created by another cluster with the `rd` command, specify the `mirror pool peer bootstrap import` command, the pool name, a file path to the created token (or '-' to read from standard input), along with an optional site name to describe the local cluster and a mirroring direction (defaults to `rx-tx` for bidirectional mirroring, but can also be set to `rx-only` for unidirectional mirroring):

```
cephuser@adm > rbd mirror pool peer bootstrap import [--site-name local-site-name] \
[--direction rx-only or rx-tx] pool-name token-path
```

For example, on the secondary cluster:

```
cephuser@adm > cat >>EOF < token
eyJmc2lkIjoiOWY1MjgyZGItYjg5OS00NTk2LTgwOTgtMzIwYzFmYzM5NmYzIiw1Y2xpZW50X2lkIjoicmJkLW1pcnJvcilwZWVyIiwia2V5IjoiQVFBUnczOWQwdkhvQmhBQVlMM1I4RmR5dHNJQU50bkFTZ0l0TVE9PSIsIm1vbl9ob3N0I
\
joiW3YyOjE5Mi4xNjguMS4z0jY4MjAsdjE6MTkyLjE2OC4xLjM6NjgyMV0ifQ==
EOF
cephuser@adm > rbd --cluster secondary mirror pool peer bootstrap import --site-name secondary image-pool token
```

6.6.5 Removing cluster peer

To remove a mirroring peer Ceph cluster with the `rbd` command, specify the `mirror pool peer remove` command, the pool name, and the peer UUID (available from the `rbd mirror pool info` command):

```
cephuser@adm > rbd mirror pool peer remove pool-name peer-uuid
```

6.6.6 Configuring pool replication in the Ceph Dashboard

The `rbd-mirror` daemon needs to have access to the primary cluster to be able to mirror RBD images. Ensure you have followed the steps in [Section 6.6.4, "Bootstrapping peers"](#) before continuing.

1. On both the *primary* and *secondary* cluster, create pools with an identical name and assign the `rbd` application to them. Refer to [Section 5.1, "Adding a new pool"](#) for more details on creating a new pool.

Create Pool

Name *

mirrored-pool ✓

Pool type *

replicated ✓ ⇅

PG Autoscale

off ✓ ⇅

Placement groups *

4 ✓

[Calculation help](#)

Replicated size *

3

Applications

 rbd ✕

CRUSH

Crush ruleset

replicated_rule ⇅ ? + ✕

Compression

Mode

none ⇅

Quotas

Max bytes ?

e.g., 10GiB

Max objects ?

0

RBD Configuration

2. On both the *primary* and *secondary* cluster's dashboards, navigate to *Block > Mirroring*. In the *Pools* table on the right, click the name of the pool to replicate, and after clicking *Edit Mode*, select the replication mode. In this example, we will work with a *pool* replication mode, which means that all images within a given pool will be replicated. Confirm with *Update*.

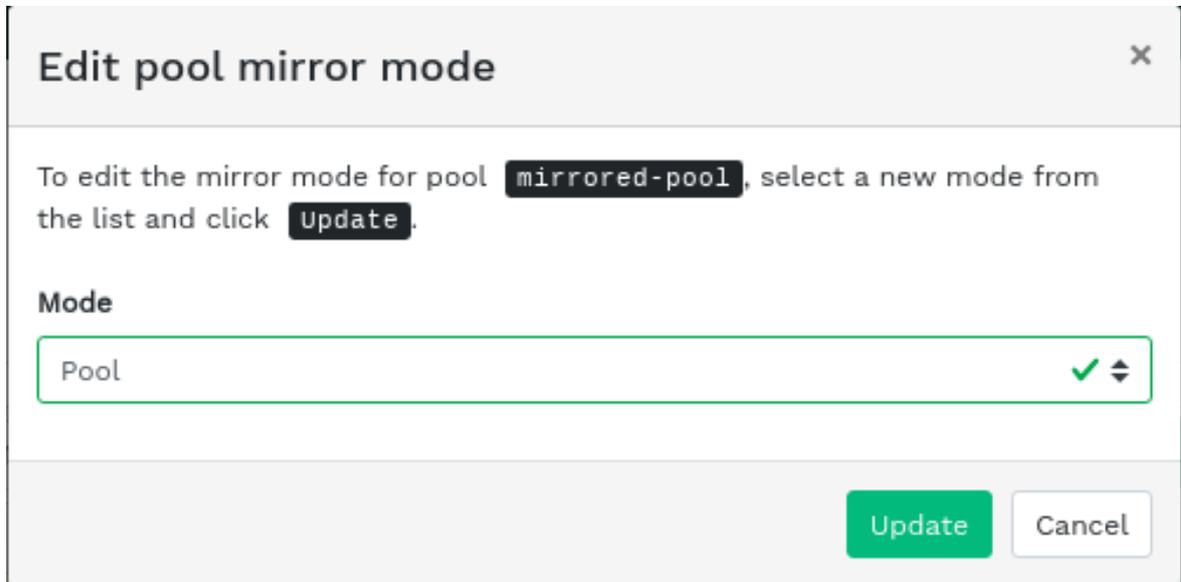


FIGURE 6.8: CONFIGURING THE REPLICATION MODE

! Important: Error or warning on the primary cluster

After updating the replication mode, an error or warning flag will appear in the corresponding right column. That is because the pool has no peer user for replication assigned yet. Ignore this flag for the *primary* cluster as we assign a peer user to the *secondary* cluster only.

3. On the *secondary* cluster's Dashboard, navigate to *Block* > *Mirroring*. Add the pool mirror peer by selecting *Add Peer*. Provide the *primary* cluster's details:

Add pool mirror peer ×

Add the pool mirror peer attributes for pool `mirrored-pool` and click `Submit`.

Cluster Name *

CephX ID *

Monitor Addresses

CephX Key

FIGURE 6.9: **ADDING PEER CREDENTIALS**

Cluster Name

An arbitrary unique string that identifies the primary cluster, such as 'primary'. The cluster name needs to be different from the real secondary cluster's name.

CephX ID

The Ceph user ID that you created as a mirroring peer. In this example it is 'rbd-mirror-peer'.

Monitor Addresses

Comma-separated list of IP addresses of the primary cluster's Ceph Monitor nodes.

CephX Key

The key related to the peer user ID. You can retrieve it by running the following example command on the primary cluster:

```
cephuser@adm > ceph auth print_key pool-mirror-peer-name
```

Confirm with *Submit*.

Pools

Name	Mode	Leader	# Local	# Remote	Health
example_rbd_pool	pool	292255	2	2	OK
mirrored-pool	pool	292255	0	0	OK
pool3	image	292255	2	2	OK
pool4	pool	292255	1	1	OK

1 selected / 4 total

FIGURE 6.10: LIST OF REPLICATED POOLS

6.6.7 Verifying that RBD image replication works

When the `rbd-mirror` daemon is running and RBD image replication is configured on the Ceph Dashboard, it is time to verify whether the replication actually works:

1. On the *primary* cluster's Ceph Dashboard, create an RBD image so that its parent pool is the pool that you already created for replication purposes. Enable the Exclusive lock and Journaling features for the image. Refer to [Section 6.3, "Creating RBDs"](#) for details on how to create RBD images.

Create RBD

Name *

Pool *

Use a dedicated data pool

Size *

Features

- Deep flatten
- Layering
- Exclusive lock
- Object map (requires exclusive-lock)
- Journaling (requires exclusive-lock)
- Fast diff (requires object-map)

[Advanced...](#)

- After you create the image that you want to replicate, open the *secondary* cluster's Ceph Dashboard and navigate to *Block > Mirroring*. The *Pools* table on the right will reflect the change in the number of *# Remote* images and synchronize the number of *# Local* images.

Pools

Name	Mode	Leader	# Local	# Remote	Health
example_rbd_pool	pool	292255	2	2	OK
mirrored-pool	pool	292255	1	1	OK
pool3	image	292255	2	2	OK
pool4	pool	292255	1	1	OK

1 selected / 4 total

FIGURE 6.12: NEW RBD IMAGE SYNCHRONIZED



Tip: Replication progress

The *Images* table at the bottom of the page shows the status of replication of RBD images. The *Issues* tab includes possible problems, the *Syncing* tab displays the progress of image replication, and the *Ready* tab lists all images with successful replication.

Images

Issues Syncing Ready

Pool	Image	Description	State
mirrored-pool	mirrored-image1	replaying, master_posillon=[object_number=3, tag_tid=1, entry_tid=3], mirror_position=[object_number=3, tag_tid=1, entry_tid=3], entries_behind_master=0	Replaying
pool3	img1	replaying, master_posillon=[object_number=1, tag_tid=2, entry_tid=6401], mirror_position=[object_number=1, tag_tid=2, entry_tid=6401], entries_behind_master=0	Replaying
pool3	new_image1	replaying, master_posillon=[object_number=1, tag_tid=3, entry_tid=641], mirror_position=[object_number=1, tag_tid=3, entry_tid=641], entries_behind_master=0	Replaying
pool4	img4	replaying, master_posillon=[object_number=3, tag_tid=1, entry_tid=3], mirror_position=[object_number=3, tag_tid=1, entry_tid=3], entries_behind_master=0	Replaying

6 total

FIGURE 6.13: RBD IMAGES' REPLICATION STATUS

- On the *primary* cluster, write data to the RBD image. On the *secondary* cluster's Ceph Dashboard, navigate to *Block > Images* and monitor whether the corresponding image's size is growing as the data on the primary cluster is written.

6.7 Managing iSCSI Gateways



Tip: More information on iSCSI Gateways

For more general information about iSCSI Gateways, refer to [Chapter 22, Ceph iSCSI gateway](#).

To list all available gateways and mapped images, click *Block* > *iSCSI* from the main menu. An *Overview* tab opens, listing currently configured iSCSI Gateways and mapped RBD images.

The *Gateways* table lists each gateway's state, number of iSCSI targets, and number of sessions. The *Images* table lists each mapped image's name, related pool name backstore type, and other statistical details.

The *Targets* tab lists currently configured iSCSI targets.

Target	Portals	Images	# Sessions
> iqn.2001-07.com.ceph:1597683071527	node1.asettle-dashboard.test:10.20.164.201	rbd/example_rbd_device_potato	0
> iqn.2001-07.com.ceph:1597683089358	node1.asettle-dashboard.test:10.20.164.201	rbd/potato-rbd	0

0 selected / 2 total

FIGURE 6.14: LIST OF ISCSI TARGETS

To view more detailed information about a target, click the drop-down arrow on the target table row. A tree-structured schema opens, listing disks, portals, initiators, and groups. Click an item to expand it and view its detailed contents, optionally with a related configuration in the table on the right.

Overview Targets

Edit Discovery authentication

Target	Portals	Images	# Sessions
iqn.2001-07.com.ceph:1597683071527	node1.asettle-dashboard.test:10.20.164.201	rbd/example_rbd_device_potato	0

iSCSI Topology

- ▼ iqn.2001-07.com.ceph:1597683071527
 - ▼ Disks
 - rbd/example_rbd_device_potato
 - ▼ Portals
 - node1.asettle-dashboard.test:10.20.164.201
 - Initiators
 - Groups

rbd/example_rbd_device_potato

Name	Current	Default
backstore	user:rbd (tcmu-runner)	rbd
hw_max_sectors	1024	1024
lun	0	
max_data_area_mb	8	8
osd_op_timeout	30	30
qfull_timeout	5	5
wwn	bf60abfd-9159-4098-bc9b-2be4daaefa5c	
7 total		

>	Target	Portals	Images	# Sessions
	iqn.2001-07.com.ceph:1597683089358	node1.asettle-dashboard.test:10.20.164.201	rbd/potato-rbd	0

1 selected / 2 total

FIGURE 6.15: iSCSI TARGET DETAILS

6.7.1 Adding iSCSI targets

To add a new iSCSI target, click *Create* in the top left of the *Targets* table and enter the required information.

Create Target

Target IQN *

Portals *

Images

Backstore: user:rbd (tcmu-runner).

ACL authentication

User

Password

Mutual User

Mutual Password

FIGURE 6.16: ADDING A NEW TARGET

1. Enter the target address of the new gateway.
2. Click *Add portal* and select one or multiple iSCSI portals from the list.
3. Click *Add image* and select one or multiple RBD images for the gateway.
4. If you need to use authentication to access the gateway, activate the *ACL Authentication* check box and enter the credentials. You can find more advanced authentication options after activating *Mutual authentication* and *Discovery authentication*.
5. Confirm with *Create Target*.

6.7.2 Editing iSCSI targets

To edit an existing iSCSI target, click its row in the *Targets* table and click *Edit* in the top left of the table.

You can then modify the iSCSI target, add or delete portals, and add or delete related RBD images. You can also adjust authentication information for the gateway.

6.7.3 Deleting iSCSI targets

To delete an iSCSI target, select the table row and click the drop-down arrow next to the *Edit* button and select *Delete*. Activate *Yes, I am sure* and confirm with *Delete iSCSI target*.

6.8 RBD Quality of Service (QoS)



Tip: For more information

For more general information and a description of RBD QoS configuration options, refer to [Section 20.6, "QoS settings"](#).

The QoS options can be configured at different levels.

- Globally
- On a per-pool basis
- On a per-image basis

The *global* configuration is at the top of the list and will be used for all newly created RBD images and for those images that do not override these values on the pool or RBD image layer. An option value specified globally can be overridden on a per-pool or per-image basis. Options specified on a pool will be applied to all RBD images of that pool unless overridden by a configuration option set on an image. Options specified on an image will override options specified on a pool and will override options specified globally.

This way it is possible to define defaults globally, adapt them for all RBD images of a specific pool, and override the pool configuration for individual RBD images.

6.8.1 Configuring options globally

To configure the RADOS Block Device options globally, select *Cluster > Configuration* from the main menu.

1. To list all available global configuration options, next to *Level*, choose *Advanced* from the drop-down menu.
2. Filter the results of the table by filtering for `rbd_qos` in the search field. This lists all available configuration options for QoS.
3. To change a value, click the row in the table, then select *Edit* at the top left of the table. The *Edit* dialog contains six different fields for specifying values. The RBD configuration option values are required in the *mgr* text box.



Note

Unlike the other dialogs, this one does not allow you to specify the value in convenient units. You need to set these values in either bytes or IOPS, depending on the option you are editing.

6.8.2 Configuring options on a new pool

To create a new pool and configure RBD configuration options on it, click *Pools > Create*. Select *replicated* as pool type. You will then need to add the `rbd` application tag to the pool to be able to configure the RBD QoS options.



Note

It is not possible to configure RBD QoS configuration options on an erasure coded pool. To configure the RBD QoS options for erasure coded pools, you need to edit the replicated metadata pool of an RBD image. The configuration will then be applied to the erasure coded data pool of that image.

6.8.3 Configuring options on an existing pool

To configure RBD QoS options on an existing pool, click *Pools*, then click the pool's table row and select *Edit* at the top left of the table.

You should see the *RBD Configuration* section in the dialog, followed by a *Quality of Service* section.



Note

If you see neither the *RBD Configuration* nor the *Quality of Service* section, you are likely either editing an *erasure coded* pool, which cannot be used to set RBD configuration options, or the pool is not configured to be used by RBD images. In the latter case, assign the *rbd* application tag to the pool and the corresponding configuration sections will show up.

6.8.4 Configuration options

Click *Quality of Service* + to expand the configuration options. A list of all available options will show up. The units of the configuration options are already shown in the text boxes. In case of any bytes per second (BPS) option, you are free to use shortcuts such as '1M' or '5G'. They will be automatically converted to '1 MB/s' and '5 GB/s' respectively.

By clicking the reset button to the right of each text box, any value set on the pool will be removed. This does not remove configuration values of options configured globally or on an RBD image.

6.8.5 Creating RBD QoS options with a new RBD image

To create an RBD image with RBD QoS options set on that image, select *Block > Images* and then click *Create*. Click *Advanced...* to expand the advanced configuration section. Click *Quality of Service* + to open all available configuration options.

6.8.6 Editing RBD QoS options on existing images

To edit RBD QoS options on an existing image, select *Block > Images*, then click the pool's table row, and lastly click *Edit*. The edit dialog will show up. Click *Advanced...* to expand the advanced configuration section. Click *Quality of Service* + to open all available configuration options.

6.8.7 Changing configuration options when copying or cloning images

If an RBD image is cloned or copied, the values set on that particular image will be copied too, by default. If you want to change them while copying or cloning, you can do so by specifying the updated configuration values in the copy/clone dialog, the same way as when creating or editing an RBD image. Doing so will only set (or reset) the values for the RBD image that is copied or cloned. This operation changes neither the source RBD image configuration, nor the global configuration.

If you choose to reset the option value on copying/cloning, no value for that option will be set on that image. This means that any value of that option specified for the parent pool will be used if the parent pool has the value configured. Otherwise, the global default will be used.

7 Manage NFS Ganesha



Important

NFS Ganesha supports NFS version 4.1 and newer. It does not support NFS version 3.



Tip: More information on NFS Ganesha

For more general information about NFS Ganesha, refer to [Chapter 25, NFS Ganesha](#).

To list all available NFS exports, click *NFS* from the main menu.

The list shows each export's directory, daemon host name, type of storage back-end, and access type.

A screenshot of a web interface showing a table of NFS exports. The table has columns for Path, Pseudo, Cluster, Daemons, Storage Backend, and Access Type. There are three rows of data. Above the table is a '+ Create' button, a grid icon, a page number '10', and a search box. Below the table, it says '0 selected / 3 total'.

Path	Pseudo	Cluster	Daemons	Storage Backend	Access Type
/potato/potato	/exportimus-maximus	ganesha-sesdev_nfs		CephFS	MDONLY_RO
/root	/exportcephfs	ganesha-sesdev_nfs		CephFS	RW
/root/potato	/exportpotato	ganesha-sesdev_nfs		CephFS	MDONLY

FIGURE 7.1: LIST OF NFS EXPORTS

To view more detailed information about an NFS export, click its table row.

A screenshot of a web interface showing the details of an NFS export. The table has two columns: the property name and its value. The properties listed are Access Type, CephFS Filesystem, CephFS User, Cluster, Daemons, NFS Protocol, Path, Pseudo, Squash, Storage Backend, and Transport.

Access Type	RW
CephFS Filesystem	sesdev_fs
CephFS User	admin
Cluster	ganesha-sesdev_nfs
Daemons	
NFS Protocol	NFSv3, NFSv4
Path	/root
Pseudo	/exportcephfs
Squash	no_root_squash
Storage Backend	CephFS
Transport	TCP, UDP

FIGURE 7.2: NFS EXPORT DETAILS

7.1 Creating NFS exports

To add a new NFS export, click *Create* in the top left of the exports table and enter the required information.

Create NFS export

Cluster *	ganesha-sesdev_nfs	⌵
Daemons	No items selected.	<input type="button" value="+ Add daemon"/>
Storage Backend *	CephFS	✓ ⌵
CephFS User ID *	admin	✓ ⌵
CephFS Name *	sesdev_fs	✓ ⌵
Security Label	<input type="checkbox"/> Enable security label	
CephFS Path *	/root	✓
	New directory will be created	
NFS Protocol *	<input checked="" type="checkbox"/> NFSv3 <input checked="" type="checkbox"/> NFSv4	
NFS Tag ?		
Pseudo * ?	/exportcephfs	✓
Access Type *	RW	✓ ⌵
	Allows all operations	
Squash *	no_root_squash	✓ ⌵
Transport Protocol *	<input checked="" type="checkbox"/> UDP <input checked="" type="checkbox"/> TCP	
Clients	Any client can access	<input type="button" value="+ Add clients"/>

FIGURE 7.3: ADDING A NEW NFS EXPORT

1. Select one or more NFS Ganesha daemons that will run the export.
2. Select a storage back-end.

Important

At this time, only NFS exports backed by CephFS are supported.

3. Select a user ID and other back-end related options.
4. Enter the directory path for the NFS export. If the directory does not exist on the server, it will be created.
5. Specify other NFS related options, such as supported NFS protocol version, pseudo, access type, squashing, or transport protocol.
6. If you need to limit access to specific clients only, click *Add clients* and add their IP addresses together with access type and squashing options.
7. Confirm with *Create NFS export*.

7.2 Deleting NFS exports

To delete an export, select and highlight the export in the table row. Click the drop-down arrow next to the *Edit* button and select *Delete*. Activate the *Yes, I am sure* check box and confirm with *Delete NFS export*.

7.3 Editing NFS exports

To edit an existing export, select and highlight the export in the table row and click *Edit* in the top left of the exports table.

You can then adjust all the details of the NFS export.

Edit NFS export

Cluster *	ganesha-sesdev_nfs
Daemons	No items selected. + Add daemon
Storage Backend *	CephFS
CephFS User ID *	admin
CephFS Name *	sesdev_fs
Security Label	<input type="checkbox"/> Enable security label
CephFS Path *	/root
NFS Protocol *	<input checked="" type="checkbox"/> NFSv3 <input checked="" type="checkbox"/> NFSv4
NFS Tag ?	
Pseudo * ?	/exportcephfs
Access Type *	RW Allows all operations
Squash *	no_root_squash
Transport Protocol *	<input checked="" type="checkbox"/> UDP <input checked="" type="checkbox"/> TCP

FIGURE 7.4: EDITING AN NFS EXPORT

8 Manage CephFS



Tip: For more information

To find detailed information about CephFS, refer to [Chapter 23, Clustered file system](#).

8.1 Viewing CephFS overview

Click *Filesystems* from the main menu to view the overview of configured file systems. The main table shows each file system's name, date of creation, and whether it is enabled or not.

By clicking a file system's table row, you reveal details about its rank and pools added to the file system.

The screenshot shows the details for the CephFS named 'sesdev_fs'. At the top, there is a table with columns: Name, Created, and Enabled. The row for 'sesdev_fs' shows it was created on 8/17/20 at 11:38:25 AM and is enabled. Below this, there are tabs for 'Details', 'Clients 2', 'Directories', and 'Performance Details'. The 'Details' tab is active. Under 'Ranks', there is a table with columns: Rank, State, Daemon, Activity, Dentries, and Inodes. One rank is shown with Rank 0, State active, Daemon sesdev_fs.node2.gzdz, Activity Reqs: 0 /s, Dentries 10, and Inodes 13. Under 'Pools', there is a table with columns: Pool, Type, and Size. Two pools are shown: 'cephfs.sesdev_fs data' and 'cephfs.sesdev_fs metadata', both with a size of 13.1 GiB. Under 'Standbys', there is a table with columns: Standby daemons and a single entry 'sesdev_fs.node3.jswzii'.

Name	Created	Enabled
sesdev_fs	8/17/20 11:38:25 AM	✓

Rank	State	Daemon	Activity	Dentries	Inodes
0	active	sesdev_fs.node2.gzdz	Reqs: 0 /s	10	13

Pool	Type	Size
cephfs.sesdev_fs data		13.1 GiB
cephfs.sesdev_fs metadata		13.1 GiB

Standby daemons
sesdev_fs.node3.jswzii

FIGURE 8.1: CEPHFS DETAILS

At the bottom of the screen, you can see statistics counting the number of related MDS inodes and client requests, collected in real time.

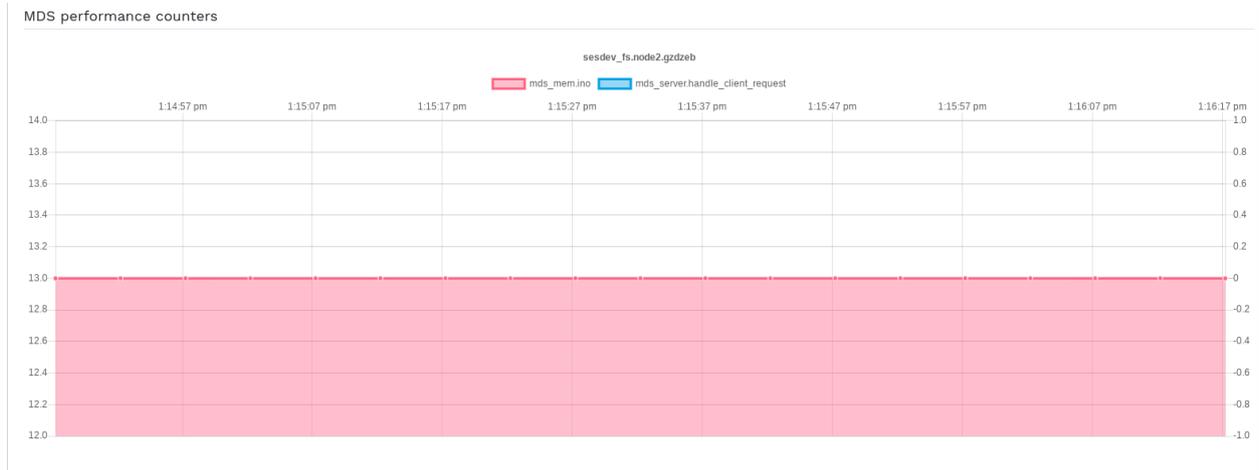


FIGURE 8.2: CEPHFS DETAILS

9 Manage the Object Gateway

Important

Before you begin, you may encounter the following notification when trying to access the Object Gateway front-end on the Ceph Dashboard:

```
Information
No RGW credentials found, please consult the documentation on how to enable RGW for
the dashboard.
Please consult the documentation on how to configure and enable the Object Gateway
management functionality.
```

This is because the Object Gateway has not been automatically configured by `cephadm` for the Ceph Dashboard. If you encounter this notification, follow the instructions at [Section 10.4, “Enabling the Object Gateway management front-end”](#) to manually enable the Object Gateway front-end for the Ceph Dashboard.

Tip: More information on Object Gateway

For more general information about Object Gateway, refer to [Chapter 21, Ceph Object Gateway](#).

9.1 Viewing Object Gateways

To view a list of configured Object Gateways, click *Object Gateway* > *Daemons*. The list includes the ID of the gateway, host name of the cluster node where the gateway daemon is running, and the gateway's version number.

Click the drop-down arrow next to the gateway's name to view detailed information about the gateway. The *Performance Counters* tab shows details about read/write operations and cache statistics.

Details Performance Counters Performance Details	
arch	x86_64
ceph_release	octopus
ceph_version	ceph version 15.2.4-557-g4ac763f0b3 (4ac763f0b3864d9168bc4a46fef26d7fa759545e) octopus (stable)
ceph_version_short	15.2.4-557-g4ac763f0b3
container_hostname	node1
container_image	registry.suse.de/devel/storage/7.0/containers/ses/7/ceph/ceph
cpu	Intel Core Processor (Haswell, no TSX)
distro	sles
distro_description	SUSE Linux Enterprise Server 15 SP2
distro_version	15.2
frontend_config#0	beast port=80
frontend_type#0	beast
hostname	node1
kernel_description	#1 SMP Wed Jul 29 18:54:11 UTC 2020 (dbe0add)
kernel_version	5.3.18-24.9-default
mem_swap_kb	0
mem_total_kb	4020668
num_handles	1
os	Linux
pid	1
zone_id	2a664005-94ad-432a-b873-d563fed68496
zone_name	default
zonegroup_id	cc4ec3c6-c611-4bfd-a155-e3e05552d5cd
zonegroup_name	default

FIGURE 9.1: GATEWAY'S DETAILS

9.2 Managing Object Gateway users

Click *Object Gateway* > *Users* to view a list of existing Object Gateway users.

Click the drop-down arrow next to the user name to view details about the user account, such as status information or the user and bucket quota details.

Details		Keys
Username	rgw-admin	
Full name	admin	
Suspended	No	
System	Yes	
Maximum buckets	1000	
User quota		
Enabled	No	
Maximum size	-	
Maximum objects	-	
Bucket quota		
Enabled	No	
Maximum size	-	
Maximum objects	-	

FIGURE 9.2: GATEWAY USERS

9.2.1 Adding a new gateway user

To add a new gateway user, click *Create* in the top left of the table heading. Fill in their credentials, details about the S3 key and user and bucket quotas, then confirm with *Create User*.

Create User

Username

*

example_rgw_user



Full name *

Example User



Email address

example@user.com



Max. buckets

Custom



1000

Suspended

S3 key

Auto-generate key

User quota

Enabled

Bucket quota

Enabled

Unlimited size

Unlimited objects

FIGURE 9.3: ADDING A NEW GATEWAY USER

Create User

Cancel

9.2.2 Deleting gateway users

To delete a gateway user, select and highlight the user. Click the drop-down button next to *Edit* and select *Delete* from the list to delete the user account. Activate the *Yes, I am sure* check box and confirm with *Delete user*.

9.2.3 Editing gateway user details

To change gateway user details, select and highlight the user. Click *Edit* in the top left of the table heading.

Modify basic or additional user information, such as their capabilities, keys, sub-users, and quota information. Confirm with *Edit User*.

The *Keys* tab includes a read-only list of the gateway's users and their access and secret keys. To view the keys, click a user name in the list and then select *Show* in the top left of the table heading. In the *S3 Key* dialog, click the 'eye' icon to unveil the keys, or click the clipboard icon to copy the related key to the clipboard.

9.3 Managing the Object Gateway buckets

Object Gateway (OGW) buckets implement the functionality of OpenStack Swift containers. Object Gateway buckets serve as containers for storing data objects.

Click *Object Gateway* > *Buckets* to view a list of Object Gateway buckets.

9.3.1 Adding a new bucket

To add a new Object Gateway bucket, click *Create* in the top left of the table heading. Enter the bucket's name, select the owner, and set the placement target. Confirm with *Create Bucket*.



Note

At this stage you can also enable locking by selecting *Enabled*; however, this is configurable after creation. See [Section 9.3.3, "Editing the bucket"](#) for more information.

9.3.2 Viewing bucket details

To view detailed information about an Object Gateway bucket, click the drop-down arrow next to the bucket name.

Details	
Name	export
ID	2a664005-94ad-432a-b873-d563fed68496.14523.1
Owner	rgw-admin
Index type	Normal
Placement rule	default-placement
Marker	2a664005-94ad-432a-b873-d563fed68496.14523.1
Maximum marker	0#1#2#3#4#5#6#7#8#9#10#
Version	0#1#1#2#1,3#1,4#1,5#1,6#1,7#1,8#1,9#1,10#1
Master version	0#0,1#0,2#0,3#0,4#0,5#0,6#0,7#0,8#0,9#0,10#0
Modification time	8/24/20 1:24:34 PM
Zonegroup	cc4ec3c6-c611-4bfd-a155-e3e05552d5cd
Versioning	Suspended
MFA Delete	Disabled

Bucket quota

Enabled	No
Maximum size	Unlimited
Maximum objects	Unlimited

Locking

Enabled	No
---------	----

FIGURE 9.4: GATEWAY BUCKET DETAILS



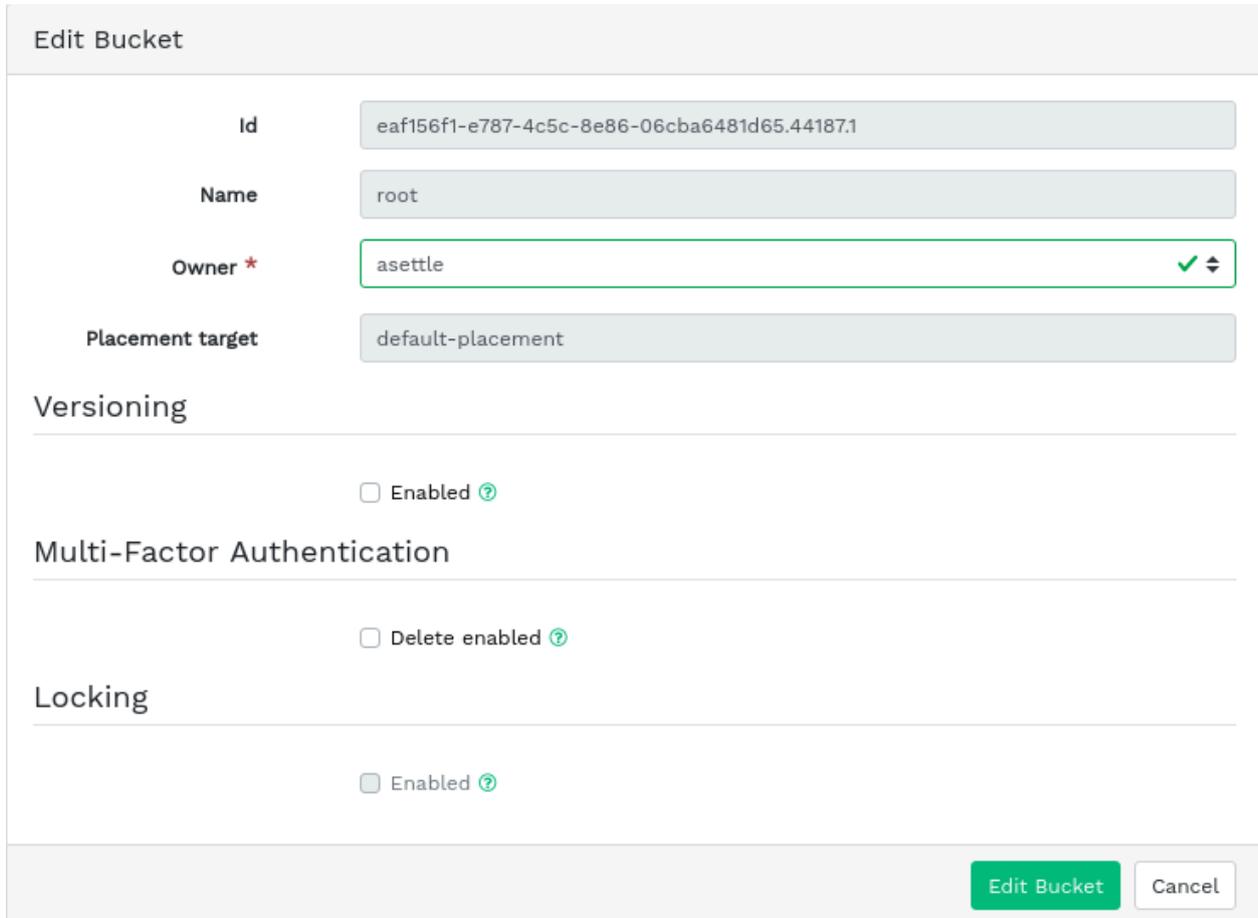
Tip: Bucket quota

Below the *Details* table, you can find details about the bucket quota and locking settings.

9.3.3 Editing the bucket

Select and highlight a bucket, then click *Edit* in the top left of the table heading.

You can update the owner of the bucket or enable versioning, multi-factor authentication or locking. Confirm any changes with *Edit Bucket*.



Edit Bucket

Id eaf156f1-e787-4c5c-8e86-06cba6481d65.44187.1

Name root

Owner * asettle ✓ ▾

Placement target default-placement

Versioning

Enabled ⓘ

Multi-Factor Authentication

Delete enabled ⓘ

Locking

Enabled ⓘ

Edit Bucket Cancel

FIGURE 9.5: EDITING THE BUCKET DETAILS

9.3.4 Deleting a bucket

To delete an Object Gateway bucket, select and highlight the bucket. Click the drop-down button next to *Edit* and select *Delete* from the list to delete the bucket. Activate the *Yes, I am sure* check box and confirm with *Delete bucket*.

10 Manual configuration

This section introduces advanced information for users that prefer configuring dashboard settings manually on the command line.

10.1 Configuring TLS/SSL support

All HTTP connections to the dashboard are secured with TLS/SSL by default. A secure connection requires an SSL certificate. You can either use a self-signed certificate, or generate a certificate and have a well known certificate authority (CA) sign it.



Tip: Disabling SSL

You may want to disable the SSL support for a specific reason. For example, if the dashboard is running behind a proxy that does not support SSL.

Use caution when disabling SSL as **user names and passwords** will be sent to the dashboard **unencrypted**.

To disable SSL, run:

```
cephuser@adm > ceph config set mgr mgr/dashboard/ssl false
```



Tip: Restarting the Ceph Manager processes

You need to restart the Ceph Manager processes manually after changing the SSL certificate and key. You can do so by either running

```
cephuser@adm > ceph mgr fail ACTIVE-MANAGER-NAME
```

or by disabling and re-enabling the dashboard module, which also triggers the manager to respawn itself:

```
cephuser@adm > ceph mgr module disable dashboard  
cephuser@adm > ceph mgr module enable dashboard
```

10.1.1 Creating self-signed certificates

Creating a self-signed certificate for secure communication is simple. This way you can get the dashboard running quickly.



Note: Web browsers complaint

Most Web browsers will complain about a self-signed certificate and require explicit confirmation before establishing a secure connection to the dashboard.

To generate and install a self-signed certificate, use the following built-in command:

```
cephuser@adm > ceph dashboard create-self-signed-cert
```

10.1.2 Using certificates signed by CA

To properly secure the connection to the dashboard and to eliminate Web browser complaints about a self-signed certificate, we recommend using a certificate that is signed by a CA.

You can generate a certificate key pair with a command similar to the following:

```
# openssl req -new -nodes -x509 \  
-subj "/O=IT/CN=ceph-mgr-dashboard" -days 3650 \  
-keyout dashboard.key -out dashboard.crt -extensions v3_ca
```

The above command outputs `dashboard.key` and `dashboard.crt` files. After you get the `dashboard.crt` file signed by a CA, enable it for all Ceph Manager instances by running the following commands:

```
cephuser@adm > ceph dashboard set-ssl-certificate -i dashboard.crt  
cephuser@adm > ceph dashboard set-ssl-certificate-key -i dashboard.key
```



Tip: Different certificates for each manager instance

If you require different certificates for each Ceph Manager instance, modify the commands and include the name of the instance as follows. Replace `NAME` with the name of the Ceph Manager instance (usually the related host name):

```
cephuser@adm > ceph dashboard set-ssl-certificate NAME -i dashboard.crt
```

```
cephuser@adm > ceph dashboard set-ssl-certificate-key NAME -i dashboard.key
```

10.2 Changing host name and port number

The Ceph Dashboard binds to a specific TCP/IP address and TCP port. By default, the currently active Ceph Manager that hosts the dashboard binds to TCP port 8443 (or 8080 when SSL is disabled).



Note

If a firewall is enabled on the hosts running Ceph Manager (and thus the Ceph Dashboard), you may need to change the configuration to enable access to these ports. For more information on firewall settings for Ceph, see *Book "Troubleshooting Guide", Chapter 13 "Hints and tips", Section 13.7 "Firewall settings for Ceph"*.

The Ceph Dashboard binds to "::*" by default, which corresponds to all available IPv4 and IPv6 addresses. You can change the IP address and port number of the Web application so that they apply to all Ceph Manager instances by using the following commands:

```
cephuser@adm > ceph config set mgr mgr/dashboard/server_addr IP_ADDRESS  
cephuser@adm > ceph config set mgr mgr/dashboard/server_port PORT_NUMBER
```



Tip: Configuring Ceph Manager instances separately

Since each `ceph-mgr` daemon hosts its own instance of the dashboard, you may need to configure them separately. Change the IP address and port number for a specific manager instance by using the following commands (replace `NAME` with the ID of the `ceph-mgr` instance):

```
cephuser@adm > ceph config set mgr mgr/dashboard/NAME/server_addr IP_ADDRESS  
cephuser@adm > ceph config set mgr mgr/dashboard/NAME/server_port PORT_NUMBER
```



Tip: Listing configured endpoints

The `ceph mgr services` command displays all endpoints that are currently configured. Look for the `dashboard` key to obtain the URL for accessing the dashboard.

10.3 Adjusting user names and passwords

If you do not want to use the default administrator account, create a different user account and associate it with at least one role. We provide a set of predefined system roles that you can use. For more details refer to [Chapter 11, Manage users and roles on the command line](#).

To create a user with administrator privileges, use the following command:

```
cephuser@adm > ceph dashboard ac-user-create USER_NAME PASSWORD administrator
```

10.4 Enabling the Object Gateway management front-end

To use the Object Gateway management functionality of the dashboard, you need to provide the login credentials of a user with the `system` flag enabled:

1. If you do not have a user with the `system` flag, create one:

```
cephuser@adm > radosgw-admin user create --uid=USER_ID --display-name=DISPLAY_NAME --system
```

Take note of the `access_key` and `secret_key` keys in the output of the command.

2. You can also obtain the credentials of an existing user by using the `radosgw-admin` command:

```
cephuser@adm > radosgw-admin user info --uid=USER_ID
```

3. Provide the received credentials to the dashboard in separate files:

```
cephuser@adm > ceph dashboard set-rgw-api-access-key ACCESS_KEY_FILE  
cephuser@adm > ceph dashboard set-rgw-api-secret-key SECRET_KEY_FILE
```



Note

By default the firewall is enabled in SUSE Linux Enterprise Server 15 SP2. For more information on firewall configuration, see *Book "Troubleshooting Guide", Chapter 13 "Hints and tips", Section 13.7 "Firewall settings for Ceph"*.

There are several points to consider:

- The host name and port number of the Object Gateway are determined automatically.
- If multiple zones are used, it will automatically determine the host within the master zonegroup and master zone. This is sufficient for most setups, but in some circumstances you may want to set the host name and port manually:

```
cephuser@adm > ceph dashboard set-rgw-api-host HOST
cephuser@adm > ceph dashboard set-rgw-api-port PORT
```

- These are additional settings that you may need:

```
cephuser@adm > ceph dashboard set-rgw-api-scheme SCHEME # http or https
cephuser@adm > ceph dashboard set-rgw-api-admin-resource ADMIN_RESOURCE
cephuser@adm > ceph dashboard set-rgw-api-user-id USER_ID
```

- If you are using a self-signed certificate ([Section 10.1, “Configuring TLS/SSL support”](#)) in your Object Gateway setup, disable certificate verification in the dashboard to avoid refused connections caused by certificates signed by an unknown CA or not matching the host name:

```
cephuser@adm > ceph dashboard set-rgw-api-ssl-verify False
```

- If the Object Gateway takes too long to process requests and the dashboard runs into timeouts, the timeout value can be adjusted (default is 45 seconds):

```
cephuser@adm > ceph dashboard set-rest-requests-timeout SECONDS
```

10.5 Enabling iSCSI management

The Ceph Dashboard manages iSCSI targets using the REST API provided by the `rbd-target-api` service of the Ceph iSCSI gateway. Ensure it is installed and enabled on iSCSI gateways.



Note

The iSCSI management functionality of the Ceph Dashboard depends on the latest version 3 of the `ceph-iscsi` project. Ensure that your operating system provides the correct version, otherwise the Ceph Dashboard will not enable the management features.

If the `ceph-iscsi` REST API is configured in HTTPS mode and it is using a self-signed certificate, configure the dashboard to avoid SSL certificate verification when accessing `ceph-iscsi` API.

Disable API SSL verification:

```
cephuser@adm > ceph dashboard set-iscsi-api-ssl-verification false
```

Define the available iSCSI gateways:

```
cephuser@adm > ceph dashboard iscsi-gateway-list
cephuser@adm > ceph dashboard iscsi-gateway-add scheme://username:password@host[:port]
cephuser@adm > ceph dashboard iscsi-gateway-rm gateway_name
```

10.6 Enabling Single Sign-On

Single Sign-On (SSO) is an access control method that enables users to log in with a single ID and password to multiple applications simultaneously.

The Ceph Dashboard supports external authentication of users via the SAML 2.0 protocol. Because *authorization* is still performed by the dashboard, you first need to create user accounts and associate them with the desired roles. However, the *authentication* process can be performed by an existing *Identity Provider* (IdP).

To configure Single Sign-On, use the following command:

```
cephuser@adm > ceph dashboard sso setup saml2 CEPH_DASHBOARD_BASE_URL \
  IDP_METADATA IDP_USERNAME_ATTRIBUTE \
  IDP_ENTITY_ID SP_X_509_CERT \
  SP_PRIVATE_KEY
```

Parameters:

CEPH_DASHBOARD_BASE_URL

Base URL where Ceph Dashboard is accessible (for example, 'https://cephdashboard.local').

IDP_METADATA

URL, file path, or content of the IdP metadata XML (for example, 'https://myidp/metadata').

IDP_USERNAME_ATTRIBUTE

Optional. Attribute that will be used to get the user name from the authentication response. Defaults to 'uid'.

IDP_ENTITY_ID

Optional. Use when more than one entity ID exists on the IdP metadata.

SP_X_509_CERT / SP_PRIVATE_KEY

Optional. File path or content of the certificate that will be used by Ceph Dashboard (Service Provider) for signing and encryption. These file paths need to be accessible from the active Ceph Manager instance.



Note: SAML requests

The issuer value of SAML requests will follow this pattern:

```
CEPH_DASHBOARD_BASE_URL/auth/saml2/metadata
```

To display the current SAML 2.0 configuration, run:

```
cephuser@adm > ceph dashboard sso show saml2
```

To disable Single Sign-On, run:

```
cephuser@adm > ceph dashboard sso disable
```

To check if SSO is enabled, run:

```
cephuser@adm > ceph dashboard sso status
```

To enable SSO, run:

```
cephuser@adm > ceph dashboard sso enable saml2
```

11 Manage users and roles on the command line

This section describes how to manage user accounts used by the Ceph Dashboard. It helps you create or modify user accounts, as well as set proper user roles and permissions.

11.1 Managing the password policy

By default the password policy feature is enabled including the following checks:

- Is the password longer than N characters?
- Are the old and new password the same?

The password policy feature can be switched on or off completely:

```
cephuser@adm > ceph dashboard set-pwd-policy-enabled true|false
```

The following individual checks can be switched on or off:

```
cephuser@adm > ceph dashboard set-pwd-policy-check-length-enabled true|false
cephuser@adm > ceph dashboard set-pwd-policy-check-oldpwd-enabled true|false
cephuser@adm > ceph dashboard set-pwd-policy-check-username-enabled true|false
cephuser@adm > ceph dashboard set-pwd-policy-check-exclusion-list-enabled true|false
cephuser@adm > ceph dashboard set-pwd-policy-check-complexity-enabled true|false
cephuser@adm > ceph dashboard set-pwd-policy-check-sequential-chars-enabled true|false
cephuser@adm > ceph dashboard set-pwd-policy-check-repetitive-chars-enabled true|false
```

In addition, the following options are available to configure the password policy behaviour.

- The minimum password length (defaults to 8):

```
cephuser@adm > ceph dashboard set-pwd-policy-min-length  $N$ 
```

- The minimum password complexity (defaults to 10):

```
cephuser@adm > ceph dashboard set-pwd-policy-min-complexity  $N$ 
```

The password complexity is calculated by classifying each character in the password.

- A list of comma-separated words that are not allowed to be used in a password:

```
cephuser@adm > ceph dashboard set-pwd-policy-exclusion-list word[,...]
```

11.2 Managing user accounts

The Ceph Dashboard supports managing multiple user accounts. Each user account consists of a user name, a password (stored in encrypted form using `bcrypt`), an optional name, and an optional e-mail address.

User accounts are stored in Ceph Monitor's configuration database and are shared globally across all Ceph Manager instances.

Use the following commands to manage user accounts:

Show existing users:

```
cephuser@adm > ceph dashboard ac-user-show [USERNAME]
```

Create a new user:

```
cephuser@adm > ceph dashboard ac-user-create USERNAME -i [PASSWORD_FILE] [ROLENAME]
[NAME] [EMAIL]
```

Delete a user:

```
cephuser@adm > ceph dashboard ac-user-delete USERNAME
```

Change a user's password:

```
cephuser@adm > ceph dashboard ac-user-set-password USERNAME -i PASSWORD_FILE
```

Modify a user's name and email:

```
cephuser@adm > ceph dashboard ac-user-set-info USERNAME NAME EMAIL
```

Disable user

```
cephuser@adm > ceph dashboard ac-user-disable USERNAME
```

Enable User

```
cephuser@adm > ceph dashboard ac-user-enable USERNAME
```

11.3 User roles and permissions

This section describes what security scopes you can assign to a user role, how to manage user roles and assign them to user accounts.

11.3.1 Defining security scopes

User accounts are associated with a set of roles that define which parts of the dashboard can be accessed by the user. The dashboard parts are grouped within a *security* scope. Security scopes are predefined and static. The following security scopes are currently available:

hosts

Includes all features related to the *Hosts* menu entry.

config-opt

Includes all features related to the management of Ceph configuration options.

pool

Includes all features related to pool management.

osd

Includes all features related to the Ceph OSD management.

monitor

Includes all features related to the Ceph Monitor management.

rbd-image

Includes all features related to the RADOS Block Device image management.

rbd-mirroring

Includes all features related to the RADOS Block Device mirroring management.

iscsi

Includes all features related to iSCSI management.

rgw

Includes all features related to the Object Gateway management.

cephfs

Includes all features related to CephFS management.

manager

Includes all features related to the Ceph Manager management.

log

Includes all features related to Ceph logs management.

grafana

Includes all features related to the Grafana proxy.

prometheus

Include all features related to Prometheus alert management.

dashboard-settings

Allows changing dashboard settings.

11.3.2 Specifying user roles

A *role* specifies a set of mappings between a *security scope* and a set of *permissions*. There are four types of permissions: 'read', 'create', 'update', and 'delete'.

The following example specifies a role where a user has 'read' and 'create' permissions for features related to pool management, and has full permissions for features related to RBD image management:

```
{
  'role': 'my_new_role',
  'description': 'My new role',
  'scopes_permissions': {
    'pool': ['read', 'create'],
    'rbd-image': ['read', 'create', 'update', 'delete']
  }
}
```

The dashboard already provides a set of predefined roles that we call *system roles*. You can instantly use them after a fresh Ceph Dashboard installation:

administrator

Provides full permissions for all security scopes.

read-only

Provides read permission for all security scopes except the dashboard settings.

block-manager

Provides full permissions for 'rbd-image', 'rbd-mirroring', and 'iscsi' scopes.

rgw-manager

Provides full permissions for the 'rgw' scope.

cluster-manager

Provides full permissions for the 'hosts', 'osd', 'monitor', 'manager', and 'config-opt' scopes.

pool-manager

Provides full permissions for the 'pool' scope.

cephfs-manager

Provides full permissions for the 'cephfs' scope.

11.3.2.1 Managing custom roles

You can create new user roles by using the following commands:

Create a new role:

```
cephuser@adm > ceph dashboard ac-role-create ROLENAME [DESCRIPTION]
```

Delete a role:

```
cephuser@adm > ceph dashboard ac-role-delete ROLENAME
```

Add scope permissions to a role:

```
cephuser@adm > ceph dashboard ac-role-add-scope-perms ROLENAME SCOPENAME PERMISSION  
[PERMISSION...]
```

Delete scope permissions from a role:

```
cephuser@adm > ceph dashboard ac-role-del-perms ROLENAME SCOPENAME
```

11.3.2.2 Assigning roles to user accounts

Use the following commands to assign roles to users:

Set user roles:

```
cephuser@adm > ceph dashboard ac-user-set-roles USERNAME ROLENAME [ROLENAME ...]
```

Add additional roles to a user:

```
cephuser@adm > ceph dashboard ac-user-add-roles USERNAME ROLENAME [ROLENAME ...]
```

Delete roles from a user:

```
cephuser@adm > ceph dashboard ac-user-del-roles USERNAME ROLENAME [ROLENAME ...]
```



Tip: Purging custom roles

If you create custom user roles and intend to remove the Ceph cluster with the **`ceph.purge`** runner later on, you need to purge the custom roles first. Find more details in [Section 13.9, “Removing an entire Ceph cluster”](#).

11.3.2.3 Example: Creating a user and a custom role

This section illustrates a procedure for creating a user account capable of managing RBD images, viewing and creating Ceph pools, and having read-only access to any other scopes.

1. Create a new user named `tux`:

```
cephuser@adm > ceph dashboard ac-user-create tux PASSWORD
```

2. Create a role and specify scope permissions:

```
cephuser@adm > ceph dashboard ac-role-create rbd/pool-manager
cephuser@adm > ceph dashboard ac-role-add-scope-perms rbd/pool-manager \
  rbd-image read create update delete
cephuser@adm > ceph dashboard ac-role-add-scope-perms rbd/pool-manager pool read
  create
```

3. Associate the roles with the `tux` user:

```
cephuser@adm > ceph dashboard ac-user-set-roles tux rbd/pool-manager read-only
```

11.4 Proxy configuration

If you want to establish a fixed URL to reach the Ceph Dashboard or if you do not want to allow direct connections to the manager nodes, you can set up a proxy that automatically forwards incoming requests to the currently active `ceph-mgr` instance.

11.4.1 Accessing the dashboard with reverse proxies

If you are accessing the dashboard via a reverse proxy configuration, you may need to service it under a URL prefix. To get the dashboard to use hyperlinks that include your prefix, you can set the `url_prefix` setting:

```
cephuser@adm > ceph config set mgr mgr/dashboard/url_prefix URL_PREFIX
```

Then you can access the dashboard at `http://HOST_NAME:PORT_NUMBER/URL_PREFIX/`.

11.4.2 Disabling re-directions

If the Ceph Dashboard is behind a load-balancing proxy such as HAProxy, disable the redirection behaviour to prevent situations where the internal (unresolvable) URLs are published to the front-end client. Use the following command to get the dashboard to respond with an HTTP error (500 by default) instead of redirecting to the active dashboard:

```
cephuser@adm > ceph config set mgr mgr/dashboard/standby_behaviour "error"
```

To reset the setting to the default redirection behaviour, use the following command:

```
cephuser@adm > ceph config set mgr mgr/dashboard/standby_behaviour "redirect"
```

11.4.3 Configuring error status codes

If the redirection behaviour is disabled, then you should customize the HTTP status code of standby dashboards. To do so, run the following command:

```
cephuser@adm > ceph config set mgr mgr/dashboard/standby_error_status_code 503
```

11.4.4 HAProxy example configuration

The following example configuration is for TLS/SSL pass through using HAProxy.



Note

The configuration works under the following conditions: If the dashboard fails over, the front-end client might receive an HTTP redirect (303) response and will be redirected to an unresolvable host.

This happens when the failover occurs during two HAProxy health checks. In this situation the previously active dashboard node will now respond with a 303 which points to the new active node. To prevent that situation you should consider disabling the redirection behaviour on standby nodes.

```
defaults
  log global
  option log-health-checks
  timeout connect 5s
  timeout client 50s
  timeout server 450s

frontend dashboard_front
  mode http
  bind *:80
  option httplog
  redirect scheme https code 301 if !{ ssl_fc }

frontend dashboard_front_ssl
  mode tcp
  bind *:443
  option tcplog
  default_backend dashboard_back_ssl

backend dashboard_back_ssl
  mode tcp
  option httpchk GET /
  http-check expect status 200
  server x HOST:PORT ssl check verify none
  server y HOST:PORT ssl check verify none
  server z HOST:PORT ssl check verify none
```

11.5 Auditing API requests

The Ceph Dashboard's REST API can log PUT, POST, and DELETE requests to the Ceph audit log. Logging is disabled by default, but you can enable it with the following command:

```
cephuser@adm > ceph dashboard set-audit-api-enabled true
```

If enabled, the following parameters are logged per each request:

from

The origin of the request, for example 'https://[::1]:44410'.

path

The REST API path, for example /api/auth.

method

'PUT', 'POST', or 'DELETE'.

user

The name of the user (or 'None').

An example log entry looks like this:

```
2019-02-06 10:33:01.302514 mgr.x [INF] [DASHBOARD] \  
from='https://[::ffff:127.0.0.1]:37022' path='/api/rgw/user/exu' method='PUT' \  
user='admin' params='{ "max_buckets": "1000", "display_name": "Example User", "uid":  
"exu", "suspended": "0", "email": "user@example.com"}'
```



Tip: Disable logging of request payload

The logging of the request payload (the list of arguments and their values) is enabled by default. You can disable it as follows:

```
cephuser@adm > ceph dashboard set-audit-api-log-payload false
```

11.6 Configuring NFS Ganesha in the Ceph Dashboard

Ceph Dashboard can manage NFS Ganesha exports that use CephFS or Object Gateway as their backstore. The dashboard manages NFS Ganesha configuration files stored in RADOS objects on the CephFS cluster. NFS Ganesha must store part of their configuration in the Ceph cluster.

Run the following command to configure the NFS Ganesha configuration object's location:

```
cephuser@adm > ceph dashboard set-ganesha-clusters-rados-pool-  
namespace pool_name[/namespace]
```

You can now manage NFS Ganesha exports using the Ceph Dashboard.

11.6.1 Configuring multiple NFS Ganesha clusters

The Ceph Dashboard supports the management of NFS Ganesha exports belonging to different NFS Ganesha clusters. We recommend each NFS Ganesha cluster store its configuration objects in a different RADOS pool/namespace to isolate the configurations from each other.

Use the following command to specify the locations of the configuration of each NFS Ganesha cluster:

```
cephuser@adm > ceph dashboard set-ganesha-clusters-rados-pool-namespace cluster_id:pool_name[/namespace](,cluster_id:pool_name[/namespace])*
```

The `cluster_id` is an arbitrary string that uniquely identifies the NFS Ganesha cluster.

When configuring the Ceph Dashboard with multiple NFS Ganesha clusters, the Web UI automatically allows you to choose to which cluster an export belongs.

11.7 Debugging plugins

Ceph Dashboard plugins extend the functionality of the dashboard. The debug plugin allows the customization of the behaviour of the dashboard according to the debug mode. It can be enabled, disabled, or checked with the following command:

```
cephuser@adm > ceph dashboard debug status
Debug: 'disabled'
cephuser@adm > ceph dashboard debug enable
Debug: 'enabled'
cephuser@adm > dashboard debug disable
Debug: 'disabled'
```

By default, this is disabled. This is the recommended setting for production deployments. If required, debug mode can be enabled without need of restarting.

II Cluster Operation

- 12 Determine the cluster state **84**
- 13 Operational tasks **112**
- 14 Operation of Ceph services **135**
- 15 Backup and restore **140**
- 16 Monitoring and alerting **143**

12 Determine the cluster state

When you have a running cluster, you may use the **ceph** tool to monitor it. Determining the cluster state typically involves checking the status of Ceph OSDs, Ceph Monitors, placement groups, and Metadata Servers.



Tip: Interactive mode

To run the **ceph** tool in an interactive mode, type **ceph** at the command line with no arguments. The interactive mode is more convenient if you are going to enter more **ceph** commands in a row. For example:

```
cephuser@adm > ceph
ceph> health
ceph> status
ceph> quorum_status
ceph> mon stat
```

12.1 Checking a cluster's status

You can find the immediate state of the cluster using **ceph status** or **ceph -s**:

```
cephuser@adm > ceph -s
cluster:
  id:      b4b30c6e-9681-11ea-ac39-525400d7702d
  health: HEALTH_OK

services:
  mon: 5 daemons, quorum ses-min1,ses-master,ses-min2,ses-min4,ses-min3 (age 2m)
  mgr: ses-min1.gpijpm(active, since 3d), standbys: ses-min2.oopvyh
  mds: my_cephfs:1 {0=my_cephfs.ses-min1.oterul=up:active}
  osd: 3 osds: 3 up (since 3d), 3 in (since 11d)
  rgw: 2 daemons active (myrealm.myzone.ses-min1.kwwazo, myrealm.myzone.ses-
min2.jngabw)

task status:
  scrub status:
    mds.my_cephfs.ses-min1.oterul: idle

data:
  pools: 7 pools, 169 pgs
```

```
objects: 250 objects, 10 KiB
usage:   3.1 GiB used, 27 GiB / 30 GiB avail
pgs:    169 active+clean
```

The output provides the following information:

- Cluster ID
- Cluster health status
- The monitor map epoch and the status of the monitor quorum
- The OSD map epoch and the status of OSDs
- The status of Ceph Managers
- The status of Object Gateways
- The placement group map version
- The number of placement groups and pools
- The *notional* amount of data stored and the number of objects stored
- The total amount of data stored.



Tip: How Ceph calculates data usage

The used value reflects the actual amount of raw storage used. The xxx GB / xxx GB value means the amount available (the lesser number) of the overall storage capacity of the cluster. The notional number reflects the size of the stored data before it is replicated, cloned or snapshot. Therefore, the amount of data actually stored typically exceeds the notional amount stored, because Ceph creates replicas of the data and may also use storage capacity for cloning and snapshotting.

Other commands that display immediate status information are:

- `ceph pg stat`
- `ceph osd pool stats`
- `ceph df`
- `ceph df detail`

To get the information updated in real time, put any of these commands (including `ceph -s`) as an argument of the `watch` command:

```
# watch -n 10 'ceph -s'
```

Press `Ctrl - C` when you are tired of watching.

12.2 Checking cluster health

After you start your cluster and before you start reading and/or writing data, check your cluster's health:

```
cephuser@adm > ceph health
HEALTH_WARN 10 pgs degraded; 100 pgs stuck unclean; 1 mons down, quorum 0,2 \
node-1,node-2,node-3
```



Tip

If you specified non-default locations for your configuration or keyring, you may specify their locations:

```
cephuser@adm > ceph -c /path/to/conf -k /path/to/keyring health
```

The Ceph cluster returns one of the following health codes:

OSD_DOWN

One or more OSDs are marked down. The OSD daemon may have been stopped, or peer OSDs may be unable to reach the OSD over the network. Common causes include a stopped or crashed daemon, a down host, or a network outage.

Verify the host is healthy, the daemon is started, and network is functioning. If the daemon has crashed, the daemon log file (`/var/log/ceph/ceph-osd.*`) may contain debugging information.

OSD_crush_type_DOWN, for example OSD_HOST_DOWN

All the OSDs within a particular CRUSH subtree are marked down, for example all OSDs on a host.

OSD_ORPHAN

An OSD is referenced in the CRUSH map hierarchy but does not exist. The OSD can be removed from the CRUSH hierarchy with:

```
cephuser@adm > ceph osd crush rm osd.ID
```

OSD_OUT_OF_ORDER_FULL

The usage thresholds for *backfillfull* (defaults to 0.90), *nearfull* (defaults to 0.85), *full* (defaults to 0.95), and/or *failsafe_full* are not ascending. In particular, we expect *backfillfull* < *nearfull*, *nearfull* < *full*, and *full* < *failsafe_full*.

To read the current values, run:

```
cephuser@adm > ceph health detail
HEALTH_ERR 1 full osd(s); 1 backfillfull osd(s); 1 nearfull osd(s)
osd.3 is full at 97%
osd.4 is backfill full at 91%
osd.2 is near full at 87%
```

The thresholds can be adjusted with the following commands:

```
cephuser@adm > ceph osd set-backfillfull-ratio ratio
cephuser@adm > ceph osd set-nearfull-ratio ratio
cephuser@adm > ceph osd set-full-ratio ratio
```

OSD_FULL

One or more OSDs has exceeded the *full* threshold and is preventing the cluster from servicing writes. Usage by pool can be checked with:

```
cephuser@adm > ceph df
```

The currently defined *full* ratio can be seen with:

```
cephuser@adm > ceph osd dump | grep full_ratio
```

A short-term workaround to restore write availability is to raise the full threshold by a small amount:

```
cephuser@adm > ceph osd set-full-ratio ratio
```

Add new storage to the cluster by deploying more OSDs, or delete existing data in order to free up space.

OSD_BACKFILLFULL

One or more OSDs has exceeded the *backfillfull* threshold, which prevents data from being allowed to rebalance to this device. This is an early warning that rebalancing may not be able to complete and that the cluster is approaching full. Usage by pool can be checked with:

```
cephuser@adm > ceph df
```

OSD_NEARFULL

One or more OSDs has exceeded the *nearfull* threshold. This is an early warning that the cluster is approaching full. Usage by pool can be checked with:

```
cephuser@adm > ceph df
```

OSDMAP_FLAGS

One or more cluster flags of interest has been set. With the exception of *full*, these flags can be set or cleared with:

```
cephuser@adm > ceph osd set flag  
cephuser@adm > ceph osd unset flag
```

These flags include:

full

The cluster is flagged as full and cannot service writes.

pauserd, pausewr

Paused reads or writes.

noup

OSDs are not allowed to start.

nodown

OSD failure reports are being ignored, such that the monitors will not mark OSDs *down*.

noin

OSDs that were previously marked *out* will not be marked back *in* when they start.

noout

Down OSDs will not automatically be marked *out* after the configured interval.

nobackfill, norecover, norebalance

Recovery or data rebalancing is suspended.

noscrub, nodeep_scrub

Scrubbing (see [Section 17.6, “Scrubbing placement groups”](#)) is disabled.

notieragent

Cache tiering activity is suspended.

OSD_FLAGS

One or more OSDs has a per-OSD flag of interest set. These flags include:

noup

OSD is not allowed to start.

nodown

Failure reports for this OSD will be ignored.

noin

If this OSD was previously marked *out* automatically after a failure, it will not be marked *in* when it starts.

noout

If this OSD is down, it will not be automatically marked *out* after the configured interval.

Per-OSD flags can be set and cleared with:

```
cephuser@adm > ceph osd add-flag osd-ID
cephuser@adm > ceph osd rm-flag osd-ID
```

OLD_CRUSH_TUNABLES

The CRUSH Map is using very old settings and should be updated. The oldest tunables that can be used (that is the oldest client version that can connect to the cluster) without triggering this health warning is determined by the mon_crush_min_required_version configuration option.

OLD_CRUSH_STRAW_CALC_VERSION

The CRUSH Map is using an older, non-optimal method for calculating intermediate weight values for straw buckets. The CRUSH Map should be updated to use the newer method (straw_calc_version = 1).

CACHE_POOL_NO_HIT_SET

One or more cache pools is not configured with a hit set to track usage, which prevents the tiering agent from identifying cold objects to flush and evict from the cache. Hit sets can be configured on the cache pool with:

```
cephuser@adm > ceph osd pool set poolname hit_set_type type
cephuser@adm > ceph osd pool set poolname hit_set_period period-in-seconds
cephuser@adm > ceph osd pool set poolname hit_set_count number-of-hitsets
cephuser@adm > ceph osd pool set poolname hit_set_fpp target-false-positive-rate
```

OSD_NO_SORTBITWISE

No pre-Luminous v12 OSDs are running but the `sortbitwise` flag has not been set. You need to set the `sortbitwise` flag before Luminous v12 or newer OSDs can start:

```
cephuser@adm > ceph osd set sortbitwise
```

POOL_FULL

One or more pools has reached its quota and is no longer allowing writes. You can set pool quotas and usage with:

```
cephuser@adm > ceph df detail
```

You can either raise the pool quota with

```
cephuser@adm > ceph osd pool set-quota poolname max_objects num-objects
cephuser@adm > ceph osd pool set-quota poolname max_bytes num-bytes
```

or delete some existing data to reduce usage.

PG_AVAILABILITY

Data availability is reduced, meaning that the cluster is unable to service potential read or write requests for some data in the cluster. Specifically, one or more PGs is in a state that does not allow I/O requests to be serviced. Problematic PG states include *peering*, *stale*, *incomplete*, and the lack of *active* (if those conditions do not clear quickly). Detailed information about which PGs are affected is available from:

```
cephuser@adm > ceph health detail
```

In most cases the root cause is that one or more OSDs is currently down. The state of specific problematic PGs can be queried with:

```
cephuser@adm > ceph tell pgid query
```

PG_DEGRADED

Data redundancy is reduced for some data, meaning the cluster does not have the desired number of replicas for all data (for replicated pools) or erasure code fragments (for erasure coded pools). Specifically, one or more PGs have either the *degraded* or *undersized* flag set (there are not enough instances of that placement group in the cluster), or have not had the *clean* flag set for some time. Detailed information about which PGs are affected is available from:

```
cephuser@adm > ceph health detail
```

In most cases the root cause is that one or more OSDs is currently down. The state of specific problematic PGs can be queried with:

```
cephuser@adm > ceph tell pgid query
```

PG_DEGRADED_FULL

Data redundancy may be reduced or at risk for some data because of a lack of free space in the cluster. Specifically, one or more PGs has the *backfill_toofull* or *recovery_toofull* flag set, meaning that the cluster is unable to migrate or recover data because one or more OSDs is above the *backfillfull* threshold.

PG_DAMAGED

Data scrubbing (see [Section 17.6, “Scrubbing placement groups”](#)) has discovered some problems with data consistency in the cluster. Specifically, one or more PGs has the *inconsistent* or *snaptrim_error* flag is set, indicating an earlier scrub operation found a problem, or that the *repair* flag is set, meaning a repair for such an inconsistency is currently in progress.

OSD_SCRUB_ERRORS

Recent OSD scrubs have uncovered inconsistencies.

CACHE_POOL_NEAR_FULL

A cache tier pool is nearly full. Full in this context is determined by the *target_max_bytes* and *target_max_objects* properties on the cache pool. When the pool reaches the target threshold, write requests to the pool may block while data is flushed and evicted from the cache, a state that normally leads to very high latencies and poor performance. The cache pool target size can be adjusted with:

```
cephuser@adm > ceph osd pool set cache-pool-name target_max_bytes bytes  
cephuser@adm > ceph osd pool set cache-pool-name target_max_objects objects
```

Normal cache flush and evict activity may also be throttled because of reduced availability or performance of the base tier, or overall cluster load.

TOO_FEW_PGS

The number of PGs in use is below the configurable threshold of `mon_pg_warn_min_per_osd` PGs per OSD. This can lead to suboptimal distribution and balance of data across the OSDs in the cluster reduce overall performance.

TOO_MANY_PGS

The number of PGs in use is above the configurable threshold of `mon_pg_warn_max_per_osd` PGs per OSD. This can lead to higher memory usage for OSD daemons, slower peering after cluster state changes (for example OSD restarts, additions, or removals), and higher load on the Ceph Managers and Ceph Monitors.

While the `pg_num` value for existing pools cannot be reduced, the `pgp_num` value can. This effectively co-locates some PGs on the same sets of OSDs, mitigating some of the negative impacts described above. The `pgp_num` value can be adjusted with:

```
cephuser@adm > ceph osd pool set pool pgp_num value
```

SMALLER_PGP_NUM

One or more pools has a `pgp_num` value less than `pg_num`. This is normally an indication that the PG count was increased without also increasing the placement behavior. This is normally resolved by setting `pgp_num` to match `pg_num`, triggering the data migration, with:

```
cephuser@adm > ceph osd pool set pool pgp_num pg_num_value
```

MANY_OBJECTS_PER_PG

One or more pools have an average number of objects per PG that is significantly higher than the overall cluster average. The specific threshold is controlled by the `mon_pg_warn_max_object_skew` configuration value. This is usually an indication that the pool(s) containing most of the data in the cluster have too few PGs, and/or that other pools that do not contain as much data have too many PGs. The threshold can be raised to silence the health warning by adjusting the `mon_pg_warn_max_object_skew` configuration option on the monitors.

POOL_APP_NOT_ENABLED

A pool exists that contains one or more objects but has not been tagged for use by a particular application. Resolve this warning by labeling the pool for use by an application. For example, if the pool is used by RBD:

```
cephuser@adm > rbd pool init pool_name
```

If the pool is being used by a custom application 'foo', you can also label it using the low-level command:

```
cephuser@adm > ceph osd pool application enable foo
```

POOL_FULL

One or more pools have reached (or is very close to reaching) its quota. The threshold to trigger this error condition is controlled by the `mon_pool_quota_crit_threshold` configuration option. Pool quotas can be adjusted up or down (or removed) with:

```
cephuser@adm > ceph osd pool set-quota pool max_bytes bytes  
cephuser@adm > ceph osd pool set-quota pool max_objects objects
```

Setting the quota value to 0 will disable the quota.

POOL_NEAR_FULL

One or more pools are approaching their quota. The threshold to trigger this warning condition is controlled by the `mon_pool_quota_warn_threshold` configuration option. Pool quotas can be adjusted up or down (or removed) with:

```
cephuser@adm > ceph osd osd pool set-quota pool max_bytes bytes  
cephuser@adm > ceph osd osd pool set-quota pool max_objects objects
```

Setting the quota value to 0 will disable the quota.

OBJECT_MISPLACED

One or more objects in the cluster are not stored on the node where the cluster wants them to be. This is an indication that data migration caused by a recent cluster change has not yet completed. Misplaced data is not a dangerous condition in itself. Data consistency is never at risk, and old copies of objects are never removed until the desired number of new copies (in the desired locations) are present.

OBJECT_UNFOUND

One or more objects in the cluster cannot be found. Specifically, the OSDs know that a new or updated copy of an object should exist, but a copy of that version of the object has not been found on the OSDs that are currently up. Read or write requests to the 'unfound' objects will be blocked. Ideally, the down OSD that has the most recent copy of the unfound object can be brought back up. Candidate OSDs can be identified from the peering state for the PG(s) responsible for the unfound object:

```
cephuser@adm > ceph tell pgid query
```

REQUEST_SLOW

One or more OSD requests is taking a long time to process. This can be an indication of extreme load, a slow storage device, or a software bug. You can query the request queue on the OSD(s) in question with the following command executed from the OSD host:

```
cephuser@adm > cephadm enter --name osd.ID -- ceph daemon osd.ID ops
```

You can see a summary of the slowest recent requests:

```
cephuser@adm > cephadm enter --name osd.ID -- ceph daemon osd.ID dump_historic_ops
```

You can find the location of an OSD with:

```
cephuser@adm > ceph osd find osd.id
```

REQUEST_STUCK

One or more OSD requests have been blocked for a relatively long time, for example 4096 seconds. This is an indication that either the cluster has been unhealthy for an extended period of time (for example, not enough running OSDs or inactive PGs) or there is some internal problem with the OSD.

PG_NOT_SCRUBBED

One or more PGs have not been scrubbed (see [Section 17.6, “Scrubbing placement groups”](#)) recently. PGs are normally scrubbed every `mon_scrub_interval` seconds, and this warning triggers when `mon_warn_not_scrubbed` such intervals have elapsed without a scrub. PGs will not scrub if they are not flagged as clean, which may happen if they are misplaced or degraded (see `PG_AVAILABILITY` and `PG_DEGRADED` above). You can manually initiate a scrub of a clean PG with:

```
cephuser@adm > ceph pg scrub pgid
```

PG_NOT_DEEP_SCRUBBED

One or more PGs has not been deep scrubbed (see [Section 17.6, “Scrubbing placement groups”](#)) recently. PGs are normally scrubbed every `osd_deep_mon_scrub_interval` seconds, and this warning triggers when `mon_warn_not_deep_scrubbed` seconds have elapsed without a scrub. PGs will not (deep) scrub if they are not flagged as clean, which may happen if they are misplaced or degraded (see `PG_AVAILABILITY` and `PG_DEGRADED` above). You can manually initiate a scrub of a clean PG with:

```
cephuser@adm > ceph pg deep-scrub pgid
```



Tip

If you specified non-default locations for your configuration or keyring, you may specify their locations:

```
# ceph -c /path/to/conf -k /path/to/keyring health
```

12.3 Checking a cluster's usage stats

To check a cluster's data usage and distribution among pools, use the `ceph df` command. To get more details, use `ceph df detail`.

```
cephuser@adm > ceph df
--- RAW STORAGE ---
CLASS  SIZE   AVAIL  USED    RAW USED  %RAW USED
hdd    30 GiB 27 GiB 121 MiB 3.1 GiB   10.40
TOTAL  30 GiB 27 GiB 121 MiB 3.1 GiB   10.40

--- POOLS ---
POOL                                ID  STORED  OBJECTS  USED    %USED  MAX AVAIL
device_health_metrics              1    0 B      0        0 B     0      8.5 GiB
cephfs.my_cephfs.meta              2  1.0 MiB  22      4.5 MiB  0.02   8.5 GiB
cephfs.my_cephfs.data              3    0 B      0        0 B     0      8.5 GiB
.rgw.root                          4  1.9 KiB  13      2.2 MiB  0      8.5 GiB
myzone.rgw.log                     5  3.4 KiB  207     6 MiB   0.02   8.5 GiB
myzone.rgw.control                 6    0 B      8        0 B     0      8.5 GiB
myzone.rgw.meta                    7    0 B      0        0 B     0      8.5 GiB
```

The `RAW STORAGE` section of the output provides an overview of the amount of storage your cluster uses for your data.

- **CLASS**: The storage class of the device. Refer to [Section 17.1.1, "Device classes"](#) for more details on device classes.
- **SIZE**: The overall storage capacity of the cluster.
- **AVAIL**: The amount of free space available in the cluster.
- **USED**: The space (accumulated over all OSDs) allocated purely for data objects kept at block device.

- RAW USED : The sum of 'USED' space and space allocated/reserved at block device for Ceph purposes, for example BlueFS part for BlueStore.
- % RAW USED : The percentage of raw storage used. Use this number in conjunction with the full ratio and near full ratio to ensure that you are not reaching your cluster's capacity. See [Section 12.8, "Storage capacity"](#) for additional details.



Note: Cluster fill level

When a raw storage fill level is getting close to 100%, you need to add new storage to the cluster. A higher usage may lead to single full OSDs and cluster health problems.

Use the command `ceph osd df tree` to list the fill level of all OSDs.

The POOLS section of the output provides a list of pools and the notional usage of each pool. The output from this section *does not* reflect replicas, clones or snapshots. For example, if you store an object with 1MB of data, the notional usage will be 1MB, but the actual usage may be 2MB or more depending on the number of replicas, clones and snapshots.

- POOL : The name of the pool.
- ID : The pool ID.
- STORED : The amount of data stored by the user.
- OBJECTS : The notional number of objects stored per pool.
- USED : The amount of space allocated purely for data by all OSD nodes in kB.
- %USED : The notional percentage of storage used per pool.
- MAX AVAIL : The maximum available space in the given pool.



Note

The numbers in the POOLS section are notional. They are not inclusive of the number of replicas, snapshots or clones. As a result, the sum of the USED and %USED amounts will not add up to the RAW USED and %RAW USED amounts in the RAW STORAGE section of the output.

12.4 Checking OSD status

You can check OSDs to ensure they are up and on by executing:

```
cephuser@adm > ceph osd stat
```

or

```
cephuser@adm > ceph osd dump
```

You can also view OSDs according to their position in the CRUSH map.

ceph osd tree will print a CRUSH tree with a host, its OSDs, whether they are up, and their weight:

```
cephuser@adm > ceph osd tree
ID CLASS WEIGHT TYPE NAME STATUS REWEIGHT PRI-AFF
-1 3 0.02939 root default
-3 3 0.00980 rack mainrack
-2 3 0.00980 host osd-host
0 1 0.00980 osd.0 up 1.00000 1.00000
1 1 0.00980 osd.1 up 1.00000 1.00000
2 1 0.00980 osd.2 up 1.00000 1.00000
```

12.5 Checking for full OSDs

Ceph prevents you from writing to a full OSD so that you do not lose data. In an operational cluster, you should receive a warning when your cluster is getting near its full ratio. The **mon osd full ratio** defaults to 0.95, or 95% of capacity before it stops clients from writing data. The **mon osd nearfull ratio** defaults to 0.85, or 85% of capacity, when it generates a health warning.

Full OSD nodes will be reported by **ceph health**:

```
cephuser@adm > ceph health
HEALTH_WARN 1 nearfull osds
osd.2 is near full at 85%
```

or

```
cephuser@adm > ceph health
HEALTH_ERR 1 nearfull osds, 1 full osds
osd.2 is near full at 85%
osd.3 is full at 97%
```

The best way to deal with a full cluster is to add new OSD hosts/disks allowing the cluster to redistribute data to the newly available storage.



Tip: Preventing full OSDs

After an OSD becomes full—it uses 100% of its disk space—it will normally crash quickly without warning. Following are a few tips to remember when administering OSD nodes.

- Each OSD's disk space (usually mounted under `/var/lib/ceph/osd/osd-{1,2..}`) needs to be placed on a dedicated underlying disk or partition.
- Check the Ceph configuration files and make sure that Ceph does not store its log file to the disks/partitions dedicated for use by OSDs.
- Make sure that no other process writes to the disks/partitions dedicated for use by OSDs.

12.6 Checking the monitor status

After you start the cluster and before first reading and/or writing data, check the Ceph Monitors' quorum status. When the cluster is already serving requests, check the Ceph Monitors' status periodically to ensure that they are running.

To display the monitor map, execute the following:

```
cephuser@adm > ceph mon stat
```

or

```
cephuser@adm > ceph mon dump
```

To check the quorum status for the monitor cluster, execute the following:

```
cephuser@adm > ceph quorum_status
```

Ceph will return the quorum status. For example, a Ceph cluster consisting of three monitors may return the following:

```
{ "election_epoch": 10,  
  "quorum": [  
    0,
```

```

    1,
    2],
  "monmap": { "epoch": 1,
    "fsid": "444b489c-4f16-4b75-83f0-cb8097468898",
    "modified": "2011-12-12 13:28:27.505520",
    "created": "2011-12-12 13:28:27.505520",
    "mons": [
      { "rank": 0,
        "name": "a",
        "addr": "192.168.1.10:6789\0"},
      { "rank": 1,
        "name": "b",
        "addr": "192.168.1.11:6789\0"},
      { "rank": 2,
        "name": "c",
        "addr": "192.168.1.12:6789\0"}
    ]
  }
}

```

12.7 Checking placement group states

Placement groups map objects to OSDs. When you monitor your placement groups, you will want them to be active and clean. For a detailed discussion, refer to [Section 12.9, “Monitoring OSDs and placement groups”](#).

12.8 Storage capacity

When a Ceph storage cluster gets close to its maximum capacity, Ceph prevents you from writing to or reading from Ceph OSDs as a safety measure to prevent data loss. Therefore, letting a production cluster approach its full ratio is not a good practice, because it sacrifices high availability. The default full ratio is set to .95, meaning 95% of capacity. This is a very aggressive setting for a test cluster with a small number of OSDs.



Tip: Increase Storage Capacity

When monitoring your cluster, be alert to warnings related to the nearfull ratio. It means that a failure of some OSDs could result in a temporary service disruption if one or more OSDs fails. Consider adding more OSDs to increase storage capacity.

A common scenario for test clusters involves a system administrator removing a Ceph OSD from the Ceph storage cluster to watch the cluster rebalance. Then removing another Ceph OSD, and so on until the cluster eventually reaches the full ratio and locks up. We recommend a bit of capacity planning even with a test cluster. Planning enables you to estimate how much spare capacity you will need in order to maintain high availability. Ideally, you want to plan for a series of Ceph OSD failures where the cluster can recover to an `active + clean` state without replacing those Ceph OSDs immediately. You can run a cluster in an `active + degraded` state, but this is not ideal for normal operating conditions.

The following diagram depicts a simplistic Ceph storage cluster containing 33 Ceph nodes with one Ceph OSD per host, each of them reading from and writing to a 3 TB drive. This exemplary cluster has a maximum actual capacity of 99 TB. The `mon osd full ratio` option is set to 0.95. If the cluster falls to 5 TB of the remaining capacity, it will not allow the clients to read and write data. Therefore the storage cluster’s operating capacity is 95 TB, not 99 TB.

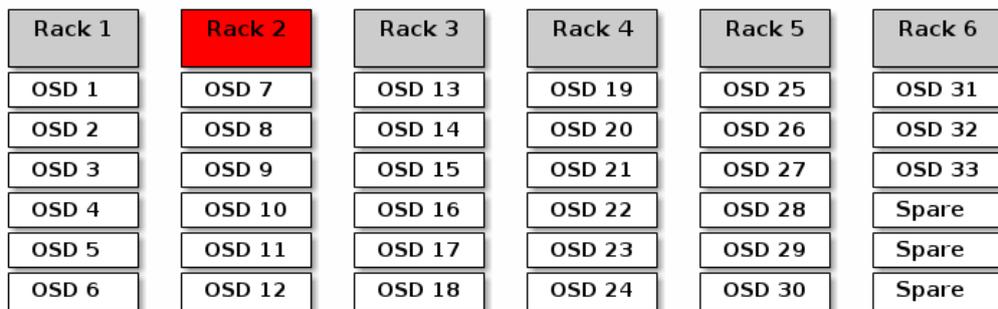


FIGURE 12.1: CEPH CLUSTER

It is normal in such a cluster for one or two OSDs to fail. A less frequent but reasonable scenario involves a rack’s router or power supply failing, which brings down multiple OSDs simultaneously (for example, OSDs 7-12). In such a scenario, you should still strive for a cluster that can remain operational and achieve an `active + clean` state—even if that means adding a few hosts with additional OSDs in short order. If your capacity usage is too high, you may not lose data. But you could still sacrifice data availability while resolving an outage within a failure domain if capacity usage of the cluster exceeds the full ratio. For this reason, we recommend at least some rough capacity planning.

Identify two numbers for your cluster:

1. The number of OSDs.
2. The total capacity of the cluster.

If you divide the total capacity of your cluster by the number of OSDs in your cluster, you will find the mean average capacity of an OSD within your cluster. Consider multiplying that number by the number of OSDs you expect will fail simultaneously during normal operations (a relatively small number). Finally, multiply the capacity of the cluster by the full ratio to arrive at a maximum operating capacity. Then, subtract the number of the amount of data from the OSDs you expect to fail to arrive at a reasonable full ratio. Repeat the foregoing process with a higher number of OSD failures (a rack of OSDs) to arrive at a reasonable number for a near full ratio. The following settings only apply on cluster creation and are then stored in the OSD map:

```
[global]
mon osd full ratio = .80
mon osd backfillfull ratio = .75
mon osd nearfull ratio = .70
```



Tip

These settings only apply during cluster creation. Afterward they need to be changed in the OSD Map using the `ceph osd set-nearfull-ratio` and `ceph osd set-full-ratio` commands.

mon osd full ratio

The percentage of disk space used before an OSD is considered `full`. Default is .95

mon osd backfillfull ratio

The percentage of disk space used before an OSD is considered too `full` to backfill. Default is .90

mon osd nearfull ratio

The percentage of disk space used before an OSD is considered `nearfull`. Default is .85



Tip: Check OSD weight

If some OSDs are `nearfull`, but others have plenty of capacity, you may have a problem with the CRUSH weight for the `nearfull` OSDs.

12.9 Monitoring OSDs and placement groups

High availability and high reliability require a fault-tolerant approach to managing hardware and software issues. Ceph has no single point-of-failure, and can service requests for data in a 'degraded' mode. Ceph's data placement introduces a layer of indirection to ensure that data does not bind directly to particular OSD addresses. This means that tracking down system faults requires finding the placement group and the underlying OSDs at root of the problem.



Tip: Access in case of failure

A fault in one part of the cluster may prevent you from accessing a particular object. That does not mean that you cannot access other objects. When you run into a fault, follow the steps for monitoring your OSDs and placement groups. Then begin troubleshooting.

Ceph is generally self-repairing. However, when problems persist, monitoring OSDs and placement groups will help you identify the problem.

12.9.1 Monitoring OSDs

An OSD's status is either *in the cluster* ('in') or *out of the cluster* ('out'). At the same time, it is either *up and running* ('up') or it is *down and not running* ('down'). If an OSD is 'up', it may be either in the cluster (you can read and write data) or out of the cluster. If it was in the cluster and recently moved out of the cluster, Ceph will migrate placement groups to other OSDs. If an OSD is out of the cluster, CRUSH will not assign placement groups to it. If an OSD is 'down', it should also be 'out'.



Note: Unhealthy state

If an OSD is 'down' and 'in', there is a problem and the cluster will not be in a healthy state.

If you execute a command such as `ceph health`, `ceph -s` or `ceph -w`, you may notice that the cluster does not always echo back `HEALTH OK`. With regard to OSDs, you should expect that the cluster will *not* echo `HEALTH OK` under the following circumstances:

- You have not started the cluster yet (it will not respond).
- You have started or restarted the cluster and it is not ready yet, because the placement groups are being created and the OSDs are in the process of peering.

- You have added or removed an OSD.
- You have modified your cluster map.

An important aspect of monitoring OSDs is to ensure that when the cluster is up and running, all the OSDs in the cluster are up and running, too. To see if all the OSDs are running, execute:

```
# ceph osd stat
x osds: y up, z in; epoch: eNNNN
```

The result should tell you the total number of OSDs (x), how many are 'up' (y), how many are 'in' (z), and the map epoch (eNNNN). If the number of OSDs that are 'in' the cluster is more than the number of OSDs that are 'up', execute the following command to identify the `ceph-osd` daemons that are not running:

```
# ceph osd tree
#ID CLASS WEIGHT  TYPE NAME                STATUS REWEIGHT PRI-AFF
-1          2.00000 pool openstack
-3          2.00000 rack dell-2950-rack-A
-2          2.00000 host dell-2950-A1
0  ssd 1.00000    osd.0                up 1.00000 1.00000
1  ssd 1.00000    osd.1                down 1.00000 1.00000
```

For example, if an OSD with ID 1 is down, start it:

```
cephuser@osd > sudo systemctl start ceph-CLUSTER_ID@osd.0.service
```

See Book "Troubleshooting Guide", Chapter 4 "Troubleshooting OSDs", Section 4.3 "OSDs not running" for problems associated with OSDs that have stopped or that will not restart.

12.9.2 Assigning placement group sets

When CRUSH assigns placement groups to OSDs, it looks at the number of replicas for the pool and assigns the placement group to OSDs such that each replica of the placement group gets assigned to a different OSD. For example, if the pool requires three replicas of a placement group, CRUSH may assign them to `osd.1`, `osd.2` and `osd.3` respectively. CRUSH actually seeks a pseudo-random placement that will take into account failure domains you set in your CRUSH Map, so you will rarely see placement groups assigned to nearest neighbor OSDs in a large cluster. We refer to the set of OSDs that should contain the replicas of a particular placement

group as the *acting set*. In some cases, an OSD in the acting set is down or otherwise not able to service requests for objects in the placement group. When these situations arise, it may match one of the following scenarios:

- You added or removed an OSD. Then, CRUSH reassigned the placement group to other OSDs and therefore changed the composition of the *acting set*, causing the migration of data with a 'backfill' process.
- An OSD was 'down', was restarted, and is now recovering.
- An OSD in the *acting set* is 'down' or unable to service requests, and another OSD has temporarily assumed its duties.

Ceph processes a client request using the *up set*, which is the set of OSDs that will actually handle the requests. In most cases, the *up set* and the *acting set* are virtually identical. When they are not, it may indicate that Ceph is migrating data, an OSD is recovering, or that there is a problem (for example, Ceph usually echoes a `HEALTH_WARN` state with a 'stuck stale' message in such scenarios).

To retrieve a list of placement groups, run:

```
cephuser@adm > ceph pg dump
```

To view which OSDs are within the *acting set* or the *up set* for a given placement group, run:

```
cephuser@adm > ceph pg map PG_NUM  
osdmap eNNN pg RAW_PG_NUM (PG_NUM) -> up [0,1,2] acting [0,1,2]
```

The result should tell you the osdmap epoch (eNNN), the placement group number (PG_NUM), the OSDs in the *up set* ('up'), and the OSDs in the *acting set* ('acting'):



Tip: Cluster problem indicator

If the *up set* and *acting set* do not match, this may be an indicator either of the cluster rebalancing itself, or of a potential problem with the cluster.

12.9.3 Peering

Before you can write data to a placement group, it must be in an active state, and it should be in a clean state. For Ceph to determine the current state of a placement group, the primary OSD of the placement group (the first OSD in the *acting set*), peers with the secondary and tertiary OSDs to establish agreement on the current state of the placement group (assuming a pool with three replicas of the PG).

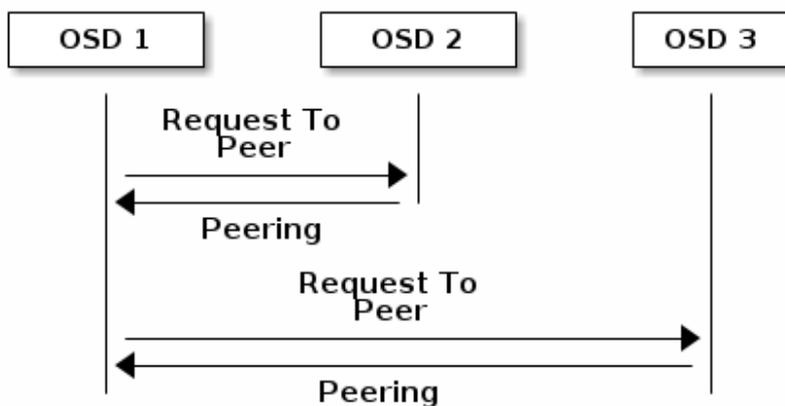


FIGURE 12.2: PEERING SCHEMA

12.9.4 Monitoring placement group states

If you execute a command such as `ceph health`, `ceph -s` or `ceph -w`, you may notice that the cluster does not always echo back the HEALTH OK message. After you check to see if the OSDs are running, you should also check placement group states.

Expect that the cluster will **not** echo HEALTH OK in a number of placement group peering-related circumstances:

- You have created a pool and placement groups have not peered yet.
- The placement groups are recovering.
- You have added an OSD to or removed an OSD from the cluster.
- You have modified your CRUSH Map and your placement groups are migrating.
- There is inconsistent data in different replicas of a placement group.

- Ceph is scrubbing a placement group's replicas.
- Ceph does not have enough storage capacity to complete backfilling operations.

If one of the above mentioned circumstances causes Ceph to echo `HEALTH_WARN`, do not panic. In many cases, the cluster will recover on its own. In some cases, you may need to take action. An important aspect of monitoring placement groups is to ensure that when the cluster is up and running, all placement groups are 'active' and preferably in the 'clean state'. To see the status of all placement groups, run:

```
cephuser@adm > ceph pg stat
x pgs: y active+clean; z bytes data, aa MB used, bb GB / cc GB avail
```

The result should tell you the total number of placement groups (x), how many placement groups are in a particular state such as 'active + clean' (y) and the amount of data stored (z).

In addition to the placement group states, Ceph will also echo back the amount of storage capacity used (aa), the amount of storage capacity remaining (bb), and the total storage capacity for the placement group. These numbers can be important in a few cases:

- You are reaching your near full ratio or full ratio.
- Your data is not getting distributed across the cluster because of an error in your CRUSH configuration.



Tip: Placement group IDs

Placement group IDs consist of the pool number (not pool name) followed by a period (.) and the placement group ID—a hexadecimal number. You can view pool numbers and their names from the output of `ceph osd lspools`. For example, the default pool `rbid` corresponds to pool number 0. A fully qualified placement group ID has the following form:

```
POOL_NUM.PG_ID
```

And it typically looks like this:

```
0.1f
```

To retrieve a list of placement groups, run the following:

```
cephuser@adm > ceph pg dump
```

You can also format the output in JSON format and save it to a file:

```
cephuser@adm > ceph pg dump -o FILE_NAME --format=json
```

To query a particular placement group, run the following:

```
cephuser@adm > ceph pg POOL_NUM.PG_ID query
```

The following list describes the common placement group states in detail.

CREATING

When you create a pool, it will create the number of placement groups you specified. Ceph will echo 'creating' when it is creating one or more placement groups. When they are created, the OSDs that are part of the placement group's *acting set* will peer. When peering is complete, the placement group status should be 'active + clean', which means that a Ceph client can begin writing to the placement group.



FIGURE 12.3: PLACEMENT GROUPS STATUS

PEERING

When Ceph is peering a placement group, it is bringing the OSDs that store the replicas of the placement group into agreement about the state of the objects and metadata in the placement group. When Ceph completes peering, this means that the OSDs that store the placement group agree about the current state of the placement group. However, completion of the peering process does **not** mean that each replica has the latest contents.



Note: Authoritative history

Ceph will **not** acknowledge a write operation to a client until all OSDs of the *acting set* persist the write operation. This practice ensures that at least one member of the *acting set* will have a record of every acknowledged write operation since the last successful peering operation.

With an accurate record of each acknowledged write operation, Ceph can construct and enlarge a new authoritative history of the placement group—a complete and fully ordered set of operations that, if performed, would bring an OSD's copy of a placement group up to date.

ACTIVE

When Ceph completes the peering process, a placement group may become active. The active state means that the data in the placement group is generally available in the primary placement group and the replicas for read and write operations.

CLEAN

When a placement group is in the clean state, the primary OSD and the replica OSDs have successfully peered and there are no stray replicas for the placement group. Ceph replicated all objects in the placement group the correct number of times.

DEGRADED

When a client writes an object to the primary OSD, the primary OSD is responsible for writing the replicas to the replica OSDs. After the primary OSD writes the object to storage, the placement group will remain in a 'degraded' state until the primary OSD has received an acknowledgement from the replica OSDs that Ceph created the replica objects successfully. The reason a placement group can be 'active + degraded' is that an OSD may be 'active' even though it does not hold all of the objects yet. If an OSD goes down, Ceph marks each placement group assigned to the OSD as 'degraded'. The OSDs must peer again when the OSD comes back up. However, a client can still write a new object to a degraded placement group if it is 'active'.

If an OSD is 'down' and the 'degraded' condition persists, Ceph may mark the down OSD as 'out' of the cluster and remap the data from the 'down' OSD to another OSD. The time between being marked 'down' and being marked 'out' is controlled by the mon osd down out interval option, which is set to 600 seconds by default.

A placement group can also be 'degraded' because Ceph cannot find one or more objects that should be in the placement group. While you cannot read or write to unfound objects, you can still access all of the other objects in the 'degraded' placement group.

RECOVERING

Ceph was designed for fault-tolerance at a scale where hardware and software problems are ongoing. When an OSD goes 'down', its contents may fall behind the current state of other replicas in the placement groups. When the OSD is back 'up', the contents of the placement groups must be updated to reflect the current state. During that time period, the OSD may reflect a 'recovering' state.

Recovery is not always trivial, because a hardware failure may cause a cascading failure of multiple OSDs. For example, a network switch for a rack or cabinet may fail, which can cause the OSDs of a number of host machines to fall behind the current state of the cluster. Each of the OSDs must recover when the fault is resolved.

Ceph provides a number of settings to balance the resource contention between new service requests and the need to recover data objects and restore the placement groups to the current state. The `osd recovery delay start` setting allows an OSD to restart, re-peer and even process some replay requests before starting the recovery process. The `osd recovery thread timeout` sets a thread timeout, because multiple OSDs may fail, restart and re-peer at staggered rates. The `osd recovery max active` setting limits the number of recovery requests an OSD will process simultaneously to prevent the OSD from failing to serve. The `osd recovery max chunk` setting limits the size of the recovered data chunks to prevent network congestion.

BACK FILLING

When a new OSD joins the cluster, CRUSH will reassign placement groups from OSDs in the cluster to the newly added OSD. Forcing the new OSD to accept the reassigned placement groups immediately can put excessive load on the new OSD. Backfilling the OSD with the placement groups allows this process to begin in the background. When backfilling is complete, the new OSD will begin serving requests when it is ready.

During the backfill operations, you may see one of several states: 'backfill_wait' indicates that a backfill operation is pending, but is not yet in progress; 'backfill' indicates that a backfill operation is in progress; 'backfill_too_full' indicates that a backfill operation was requested, but could not be completed because of insufficient storage capacity. When a placement group cannot be backfilled, it may be considered 'incomplete'.

Ceph provides a number of settings to manage the load associated with reassigning placement groups to an OSD (especially a new OSD). By default, `osd max backfills` sets the maximum number of concurrent backfills to or from an OSD to 10. The `backfill full ratio` enables an OSD to refuse a backfill request if the OSD is approaching its full ratio (90%, by default) and change with `ceph osd set-backfillfull-ratio` command. If an OSD refuses a backfill request, the `osd backfill retry interval` enables an OSD to retry the request (after 10 seconds, by default). OSDs can also set `osd backfill scan min` and `osd backfill scan max` to manage scan intervals (64 and 512, by default).

REMAPPED

When the *acting set* that services a placement group changes, the data migrates from the old *acting set* to the new *acting set*. It may take some time for a new primary OSD to service requests. So it may ask the old primary to continue to service requests until the placement group migration is complete. When data migration completes, the mapping uses the primary OSD of the new *acting set*.

STALE

While Ceph uses heartbeats to ensure that hosts and daemons are running, the `ceph-osd` daemons may also get into a 'stuck' state where they are not reporting statistics in a timely manner (for example, a temporary network fault). By default, OSD daemons report their placement group, boot and failure statistics every half second (0.5), which is more frequent than the heartbeat thresholds. If the primary OSD of a placement group's *acting set* fails to report to the monitor or if other OSDs have reported the primary OSD as 'down', the monitors will mark the placement group as 'stale'.

When you start your cluster, it is common to see the 'stale' state until the peering process completes. After your cluster has been running for a while, seeing placement groups in the 'stale' state indicates that the primary OSD for those placement groups is down or not reporting placement group statistics to the monitor.

12.9.5 Finding an object location

To store object data in the Ceph Object Store, a Ceph client needs to set an object name and specify a related pool. The Ceph client retrieves the latest cluster map and the CRUSH algorithm calculates how to map the object to a placement group, and then calculates how to assign the placement group to an OSD dynamically. To find the object location, all you need is the object name and the pool name. For example:

```
cephuser@adm > ceph osd map POOL_NAME OBJECT_NAME [NAMESPACE]
```

EXAMPLE 12.1: LOCATING AN OBJECT

As an example, let us create an object. Specify an object name 'test-object-1', a path to an example file 'testfile.txt' containing some object data, and a pool name 'data' using the **`rados put`** command on the command line:

```
cephuser@adm > rados put test-object-1 testfile.txt --pool=data
```

To verify that the Ceph Object Store stored the object, run the following:

```
cephuser@adm > rados -p data ls
```

Now, identify the object location. Ceph will output the object's location:

```
cephuser@adm > ceph osd map data test-object-1
osdmap e537 pool 'data' (0) object 'test-object-1' -> pg 0.d1743484 \
(0.4) -> up ([1,0], p0) acting ([1,0], p0)
```

To remove the example object, simply delete it using the `rados rm` command:

```
cephuser@adm > rados rm test-object-1 --pool=data
```

13 Operational tasks

13.1 Modifying the cluster configuration

To modify the configuration of an existing Ceph cluster, follow these steps:

1. Export the current configuration of the cluster to a file:

```
cephuser@adm > ceph orch ls --export --format yaml > cluster.yaml
```

2. Edit the file with the configuration and update the relevant lines. Find specification examples in *Book "Deployment Guide", Chapter 8 "Deploying the remaining core services using cephadm"* and *Section 13.4.3, "Adding OSDs using DriveGroups specification"*.

3. Apply the new configuration:

```
cephuser@adm > ceph orch apply -i cluster.yaml
```

13.2 Adding nodes

To add a new node to a Ceph cluster, follow these steps:

1. Install SUSE Linux Enterprise Server and SUSE Enterprise Storage on the new host. Refer to *Book "Deployment Guide", Chapter 5 "Installing and configuring SUSE Linux Enterprise Server"* for more information.
2. Configure the host as a Salt Minion of an already existing Salt Master. Refer to *Book "Deployment Guide", Chapter 6 "Deploying Salt"* for more information.
3. Add the new host to `ceph-salt` and make `cephadm` aware of it, for example:

```
root@master # ceph-salt config /ceph_cluster/minions add ses-min5.example.com
root@master # ceph-salt config /ceph_cluster/roles/cephadm add ses-min5.example.com
```

Refer to *Book "Deployment Guide", Chapter 7 "Deploying the bootstrap cluster using ceph-salt", Section 7.2.2 "Adding Salt Minions"* for more information.

4. Verify that the node was added to `ceph-salt`:

```
root@master # ceph-salt config /ceph_cluster/minions ls
o- minions ..... [Minions: 5]
```

```
[...]
o- ses-min5.example.com ..... [no roles]
```

5. Apply the configuration to the new cluster host:

```
root@master # ceph-salt apply ses-min5.example.com
```

6. Verify that the newly added host now belongs to the cephadm environment:

```
cephuser@adm > ceph orch host ls
HOST                ADDR                LABELS  STATUS
[...]
ses-min5.example.com  ses-min5.example.com
```

13.3 Removing nodes



Tip: Remove OSDs

If the node that you are going to remove runs OSDs, remove the OSDs from it first and check that no OSDs are running on that node. Refer to [Section 13.4.4, “Removing OSDs”](#) for more details on removing OSDs.

To remove a node from a cluster, do the following:

1. For all Ceph service types except for `node-exporter` and `crash`, remove the node's host name from the cluster placement specification file (for example, `cluster.yml`). Refer to *Book “Deployment Guide”, Chapter 8 “Deploying the remaining core services using cephadm”, Section 8.2 “Service and placement specification”* for more details. For example, if you are removing the host named `ses-min2`, remove all occurrences of `- ses-min2` from all `placement:` sections:

Update

```
service_type: rgw
service_id: EXAMPLE_NFS
placement:
  hosts:
    - ses-min2
    - ses-min3
```

to

```
service_type: rgw
service_id: EXAMPLE_NFS
placement:
  hosts:
  - ses-min3
```

Apply your changes to the configuration file:

```
cephuser@adm > ceph orch apply -i rgw-example.yaml
```

2. Remove the node from cephadm's environment:

```
cephuser@adm > ceph orch host rm ses-min2
```

3. If the node is running `crash.osd.1` and `crash.osd.2` services, remove them by running the following command on the host:

```
root@minion > cephadm rm-daemon --fsid CLUSTER_ID --name SERVICE_NAME
```

For example:

```
root@minion > cephadm rm-daemon --fsid b4b30c6e... --name crash.osd.1
root@minion > cephadm rm-daemon --fsid b4b30c6e... --name crash.osd.2
```

4. Remove all the roles from the minion you want to delete:

```
cephuser@adm > ceph-salt config /ceph_cluster/roles/tuned/throughput remove ses-min2
cephuser@adm > ceph-salt config /ceph_cluster/roles/tuned/latency remove ses-min2
cephuser@adm > ceph-salt config /ceph_cluster/roles/cephadm remove ses-min2
cephuser@adm > ceph-salt config /ceph_cluster/roles/admin remove ses-min2
```

If the minion you want to remove is the bootstrap minion, you also need to remove the bootstrap role:

```
cephuser@adm > ceph-salt config /ceph_cluster/roles/bootstrap reset
```

5. After removing all OSDs on a single host, remove the host from the CRUSH map:

```
cephuser@adm > ceph osd crush remove bucket-name
```



Note

The bucket name should be the same as the host name.

6. You can now remove the minion from the cluster:

```
cephuser@adm > ceph-salt config /ceph_cluster/minions remove ses-min2
```

! Important

In the event of a failure and the minion you are trying to remove is in a permanently powered-off state, you will need to remove the node from the Salt Master:

```
root@master # salt-key -d minion_id
```

Then, manually remove the node from `pillar_root/ceph-salt.sls`. This is typically located in `/srv/pillar/ceph-salt.sls`.

13.4 OSD management

This section describes how to add, erase, or remove OSDs in a Ceph cluster.

13.4.1 Listing disk devices

To identify used and unused disk devices on all cluster nodes, list them by running the following command:

```
cephuser@adm > ceph orch device ls
HOST      PATH      TYPE SIZE  DEVICE  AVAIL REJECT REASONS
ses-master /dev/vda  hdd  42.0G      False locked
ses-min1  /dev/vda  hdd  42.0G      False locked
ses-min1  /dev/vdb  hdd  8192M  387836 False locked, LVM detected, Insufficient space
(<5GB) on vgs
ses-min2  /dev/vdc  hdd  8192M  450575 True
```

13.4.2 Erasing disk devices

To re-use a disk device, you need to erase (or *zap*) it first:

```
ceph orch device zap HOST_NAME DISK_DEVICE
```

For example:

```
cephuser@adm > ceph orch device zap ses-min2 /dev/vdc
```



Note

If you previously deployed OSDs by using DriveGroups or the `--all-available-devices` option while the `unmanaged` flag was not set, cephadm will deploy these OSDs automatically after you erase them.

13.4.3 Adding OSDs using DriveGroups specification

DriveGroups specify the layouts of OSDs in the Ceph cluster. They are defined in a single YAML file. In this section, we will use `drive_groups.yml` as an example.

An administrator should manually specify a group of OSDs that are interrelated (hybrid OSDs that are deployed on a mixture of HDDs and SSDs) or share identical deployment options (for example, the same object store, same encryption option, stand-alone OSDs). To avoid explicitly listing devices, DriveGroups use a list of filter items that correspond to a few selected fields of `ceph-volume`'s inventory reports. cephadm will provide code that translates these DriveGroups into actual device lists for inspection by the user.

The command to apply the OSD specification to the cluster is:

```
cephuser@adm > ceph orch apply osd -i drive_groups.yml
```

To see a preview of actions and test your application, you can use the `--dry-run` option together with the `ceph orch apply osd` command. For example:

```
cephuser@adm > ceph orch apply osd -i drive_groups.yml --dry-run
...
+-----+-----+-----+-----+-----+
|SERVICE|NAME  |HOST  |DATA      |DB  |WAL  |
+-----+-----+-----+-----+-----+
|osd     |test  |mgr0  |/dev/sda  |-  |-   |
|osd     |test  |mgr0  |/dev/sdb  |-  |-   |
+-----+-----+-----+-----+-----+
```

If the `--dry-run` output matches your expectations, then simply re-run the command without the `--dry-run` option.

13.4.3.1 Unmanaged OSDs

All available clean disk devices that match the DriveGroups specification will be used as OSDs automatically after you add them to the cluster. This behavior is called a *managed* mode.

To disable the *managed* mode, add the `unmanaged: true` line to the relevant specifications, for example:

```
service_type: osd
service_id: example_drvgrp_name
placement:
  hosts:
    - ses-min2
    - ses-min3
encrypted: true
unmanaged: true
```



Tip

To change already deployed OSDs from the *managed* to *unmanaged* mode, add the `unmanaged: true` lines where applicable during the procedure described in [Section 13.1, "Modifying the cluster configuration"](#).

13.4.3.2 DriveGroups specification

Following is an example DriveGroups specification file:

```
service_type: osd
service_id: example_drvgrp_name
placement:
  host_pattern: '*'
data_devices:
  drive_spec: DEVICE_SPECIFICATION
db_devices:
  drive_spec: DEVICE_SPECIFICATION
wal_devices:
  drive_spec: DEVICE_SPECIFICATION
block_wal_size: '5G' # (optional, unit suffixes permitted)
block_db_size: '5G' # (optional, unit suffixes permitted)
encrypted: true # 'True' or 'False' (defaults to 'False')
```



Note

The option previously called "encryption" in DeepSea has been renamed to "encrypted". When applying DriveGroups in SUSE Enterprise Storage 7, ensure you use this new terminology in your service specification, otherwise the `ceph orch apply` operation will fail.

13.4.3.3 Matching disk devices

You can describe the specification using the following filters:

- By a disk model:

```
model: DISK_MODEL_STRING
```

- By a disk vendor:

```
vendor: DISK_VENDOR_STRING
```



Tip

Always enter the `DISK_VENDOR_STRING` in lowercase.

To obtain details about disk model and vendor, examine the output of the following command:

```
cephuser@adm > ceph orch device ls
HOST    PATH    TYPE  SIZE DEVICE_ID          MODEL      VENDOR
ses-min1 /dev/sdb  ssd   29.8G SATA_SSD_AF34075704240015  SATA SSD   ATA
ses-min2 /dev/sda  ssd   223G Micron_5200_MTFDDAK240TDN  Micron_5200_MTFD ATA
[...]
```

- Whether a disk is rotational or not. SSDs and NVMe drives are not rotational.

```
rotational: 0
```

- Deploy a node using *all* available drives for OSDs:

```
data_devices:
  all: true
```

- Additionally, by limiting the number of matching disks:

```
limit: 10
```

13.4.3.4 Filtering devices by size

You can filter disk devices by their size—either by an exact size, or a size range. The `size:` parameter accepts arguments in the following form:

- `'10G'` - Includes disks of an exact size.
- `'10G:40G'` - Includes disks whose size is within the range.
- `':10G'` - Includes disks less than or equal to 10 GB in size.
- `'40G:'` - Includes disks equal to or greater than 40 GB in size.

EXAMPLE 13.1: MATCHING BY DISK SIZE

```
service_type: osd
service_id: example_drvgrp_name
placement:
  host_pattern: '*'
data_devices:
  size: '40TB:'
db_devices:
  size: ':2TB'
```



Note: Quotes required

When using the `'` delimiter, you need to enclose the size in quotes, otherwise the `'` sign will be interpreted as a new configuration hash.



Tip: Unit shortcuts

Instead of Gigabytes (G), you can specify the sizes in Megabytes (M) or Terabytes (T).

13.4.3.5 DriveGroups examples

This section includes examples of different OSD setups.

EXAMPLE 13.2: SIMPLE SETUP

This example describes two nodes with the same setup:

- 20 HDDs

- Vendor: Intel
- Model: SSD-123-foo
- Size: 4 TB
- 2 SSDs
 - Vendor: Micron
 - Model: MC-55-44-ZX
 - Size: 512 GB

The corresponding `drive_groups.yml` file will be as follows:

```
service_type: osd
service_id: example_drvgrp_name
placement:
  host_pattern: '*'
data_devices:
  model: SSD-123-foo
db_devices:
  model: MC-55-44-XZ
```

Such a configuration is simple and valid. The problem is that an administrator may add disks from different vendors in the future, and these will not be included. You can improve it by reducing the filters on core properties of the drives:

```
service_type: osd
service_id: example_drvgrp_name
placement:
  host_pattern: '*'
data_devices:
  rotational: 1
db_devices:
  rotational: 0
```

In the previous example, we are enforcing all rotating devices to be declared as 'data devices' and all non-rotating devices will be used as 'shared devices' (wal, db).

If you know that drives with more than 2 TB will always be the slower data devices, you can filter by size:

```
service_type: osd
service_id: example_drvgrp_name
placement:
```

```
host_pattern: '*'
data_devices:
  size: '2TB:'
db_devices:
  size: ':2TB'
```

EXAMPLE 13.3: ADVANCED SETUP

This example describes two distinct setups: 20 HDDs should share 2 SSDs, while 10 SSDs should share 2 NVMeS.

- 20 HDDs
 - Vendor: Intel
 - Model: SSD-123-foo
 - Size: 4 TB
- 12 SSDs
 - Vendor: Micron
 - Model: MC-55-44-ZX
 - Size: 512 GB
- 2 NVMeS
 - Vendor: Samsung
 - Model: NVME-QQQQ-987
 - Size: 256 GB

Such a setup can be defined with two layouts as follows:

```
service_type: osd
service_id: example_drvgrp_name
placement:
  host_pattern: '*'
data_devices:
  rotational: 0
db_devices:
  model: MC-55-44-XZ
```

```
service_type: osd
service_id: example_drvgrp_name2
```

```
placement:
  host_pattern: '*'
data_devices:
  model: MC-55-44-XZ
db_devices:
  vendor: samsung
  size: 256GB
```

EXAMPLE 13.4: ADVANCED SETUP WITH NON-UNIFORM NODES

The previous examples assumed that all nodes have the same drives. However, that is not always the case:

Nodes 1-5:

- 20 HDDs
 - Vendor: Intel
 - Model: SSD-123-foo
 - Size: 4 TB
- 2 SSDs
 - Vendor: Micron
 - Model: MC-55-44-ZX
 - Size: 512 GB

Nodes 6-10:

- 5 NVMe
 - Vendor: Intel
 - Model: SSD-123-foo
 - Size: 4 TB
- 20 SSDs
 - Vendor: Micron
 - Model: MC-55-44-ZX
 - Size: 512 GB

You can use the 'target' key in the layout to target specific nodes. Salt target notation helps to keep things simple:

```
service_type: osd
service_id: example_drvgrp_one2five
placement:
  host_pattern: 'node[1-5]'
data_devices:
  rotational: 1
db_devices:
  rotational: 0
```

followed by

```
service_type: osd
service_id: example_drvgrp_rest
placement:
  host_pattern: 'node[6-10]'
data_devices:
  model: MC-55-44-XZ
db_devices:
  model: SSD-123-foo
```

EXAMPLE 13.5: EXPERT SETUP

All previous cases assumed that the WALs and DBs use the same device. It is however possible to deploy the WAL on a dedicated device as well:

- 20 HDDs
 - Vendor: Intel
 - Model: SSD-123-foo
 - Size: 4 TB
- 2 SSDs
 - Vendor: Micron
 - Model: MC-55-44-ZX
 - Size: 512 GB
- 2 NVMeS

- Vendor: Samsung
- Model: NVME-QQQQ-987
- Size: 256 GB

```
service_type: osd
service_id: example_drvgrp_name
placement:
  host_pattern: '*'
data_devices:
  model: MC-55-44-XZ
db_devices:
  model: SSD-123-foo
wal_devices:
  model: NVME-QQQQ-987
```

EXAMPLE 13.6: COMPLEX (AND UNLIKELY) SETUP

In the following setup, we are trying to define:

- 20 HDDs backed by 1 NVMe
- 2 HDDs backed by 1 SSD(db) and 1 NVMe (wal)
- 8 SSDs backed by 1 NVMe
- 2 SSDs stand-alone (encrypted)
- 1 HDD is spare and should not be deployed

The summary of used drives is as follows:

- 23 HDDs
 - Vendor: Intel
 - Model: SSD-123-foo
 - Size: 4 TB
- 10 SSDs

- Vendor: Micron
- Model: MC-55-44-ZX
- Size: 512 GB
- 1 NVMe
 - Vendor: Samsung
 - Model: NVME-QQQQ-987
 - Size: 256 GB

The DriveGroups definition will be the following:

```
service_type: osd
service_id: example_drvgrp_hdd_nvme
placement:
  host_pattern: '*'
data_devices:
  rotational: 0
db_devices:
  model: NVME-QQQQ-987
```

```
service_type: osd
service_id: example_drvgrp_hdd_ssd_nvme
placement:
  host_pattern: '*'
data_devices:
  rotational: 0
db_devices:
  model: MC-55-44-XZ
wal_devices:
  model: NVME-QQQQ-987
```

```
service_type: osd
service_id: example_drvgrp_ssd_nvme
placement:
  host_pattern: '*'
data_devices:
  model: SSD-123-foo
db_devices:
  model: NVME-QQQQ-987
```

```
service_type: osd
```

```
service_id: example_drvgrp_standalone_encrypted
placement:
  host_pattern: '*'
data_devices:
  model: SSD-123-foo
encrypted: True
```

One HDD will remain as the file is being parsed from top to bottom.

13.4.4 Removing OSDs

Before removing an OSD node from the cluster, verify that the cluster has more free disk space than the OSD disk you are going to remove. Be aware that removing an OSD results in rebalancing of the whole cluster.

1. Identify which OSD to remove by getting its ID:

```
cephuser@adm > ceph orch ps --daemon_type osd
NAME   HOST           STATUS          REFRESHED  AGE  VERSION
osd.0  target-ses-090 running (3h)    7m ago     3h   15.2.7.689 ...
osd.1  target-ses-090 running (3h)    7m ago     3h   15.2.7.689 ...
osd.2  target-ses-090 running (3h)    7m ago     3h   15.2.7.689 ...
osd.3  target-ses-090 running (3h)    7m ago     3h   15.2.7.689 ...
```

2. Remove one or more OSDs from the cluster:

```
cephuser@adm > ceph orch osd rm OSD1_ID OSD2_ID ...
```

For example:

```
cephuser@adm > ceph orch osd rm 1 2
```

3. You can query the state of the removal operation:

```
cephuser@adm > ceph orch osd rm status
OSD_ID  HOST           STATE                                PG_COUNT  REPLACE  FORCE  STARTED_AT
2       cephadm-dev   done, waiting for purge             0         True     False 2020-07-17
13:01:43.147684
3       cephadm-dev   draining                             17        False   True  2020-07-17
13:01:45.162158
4       cephadm-dev   started                              42        False   True  2020-07-17
13:01:45.162158
```

13.4.4.1 Stopping OSD removal

After you have scheduled an OSD removal, you can stop the removal if needed. The following command will reset the initial state of the OSD and remove it from the queue:

```
cephuser@adm > ceph orch osd rm stop OSD_SERVICE_ID
```

13.4.5 Replacing OSDs

There are several reasons why you may need to replace an OSD disk. For example:

- The OSD disk failed or is soon going to fail based on SMART information, and can no longer be used to store data safely.
- You need to upgrade the OSD disk, for example to increase its size.
- You need to change the OSD disk layout.
- You plan to move from a non-LVM to a LVM-based layout.

To replace an OSD while preserving its ID, run:

```
cephuser@adm > ceph orch osd rm OSD_SERVICE_ID --replace
```

For example:

```
cephuser@adm > ceph orch osd rm 4 --replace
```

Replacing an OSD is identical to removing an OSD (see [Section 13.4.4, “Removing OSDs”](#) for more details) with the exception that the OSD is not permanently removed from the CRUSH hierarchy and is assigned a `destroyed` flag instead.

The `destroyed` flag is used to determined OSD IDs that will be reused during the next OSD deployment. Newly added disks that match the DriveGroups specification (see [Section 13.4.3, “Adding OSDs using DriveGroups specification”](#) for more details) will be assigned OSD IDs of their replaced counterpart.



Tip

Appending the `--dry-run` option will not execute the actual replacement, but will preview the steps that would normally happen.



Note

In the case of replacing an OSD after a failure, we highly recommend triggering a deep scrub of the placement groups. See [Section 17.6, “Scrubbing placement groups”](#) for more details.

Run the following command to initiate a deep scrub:

```
cephuser@adm > ceph osd deep-scrub osd.OSD_NUMBER
```



Important: Shared device failure

If a shared device for DB/WAL fails you will need to perform the replacement procedure for all OSDs that share the failed device.

13.5 Moving the Salt Master to a new node

If you need to replace the Salt Master host with a new one, follow these steps:

1. Export the cluster configuration and back up the exported JSON file. Find more details in *Book “Deployment Guide”, Chapter 7 “Deploying the bootstrap cluster using ceph-salt”, Section 7.2.14 “Exporting cluster configurations”*.
2. If the old Salt Master is also the only administration node in the cluster, then manually move `/etc/ceph/ceph.client.admin.keyring` and `/etc/ceph/ceph.conf` to the new Salt Master.
3. Stop and disable the Salt Master `systemd` service on the old Salt Master node:

```
root@master # systemctl stop salt-master.service
root@master # systemctl disable salt-master.service
```

4. If the old Salt Master node is no longer in the cluster, also stop and disable the Salt Minion `systemd` service:

```
root@master # systemctl stop salt-minion.service
root@master # systemctl disable salt-minion.service
```



Warning

Do not stop or disable the `salt-minion.service` if the old Salt Master node has any Ceph daemons (MON, MGR, OSD, MDS, gateway, monitoring) running on it.

5. Install SUSE Linux Enterprise Server 15 SP2 on the new Salt Master following the procedure described in *Book "Deployment Guide", Chapter 5 "Installing and configuring SUSE Linux Enterprise Server"*.



Tip: Transition of Salt Minion

To simplify the transition of Salt Minions to the new Salt Master, remove the original Salt Master's public key from each of them:

```
root@minion > rm /etc/salt/pki/minion/minion_master.pub
root@minion > systemctl restart salt-minion.service
```

6. Install the `salt-master` package and, if applicable, the `salt-minion` package on the new Salt Master.
7. Install `ceph-salt` on the new Salt Master node:

```
root@master # zypper install ceph-salt
root@master # systemctl restart salt-master.service
root@master # salt '*' saltutil.sync_all
```



Important

Make sure to run all three commands before continuing. The commands are idempotent; it does not matter if they get repeated.

8. Include the new Salt Master in the cluster as described in *Book "Deployment Guide", Chapter 7 "Deploying the bootstrap cluster using ceph-salt", Section 7.1 "Installing ceph-salt", Book "Deployment Guide", Chapter 7 "Deploying the bootstrap cluster using ceph-salt", Section 7.2.2 "Adding Salt Minions" and Book "Deployment Guide", Chapter 7 "Deploying the bootstrap cluster using ceph-salt", Section 7.2.4 "Specifying Admin Node"*.

9. Import the backed up cluster configuration and apply it:

```
root@master # ceph-salt import CLUSTER_CONFIG.json
root@master # ceph-salt apply
```



Important

Rename the Salt Master's `minion_id` in the exported `CLUSTER_CONFIG.json` file before importing it.

13.6 Updating the cluster nodes

Keep the Ceph cluster nodes up-to-date by applying rolling updates regularly.

13.6.1 Software repositories

Before patching the cluster with the latest software packages, verify that all the cluster's nodes have access to the relevant repositories.

13.6.2 Repository staging

If you use a staging tool—for example, SUSE Manager, Subscription Management Tool, or RMT—that serves software repositories to the cluster nodes, verify that stages for both 'Updates' repositories for SUSE Linux Enterprise Server and SUSE Enterprise Storage are created at the same point in time.

We strongly recommend to use a staging tool to apply patches which have `frozen` or `staged` patch levels. This ensures that new nodes joining the cluster have the same patch level as the nodes already running in the cluster. This way you avoid the need to apply the latest patches to all the cluster's nodes before new nodes can join the cluster.

13.6.3 Downtime of Ceph services

Depending on the configuration, cluster nodes may be rebooted during the update. If there is a single point of failure for services such as Object Gateway, Samba Gateway, NFS Ganesha, or iSCSI, the client machines may be temporarily disconnected from services whose nodes are being rebooted.

13.6.4 Running the update

To update the software packages on all cluster nodes to the latest version, run the following command:

```
root@master # ceph-salt update
```

13.7 Updating Ceph

You can instruct `cephadm` to update Ceph from one bugfix release to another. The automated update of Ceph services respects the recommended order—it starts with Ceph Managers, Ceph Monitors, and then continues on to other services such as Ceph OSDs, Metadata Servers, and Object Gateways. Each daemon is restarted only after Ceph indicates that the cluster will remain available.



Note

The following update procedure uses the `ceph orch upgrade` command. Keep in mind that the following instructions detail how to update your Ceph cluster with a product version (for example, a maintenance update), and *does not* provide instructions on how to upgrade your cluster from one product version to another.

13.7.1 Starting the update

Before you start the update, verify that all nodes are currently online and your cluster is healthy:

```
cephuser@adm > cephadm shell -- ceph -s
```

To update to a specific Ceph release:

```
cephuser@adm > ceph orch upgrade start --image REGISTRY_URL
```

For example:

```
cephuser@adm > ceph orch upgrade start --image registry.suse.com/ses/7/ceph/ceph:latest
```

Upgrade packages on the hosts:

```
cephuser@adm > ceph-salt update
```

13.7.2 Monitoring the update

Run the following command to determine whether an update is in progress:

```
cephuser@adm > ceph orch upgrade status
```

While the update is in progress, you will see a progress bar in the Ceph status output:

```
cephuser@adm > ceph -s
[...]
progress:
  Upgrade to registry.suse.com/ses/7/ceph/ceph:latest (00h 20m 12s)
    [=====.....] (time remaining: 01h 43m 31s)
```

You can also watch the cephadm log:

```
cephuser@adm > ceph -W cephadm
```

13.7.3 Cancelling an update

You can stop the update process at any time:

```
cephuser@adm > ceph orch upgrade stop
```

13.8 Halting or rebooting cluster

In some cases it may be necessary to halt or reboot the whole cluster. We recommended carefully checking for dependencies of running services. The following steps provide an outline for stopping and starting the cluster:

1. Tell the Ceph cluster not to mark OSDs as out:

```
cephuser@adm > ceph osd set noout
```

2. Stop daemons and nodes in the following order:
 1. Storage clients
 2. Gateways, for example NFS Ganesha or Object Gateway
 3. Metadata Server
 4. Ceph OSD
 5. Ceph Manager
 6. Ceph Monitor
3. If required, perform maintenance tasks.
4. Start the nodes and servers in the reverse order of the shutdown process:
 1. Ceph Monitor
 2. Ceph Manager
 3. Ceph OSD
 4. Metadata Server
 5. Gateways, for example NFS Ganesha or Object Gateway
 6. Storage clients
5. Remove the noout flag:

```
cephuser@adm > ceph osd unset noout
```

13.9 Removing an entire Ceph cluster

The `ceph-salt purge` command removes the entire Ceph cluster. If there are more Ceph clusters deployed, the one reported by `ceph -s` is purged. This way you can clean the cluster environment when testing different setups.

To prevent accidental deletion, the orchestration checks if the safety is disengaged. You can disengage the safety measures and remove the Ceph cluster by running:

```
root@master # ceph-salt disengage-safety
```

```
root@master # ceph-salt purge
```

14 Operation of Ceph services

You can operate Ceph services on a daemon, node, or cluster level. Depending on which approach you need, use `cephadm` or the `systemctl` command.

14.1 Operating individual services

If you need to operate an individual service, identify it first:

```
cephuser@adm > ceph orch ps
```

NAME	HOST	STATUS	REFRESHED	[...]
mds.my_cephfs.ses-min1.oterul	ses-min1	running (5d)	8m ago	
mgr.ses-min1.gpijpm	ses-min1	running (5d)	8m ago	
mgr.ses-min2.oopvyh	ses-min2	running (5d)	8m ago	
mon.ses-min1	ses-min1	running (5d)	8m ago	
mon.ses-min2	ses-min2	running (5d)	8m ago	
mon.ses-min4	ses-min4	running (5d)	7m ago	
osd.0	ses-min2	running (61m)	8m ago	
osd.1	ses-min3	running (61m)	7m ago	
osd.2	ses-min4	running (61m)	7m ago	
rgw.myrealm.myzone.ses-min1.kwwazo	ses-min1	running (5d)	8m ago	
rgw.myrealm.myzone.ses-min2.jngabw	ses-min2	error	8m ago	

To identify a service on a specific node, run:

```
ceph orch ps NODE_HOST_NAME
```

For example:

```
cephuser@adm > ceph orch ps ses-min2
```

NAME	HOST	STATUS	REFRESHED
mgr.ses-min2.oopvyh	ses-min2	running (5d)	3m ago
mon.ses-min2	ses-min2	running (5d)	3m ago
osd.0	ses-min2	running (67m)	3m ago



Tip

The `ceph orch ps` command supports several output formats. To change it, append the `--format FORMAT` option where *FORMAT* is one of `json`, `json-pretty`, or `yaml`. For example:

```
cephuser@adm > ceph orch ps --format yaml
```

Once you know the name of the service you can start, restart, or stop it:

```
ceph orch daemon COMMAND SERVICE_NAME
```

For example, to restart the OSD service with ID 0, run:

```
cephuser@adm > ceph orch daemon restart osd.0
```

14.2 Operating service types

If you need to operate a specific type of service across the whole Ceph cluster, use the following command:

```
ceph orch COMMAND SERVICE_TYPE
```

Replace *COMMAND* with either start, stop, or restart.

For example, the following command restarts all MONs in the cluster, regardless of which nodes they actually run on:

```
cephuser@adm > ceph orch restart mon
```

14.3 Operating services on a single node

By using the **systemctl** command, you can operate Ceph related systemd services and targets on a single node.

14.3.1 Identifying services and targets

Before operating Ceph related systemd services and targets, you need to identify the file names of their unit files. File names of the services have the following pattern:

```
ceph-FSID@SERVICE_TYPE.ID.service
```

For example:

```
ceph-b4b30c6e-9681-11ea-ac39-525400d7702d@mon.doc-ses-min1.service
```

```
ceph-b4b30c6e-9681-11ea-ac39-525400d7702d@rgw.myrealm.myzone.doc-ses-min1.kwwazo.service
```

FSID

Unique ID of the Ceph cluster. You can find it in the output of the `ceph fsid` command.

SERVICE_TYPE

Type of the service, for example `osd`, `mon`, or `rgw`.

ID

Identification string of the service. For OSDs, it is the ID number of the service. For other services, it can be either a host name of the node, or additional strings relevant for the service type.



Tip

The `SERVICE_TYPE.ID` part is identical to the content of the `NAME` column in the output of the `ceph orch ps` command.

14.3.2 Operating all services on a node

By using Ceph's `systemd` targets, you can simultaneously operate either *all* services on a node, or all services that *belong to a cluster* identified by its `FSID`.

For example, to stop all Ceph services on a node regardless to which cluster the services belong to, run:

```
root@minion > systemctl stop ceph.target
```

To restart all services that belong to a Ceph cluster with ID `b4b30c6e-9681-11ea-ac39-525400d7702d`, run:

```
root@minion > systemctl restart ceph-b4b30c6e-9681-11ea-ac39-525400d7702d.target
```

14.3.3 Operating an individual service on a node

After you have identified the name of a specific service, operate it the following way:

```
systemctl COMMAND SERVICE_NAME
```

For example, to restart a single OSD service with ID 1 on a cluster with ID `b4b30c6e-9681-11ea-ac39-525400d7702d`, run:

```
# systemctl restart ceph-b4b30c6e-9681-11ea-ac39-525400d7702d@osd.1.service
```

14.3.4 Querying the service status

You can query `systemd` for the status of services. For example:

```
# systemctl status ceph-b4b30c6e-9681-11ea-ac39-525400d7702d@osd.0.service
```

14.4 Shutting down and restarting the whole Ceph cluster

Shutting down and restarting the cluster may be necessary in the case of a planned power outage. To stop all Ceph related services and restart without issue, follow the steps below.

PROCEDURE 14.1: SHUTTING DOWN THE WHOLE CEPH CLUSTER

1. Shut down or disconnect any clients accessing the cluster.
2. To prevent CRUSH from automatically rebalancing the cluster, set the cluster to `noout`:

```
cephuser@adm > ceph osd set noout
```

3. Stop all Ceph services on all cluster nodes:

```
root@master # ceph-salt stop
```

4. Power off all cluster nodes:

```
root@master # salt -G 'ceph-salt:member' cmd.run "shutdown -h"
```

PROCEDURE 14.2: STARTING THE WHOLE CEPH CLUSTER

1. Power on the Admin Node.
2. Power on the Ceph Monitor nodes.
3. Power on the Ceph OSD nodes.

4. Unset the previously set `noout` flag:

```
root@master # ceph osd unset noout
```

5. Power on all configured gateways.
6. Power on or connect cluster clients.

15 Backup and restore

This chapter explains which parts of the Ceph cluster you should back up in order to be able to restore its functionality.

15.1 Back Up Cluster Configuration and Data

15.1.1 Back up ceph-salt configuration

Export the cluster configuration. Find more information in *Book "Deployment Guide", Chapter 7 "Deploying the bootstrap cluster using ceph-salt", Section 7.2.14 "Exporting cluster configurations"*.

15.1.2 Back up Ceph configuration

Back up the `/etc/ceph` directory. It contains crucial cluster configuration. For example, you will need a backup of `/etc/ceph` when you need to replace the Admin Node.

15.1.3 Back up Salt configuration

You need to back up the `/etc/salt/` directory. It contains the Salt configuration files, for example the Salt Master key and accepted client keys.

The Salt files are not strictly required for backing up the Admin Node, but make redeploying the Salt cluster easier. If there is no backup of these files, the Salt minions need to be registered again at the new Admin Node.



Note: Security of the Salt Master Private Key

Make sure that the backup of the Salt Master private key is stored in a safe location. The Salt Master key can be used to manipulate all cluster nodes.

15.1.4 Back up custom configurations

- Prometheus data and customization.
- Grafana customization.
- Manual changes to the iSCSI configuration.
- Ceph keys.
- CRUSH Map and CRUSH rules. Save the decompiled CRUSH Map including CRUSH rules into `crushmap-backup.txt` by running the following command:

```
cephuser@adm > ceph osd getcrushmap | crushtool -d - -o crushmap-backup.txt
```

- Samba Gateway configuration. If you are using a single gateway, backup `/etc/samba/smb.conf`. If you are using an HA setup, also back up the CTDB and Pacemaker configuration files. Refer to [Chapter 24, Export Ceph data via Samba](#) for details on what configuration is used by Samba Gateways.
- NFS Ganesha configuration. Only needed when using an HA setup. Refer to [Chapter 25, NFS Ganesha](#) for details on what configuration is used by NFS Ganesha.

15.2 Restoring a Ceph node

The procedure to recover a node from backup is to reinstall the node, replace its configuration files, and then re-orchestrate the cluster so that the replacement node is re-added.

If you need to redeploy the Admin Node, refer to [Section 13.5, “Moving the Salt Master to a new node”](#).

For minions, it is usually easier to simply rebuild and redeploy.

1. Re-install the node. Find more information in *Book “Deployment Guide”, Chapter 5 “Installing and configuring SUSE Linux Enterprise Server”*
2. Install Salt Find more information in *Book “Deployment Guide”, Chapter 6 “Deploying Salt”*
3. After restoring the `/etc/salt` directory from a backup, enable and restart applicable Salt services, for example:

```
root@master # systemctl enable salt-master
root@master # systemctl start salt-master
root@master # systemctl enable salt-minion
```

```
root@master # systemctl start salt-minion
```

4. Remove the public master key for the old Salt Master node from all the minions.

```
root@master # rm /etc/salt/pki/minion/minion_master.pub  
root@master # systemctl restart salt-minion
```

5. Restore anything that was local to the Admin Node.
6. Import the cluster configuration from the previously exported JSON file. Refer to *Book "Deployment Guide", Chapter 7 "Deploying the bootstrap cluster using ceph-salt", Section 7.2.14 "Exporting cluster configurations"* for more details.
7. Apply the imported cluster configuration:

```
root@master # ceph-salt apply
```

16 Monitoring and alerting

In SUSE Enterprise Storage 7, cephadm deploys a monitoring and alerting stack. Users need to either define the services (such as Prometheus, Alertmanager, and Grafana) that they want to deploy with cephadm in a YAML configuration file, or they can use the CLI to deploy them. When multiple services of the same type are deployed, a highly-available setup is deployed. The node exporter is an exception to this rule.

The following monitoring services can be deployed with cephadm:

- **Prometheus** is the monitoring and alerting toolkit. It collects the data provided by Prometheus exporters and fires preconfigured alerts if predefined thresholds have been reached.
- **Alertmanager** handles alerts sent by the Prometheus server. It deduplicates, groups, and routes the alerts to the correct receiver. By default, the Ceph Dashboard will automatically be configured as the receiver.
- **Grafana** is the visualization and alerting software. The alerting functionality of Grafana is not used by this monitoring stack. For alerting, the Alertmanager is used.
- **Node exporter** is an exporter for Prometheus which provides data about the node it is installed on. It is recommended to install the node exporter on all nodes.

The Prometheus Manager Module provides a Prometheus exporter to pass on Ceph performance counters from the collection point in `ceph-mgr`.

The Prometheus configuration, including *scrape* targets (metrics providing daemons), is set up automatically by cephadm. cephadm also deploys a list of default alerts, for example `health error`, `10% OSDs down`, or `pgs inactive`.

By default, traffic to Grafana is encrypted with TLS. You can either supply your own TLS certificate or use a self-signed one. If no custom certificate has been configured before Grafana has been deployed, then a self-signed certificate is automatically created and configured for Grafana.

You can configure custom certificates for Grafana by following these steps:

1. Configure certificate files:

```
cephuser@adm > ceph config-key set mgr/cephadm/grafana_key -i $PWD/key.pem
cephuser@adm > ceph config-key set mgr/cephadm/grafana_cert -i $PWD/certificate.pem
```

2. Restart the Ceph Manager service:

```
cephuser@adm > ceph orch restart mgr
```

3. Reconfigure the Grafana service to reflect the new certificate paths and set the right URL for the Ceph Dashboard:

```
cephuser@adm > ceph orch reconfig grafana
```

The Alertmanager handles alerts sent by the Prometheus server. It takes care of deduplicating, grouping, and routing them to the correct receiver. Alerts can be silenced using the Alertmanager, but silences can also be managed using the Ceph Dashboard.

We recommend that the `Node exporter` is deployed on all nodes. This can be done using the `monitoring.yaml` file with the `node-exporter` service type. See *Book "Deployment Guide", Chapter 8 "Deploying the remaining core services using cephadm", Section 8.3.8 "Deploying the monitoring stack"* for more information on deploying services.

16.1 Configuring custom or local images



Tip

This section describes how to change the configuration of container images which are used when services are deployed or updated. It does not include the commands necessary to deploy or re-deploy services.

The recommended method to deploy the monitoring stack is by applying its specification as described in *Book "Deployment Guide", Chapter 8 "Deploying the remaining core services using cephadm", Section 8.3.8 "Deploying the monitoring stack"*.

To deploy custom or local container images, the images need to be set in `cephadm`. To do so, you will need to run the following command:

```
cephuser@adm > ceph config set mgr mgr/cephadm/OPTION_NAME VALUE
```

Where `OPTION_NAME` is any of the following names:

- `container_image_prometheus`
- `container_image_node_exporter`

- `container_image_alertmanager`
- `container_image_grafana`

If no option is set or if the setting has been removed, the following images are used as VALUE:

- `registry.suse.com/ses/7/ceph/prometheus-server:2.27.1`
- `registry.suse.com/ses/7/ceph/prometheus-node-exporter:1.1.2`
- `registry.suse.com/ses/7/ceph/prometheus-alertmanager:0.21.0`
- `registry.suse.com/ses/7/ceph/grafana:7.3.1`

For example:

```
cephuser@adm > ceph config set mgr mgr/cephadm/container_image_prometheus prom/  
prometheus:v1.4.1
```



Note

By setting a custom image, the default value will be overridden (but not overwritten). The default value changes when updates become available. By setting a custom image, you will not be able to update the component you have set the custom image for automatically. You will need to manually update the configuration (image name and tag) to be able to install updates.

If you choose to go with the recommendations instead, you can reset the custom image you have set before. After that, the default value will be used again. Use **`ceph config rm`** to reset the configuration option:

```
cephuser@adm > ceph config rm mgr mgr/cephadm/OPTION_NAME
```

For example:

```
cephuser@adm > ceph config rm mgr mgr/cephadm/container_image_prometheus
```

16.2 Updating monitoring services

As mentioned in *Section 16.1, “Configuring custom or local images”*, `cephadm` is shipped with the URLs of the recommended and tested container images, and they are used by default.

By updating the Ceph packages, new versions of these URLs may be shipped. This just updates where the container images are pulled from but does not update any services.

After the URLs to the new container images have been updated, either manually as described in [Section 16.1, “Configuring custom or local images”](#), or automatically through an update of the Ceph package, the monitoring services can be updated.

To do so, use `ceph orch reconfig` like so:

```
cephuser@adm > ceph orch reconfig node-exporter
cephuser@adm > ceph orch reconfig prometheus
cephuser@adm > ceph orch reconfig alertmanager
cephuser@adm > ceph orch reconfig grafana
```

Currently no single command to update all monitoring services exists. The order in which these services are updated is not important.



Note

If you use custom container images, the URLs specified for the monitoring services will not change automatically if the Ceph packages are updated. If you have specified custom container images, you will need to specify the URLs of the new container images manually. This may be the case if you use a local container registry.

You can find the URLs of the recommended container images to be used in the [Section 16.1, “Configuring custom or local images”](#) section.

16.3 Disabling monitoring

To disable the monitoring stack, run the following commands:

```
cephuser@adm > ceph orch rm grafana
cephuser@adm > ceph orch rm prometheus --force # this will delete metrics data
collected so far
cephuser@adm > ceph orch rm node-exporter
cephuser@adm > ceph orch rm alertmanager
cephuser@adm > ceph mgr module disable prometheus
```

16.4 Configuring Grafana

The Ceph Dashboard back-end requires the Grafana URL to be able to verify the existence of Grafana Dashboards before the front-end even loads them. Because of the nature of how Grafana is implemented in Ceph Dashboard, this means that two working connections are required in order to be able to see Grafana graphs in Ceph Dashboard:

- The back-end (Ceph MGR module) needs to verify the existence of the requested graph. If this request succeeds, it lets the front-end know that it can safely access Grafana.
- The front-end then requests the Grafana graphs directly from the user's browser using an `iframe`. The Grafana instance is accessed directly without any detour through Ceph Dashboard.

Now, it might be the case that your environment makes it difficult for the user's browser to directly access the URL configured in Ceph Dashboard. To solve this issue, a separate URL can be configured which will solely be used to tell the front-end (the user's browser) which URL it should use to access Grafana.

To change the URL that is returned to the front-end issue the following command:

```
cephuser@adm > ceph dashboard set-grafana-frontend-api-url GRAFANA-SERVER-URL
```

If no value is set for that option, it will simply fall back to the value of the `GRAFANA_API_URL` option, which is set automatically and periodically updated by `cephadm`. If set, it will instruct the browser to use this URL to access Grafana.

16.5 Configuring the Prometheus Manager Module

The Prometheus Manager Module is a module inside Ceph that extends Ceph's functionality. The module reads (meta-)data from Ceph about its state and health, providing the (scraped) data in a consumable format to Prometheus.



Note

The Prometheus Manager Module needs to be restarted for the configuration changes to be applied.

16.5.1 Configuring the network interface

By default, the Prometheus Manager Module accepts HTTP requests on port 9283 on all IPv4 and IPv6 addresses on the host. The port and listen address are both configurable with `ceph config-key set`, with keys `mgr/prometheus/server_addr` and `mgr/prometheus/server_port`. This port is registered with Prometheus's registry.

To update the `server_addr` execute the following command:

```
cephuser@adm > ceph config set mgr mgr/prometheus/server_addr 0.0.0.0
```

To update the `server_port` execute the following command:

```
cephuser@adm > ceph config set mgr mgr/prometheus/server_port 9283
```

16.5.2 Configuring scrape_interval

By default, the Prometheus Manager Module is configured with a scrape interval of 15 seconds. We do not recommend using a scrape interval below 10 seconds. To set a different scrape interval in the Prometheus module, set `scrape_interval` to the desired value:



Important

To work properly and not cause any issues, the `scrape_interval` of this module should always be set to match the Prometheus scrape interval.

```
cephuser@adm > ceph config set mgr mgr/prometheus/scrape_interval 15
```

16.5.3 Configuring the cache

On large clusters (more than 1000 OSDs), the time to fetch the metrics may become significant. Without the cache, the Prometheus Manager Module can overload the manager and lead to unresponsive or crashing Ceph Manager instances. As a result, the cache is enabled by default and cannot be disabled, but this does mean that the cache can become stale. The cache is considered stale when the time to fetch the metrics from Ceph exceeds the configured `scrape_interval`.

If this is the case, a warning will be logged and the module will either:

- Respond with a 503 HTTP status code (service unavailable).
- Return the content of the cache, even though it might be stale.

This behavior can be configured using the `ceph config set` commands.

To tell the module to respond with possibly-stale data, set it to `return`:

```
cephuser@adm > ceph config set mgr mgr/prometheus/stale_cache_strategy return
```

To tell the module to respond with `service unavailable`, set it to `fail`:

```
cephuser@adm > ceph config set mgr mgr/prometheus/stale_cache_strategy fail
```

16.5.4 Enabling RBD-image monitoring

The Prometheus Manager Module can optionally collect RBD per-image IO statistics by enabling dynamic OSD performance counters. The statistics are gathered for all images in the pools that are specified in the `mgr/prometheus/rbd_stats_pools` configuration parameter.

The parameter is a comma- or space-separated list of `pool[/namespace]` entries. If the namespace is not specified, the statistics are collected for all namespaces in the pool.

For example:

```
cephuser@adm > ceph config set mgr mgr/prometheus/rbd_stats_pools "pool1,pool2,poolN"
```

The module scans the specified pools and namespaces and makes a list of all available images, and refreshes it periodically. The interval is configurable via the `mgr/prometheus/rbd_stats_pools_refresh_interval` parameter (in seconds), and is 300 seconds (five minutes) by default.

For example, if you changed the synchronization interval to 10 minutes:

```
cephuser@adm > ceph config set mgr mgr/prometheus/rbd_stats_pools_refresh_interval 600
```

16.6 Prometheus security model

Prometheus' security model presumes that untrusted users have access to the Prometheus HTTP endpoint and logs. Untrusted users have access to all the (meta-)data Prometheus collects that is contained in the database, plus a variety of operational and debugging information.

However, Prometheus' HTTP API is limited to read-only operations. Configurations cannot be changed using the API, and secrets are not exposed. Moreover, Prometheus has some built-in measures to mitigate the impact of denial-of-service attacks.

16.7 Prometheus Alertmanager SNMP webhook

If you want to get notified about Prometheus alerts via SNMP traps, then you can install the Prometheus Alertmanager SNMP webhook via cephadm. To do so, you need to create a service and placement specification file with the following content:



Note

For more information on service and placement files, see *Book "Deployment Guide", Chapter 8 "Deploying the remaining core services using cephadm", Section 8.2 "Service and placement specification"*.

```
service_type: container
service_id: prometheus-webhook-snmp
placement:
  ADD_PLACEMENT_HERE
image: registry.suse.com/ses/7/prometheus-webhook-snmp:latest
args:
  - "--publish 9099:9099"
envs:
  - ARGS="--debug --snmp-host=ADD_HOST_GATEWAY_HERE"
  - RUN_ARGS="--metrics"
EOF
```

Use this service specification to get the service running using its default settings.

You need to publish the port the Prometheus receiver is listening on by using the command line argument `--publish HOST_PORT:CONTAINER_PORT` when running the service, because the port is not exposed automatically by the container. This can be done by adding the following lines to the specification:

```
args:
  - "--publish 9099:9099"
```

Alternatively, connect the container to the host network by using the command line argument `--network=host`.

```
args:
  - "--network=host"
```

If the SNMP trap receiver is not installed on the same host as the container, then you must also specify the FQDN of the SNMP host. Use the container's network gateway to be able to receive SNMP traps outside the container/host:

```
envs:
  - ARGS="--debug --snmp-host=CONTAINER_GATEWAY"
```

16.7.1 Configuring the prometheus-webhook-snmp service

The container can be configured by environment variables or by using a configuration file.

For the environment variables, use `ARGS` to set global options and `RUN_ARGS` for the `run` command options. You need to adapt the service specification the following way:

```
envs:
  - ARGS="--debug --snmp-host=CONTAINER_GATEWAY"
  - RUN_ARGS="--metrics --port=9101"
```

To use a configuration file, the service specification must be adapted the following way:

```
files:
  etc/prometheus-webhook-snmp.conf:
    - "debug: True"
    - "snmp_host: ADD_HOST_GATEWAY_HERE"
    - "metrics: True"
volume_mounts:
  etc/prometheus-webhook-snmp.conf: /etc/prometheus-webhook-snmp.conf
```

To deploy, run the following command:

```
cephuser@adm > ceph orch apply -i SERVICE_SPEC_FILE
```

See Book "Deployment Guide", Chapter 8 "Deploying the remaining core services using cephadm", Section 8.3 "Deploy Ceph services" for more information.

16.7.2 Configuring the Prometheus Alertmanager for SNMP

Finally, the Prometheus Alertmanager needs to be configured specifically for SNMP traps. If this service has not been deployed already, create a service specification file. You need to replace `IP_OR_FQDN` with the IP address or FQDN of the host where the Prometheus Alertmanager SNMP webhook has been installed. For example:



Note

If you have already deployed this service, then to ensure the Alertmanager is set up correctly for SNMP, re-deploy with the following settings.

```
service_type: alertmanager
placement:
  hosts:
    - HOSTNAME
user_data:
  default_webhook_urls:
    - 'http://IP_OR_FQDN:9099/'
```

Apply the service specification with the following command:

```
cephuser@adm > ceph orch apply -i SERVICE_SPEC_FILE
```

III Storing Data in a Cluster

- 17 Stored data management 154
- 18 Manage storage pools 184
- 19 Erasure coded pools 204
- 20 RADOS Block Device 210

17 Stored data management

The CRUSH algorithm determines how to store and retrieve data by computing data storage locations. CRUSH empowers Ceph clients to communicate with OSDs directly rather than through a centralized server or broker. With an algorithmically determined method of storing and retrieving data, Ceph avoids a single point of failure, a performance bottleneck, and a physical limit to its scalability.

CRUSH requires a map of your cluster, and uses the CRUSH Map to pseudo-randomly store and retrieve data in OSDs with a uniform distribution of data across the cluster.

CRUSH maps contain a list of OSDs, a list of 'buckets' for aggregating the devices into physical locations, and a list of rules that tell CRUSH how it should replicate data in a Ceph cluster's pools. By reflecting the underlying physical organization of the installation, CRUSH can model—and thereby address—potential sources of correlated device failures. Typical sources include physical proximity, a shared power source, and a shared network. By encoding this information into the cluster map, CRUSH placement policies can separate object replicas across different failure domains while still maintaining the desired distribution. For example, to address the possibility of concurrent failures, it may be desirable to ensure that data replicas are on devices using different shelves, racks, power supplies, controllers, and/or physical locations.

After you deploy a Ceph cluster, a default CRUSH Map is generated. It is fine for your Ceph sandbox environment. However, when you deploy a large-scale data cluster, you should give significant consideration to developing a custom CRUSH Map, because it will help you manage your Ceph cluster, improve performance and ensure data safety.

For example, if an OSD goes down, a CRUSH Map can help you locate the physical data center, room, row and rack of the host with the failed OSD in the event you need to use on-site support or replace hardware.

Similarly, CRUSH may help you identify faults more quickly. For example, if all OSDs in a particular rack go down simultaneously, the fault may lie with a network switch or power to the rack or the network switch rather than the OSDs themselves.

A custom CRUSH Map can also help you identify the physical locations where Ceph stores redundant copies of data when the placement group(s) (refer to [Section 17.4, "Placement groups"](#)) associated with a failed host are in a degraded state.

There are three main sections to a CRUSH Map.

- *OSD devices* consist of any object storage device corresponding to a `ceph-osd` daemon.
- *Buckets* consist of a hierarchical aggregation of storage locations (for example rows, racks, hosts, etc.) and their assigned weights.
- *Rule sets* consist of the manner of selecting buckets.

17.1 OSD devices

To map placement groups to OSDs, a CRUSH Map requires a list of OSD devices (the name of the OSD daemon). The list of devices appears first in the CRUSH Map.

```
#devices
device NUM osd.OSD_NAME class CLASS_NAME
```

For example:

```
#devices
device 0 osd.0 class hdd
device 1 osd.1 class ssd
device 2 osd.2 class nvme
device 3 osd.3 class ssd
```

As a general rule, an OSD daemon maps to a single disk.

17.1.1 Device classes

The flexibility of the CRUSH Map in controlling data placement is one of the Ceph's strengths. It is also one of the most difficult parts of the cluster to manage. *Device classes* automate the most common changes to CRUSH Maps that the administrator needed to do manually previously.

17.1.1.1 The CRUSH management problem

Ceph clusters are frequently built with multiple types of storage devices: HDD, SSD, NVMe, or even mixed classes of the above. We call these different types of storage devices *device classes* to avoid confusion between the *type* property of CRUSH buckets (for example, host, rack, row, see [Section 17.2, "Buckets"](#) for more details). Ceph OSDs backed by SSDs are much faster than those backed by spinning disks, making them better suited for certain workloads. Ceph makes it easy to create RADOS pools for different data sets or workloads and to assign different CRUSH rules to control data placement for those pools.

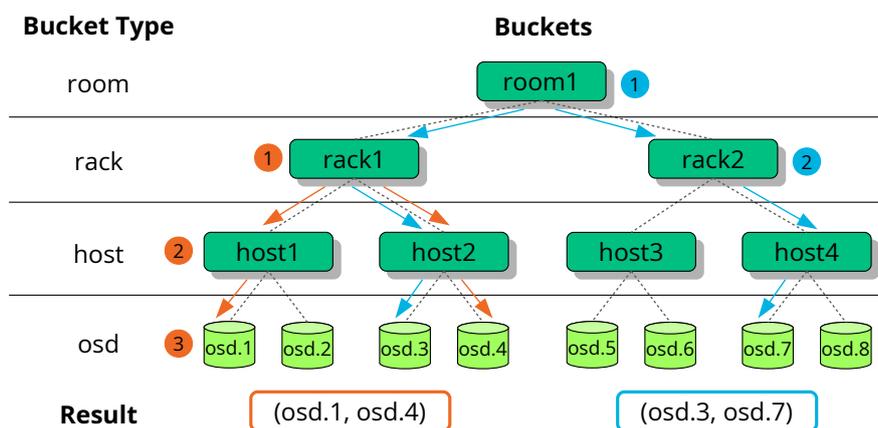


FIGURE 17.1: OSDS WITH MIXED DEVICE CLASSES

However, setting up the CRUSH rules to place data only on a certain class of device is tedious. Rules work in terms of the CRUSH hierarchy, but if the devices are mixed into the same hosts or racks (as in the sample hierarchy above), they will (by default) be mixed together and appear in the same sub-trees of the hierarchy. Manually separating them out into separate trees involved creating multiple versions of each intermediate node for each device class in previous versions of SUSE Enterprise Storage.

17.1.1.2 Device classes

An elegant solution that Ceph offers is to add a property called *device class* to each OSD. By default, OSDs will automatically set their device classes to either 'hdd', 'ssd', or 'nvme' based on the hardware properties exposed by the Linux kernel. These device classes are reported in a new column of the `ceph osd tree` command output:

```
cephuser@adm > ceph osd tree
ID CLASS WEIGHT  TYPE NAME        STATUS REWEIGHT PRI-AFF
-1          83.17899  root default
-4          23.86200  host cpach
 2  hdd  1.81898    osd.2    up  1.00000 1.00000
 3  hdd  1.81898    osd.3    up  1.00000 1.00000
 4  hdd  1.81898    osd.4    up  1.00000 1.00000
 5  hdd  1.81898    osd.5    up  1.00000 1.00000
 6  hdd  1.81898    osd.6    up  1.00000 1.00000
 7  hdd  1.81898    osd.7    up  1.00000 1.00000
 8  hdd  1.81898    osd.8    up  1.00000 1.00000
15  hdd  1.81898    osd.15   up  1.00000 1.00000
10  nvme 0.93100    osd.10   up  1.00000 1.00000
 0  ssd  0.93100    osd.0    up  1.00000 1.00000
```

```
9  ssd  0.93100      osd.9      up  1.00000  1.00000
```

If the automatic device class detection fails, for example because the device driver is not properly exposing information about the device via `/sys/block`, you can adjust device classes from the command line:

```
cephuser@adm > ceph osd crush rm-device-class osd.2 osd.3
done removing class of osd(s): 2,3
cephuser@adm > ceph osd crush set-device-class ssd osd.2 osd.3
set osd(s) 2,3 to class 'ssd'
```

17.1.1.3 Setting CRUSH placement rules

CRUSH rules can restrict placement to a specific device class. For example, you can create a 'fast' **replicated** pool that distributes data only over SSD disks by running the following command:

```
cephuser@adm > ceph osd crush rule create-
replicated RULE_NAME ROOT FAILURE_DOMAIN_TYPE DEVICE_CLASS
```

For example:

```
cephuser@adm > ceph osd crush rule create-replicated fast default host ssd
```

Create a pool named 'fast_pool' and assign it to the 'fast' rule:

```
cephuser@adm > ceph osd pool create fast_pool 128 128 replicated fast
```

The process for creating **erasure code** rules is slightly different. First, you create an erasure code profile that includes a property for your desired device class. Then, use that profile when creating the erasure coded pool:

```
cephuser@adm > ceph osd erasure-code-profile set myprofile \
k=4 m=2 crush-device-class=ssd crush-failure-domain=host
cephuser@adm > ceph osd pool create mypool 64 erasure myprofile
```

In case you need to manually edit the CRUSH Map to customize your rule, the syntax has been extended to allow the device class to be specified. For example, the CRUSH rule generated by the above commands looks as follows:

```
rule ecpool {
  id 2
  type erasure
  min_size 3
  max_size 6
  step set_chooseleaf_tries 5
  step set_choose_tries 100
```

```
step take default class ssd
step chooseleaf indep 0 type host
step emit
}
```

The important difference here is that the 'take' command includes the additional 'class *CLASS_NAME*' suffix.

17.1.1.4 Additional commands

To list device classes used in a CRUSH Map, run:

```
cephuser@adm > ceph osd crush class ls
[
  "hdd",
  "ssd"
]
```

To list existing CRUSH rules, run:

```
cephuser@adm > ceph osd crush rule ls
replicated_rule
fast
```

To view details of the CRUSH rule named 'fast', run:

```
cephuser@adm > ceph osd crush rule dump fast
{
  "rule_id": 1,
  "rule_name": "fast",
  "ruleset": 1,
  "type": 1,
  "min_size": 1,
  "max_size": 10,
  "steps": [
    {
      "op": "take",
      "item": -21,
      "item_name": "default~ssd"
    },
    {
      "op": "chooseleaf_firstn",
      "num": 0,
      "type": "host"
    },
    {
      "op": "emit"
    }
  ]
}
```

```
}  
]  
}
```

To list OSDs that belong to an 'ssd' class, run:

```
cephuser@adm > ceph osd crush class ls-osd ssd  
0  
1
```

17.1.1.5 Migrating from a legacy SSD rule to device classes

In SUSE Enterprise Storage prior to version 5, you needed to manually edit the CRUSH Map and maintain a parallel hierarchy for each specialized device type (such as SSD) in order to write rules that apply to these devices. Since SUSE Enterprise Storage 5, the device class feature has enabled this transparently.

You can transform a legacy rule and hierarchy to the new class-based rules by using the **crush-tool** command. There are several types of transformation possible:

crushtool --reclassify-root *ROOT_NAME* *DEVICE_CLASS*

This command takes everything in the hierarchy beneath *ROOT_NAME* and adjusts any rules that reference that root via

```
take ROOT_NAME
```

to instead

```
take ROOT_NAME class DEVICE_CLASS
```

It renumbers the buckets so that the old IDs are used for the specified class's 'shadow tree'. As a consequence, no data movement occurs.

EXAMPLE 17.1: **crushtool --reclassify-root**

Consider the following existing rule:

```
rule replicated_ruleset {  
  id 0  
  type replicated  
  min_size 1  
  max_size 10  
  step take default  
  step chooseleaf firstn 0 type rack  
  step emit  
}
```

If you reclassify the root 'default' as class 'hdd', the rule will become

```
rule replicated_ruleset {
  id 0
  type replicated
  min_size 1
  max_size 10
  step take default class hdd
  step chooseleaf firstn 0 type rack
  step emit
}
```

crushtool --set-subtree-class *BUCKET_NAME DEVICE_CLASS*

This method marks every device in the subtree rooted at *BUCKET_NAME* with the specified device class.

`--set-subtree-class` is normally used in conjunction with the `--reclassify-root` option to ensure that all devices in that root are labeled with the correct class. However, some of those devices may intentionally have a different class, and therefore you do not want to relabel them. In such cases, exclude the `--set-subtree-class` option. Keep in mind that such remapping will not be perfect, because the previous rule is distributed across devices of multiple classes but the adjusted rules will only map to devices of the specified device class.

crushtool --reclassify-bucket *MATCH_PATTERN DEVICE_CLASS DEFAULT_PATTERN*

This method allows merging a parallel type-specific hierarchy with the normal hierarchy. For example, many users have CRUSH Maps similar to the following one:

EXAMPLE 17.2: **crushtool --reclassify-bucket**

```
host node1 {
  id -2          # do not change unnecessarily
  # weight 109.152
  alg straw
  hash 0 # rjenkins1
  item osd.0 weight 9.096
  item osd.1 weight 9.096
  item osd.2 weight 9.096
  item osd.3 weight 9.096
  item osd.4 weight 9.096
  item osd.5 weight 9.096
  [...]
}

host node1-ssd {
```

```

    id -10          # do not change unnecessarily
    # weight 2.000
    alg straw
    hash 0 # rjenkins1
    item osd.80 weight 2.000
    [...]
}

root default {
    id -1          # do not change unnecessarily
    alg straw
    hash 0 # rjenkins1
    item node1 weight 110.967
    [...]
}

root ssd {
    id -18        # do not change unnecessarily
    # weight 16.000
    alg straw
    hash 0 # rjenkins1
    item node1-ssd weight 2.000
    [...]
}

```

This function reclassifies each bucket that matches a given pattern. The pattern can look like `%suffix` or `prefix%`. In the above example, you would use the pattern `%-ssd`. For each matched bucket, the remaining portion of the name that matches the `'%'` wild card specifies the base bucket. All devices in the matched bucket are labeled with the specified device class and then moved to the base bucket. If the base bucket does not exist (for example, if `'node12-ssd'` exists but `'node12'` does not), then it is created and linked underneath the specified default parent bucket. The old bucket IDs are preserved for the new shadow buckets to prevent data movement. Rules with the `take` steps that reference the old buckets are adjusted.

crushtool --reclassify-bucket *BUCKET_NAME DEVICE_CLASS BASE_BUCKET*

You can use the `--reclassify-bucket` option without a wild card to map a single bucket. For example, in the previous example, we want the `'ssd'` bucket to be mapped to the default bucket.

The final command to convert the map comprised of the above fragments would be as follows:

```
cephuser@adm > ceph osd getcrushmap -o original
```

```
cephuser@adm > crushtool -i original --reclassify \  
--set-subtree-class default hdd \  
--reclassify-root default hdd \  
--reclassify-bucket %-ssd ssd default \  
--reclassify-bucket ssd ssd default \  
-o adjusted
```

In order to verify that the conversion is correct, there is a `--compare` option that tests a large sample of inputs to the CRUSH Map and compares if the same result comes back out. These inputs are controlled by the same options that apply to the `--test`. For the above example, the command would be as follows:

```
cephuser@adm > crushtool -i original --compare adjusted  
rule 0 had 0/10240 mismatched mappings (0)  
rule 1 had 0/10240 mismatched mappings (0)  
maps appear equivalent
```



Tip

If there were differences, you would see what ratio of inputs are remapped in the parentheses.

If you are satisfied with the adjusted CRUSH Map, you can apply it to the cluster:

```
cephuser@adm > ceph osd setcrushmap -i adjusted
```

17.1.1.6 For more information

Find more details on CRUSH Maps in [Section 17.5, “CRUSH Map manipulation”](#).

Find more details on Ceph pools in general in [Chapter 18, Manage storage pools](#).

Find more details about erasure coded pools in [Chapter 19, Erasure coded pools](#).

17.2 Buckets

CRUSH maps contain a list of OSDs, which can be organized into a tree-structured arrangement of buckets for aggregating the devices into physical locations. Individual OSDs comprise the leaves on the tree.

0	osd	A specific device or OSD (<code>osd.1</code> , <code>osd.2</code> , etc.).
1	host	The name of a host containing one or more OSDs.
2	chassis	Identifier for which chassis in the rack contains the <code>host</code> .
3	rack	A computer rack. The default is <code>unknownrack</code> .
4	row	A row in a series of racks.
5	pdu	Abbreviation for "Power Distribution Unit".
6	pod	Abbreviation for "Point of Delivery": in this context, a group of PDUs, or a group of rows of racks.
7	room	A room containing rows of racks.
8	datacenter	A physical data center containing one or more rooms.
9	region	Geographical region of the world (for example, NAM, LAM, EMEA, APAC etc.)
10	root	The root node of the tree of OSD buckets (normally set to <code>default</code>).



Tip

You can modify the existing types and create your own bucket types.

Ceph's deployment tools generate a CRUSH Map that contains a bucket for each host, and a root named 'default', which is useful for the default `rd` pool. The remaining bucket types provide a means for storing information about the physical location of nodes/buckets, which makes cluster administration much easier when OSDs, hosts, or network hardware malfunction and the administrator needs access to physical hardware.

A bucket has a type, a unique name (string), a unique ID expressed as a negative integer, a weight relative to the total capacity/capability of its item(s), the bucket algorithm (`straw2` by default), and the hash (`0` by default, reflecting CRUSH Hash `rjenkins1`). A bucket may have one or more items. The items may consist of other buckets or OSDs. Items may have a weight that reflects the relative weight of the item.

```
[bucket-type] [bucket-name] {
  id [a unique negative numeric ID]
  weight [the relative capacity/capability of the item(s)]
  alg [the bucket type: uniform | list | tree | straw2 | straw ]
  hash [the hash type: 0 by default]
  item [item-name] weight [weight]
}
```

The following example illustrates how you can use buckets to aggregate a pool and physical locations like a data center, a room, a rack and a row.

```
host ceph-osd-server-1 {
  id -17
  alg straw2
  hash 0
  item osd.0 weight 0.546
  item osd.1 weight 0.546
}

row rack-1-row-1 {
  id -16
  alg straw2
  hash 0
  item ceph-osd-server-1 weight 2.00
}

rack rack-3 {
  id -15
  alg straw2
  hash 0
  item rack-3-row-1 weight 2.00
  item rack-3-row-2 weight 2.00
  item rack-3-row-3 weight 2.00
  item rack-3-row-4 weight 2.00
  item rack-3-row-5 weight 2.00
}

rack rack-2 {
  id -14
  alg straw2
  hash 0
  item rack-2-row-1 weight 2.00
  item rack-2-row-2 weight 2.00
  item rack-2-row-3 weight 2.00
  item rack-2-row-4 weight 2.00
  item rack-2-row-5 weight 2.00
}
```

```

rack rack-1 {
    id -13
    alg straw2
    hash 0
    item rack-1-row-1 weight 2.00
    item rack-1-row-2 weight 2.00
    item rack-1-row-3 weight 2.00
    item rack-1-row-4 weight 2.00
    item rack-1-row-5 weight 2.00
}

room server-room-1 {
    id -12
    alg straw2
    hash 0
    item rack-1 weight 10.00
    item rack-2 weight 10.00
    item rack-3 weight 10.00
}

datacenter dc-1 {
    id -11
    alg straw2
    hash 0
    item server-room-1 weight 30.00
    item server-room-2 weight 30.00
}

root data {
    id -10
    alg straw2
    hash 0
    item dc-1 weight 60.00
    item dc-2 weight 60.00
}

```

17.3 Rule sets

CRUSH maps support the notion of 'CRUSH rules', which are the rules that determine data placement for a pool. For large clusters, you will likely create many pools where each pool may have its own CRUSH ruleset and rules. The default CRUSH Map has a rule for the default root. If you want more roots and more rules, you need to create them later or they will be created automatically when new pools are created.



Note

In most cases, you will not need to modify the default rules. When you create a new pool, its default ruleset is 0.

A rule takes the following form:

```
rule rulename {  
  
    ruleset ruleset  
    type type  
    min_size min-size  
    max_size max-size  
    step step  
  
}
```

ruleset

An integer. Classifies a rule as belonging to a set of rules. Activated by setting the ruleset in a pool. This option is required. Default is 0.

type

A string. Describes a rule for either a 'replicated' or 'erasure' coded pool. This option is required. Default is replicated.

min_size

An integer. If a pool group makes fewer replicas than this number, CRUSH will NOT select this rule. This option is required. Default is 2.

max_size

An integer. If a pool group makes more replicas than this number, CRUSH will NOT select this rule. This option is required. Default is 10.

step take bucket

Takes a bucket specified by a name, and begins iterating down the tree. This option is required. For an explanation about iterating through the tree, see [Section 17.3.1, "Iterating the node tree"](#).

step target mode num type bucket-type

target can either be choose or chooseleaf. When set to choose, a number of buckets is selected. chooseleaf directly selects the OSDs (leaf nodes) from the sub-tree of each bucket in the set of buckets.

mode can either be firstn or indep. See [Section 17.3.2, "firstn and indep"](#).

Selects the number of buckets of the given type. Where N is the number of options available, if num > 0 && < N, choose that many buckets; if num < 0, it means N - num; and, if num == 0, choose N buckets (all available). Follows step take or step choose.

step emit

Outputs the current value and empties the stack. Typically used at the end of a rule, but may also be used to form different trees in the same rule. Follows step choose.

17.3.1 Iterating the node tree

The structure defined with the buckets can be viewed as a node tree. Buckets are nodes and OSDs are leaves in this tree.

Rules in the CRUSH Map define how OSDs are selected from this tree. A rule starts with a node and then iterates down the tree to return a set of OSDs. It is not possible to define which branch needs to be selected. Instead the CRUSH algorithm assures that the set of OSDs fulfills the replication requirements and evenly distributes the data.

With step take bucket the iteration through the node tree begins at the given bucket (not bucket type). If OSDs from all branches in the tree are to be returned, the bucket must be the root bucket. Otherwise the following steps are only iterating through a sub-tree.

After step take one or more step choose entries follow in the rule definition. Each step choose chooses a defined number of nodes (or branches) from the previously selected upper node.

In the end the selected OSDs are returned with step emit.

step chooseleaf is a convenience function that directly selects OSDs from branches of the given bucket.

[Figure 17.2, "Example tree"](#) provides an example of how step is used to iterate through a tree. The orange arrows and numbers correspond to example1a and example1b, while blue corresponds to example2 in the following rule definitions.

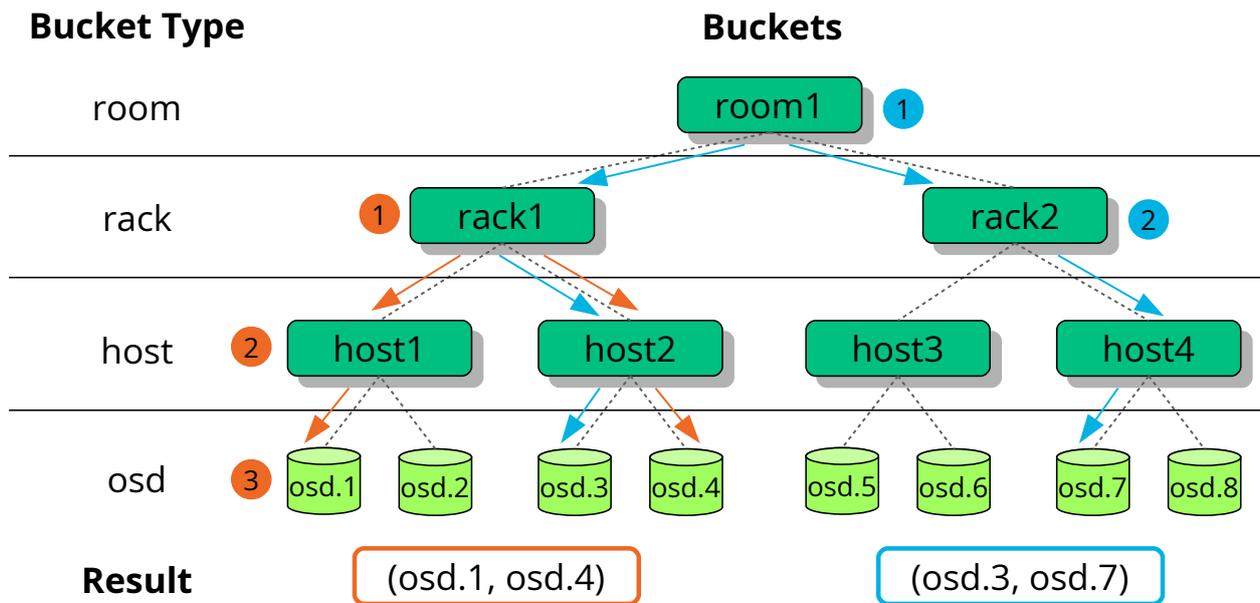


FIGURE 17.2: EXAMPLE TREE

```
# orange arrows
rule example1a {
  ruleset 0
  type replicated
  min_size 2
  max_size 10
  # orange (1)
  step take rack1
  # orange (2)
  step choose firstn 0 host
  # orange (3)
  step choose firstn 1 osd
  step emit
}

rule example1b {
  ruleset 0
  type replicated
  min_size 2
  max_size 10
  # orange (1)
  step take rack1
  # orange (2) + (3)
  step chooseleaf firstn 0 host
  step emit
}
```

```

# blue arrows
rule example2 {
    ruleset 0
    type replicated
    min_size 2
    max_size 10
    # blue (1)
    step take room1
    # blue (2)
    step chooseleaf firstn 0 rack
    step emit
}

```

17.3.2 `firstn` and `indep`

A CRUSH rule defines replacements for failed nodes or OSDs (see [Section 17.3, “Rule sets”](#)). The keyword `step` requires either `firstn` or `indep` as parameter. [Figure 17.3, “Node replacement methods”](#) provides an example.

`firstn` adds replacement nodes to the end of the list of active nodes. In case of a failed node, the following healthy nodes are shifted to the left to fill the gap of the failed node. This is the default and desired method for *replicated pools*, because a secondary node already has all data and therefore can take over the duties of the primary node immediately.

`indep` selects fixed replacement nodes for each active node. The replacement of a failed node does not change the order of the remaining nodes. This is desired for *erasure coded pools*. In erasure coded pools the data stored on a node depends on its position in the node selection. When the order of nodes changes, all data on affected nodes needs to be relocated.

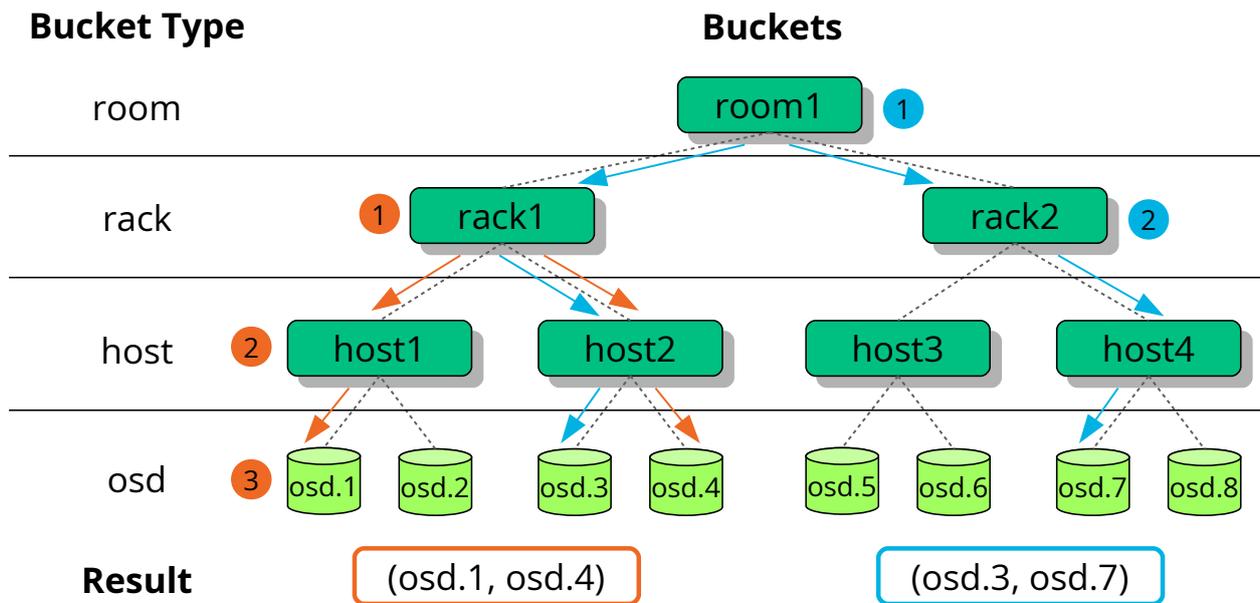


FIGURE 17.3: NODE REPLACEMENT METHODS

17.4 Placement groups

Ceph maps objects to placement groups (PGs). Placement groups are shards or fragments of a logical object pool that place objects as a group into OSDs. Placement groups reduce the amount of per-object metadata when Ceph stores the data in OSDs. A larger number of placement groups—for example, 100 per OSD—leads to better balancing.

17.4.1 Using placement groups

A placement group (PG) aggregates objects within a pool. The main reason is that tracking object placement and metadata on a per-object basis is computationally expensive. For example, a system with millions of objects cannot track placement of each of its objects directly.

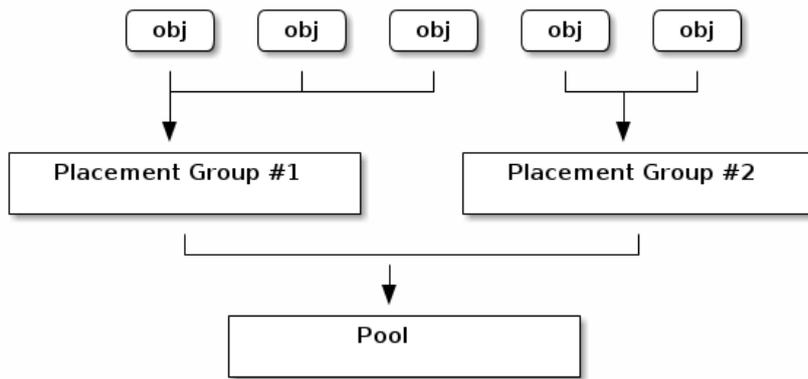


FIGURE 17.4: PLACEMENT GROUPS IN A POOL

The Ceph client will calculate to which placement group an object will belong to. It does this by hashing the object ID and applying an operation based on the number of PGs in the defined pool and the ID of the pool.

The object's contents within a placement group are stored in a set of OSDs. For example, in a replicated pool of size two, each placement group will store objects on two OSDs:

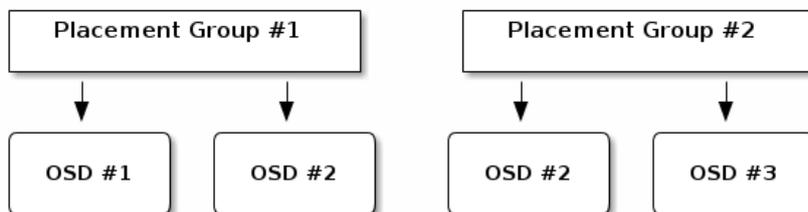


FIGURE 17.5: PLACEMENT GROUPS AND OSDS

If OSD #2 fails, another OSD will be assigned to placement group #1 and will be filled with copies of all objects in OSD #1. If the pool size is changed from two to three, an additional OSD will be assigned to the placement group and will receive copies of all objects in the placement group.

Placement groups do not own the OSD, they share it with other placement groups from the same pool or even other pools. If OSD #2 fails, the placement group #2 will also need to restore copies of objects, using OSD #3.

When the number of placement groups increases, the new placement groups will be assigned OSDs. The result of the CRUSH function will also change and some objects from the former placement groups will be copied over to the new placement groups and removed from the old ones.

17.4.2 Determining the value of *PG_NUM*



Note

Since Ceph Nautilus (v14.x), you can use the Ceph Manager `pg_autoscaler` module to auto-scale the PGs as needed. If you want to enable this feature, refer to *Book “Deploying and Administering SUSE Enterprise Storage with Rook”, Chapter 7 “Configuration”, Section 7.1.1.1 “Default PG and PGP counts”*.

When creating a new pool, you can still choose the value of *PG_NUM* manually:

```
# ceph osd pool create POOL_NAME PG_NUM
```

PG_NUM cannot be calculated automatically. Following are a few commonly used values, depending on the number of OSDs in the cluster:

Less than 5 OSDs:

Set *PG_NUM* to 128.

Between 5 and 10 OSDs:

Set *PG_NUM* to 512.

Between 10 and 50 OSDs:

Set *PG_NUM* to 1024.

As the number of OSDs increases, choosing the right value for *PG_NUM* becomes more important. *PG_NUM* strongly affects the behavior of the cluster as well as the durability of the data in case of OSD failure.

17.4.2.1 Calculating placement groups for more than 50 OSDs

If you have less than 50 OSDs, use the preselection described in [Section 17.4.2, “Determining the value of PG_NUM”](#). If you have more than 50 OSDs, we recommend approximately 50-100 placement groups per OSD to balance out resource usage, data durability, and distribution. For a single pool of objects, you can use the following formula to get a baseline:

```
total PGs = (OSDs * 100) / POOL_SIZE
```

Where `POOL_SIZE` is either the number of replicas for replicated pools, or the 'k'+ 'm' sum for erasure coded pools as returned by the `ceph osd erasure-code-profile get` command. You should round the result up to the nearest power of 2. Rounding up is recommended for the CRUSH algorithm to evenly balance the number of objects among placement groups.

As an example, for a cluster with 200 OSDs and a pool size of 3 replicas, you would estimate the number of PGs as follows:

```
(200 * 100) / 3 = 6667
```

The nearest power of 2 is **8192**.

When using multiple data pools for storing objects, you need to ensure that you balance the number of placement groups per pool with the number of placement groups per OSD. You need to reach a reasonable total number of placement groups that provides reasonably low variance per OSD without taxing system resources or making the peering process too slow.

For example, a cluster of 10 pools, each with 512 placement groups on 10 OSDs, is a total of 5,120 placement groups spread over 10 OSDs, that is 512 placement groups per OSD. Such a setup does not use too many resources. However, if 1000 pools were created with 512 placement groups each, the OSDs would handle approximately 50,000 placement groups each and it would require significantly more resources and time for peering.

17.4.3 Setting the number of placement groups



Note

Since Ceph Nautilus (v14.x), you can use the Ceph Manager `pg_autoscaler` module to auto-scale the PGs as needed. If you want to enable this feature, refer to *Book “Deploying and Administering SUSE Enterprise Storage with Rook”, Chapter 7 “Configuration”, Section 7.1.1.1 “Default PG and PGP counts”*.

If you still need to specify the number of placement groups in a pool manually, you need to specify them at the time of pool creation (see [Section 18.1, “Creating a pool”](#)). Once you have set placement groups for a pool, you may increase the number of placement groups by running the following command:

```
# ceph osd pool set POOL_NAME pg_num PG_NUM
```

After you increase the number of placement groups, you also need to increase the number of placement groups for placement (`PGP_NUM`) before your cluster will rebalance. `PGP_NUM` will be the number of placement groups that will be considered for placement by the CRUSH algorithm. Increasing `PG_NUM` splits the placement groups but data will not be migrated to the newer placement groups until `PGP_NUM` is increased. `PGP_NUM` should be equal to `PG_NUM`. To increase the number of placement groups for placement, run the following:

```
# ceph osd pool set POOL_NAME pgp_num PGP_NUM
```

17.4.4 Finding the number of placement groups

To find out the number of placement groups in a pool, run the following `get` command:

```
# ceph osd pool get POOL_NAME pg_num
```

17.4.5 Finding a cluster's PG statistics

To find out the statistics for the placement groups in your cluster, run the following command:

```
# ceph pg dump [--format FORMAT]
```

Valid formats are 'plain' (default) and 'json'.

17.4.6 Finding statistics for stuck PGs

To find out the statistics for all placement groups stuck in a specified state, run the following:

```
# ceph pg dump_stuck STATE \  
[--format FORMAT] [--threshold THRESHOLD]
```

STATE is one of 'inactive' (PGs cannot process reads or writes because they are waiting for an OSD with the most up-to-date data to come up), 'unclean' (PGs contain objects that are not replicated the desired number of times), 'stale' (PGs are in an unknown state—the OSDs that host them have not reported to the monitor cluster in a time interval specified by the `mon_osd_report_timeout` option), 'undersized', or 'degraded'.

Valid formats are 'plain' (default) and 'json'.

The threshold defines the minimum number of seconds the placement group is stuck before including it in the returned statistics (300 seconds by default).

17.4.7 Searching a placement group map

To search for the placement group map for a particular placement group, run the following:

```
# ceph pg map PG_ID
```

Ceph will return the placement group map, the placement group, and the OSD status:

```
# ceph pg map 1.6c
osdmap e13 pg 1.6c (1.6c) -> up [1,0] acting [1,0]
```

17.4.8 Retrieving a placement groups statistics

To retrieve statistics for a particular placement group, run the following:

```
# ceph pg PG_ID query
```

17.4.9 Scrubbing a placement group

To scrub ([Section 17.6, “Scrubbing placement groups”](#)) a placement group, run the following:

```
# ceph pg scrub PG_ID
```

Ceph checks the primary and replica nodes, generates a catalog of all objects in the placement group, and compares them to ensure that no objects are missing or mismatched and their contents are consistent. Assuming the replicas all match, a final semantic sweep ensures that all of the snapshot-related object metadata is consistent. Errors are reported via logs.

17.4.10 Prioritizing backfill and recovery of placement groups

You may run into a situation where several placement groups require recovery and/or backfill, while some groups hold data more important than others. For example, those PGs may hold data for images used by running machines and other PGs may be used by inactive machines or less relevant data. In that case, you may want to prioritize recovery of those groups so that performance and availability of data stored on those groups is restored earlier. To mark particular placement groups as prioritized during backfill or recovery, run the following:

```
# ceph pg force-recovery PG_ID1 [PG_ID2 ... ]
# ceph pg force-backfill PG_ID1 [PG_ID2 ... ]
```

This will cause Ceph to perform recovery or backfill on specified placement groups first, before other placement groups. This does not interrupt currently ongoing backfills or recovery, but causes specified PGs to be processed as soon as possible. If you change your mind or prioritize wrong groups, cancel the prioritization:

```
# ceph pg cancel-force-recovery PG_ID1 [PG_ID2 ... ]
# ceph pg cancel-force-backfill PG_ID1 [PG_ID2 ... ]
```

The **cancel-*** commands remove the 'force' flag from the PGs so that they are processed in default order. Again, this does not affect placement groups currently being processed, only those that are still queued. The 'force' flag is cleared automatically after recovery or backfill of the group is done.

17.4.11 Reverting lost objects

If the cluster has lost one or more objects and you have decided to abandon the search for the lost data, you need to mark the unfound objects as 'lost'.

If the objects are still lost after having queried all possible locations, you may need to give up on the lost objects. This is possible given unusual combinations of failures that allow the cluster to learn about writes that were performed before the writes themselves are recovered.

Currently the only supported option is 'revert', which will either roll back to a previous version of the object, or forget about it entirely in case of a new object. To mark the 'unfound' objects as 'lost', run the following:

```
cephuser@adm > ceph pg PG_ID mark_unfound_lost revert|delete
```

17.4.12 Enabling the PG auto-scaler

Placement groups (PGs) are an internal implementation detail of how Ceph distributes data. By enabling `pg-autoscaling`, you can allow the cluster to either make or automatically tune PGs based on how the cluster is used.

Each pool in the system has a `pg_autoscale_mode` property that can be set to `off`, `on`, or `warn`:

The autoscaler is configured on a per-pool basis, and can run in three modes:

`off`

Disable autoscaling for this pool. It is up to the administrator to choose an appropriate PG number for each pool.

`on`

Enable automated adjustments of the PG count for the given pool.

`warn`

Raise health alerts when the PG count should be adjusted.

To set the autoscaling mode for existing pools:

```
cephuser@adm > ceph osd pool set POOL_NAME pg_autoscale_mode mode
```

You can also configure the default `pg_autoscale_mode` that is applied to any pools that are created in the future with:

```
cephuser@adm > ceph config set global osd_pool_default_pg_autoscale_mode MODE
```

You can view each pool, its relative utilization, and any suggested changes to the PG count with this command:

```
cephuser@adm > ceph osd pool autoscale-status
```

17.5 CRUSH Map manipulation

This section introduces ways to basic CRUSH Map manipulation, such as editing a CRUSH Map, changing CRUSH Map parameters, and adding/moving/removing an OSD.

17.5.1 Editing a CRUSH Map

To edit an existing CRUSH map, do the following:

1. Get a CRUSH Map. To get the CRUSH Map for your cluster, execute the following:

```
cephuser@adm > ceph osd getcrushmap -o compiled-crushmap-filename
```

Ceph will output (-o) a compiled CRUSH Map to the file name you specified. Since the CRUSH Map is in a compiled form, you must decompile it first before you can edit it.

2. Decompile a CRUSH Map. To decompile a CRUSH Map, execute the following:

```
cephuser@adm > crushtool -d compiled-crushmap-filename \  
-o decompiled-crushmap-filename
```

Ceph will decompile (-d) the compiled CRUSH Map and output (-o) it to the file name you specified.

3. Edit at least one of Devices, Buckets and Rules parameters.

4. Compile a CRUSH Map. To compile a CRUSH Map, execute the following:

```
cephuser@adm > crushtool -c decompiled-crush-map-filename \  
-o compiled-crush-map-filename
```

Ceph will store a compiled CRUSH Map to the file name you specified.

5. Set a CRUSH Map. To set the CRUSH Map for your cluster, execute the following:

```
cephuser@adm > ceph osd setcrushmap -i compiled-crushmap-filename
```

Ceph will input the compiled CRUSH Map of the file name you specified as the CRUSH Map for the cluster.



Tip: Use versioning system

Use a versioning system—such as git or svn—for the exported and modified CRUSH Map files. It makes a possible rollback simple.



Tip: Test the new CRUSH Map

Test the new adjusted CRUSH Map using the `crushtool --test` command, and compare to the state before applying the new CRUSH Map. You may find the following command switches useful: `--show-statistics`, `--show-mappings`, `--show-bad-mappings`, `--show-utilization`, `--show-utilization-all`, `--show-choose-tries`

17.5.2 Adding or moving an OSD

To add or move an OSD in the CRUSH Map of a running cluster, execute the following:

```
cephuser@adm > ceph osd crush set id_or_name weight root=pool-name
bucket-type=bucket-name ...
```

id

An integer. The numeric ID of the OSD. This option is required.

name

A string. The full name of the OSD. This option is required.

weight

A double. The CRUSH weight for the OSD. This option is required.

root

A key/value pair. By default, the CRUSH hierarchy contains the pool default as its root. This option is required.

bucket-type

Key/value pairs. You may specify the OSD's location in the CRUSH hierarchy.

The following example adds `osd.0` to the hierarchy, or moves the OSD from a previous location.

```
cephuser@adm > ceph osd crush set osd.0 1.0 root=data datacenter=dc1 room=room1 \
row=foo rack=bar host=foo-bar-1
```

17.5.3 Difference between `ceph osd reweight` and `ceph osd crush reweight`

There are two similar commands that change the 'weight' of a Ceph OSD. The context of their usage is different and may cause confusion.

17.5.3.1 **ceph osd reweight**

Usage:

```
cephuser@adm > ceph osd reweight OSD_NAME NEW_WEIGHT
```

ceph osd reweight sets an override weight on the Ceph OSD. This value is in the range of 0 to 1, and forces CRUSH to reposition the data that would otherwise live on this drive. It does **not** change the weights assigned to the buckets above the OSD, and is a corrective measure in case the normal CRUSH distribution is not working out quite right. For example, if one of your OSDs is at 90% and the others are at 40%, you could reduce this weight to try and compensate for it.



Note: OSD weight is temporary

Note that **ceph osd reweight** is not a persistent setting. When an OSD gets marked out, its weight will be set to 0 and when it gets marked in again, the weight will be changed to 1.

17.5.3.2 **ceph osd crush reweight**

Usage:

```
cephuser@adm > ceph osd crush reweight OSD_NAME NEW_WEIGHT
```

ceph osd crush reweight sets the **CRUSH** weight of the OSD. This weight is an arbitrary value—generally the size of the disk in TB—and controls how much data the system tries to allocate to the OSD.

17.5.4 Removing an OSD

To remove an OSD from the CRUSH Map of a running cluster, execute the following:

```
cephuser@adm > ceph osd crush remove OSD_NAME
```

17.5.5 Adding a bucket

To add a bucket to the CRUSH Map of a running cluster, execute the **ceph osd crush add-bucket** command:

```
cephuser@adm > ceph osd crush add-bucket BUCKET_NAME BUCKET_TYPE
```

17.5.6 Moving a bucket

To move a bucket to a different location or position in the CRUSH Map hierarchy, execute the following:

```
cephuser@adm > ceph osd crush move BUCKET_NAME BUCKET_TYPE=BUCKET_NAME [...]
```

For example:

```
cephuser@adm > ceph osd crush move bucket1 datacenter=dc1 room=room1 row=foo rack=bar  
host=foo-bar-1
```

17.5.7 Removing a bucket

To remove a bucket from the CRUSH Map hierarchy, execute the following:

```
cephuser@adm > ceph osd crush remove BUCKET_NAME
```



Note: Empty bucket only

A bucket must be empty before removing it from the CRUSH hierarchy.

17.6 Scrubbing placement groups

In addition to making multiple copies of objects, Ceph ensures data integrity by *scrubbing* placement groups (find more information about placement groups in *Book "Deployment Guide", Chapter 1 "SES and Ceph", Section 1.3.2 "Placement groups"*). Ceph scrubbing is analogous to running **fsck** on the object storage layer. For each placement group, Ceph generates a catalog of all objects and compares each primary object and its replicas to ensure that no objects are missing or mismatched. Daily light scrubbing checks the object size and attributes, while weekly deep scrubbing reads the data and uses checksums to ensure data integrity.

Scrubbing is important for maintaining data integrity, but it can reduce performance. You can adjust the following settings to increase or decrease scrubbing operations:

```
osd max scrubs
```

The maximum number of simultaneous scrub operations for a Ceph OSD. Default is 1.

osd scrub begin hour, osd scrub end hour

The hours of day (0 to 24) that define a time window during which the scrubbing can happen. By default, begins at 0 and ends at 24.



Important

If the placement group's scrub interval exceeds the osd scrub max interval setting, the scrub will happen no matter what time window you define for scrubbing.

osd scrub during recovery

Allows scrubs during recovery. Setting this to 'false' will disable scheduling new scrubs while there is an active recovery. Already running scrubs will continue. This option is useful for reducing load on busy clusters. Default is 'true'.

osd scrub thread timeout

The maximum time in seconds before a scrub thread times out. Default is 60.

osd scrub finalize thread timeout

The maximum time in seconds before a scrub finalize thread times out. Default is 60*10.

osd scrub load threshold

The normalized maximum load. Ceph will not scrub when the system load (as defined by the ratio of getloadavg() / number of online cpus) is higher than this number. Default is 0.5.

osd scrub min interval

The minimal interval in seconds for scrubbing Ceph OSD when the Ceph cluster load is low. Default is 60*60*24 (once a day).

osd scrub max interval

The maximum interval in seconds for scrubbing Ceph OSD, irrespective of cluster load. Default is 7*60*60*24 (once a week).

osd scrub chunk min

The minimum number of object store chunks to scrub during a single operation. Ceph blocks writes to a single chunk during a scrub. Default is 5.

osd scrub chunk max

The maximum number of object store chunks to scrub during a single operation. Default is 25.

osd scrub sleep

Time to sleep before scrubbing the next group of chunks. Increasing this value slows down the whole scrub operation, while client operations are less impacted. Default is 0.

osd deep scrub interval

The interval for 'deep' scrubbing (fully reading all data). The osd scrub load threshold option does not affect this setting. Default is 60*60*24*7 (once a week).

osd scrub interval randomize ratio

Add a random delay to the osd scrub min interval value when scheduling the next scrub job for a placement group. The delay is a random value smaller than the result of osd scrub min interval * osd scrub interval randomized ratio. Therefore, the default setting practically randomly spreads the scrubs out in the allowed time window of $[1, 1.5] * \text{osd scrub min interval}$. Default is 0.5.

osd deep scrub stride

Read size when doing a deep scrub. Default is 524288 (512 kB).

18 Manage storage pools

Ceph stores data within pools. Pools are logical groups for storing objects. When you first deploy a cluster without creating a pool, Ceph uses the default pools for storing data. The following important highlights relate to Ceph pools:

- *Resilience*: Ceph pools provide resilience by replicating or encoding the data contained within them. Each pool can be set to either replicated or erasure coding. For replicated pools, you further set the number of replicas, or copies, which each data object within the pool will have. The number of copies (OSDs, CRUSH buckets/leaves) that can be lost is one less than the number of replicas. With erasure coding, you set the values of k and m, where k is the number of data chunks and m is the number of coding chunks. For erasure coded pools, it is the number of coding chunks that determines how many OSDs (CRUSH buckets/leaves) can be lost without losing data.
- *Placement Groups*: You can set the number of placement groups for the pool. A typical configuration uses approximately 100 placement groups per OSD to provide optimal balancing without using up too many computing resources. When setting up multiple pools, be careful to ensure you set a reasonable number of placement groups for both the pool and the cluster as a whole.
- *CRUSH Rules*: When you store data in a pool, objects and its replicas (or chunks in case of erasure coded pools) are placed according to the CRUSH ruleset mapped to the pool. You can create a custom CRUSH rule for your pool.
- *Snapshots*: When you create snapshots with `ceph osd pool mksnap`, you effectively take a snapshot of a particular pool.

To organize data into pools, you can list, create, and remove pools. You can also view the usage statistics for each pool.

18.1 Creating a pool

A pool can be created as either replicated to recover from lost OSDs by keeping multiple copies of the objects or erasure to have generalized RAID 5 or 6 capability. Replicated pools require more raw storage, while erasure coded pools require less raw storage. The default setting is replicated. For more information on erasure coded pools, see [Chapter 19, Erasure coded pools](#).

To create a replicated pool, execute:

```
cephuser@adm > ceph osd pool create POOL_NAME
```



Note

The autoscaler will take care of the remaining optional arguments. For more information, see [Section 17.4.12, “Enabling the PG auto-scaler”](#).

To create an erasure coded pool, execute:

```
cephuser@adm > ceph osd pool create POOL_NAME erasure CRUSH_RULESET_NAME \
EXPECTED_NUM_OBJECTS
```

The `ceph osd pool create` command can fail if you exceed the limit of placement groups per OSD. The limit is set with the option `mon_max_pg_per_osd`.

POOL_NAME

The name of the pool. It must be unique. This option is required.

POOL_TYPE

The pool type which may either be `replicated` to recover from lost OSDs by keeping multiple copies of the objects or `erasure` to get a kind of generalized RAID 5 capability. The replicated pools require more raw storage but implement all Ceph operations. The erasure pools require less raw storage but only implement a subset of the available operations. The default `POOL_TYPE` is `replicated`.

CRUSH_RULESET_NAME

The name of the CRUSH ruleset for this pool. If the specified ruleset does not exist, the creation of replicated pools will fail with `-ENOENT`. For replicated pools it is the ruleset specified by the `osd pool default CRUSH replicated ruleset` configuration variable. This ruleset must exist. For erasure pools it is 'erasure-code' if the default erasure code profile is used or `POOL_NAME` otherwise. This ruleset will be created implicitly if it does not exist already.

erasure_code_profile=profile

For erasure coded pools only. Use the erasure code profile. It must be an existing profile as defined by `osd erasure-code-profile set`.



Note

If for any reason the autoscaler has been disabled (`pg_autoscale_mode` set to off) on a pool, you can calculate and set the PG numbers manually. See [Section 17.4, “Placement groups”](#) for details on calculating an appropriate number of placement groups for your pool.

EXPECTED_NUM_OBJECTS

The expected number of objects for this pool. By setting this value (together with a negative `filestore merge threshold`), the PG folder splitting happens at the pool creation time. This avoids the latency impact with a runtime folder splitting.

18.2 Listing pools

To list your cluster’s pools, execute:

```
cephuser@adm > ceph osd pool ls
```

18.3 Renaming a pool

To rename a pool, execute:

```
cephuser@adm > ceph osd pool rename CURRENT_POOL_NAME NEW_POOL_NAME
```

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.

18.4 Deleting a pool



Warning: Pool deletion is not reversible

Pools may contain important data. Deleting a pool causes all data in the pool to disappear, and there is no way to recover it.

Because inadvertent pool deletion is a real danger, Ceph implements two mechanisms that prevent pools from being deleted. Both mechanisms must be disabled before a pool can be deleted.

The first mechanism is the `NODELETE` flag. Each pool has this flag, and its default value is 'false'. To find out the value of this flag on a pool, run the following command:

```
cephuser@adm > ceph osd pool get pool_name nodelete
```

If it outputs `nodelete: true`, it is not possible to delete the pool until you change the flag using the following command:

```
cephuser@adm > ceph osd pool set pool_name nodelete false
```

The second mechanism is the cluster-wide configuration parameter `mon allow pool delete`, which defaults to 'false'. This means that, by default, it is not possible to delete a pool. The error message displayed is:

```
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
```

To delete the pool in spite of this safety setting, you can temporarily set `mon allow pool delete` to 'true', delete the pool, and then return the parameter to 'false':

```
cephuser@adm > ceph tell mon.* injectargs --mon-allow-pool-delete=true
cephuser@adm > ceph osd pool delete pool_name pool_name --yes-i-really-really-mean-it
cephuser@adm > ceph tell mon.* injectargs --mon-allow-pool-delete=false
```

The `injectargs` command displays the following message:

```
injectargs:mon_allow_pool_delete = 'true' (not observed, change may require restart)
```

This is merely confirming that the command was executed successfully. It is not an error.

If you created your own rulesets and rules for a pool you created, you should consider removing them when you no longer need your pool.

18.5 Other operations

18.5.1 Associating pools with an application

Before using pools, you need to associate them with an application. Pools that will be used with CephFS, or pools that are automatically created by Object Gateway are automatically associated.

For other cases, you can manually associate a free-form application name with a pool:

```
cephuser@adm > ceph osd pool application enable POOL_NAME APPLICATION_NAME
```



Tip: Default application names

CephFS uses the application name `cephfs`, RADOS Block Device uses `rbd`, and Object Gateway uses `rgw`.

A pool can be associated with multiple applications, and each application can have its own metadata. To list the application (or applications) associated with a pool, issue the following command:

```
cephuser@adm > ceph osd pool application get pool_name
```

18.5.2 Setting pool quotas

You can set pool quotas for the maximum number of bytes and/or the maximum number of objects per pool.

```
cephuser@adm > ceph osd pool set-quota POOL_NAME MAX_OBJECTS OBJ_COUNT MAX_BYTES BYTES
```

For example:

```
cephuser@adm > ceph osd pool set-quota data max_objects 10000
```

To remove a quota, set its value to 0.

18.5.3 Showing pool statistics

To show a pool's usage statistics, execute:

```
cephuser@adm > rados df
```

POOL_NAME	DEGRADED	RD_OPS	RD	WR_OPS	WR	USED	OBJECTS	CLONES	COPIES	MISSING_ON_PRIMARY	UNFOUND
.rgw.root				768 KiB		4	0	12		0	0
0	44	44 KiB	4	4 KiB		0 B		0 B			
cephfs_data				960 KiB		5	0	15		0	0
0	5502	2.1 MiB	14	11 KiB		0 B		0 B			
cephfs_metadata				1.5 MiB		22	0	66		0	0
0	26	78 KiB	176	147 KiB		0 B		0 B			
default.rgw.buckets.index				0 B		1	0	3		0	0
0	4	4 KiB	1	0 B		0 B		0 B			

default.rgw.control	0 B	8	0	24	0	0
0	0	0 B	0	0 B	0 B	0 B
default.rgw.log	0 B	207	0	621	0	0
0	5372132	5.1 GiB	3579618	0 B	0 B	0 B
default.rgw.meta	961 KiB	6	0	18	0	0
0	155	140 KiB	14	7 KiB	0 B	0 B
example_rbd_pool	2.1 MiB	18	0	54	0	0
0	3350841	2.7 GiB	118	98 KiB	0 B	0 B
iscsi-images	769 KiB	8	0	24	0	0
0	1559261	1.3 GiB	61	42 KiB	0 B	0 B
mirrored-pool	1.1 MiB	10	0	30	0	0
0	475724	395 MiB	54	48 KiB	0 B	0 B
pool2	0 B	0	0	0	0	0
0	0	0 B	0	0 B	0 B	0 B
pool3	333 MiB	37	0	111	0	0
0	3169308	2.5 GiB	14847	118 MiB	0 B	0 B
pool4	1.1 MiB	13	0	39	0	0
0	1379568	1.1 GiB	16840	16 MiB	0 B	0 B

A description of individual columns follow:

USED

Number of bytes used by the pool.

OBJECTS

Number of objects stored in the pool.

CLONES

Number of clones stored in the pool. When a snapshot is created and one writes to an object, instead of modifying the original object its clone is created so the original snapshotted object content is not modified.

COPIES

Number of object replicas. For example, if a replicated pool with the replication factor 3 has 'x' objects, it will normally have 3 * x copies.

MISSING_ON_PRIMARY

Number of objects in the degraded state (not all copies exist) while the copy is missing on the primary OSD.

UNFOUND

Number of unfound objects.

DEGRADED

Number of degraded objects.

RD_OPS

Total number of read operations requested for this pool.

RD

Total number of bytes read from this pool.

WR_OPS

Total number of write operations requested for this pool.

WR

Total number of bytes written to the pool. Note that it is not the same as the pool's usage because you can write to the same object many times. The result is that the pool's usage will remain the same but the number of bytes written to the pool will grow.

USED COMPR

Number of bytes allocated for compressed data.

UNDER COMPR

Number of bytes that the compressed data occupy when it is not compressed.

18.5.4 Getting pool values

To get a value from a pool, run the following **get** command:

```
cephuser@adm > ceph osd pool get POOL_NAME KEY
```

You can get values for keys listed in [Section 18.5.5, "Setting pool values"](#) plus the following keys:

PG_NUM

The number of placement groups for the pool.

PGP_NUM

The effective number of placement groups to use when calculating data placement. Valid range is equal to or less than PG_NUM.



Tip: All of a pool's values

To list all values related to a specific pool, run:

```
cephuser@adm > ceph osd pool get POOL_NAME all
```

18.5.5 Setting pool values

To set a value to a pool, execute:

```
cephuser@adm > ceph osd pool set POOL_NAME KEY VALUE
```

The following is a list of pool values sorted by a pool type:

COMMON POOL VALUES

crash_replay_interval

The number of seconds to allow clients to replay acknowledged, but uncommitted requests.

pg_num

The number of placement groups for the pool. If you add new OSDs to the cluster, verify the value for placement groups on all pools targeted for the new OSDs.

pgp_num

The effective number of placement groups to use when calculating data placement.

crush_ruleset

The ruleset to use for mapping object placement in the cluster.

hashpspool

Set (1) or unset (0) the HASHPSPOOL flag on a given pool. Enabling this flag changes the algorithm to better distribute PGs to OSDs. After enabling this flag on a pool whose HASHPSPOOL flag was set to the default 0, the cluster starts backfilling to have a correct placement of all PGs again. Be aware that this can create quite substantial I/O load on a cluster, therefore do not enable the flag from 0 to 1 on highly loaded production clusters.

nodelete

Prevents the pool from being removed.

nopgchange

Prevents the pool's `pg_num` and `pgp_num` from being changed.

noscrub,nodeep-scrub

Disables (deep) scrubbing of the data for the specific pool to resolve temporary high I/O load.

write_fadvise_dontneed

Set or unset the `WRITE_FADVISE_DONTNEED` flag on a given pool's read/write requests to bypass putting data into cache. Default is `false`. Applies to both replicated and EC pools.

scrub_min_interval

The minimum interval in seconds for pool scrubbing when the cluster load is low. The default `0` means that the `osd_scrub_min_interval` value from the Ceph configuration file is used.

scrub_max_interval

The maximum interval in seconds for pool scrubbing, regardless of the cluster load. The default `0` means that the `osd_scrub_max_interval` value from the Ceph configuration file is used.

deep_scrub_interval

The interval in seconds for the pool *deep* scrubbing. The default `0` means that the `osd_deep_scrub` value from the Ceph configuration file is used.

REPLICATED POOL VALUES

size

Sets the number of replicas for objects in the pool. See [Section 18.5.6, “Setting the number of object replicas”](#) for further details. Replicated pools only.

min_size

Sets the minimum number of replicas required for I/O. See [Section 18.5.6, “Setting the number of object replicas”](#) for further details. Replicated pools only.

nosizechange

Prevents the pool's size from being changed. When a pool is created, the default value is taken from the value of the `osd_pool_default_flag_nosizechange` parameter which is `false` by default. Applies to replicated pools only because you cannot change size for EC pools.

hit_set_type

Enables hit set tracking for cache pools. See [Bloom Filter \(http://en.wikipedia.org/wiki/Bloom_filter\)](http://en.wikipedia.org/wiki/Bloom_filter) for additional information. This option can have the following values: `bloom`, `explicit_hash`, `explicit_object`. Default is `bloom`, other values are for testing only.

hit_set_count

The number of hit sets to store for cache pools. The higher the number, the more RAM consumed by the `ceph-osd` daemon. Default is `0`.

hit_set_period

The duration of a hit set period in seconds for cache pools. The higher the number, the more RAM consumed by the `ceph-osd` daemon. When a pool is created, the default value is taken from the value of the `osd_tier_default_cache_hit_set_period` parameter, which is `1200` by default. Applies to replicated pools only because EC pools cannot be used as a cache tier.

hit_set_fpp

The false positive probability for the bloom hit set type. See [Bloom Filter \(http://en.wikipedia.org/wiki/Bloom_filter\)](http://en.wikipedia.org/wiki/Bloom_filter) for additional information. Valid range is 0.0 - 1.0 Default is `0.05`

use_gmt_hitset

Force OSDs to use GMT (Greenwich Mean Time) time stamps when creating a hit set for cache tiering. This ensures that nodes in different time zones return the same result. Default is `1`. This value should not be changed.

cache_target_dirty_ratio

The percentage of the cache pool containing modified (dirty) objects before the cache tiering agent will flush them to the backing storage pool. Default is `0.4`.

cache_target_dirty_high_ratio

The percentage of the cache pool containing modified (dirty) objects before the cache tiering agent will flush them to the backing storage pool with a higher speed. Default is `0.6`.

cache_target_full_ratio

The percentage of the cache pool containing unmodified (clean) objects before the cache tiering agent will evict them from the cache pool. Default is `0.8`.

target_max_bytes

Ceph will begin flushing or evicting objects when the `max_bytes` threshold is triggered.

target_max_objects

Ceph will begin flushing or evicting objects when the `max_objects` threshold is triggered.

hit_set_grade_decay_rate

Temperature decay rate between two successive `hit_set`s. Default is `20`.

hit_set_search_last_n

Count at most `N` appearances in `hit_set`s for temperature calculation. Default is `1`.

cache_min_flush_age

The time (in seconds) before the cache tiering agent will flush an object from the cache pool to the storage pool.

cache_min_evict_age

The time (in seconds) before the cache tiering agent will evict an object from the cache pool.

ERASURE CODED POOL VALUES

fast_read

If this flag is enabled on erasure coding pools, then the read request issues sub-reads to all shards, and waits until it receives enough shards to decode to serve the client. In the case of *jerasure* and *isa* erasure plug-ins, when the first K replies return, then the client's request is served immediately using the data decoded from these replies. This approach causes more CPU load and less disk/network load. Currently, this flag is only supported for erasure coding pools. Default is 0.

18.5.6 Setting the number of object replicas

To set the number of object replicas on a replicated pool, execute the following:

```
cephuser@adm > ceph osd pool set poolname size num-replicas
```

The *num-replicas* includes the object itself. For example if you want the object and two copies of the object for a total of three instances of the object, specify 3.



Warning: Do not set less than 3 replicas

If you set the *num-replicas* to 2, there will be only *one* copy of your data. If you lose one object instance, you need to trust that the other copy has not been corrupted, for example since the last scrubbing during recovery (refer to [Section 17.6, "Scrubbing placement groups"](#) for details).

Setting a pool to one replica means that there is exactly *one* instance of the data object in the pool. If the OSD fails, you lose the data. A possible usage for a pool with one replica is storing temporary data for a short time.



Tip: Setting more than 3 replicas

Setting 4 replicas for a pool increases the reliability by 25%.

In case of two data centers, you need to set at least 4 replicas for a pool to have two copies in each data center so that if one data center is lost, two copies still exist and you can still lose one disk without losing data.



Note

An object might accept I/Os in degraded mode with fewer than `pool_size` replicas. To set a minimum number of required replicas for I/O, you should use the `min_size` setting. For example:

```
cephuser@adm > ceph osd pool set data min_size 2
```

This ensures that no object in the data pool will receive I/O with fewer than `min_size` replicas.



Tip: Get the number of object replicas

To get the number of object replicas, execute the following:

```
cephuser@adm > ceph osd dump | grep 'replicated size'
```

Ceph will list the pools, with the `replicated_size` attribute highlighted. By default, Ceph creates two replicas of an object (a total of three copies, or a size of 3).

18.6 Pool migration

When creating a pool (see [Section 18.1, “Creating a pool”](#)) you need to specify its initial parameters, such as the pool type or the number of placement groups. If you later decide to change any of these parameters—for example when converting a replicated pool into an erasure coded one, or decreasing the number of placement groups—you need to migrate the pool data to another one whose parameters suit your deployment.

This section describes two migration methods—a *cache tier* method for general pool data migration, and a method using `rbd migrate` sub-commands to migrate RBD images to a new pool. Each method has its specifics and limitations.

18.6.1 Limitations

- You can use the *cache tier* method to migrate from a replicated pool to either an EC pool or another replicated pool. Migrating from an EC pool is not supported.
- You cannot migrate RBD images and CephFS exports from a replicated pool to an EC pool. The reason is that EC pools do not support `omap`, while RBD and CephFS use `omap` to store its metadata. For example, the header object of the RBD will fail to be flushed. But you can migrate data to EC pool, leaving metadata in replicated pool.
- The `rbd migration` method allows migrating images with minimal client downtime. You only need to stop the client before the `prepare` step and start it afterward. Note that only a `librbd` client that supports this feature (Ceph Nautilus or newer) will be able to open the image just after the `prepare` step, while older `librbd` clients or the `krbd` clients will not be able to open the image until the `commit` step is executed.

18.6.2 Migrating using cache tier

The principle is simple—include the pool that you need to migrate into a cache tier in reverse order. The following example migrates a replicated pool named 'testpool' to an erasure coded pool:

PROCEDURE 18.1: MIGRATING REPLICATED TO ERASURE CODED POOL

1. Create a new erasure coded pool named 'newpool'. Refer to [Section 18.1, “Creating a pool”](#) for a detailed explanation of pool creation parameters.

```
cephuser@adm > ceph osd pool create newpool erasure default
```

Verify that the used client keyring provides at least the same capabilities for 'newpool' as it does for 'testpool'.

Now you have two pools: the original replicated 'testpool' filled with data, and the new empty erasure coded 'newpool':

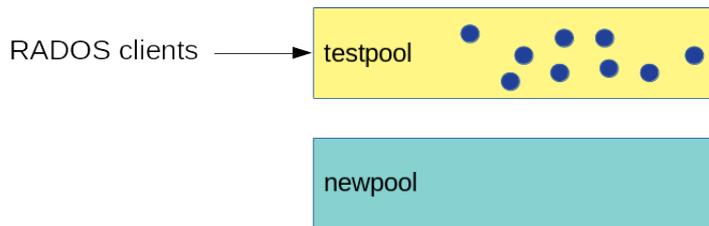


FIGURE 18.1: POOLS BEFORE MIGRATION

2. Set up the cache tier and configure the replicated pool 'testpool' as a cache pool. The `--force-nonempty` option allows adding a cache tier even if the pool already has data:

```
cephuser@adm > ceph tell mon.* injectargs \
 '--mon_debug_unsafe_allow_tier_with_nonempty_snaps=1'
cephuser@adm > ceph osd tier add newpool testpool --force-nonempty
cephuser@adm > ceph osd tier cache-mode testpool proxy
```

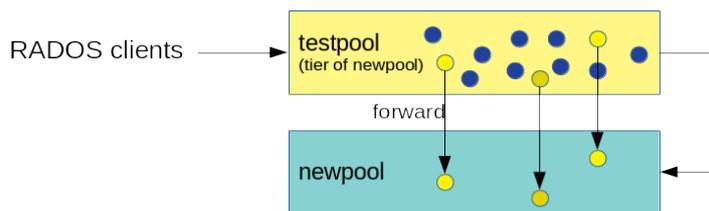


FIGURE 18.2: CACHE TIER SETUP

3. Force the cache pool to move all objects to the new pool:

```
cephuser@adm > rados -p testpool cache-flush-evict-all
```

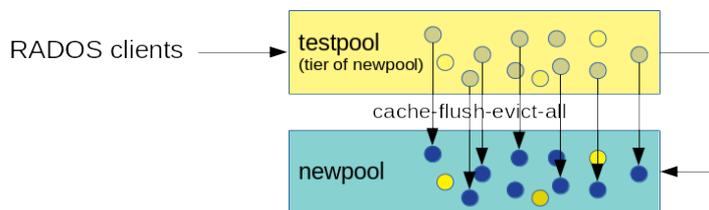


FIGURE 18.3: DATA FLUSHING

4. Until all the data has been flushed to the new erasure coded pool, you need to specify an overlay so that objects are searched on the old pool:

```
cephuser@adm > ceph osd tier set-overlay newpool testpool
```

With the overlay, all operations are forwarded to the old replicated 'testpool':

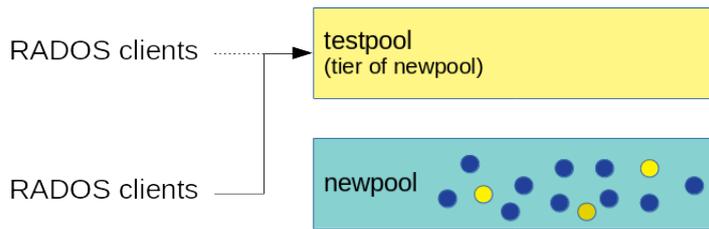


FIGURE 18.4: SETTING OVERLAY

Now you can switch all the clients to access objects on the new pool.

5. After all data is migrated to the erasure coded 'newpool', remove the overlay and the old cache pool 'testpool':

```
cephuser@adm > ceph osd tier remove-overlay newpool
cephuser@adm > ceph osd tier remove newpool testpool
```

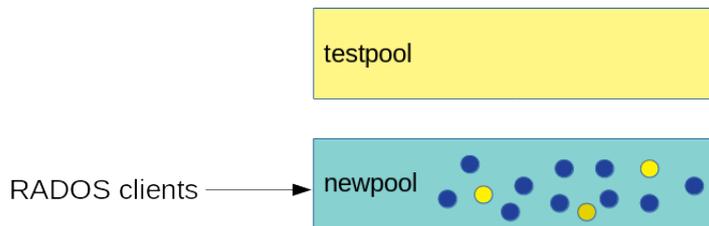


FIGURE 18.5: MIGRATION COMPLETE

6. Run

```
cephuser@adm > ceph tell mon.* injectargs \
'--mon_debug_unsafe_allow_tier_with_nonempty_snaps=0'
```

18.6.3 Migrating RBD images

The following is the recommended way to migrate RBD images from one replicated pool to another replicated pool.

1. Stop clients (such as a virtual machine) from accessing the RBD image.
2. Create a new image in the target pool, with the parent set to the source image:

```
cephuser@adm > rbd migration prepare SRC_POOL/IMAGE TARGET_POOL/IMAGE
```



Tip: Migrate only data to an erasure coded pool

If you need to migrate only the image data to a new EC pool and leave the metadata in the original replicated pool, run the following command instead:

```
cephuser@adm > rbd migration prepare SRC_POOL/IMAGE \  
--data-pool TARGET_POOL/IMAGE
```

3. Let clients access the image in the target pool.
4. Migrate data to the target pool:

```
cephuser@adm > rbd migration execute SRC_POOL/IMAGE
```

5. Remove the old image:

```
cephuser@adm > rbd migration commit SRC_POOL/IMAGE
```

18.7 Pool snapshots

Pool snapshots are snapshots of the state of the whole Ceph pool. With pool snapshots, you can retain the history of the pool's state. Creating pool snapshots consumes storage space proportional to the pool size. Always check the related storage for enough disk space before creating a snapshot of a pool.

18.7.1 Making a snapshot of a pool

To make a snapshot of a pool, run:

```
cephuser@adm > ceph osd pool mksnap POOL-NAME SNAP-NAME
```

For example:

```
cephuser@adm > ceph osd pool mksnap pool1 snap1  
created pool pool1 snap snap1
```

18.7.2 Listing snapshots of a pool

To list existing snapshots of a pool, run:

```
cephuser@adm > rados lssnap -p POOL_NAME
```

For example:

```
cephuser@adm > rados lssnap -p pool1
1 snap1 2018.12.13 09:36:20
2 snap2 2018.12.13 09:46:03
2 snaps
```

18.7.3 Removing a snapshot of a pool

To remove a snapshot of a pool, run:

```
cephuser@adm > ceph osd pool rmsnap POOL-NAME SNAP-NAME
```

18.8 Data compression

BlueStore (find more details in *Book "Deployment Guide", Chapter 1 "SES and Ceph", Section 1.4 "BlueStore"*) provides on-the-fly data compression to save disk space. The compression ratio depends on the data stored in the system. Note that compression/decompression requires additional CPU power.

You can configure data compression globally (see [Section 18.8.3, "Global compression options"](#)) and then override specific compression settings for each individual pool.

You can enable or disable pool data compression, or change the compression algorithm and mode at any time, regardless of whether the pool contains data or not.

No compression will be applied to existing data after enabling the pool compression.

After disabling the compression of a pool, all its data will be decompressed.

18.8.1 Enabling compression

To enable data compression for a pool named *POOL_NAME*, run the following command:

```
cephuser@adm > ceph osd pool set POOL_NAME compression_algorithm COMPRESSION_ALGORITHM
```

```
cephuser@adm > ceph osd pool set POOL_NAME compression_mode COMPRESSION_MODE
```



Tip: Disabling pool compression

To disable data compression for a pool, use 'none' as the compression algorithm:

```
cephuser@adm > ceph osd pool set POOL_NAME compression_algorithm none
```

18.8.2 Pool compression options

A full list of compression settings:

compression_algorithm

Possible values are none, zstd, snappy. Default is snappy.

Which compression algorithm to use depends on the specific use case. Several recommendations follow:

- Use the default snappy as long as you do not have a good reason to change it.
- zstd offers a good compression ratio, but causes high CPU overhead when compressing small amounts of data.
- Run a benchmark of these algorithms on a sample of your actual data while keeping an eye on the CPU and memory usage of your cluster.

compression_mode

Possible values are none, aggressive, passive, force. Default is none.

- none: compress never
- passive: compress if hinted COMPRESSIBLE
- aggressive: compress unless hinted INCOMPRESSIBLE
- force: compress always

compression_required_ratio

Value: Double, Ratio = $\text{SIZE_COMPRESSED} / \text{SIZE_ORIGINAL}$. Default is 0.875, which means that if the compression does not reduce the occupied space by at least 12.5%, the object will not be compressed.

Objects above this ratio will not be stored compressed because of the low net gain.

`compression_max_blob_size`

Value: Unsigned Integer, size in bytes. Default: 0

Maximum size of objects that are compressed.

`compression_min_blob_size`

Value: Unsigned Integer, size in bytes. Default: 0

Minimum size of objects that are compressed.

18.8.3 Global compression options

The following configuration options can be set in the Ceph configuration and apply to all OSDs and not only a single pool. The pool specific configuration listed in [Section 18.8.2, “Pool compression options”](#) takes precedence.

`bluestore_compression_algorithm`

See [compression_algorithm](#)

`bluestore_compression_mode`

See [compression_mode](#)

`bluestore_compression_required_ratio`

See [compression_required_ratio](#)

`bluestore_compression_min_blob_size`

Value: Unsigned Integer, size in bytes. Default: 0

Minimum size of objects that are compressed. The setting is ignored by default in favor of `bluestore_compression_min_blob_size_hdd` and `bluestore_compression_min_blob_size_ssd`. It takes precedence when set to a non-zero value.

`bluestore_compression_max_blob_size`

Value: Unsigned Integer, size in bytes. Default: 0

Maximum size of objects that are compressed before they will be split into smaller chunks. The setting is ignored by default in favor of `bluestore_compression_max_blob_size_hdd` and `bluestore_compression_max_blob_size_ssd`. It takes precedence when set to a non-zero value.

`bluestore_compression_min_blob_size_ssd`

Value: Unsigned Integer, size in bytes. Default: 8K

Minimum size of objects that are compressed and stored on solid-state drive.

bluestore_compression_max_blob_size_ssd

Value: Unsigned Integer, size in bytes. Default: 64K

Maximum size of objects that are compressed and stored on solid-state drive before they will be split into smaller chunks.

bluestore_compression_min_blob_size_hdd

Value: Unsigned Integer, size in bytes. Default: 128K

Minimum size of objects that are compressed and stored on hard disks.

bluestore_compression_max_blob_size_hdd

Value: Unsigned Integer, size in bytes. Default: 512K

Maximum size of objects that are compressed and stored on hard disks before they will be split into smaller chunks.

19 Erasure coded pools

Ceph provides an alternative to the normal replication of data in pools, called *erasure* or *erasure coded* pool. Erasure pools do not provide all functionality of *replicated* pools (for example, they cannot store metadata for RBD pools), but require less raw storage. A default erasure pool capable of storing 1 TB of data requires 1.5 TB of raw storage, allowing a single disk failure. This compares favorably to a replicated pool, which needs 2 TB of raw storage for the same purpose. For background information on Erasure Code, see https://en.wikipedia.org/wiki/Erasure_code. For a list of pool values related to EC pools, refer to *Erasure coded pool values*.

19.1 Prerequisite for erasure coded Pools

To make use of erasure coding, you need to:

- Define an erasure rule in the CRUSH Map.
- Define an erasure code profile that specifies the coding algorithm to be used.
- Create a pool using the previously mentioned rule and profile.

Keep in mind that changing the profile and the details in the profile will not be possible after the pool is created and has data.

Ensure that the CRUSH rules for *erasure pools* use `indep` for `step`. For details see [Section 17.3.2, "firstn and indep"](#).

19.2 Creating a sample erasure coded pool

The simplest erasure coded pool is equivalent to RAID5 and requires at least three hosts. This procedure describes how to create a pool for testing purposes.

1. The command `ceph osd pool create` is used to create a pool with type *erasure*. The `12` stands for the number of placement groups. With default parameters, the pool is able to handle the failure of one OSD.

```
cephuser@adm > ceph osd pool create ecpool 12 12 erasure
pool 'ecpool' created
```

2. The string `ABCDEFGHI` is written into an object called `NYAN`.

```
cephuser@adm > echo ABCDEFGHI | rados --pool ecpool put NYAN -
```

3. For testing purposes OSDs can now be disabled, for example by disconnecting them from the network.
4. To test whether the pool can handle the failure of devices, the content of the file can be accessed with the `rados` command.

```
cephuser@adm > rados --pool ecpool get NYAN -  
ABCDEFGHI
```

19.3 Erasure code profiles

When the `ceph osd pool create` command is invoked to create an *erasure pool*, the default profile is used, unless another profile is specified. Profiles define the redundancy of data. This is done by setting two parameters, arbitrarily named `k` and `m`. `k` and `m` define in how many chunks a piece of data is split and how many coding chunks are created. Redundant chunks are then stored on different OSDs.

Definitions required for erasure pool profiles:

chunk

when the encoding function is called, it returns chunks of the same size: data chunks which can be concatenated to reconstruct the original object and coding chunks which can be used to rebuild a lost chunk.

k

the number of data chunks, that is the number of chunks into which the original object is divided. For example, if `k = 2` a 10 kB object will be divided into `k` objects of 5 kB each. The default `min_size` on erasure coded pools is `k + 1`. However, we recommend `min_size` to be `k + 2` or more to prevent loss of writes and data.

m

the number of coding chunks, that is the number of additional chunks computed by the encoding functions. If there are 2 coding chunks, it means 2 OSDs can be out without losing data.

crush-failure-domain

defines to which devices the chunks are distributed. A bucket type needs to be set as value. For all bucket types, see [Section 17.2, “Buckets”](#). If the failure domain is `rack`, the chunks will be stored on different racks to increase the resilience in case of rack failures. Keep in mind that this requires $k + m$ racks.

With the default erasure code profile used in [Section 19.2, “Creating a sample erasure coded pool”](#), you will not lose cluster data if a single OSD or host fails. Therefore, to store 1 TB of data it needs another 0.5 TB of raw storage. That means 1.5 TB of raw storage is required for 1 TB of data (because of $k = 2$, $m = 1$). This is equivalent to a common RAID 5 configuration. For comparison, a replicated pool needs 2 TB of raw storage to store 1 TB of data.

The settings of the default profile can be displayed with:

```
cephuser@adm > ceph osd erasure-code-profile get default
directory=.libs
k=2
m=1
plugin=jerasure
crush-failure-domain=host
technique=reed_sol_van
```

Choosing the right profile is important because it cannot be modified after the pool is created. A new pool with a different profile needs to be created and all objects from the previous pool moved to the new one (see [Section 18.6, “Pool migration”](#)).

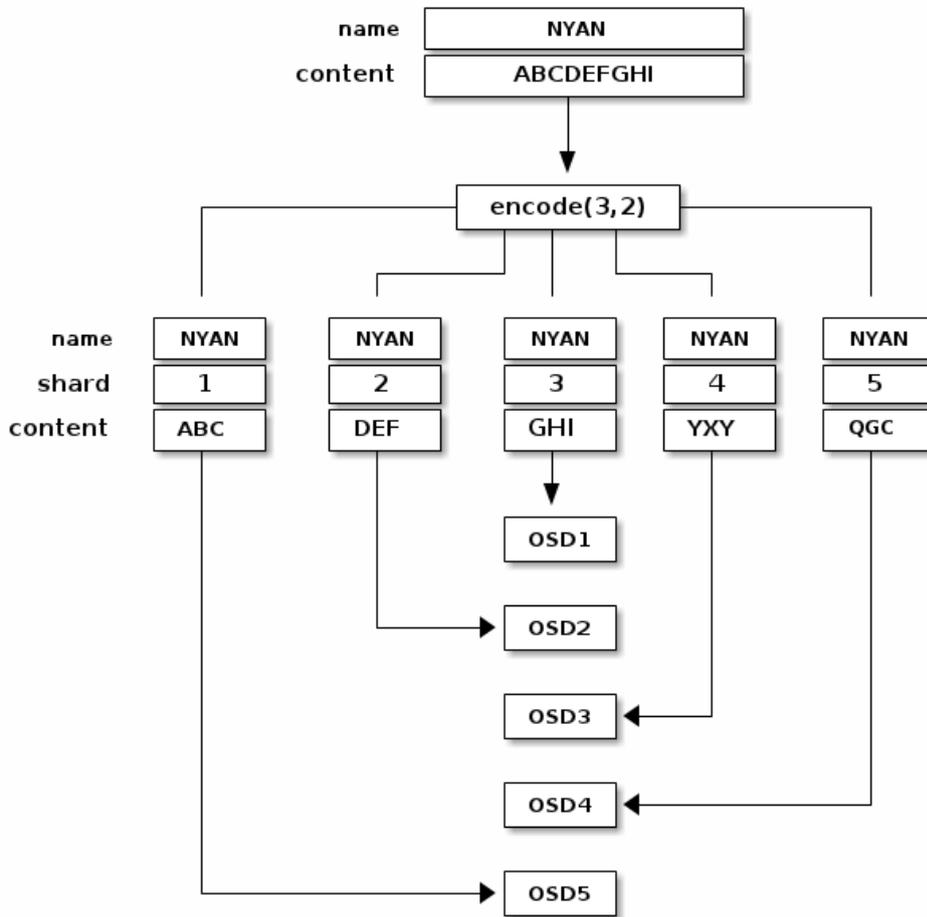
The most important parameters of the profile are `k`, `m` and `crush-failure-domain` because they define the storage overhead and the data durability. For example, if the desired architecture must sustain the loss of two racks with a storage overhead of 66%, the following profile can be defined. Note that this is only valid with a CRUSH Map that has buckets of type 'rack':

```
cephuser@adm > ceph osd erasure-code-profile set myprofile \
k=3 \
m=2 \
crush-failure-domain=rack
```

The example [Section 19.2, “Creating a sample erasure coded pool”](#) can be repeated with this new profile:

```
cephuser@adm > ceph osd pool create ecpool 12 12 erasure myprofile
cephuser@adm > echo ABCDEFGHI | rados --pool ecpool put NYAN -
cephuser@adm > rados --pool ecpool get NYAN -
ABCDEFGHI
```

The NYAN object will be divided in three ($k=3$) and two additional chunks will be created ($m=2$). The value of m defines how many OSDs can be lost simultaneously without losing any data. The `crush-failure-domain=rack` will create a CRUSH ruleset that ensures no two chunks are stored in the same rack.



19.3.1 Creating a new erasure code profile

The following command creates a new erasure code profile:

```
# ceph osd erasure-code-profile set NAME \
  directory=DIRECTORY \
  plugin=PLUGIN \
  stripe_unit=STRIPE_UNIT \
  KEY=VALUE ... \
  --force
```

DIRECTORY

Optional. Set the directory name from which the erasure code plugin is loaded. Default is `/usr/lib/ceph/erasure-code`.

PLUGIN

Optional. Use the erasure code plugin to compute coding chunks and recover missing chunks. Available plugins are 'jerasure', 'isa', 'lrc', and 'shes'. Default is 'jerasure'.

STRIPE_UNIT

Optional. The amount of data in a data chunk, per stripe. For example, a profile with 2 data chunks and `stripe_unit=4K` would put the range 0-4K in chunk 0, 4K-8K in chunk 1, then 8K-12K in chunk 0 again. This should be a multiple of 4K for best performance. The default value is taken from the monitor configuration option `osd_pool_erasure_code_stripe_unit` when a pool is created. The 'stripe_width' of a pool using this profile will be the number of data chunks multiplied by this 'stripe_unit'.

KEY=VALUE

Key/value pairs of options specific to the selected erasure code plugin.

--force

Optional. Override an existing profile by the same name, and allow setting a non-4K-aligned `stripe_unit`.

19.3.2 Removing an erasure code profile

The following command removes an erasure code profile as identified by its `NAME`:

```
# ceph osd erasure-code-profile rm NAME
```



Important

If the profile is referenced by a pool, the deletion will fail.

19.3.3 Displaying an erasure code profile's details

The following command displays details of an erasure code profile as identified by its `NAME`:

```
# ceph osd erasure-code-profile get NAME
```

19.3.4 Listing erasure code profiles

The following command lists the names of all erasure code profiles:

```
# ceph osd erasure-code-profile ls
```

19.4 Marking erasure coded pools with RADOS Block Device

To mark an EC pool as an RBD pool, tag it accordingly:

```
cephuser@adm > ceph osd pool application enable rbd ec_pool_name
```

RBD can store image *data* in EC pools. However, the image header and metadata still need to be stored in a replicated pool. Assuming you have the pool named 'rbd' for this purpose:

```
cephuser@adm > rbd create rbd/image_name --size 1T --data-pool ec_pool_name
```

You can use the image normally like any other image, except that all of the data will be stored in the *ec_pool_name* pool instead of 'rbd' pool.

20 RADOS Block Device

A block is a sequence of bytes, for example a 4 MB block of data. Block-based storage interfaces are the most common way to store data with rotating media, such as hard disks, CDs, floppy disks. The ubiquity of block device interfaces makes a virtual block device an ideal candidate to interact with a mass data storage system like Ceph.

Ceph block devices allow sharing of physical resources, and are resizable. They store data striped over multiple OSDs in a Ceph cluster. Ceph block devices leverage RADOS capabilities such as snapshotting, replication, and consistency. Ceph's RADOS Block Devices (RBD) interact with OSDs using kernel modules or the `librbd` library.

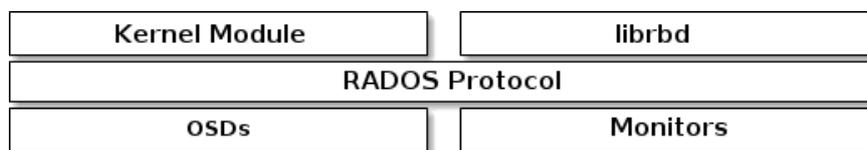


FIGURE 20.1: RADOS PROTOCOL

Ceph's block devices deliver high performance with infinite scalability to kernel modules. They support virtualization solutions such as QEMU, or cloud-based computing systems such as OpenStack that rely on `libvirt`. You can use the same cluster to operate the Object Gateway, CephFS, and RADOS Block Devices simultaneously.

20.1 Block device commands

The `rbd` command enables you to create, list, introspect, and remove block device images. You can also use it, for example, to clone images, create snapshots, rollback an image to a snapshot, or view a snapshot.

20.1.1 Creating a block device image in a replicated pool

Before you can add a block device to a client, you need to create a related image in an existing pool (see [Chapter 18, Manage storage pools](#)):

```
cephuser@adm > rbd create --size MEGABYTES POOL-NAME/IMAGE-NAME
```

For example, to create a 1 GB image named 'myimage' that stores information in a pool named 'mypool', execute the following:

```
cephuser@adm > rbd create --size 1024 mypool/myimage
```



Tip: Image size units

If you omit a size unit shortcut ('G' or 'T'), the image's size is in megabytes. Use 'G' or 'T' after the size number to specify gigabytes or terabytes.

20.1.2 Creating a block device image in an erasure coded pool

It is possible to store data of a block device image directly in erasure coded (EC) pools. A RADOS Block Device image consists of *data* and *metadata* parts. You can store only the data part of a RADOS Block Device image in an EC pool. The pool needs to have the `overwrite` flag set to `true`, and that is only possible if all OSDs where the pool is stored use BlueStore.

You cannot store the image's metadata part in an EC pool. You can specify the replicated pool for storing the image's metadata with the `--pool=` option of the `rbd create` command or specify `pool/` as a prefix to the image name.

Create an EC pool:

```
cephuser@adm > ceph osd pool create EC_POOL 12 12 erasure
cephuser@adm > ceph osd pool set EC_POOL allow_ec_overwrites true
```

Specify the replicated pool for storing metadata:

```
cephuser@adm > rbd create IMAGE_NAME --size=1G --data-pool EC_POOL --pool=POOL
```

Or:

```
cephuser@adm > rbd create POOL/IMAGE_NAME --size=1G --data-pool EC_POOL
```

20.1.3 Listing block device images

To list block devices in a pool named 'mypool', execute the following:

```
cephuser@adm > rbd ls mypool
```

20.1.4 Retrieving image information

To retrieve information from an image 'myimage' within a pool named 'mypool', run the following:

```
cephuser@adm > rbd info mypool/myimage
```

20.1.5 Resizing a block device image

RADOS Block Device images are thin provisioned—they do not actually use any physical storage until you begin saving data to them. However, they do have a maximum capacity that you set with the `--size` option. If you want to increase (or decrease) the maximum size of the image, run the following:

```
cephuser@adm > rbd resize --size 2048 POOL_NAME/IMAGE_NAME # to increase
cephuser@adm > rbd resize --size 2048 POOL_NAME/IMAGE_NAME --allow-shrink # to decrease
```

20.1.6 Removing a block device image

To remove a block device that corresponds to an image 'myimage' in a pool named 'mypool', run the following:

```
cephuser@adm > rbd rm mypool/myimage
```

20.2 Mounting and unmounting

After you create a RADOS Block Device, you can use it like any other disk device: format it, mount it to be able to exchange files, and unmount it when done.

The `rbd` command defaults to accessing the cluster using the Ceph `admin` user account. This account has full administrative access to the cluster. This runs the risk of accidentally causing damage, similarly to logging into a Linux workstation as `root`. Thus, it is preferable to create user accounts with fewer privileges and use these accounts for normal read/write RADOS Block Device access.

20.2.1 Creating a Ceph user account

To create a new user account with Ceph Manager, Ceph Monitor, and Ceph OSD capabilities, use the `ceph` command with the `auth get-or-create` subcommand:

```
cephuser@adm > ceph auth get-or-create client.ID mon 'profile rbd' osd 'profile profile
name \
  [pool=pool-name] [, profile ...]' mgr 'profile rbd [pool=pool-name]'
```

For example, to create a user called `qemu` with read-write access to the pool `vms` and read-only access to the pool `images`, execute the following:

```
ceph auth get-or-create client.qemu mon 'profile rbd' osd 'profile rbd pool=vms, profile
rbd-read-only pool=images' \
  mgr 'profile rbd pool=images'
```

The output from the `ceph auth get-or-create` command will be the keyring for the specified user, which can be written to `/etc/ceph/ceph.client.ID.keyring`.



Note

When using the `rbd` command, you can specify the user ID by providing the optional `--id ID` argument.

For more details on managing Ceph user accounts, refer to [Chapter 30, Authentication with cephx](#).

20.2.2 User authentication

To specify a user name, use `--id user-name`. If you use `cephx` authentication, you also need to specify a secret. It may come from a keyring or a file containing the secret:

```
cephuser@adm > rbd device map --pool rbd myimage --id admin --keyring /path/to/keyring
```

or

```
cephuser@adm > rbd device map --pool rbd myimage --id admin --keyfile /path/to/file
```

20.2.3 Preparing a RADOS Block Device for use

1. Make sure your Ceph cluster includes a pool with the disk image you want to map. Assume the pool is called `mypool` and the image is `myimage`.

```
cephuser@adm > rbd list mypool
```

2. Map the image to a new block device:

```
cephuser@adm > rbd device map --pool mypool myimage
```

3. List all mapped devices:

```
cephuser@adm > rbd device list
id pool  image  snap device
0  mypool myimage -  /dev/rbd0
```

The device we want to work on is /dev/rbd0.



Tip: RBD device path

Instead of /dev/rbdDEVICE_NUMBER, you can use /dev/rbd/POOL_NAME/IMAGE_NAME as a persistent device path. For example:

```
/dev/rbd/mypool/myimage
```

4. Make an XFS file system on the /dev/rbd0 device:

```
# mkfs.xfs /dev/rbd0
log stripe unit (4194304 bytes) is too large (maximum is 256KiB)
log stripe unit adjusted to 32KiB
meta-data=/dev/rbd0          isize=256    agcount=9, agsize=261120 blks
=                               sectsz=512   attr=2, projid32bit=1
=                               crc=0      finobt=0
data      =                               bsize=4096  blocks=2097152, imaxpct=25
=                               sunit=1024  swidth=1024 blks
naming    =version 2                bsize=4096  ascii-ci=0  ftype=0
log       =internal log            bsize=4096  blocks=2560, version=2
=                               sectsz=512   sunit=8 blks, lazy-count=1
realtime  =none                    extsz=4096  blocks=0, rtextents=0
```

5. Replacing /mnt with your mount point, mount the device and check it is correctly mounted:

```
# mount /dev/rbd0 /mnt
# mount | grep rbd0
/dev/rbd0 on /mnt type xfs (rw,relatime,attr2,inode64,sunit=8192,...
```

Now you can move data to and from the device as if it was a local directory.



Tip: Increasing the size of RBD device

If you find that the size of the RBD device is no longer enough, you can easily increase it.

1. Increase the size of the RBD image, for example up to 10 GB.

```
cephuser@adm > rbd resize --size 10000 mypool/myimage
Resizing image: 100% complete...done.
```

2. Grow the file system to fill up the new size of the device:

```
# xfs_growfs /mnt
[...]
data blocks changed from 2097152 to 2560000
```

6. After you finish accessing the device, you can unmap and unmount it.

```
cephuser@adm > rbd device unmap /dev/rbd0
# unmount /mnt
```



Tip: Manual mounting and unmounting

A `rbdmap` script and `systemd` unit is provided to make the process of mapping and mounting RBDs after boot, and unmounting them before shutdown, smoother. Refer to [Section 20.2.4, “`rbdmap` Map RBD devices at boot time”](#).

20.2.4 `rbdmap` Map RBD devices at boot time

`rbdmap` is a shell script that automates `rbd map` and `rbd device unmap` operations on one or more RBD images. Although you can run the script manually at any time, the main advantage is automatic mapping and mounting of RBD images at boot time (and unmounting and unmapping at shutdown), as triggered by the Init system. A `systemd` unit file, `rbdmap.service` is included with the `ceph-common` package for this purpose.

The script takes a single argument, which can be either `map` or `unmap`. In either case, the script parses a configuration file. It defaults to `/etc/ceph/rbdmap`, but can be overridden via an environment variable `RBDMAPFILE`. Each line of the configuration file corresponds to an RBD image which is to be mapped, or unmapped.

The configuration file has the following format:

```
image_specification rbd_options
```

image_specification

Path to an image within a pool. Specify as `pool_name / image_name`.

rbd_options

An optional list of parameters to be passed to the underlying `rbd device map` command. These parameters and their values should be specified as a comma-separated string, for example:

```
PARAM1=VAL1,PARAM2=VAL2,...
```

The example makes the `rbdmap` script run the following command:

```
cephuser@adm > rbd device map POOL_NAME/IMAGE_NAME --PARAM1 VAL1 --PARAM2 VAL2
```

In the following example you can see how to specify a user name and a keyring with a corresponding secret:

```
cephuser@adm > rbdmap device map mypool/myimage id=rbd_user,keyring=/etc/ceph/ceph.client.rbd.keyring
```

When run as `rbdmap map`, the script parses the configuration file, and for each specified RBD image, it attempts to first map the image (using the `rbd device map` command) and then mount the image.

When run as `rbdmap unmap`, images listed in the configuration file will be unmounted and unmapped.

`rbdmap unmap-all` attempts to unmount and subsequently unmap all currently mapped RBD images, regardless of whether they are listed in the configuration file.

If successful, the `rbd device map` operation maps the image to a `/dev/rbdX` device, at which point a udev rule is triggered to create a friendly device name symbolic link `/dev/rbd/pool_name/image_name` pointing to the real mapped device.

In order for mounting and unmounting to succeed, the 'friendly' device name needs to have a corresponding entry in `/etc/fstab`. When writing `/etc/fstab` entries for RBD images, specify the 'noauto' (or 'nofail') mount option. This prevents the Init system from trying to mount the device too early—before the device in question even exists, as `rbdmap.service` is typically triggered quite late in the boot sequence.

For a complete list of `rbd` options, see the `rbd` manual page (`man 8 rbd`).

For examples of the `rbdmap` usage, see the `rbdmap` manual page (`man 8 rbdmap`).

20.2.5 Increasing the size of RBD devices

If you find that the size of the RBD device is no longer enough, you can easily increase it.

1. Increase the size of the RBD image, for example up to 10GB.

```
cephuser@adm > rbd resize --size 10000 mypool/myimage
Resizing image: 100% complete...done.
```

2. Grow the file system to fill up the new size of the device.

```
# xfs_growfs /mnt
[...]
data blocks changed from 2097152 to 2560000
```

20.3 Snapshots

An RBD snapshot is a snapshot of a RADOS Block Device image. With snapshots, you retain a history of the image's state. Ceph also supports snapshot layering, which allows you to clone VM images quickly and easily. Ceph supports block device snapshots using the `rbd` command and many higher-level interfaces, including QEMU, `libvirt`, OpenStack, and CloudStack.



Note

Stop input and output operations and flush all pending writes before snapshotting an image. If the image contains a file system, the file system must be in a consistent state at the time of snapshotting.

20.3.1 Enabling and configuring cephx

When `cephx` is enabled, you must specify a user name or ID and a path to the keyring containing the corresponding key for the user. See [Chapter 30, Authentication with cephx](#) for more details. You may also add the `CEPH_ARGS` environment variable to avoid re-entry of the following parameters.

```
cephuser@adm > rbd --id user-ID --keyring=/path/to/secret commands
cephuser@adm > rbd --name username --keyring=/path/to/secret commands
```

For example:

```
cephuser@adm > rbd --id admin --keyring=/etc/ceph/ceph.keyring commands
cephuser@adm > rbd --name client.admin --keyring=/etc/ceph/ceph.keyring commands
```



Tip

Add the user and secret to the `CEPH_ARGS` environment variable so that you do not need to enter them each time.

20.3.2 Snapshot basics

The following procedures demonstrate how to create, list, and remove snapshots using the `rbd` command on the command line.

20.3.2.1 Creating snapshots

To create a snapshot with `rbd`, specify the `snap create` option, the pool name, and the image name.

```
cephuser@adm > rbd --pool pool-name snap create --snap snap-name image-name
cephuser@adm > rbd snap create pool-name/image-name@snap-name
```

For example:

```
cephuser@adm > rbd --pool rbd snap create --snap snapshot1 image1
cephuser@adm > rbd snap create rbd/image1@snapshot1
```

20.3.2.2 Listing snapshots

To list snapshots of an image, specify the pool name and the image name.

```
cephuser@adm > rbd --pool pool-name snap ls image-name
cephuser@adm > rbd snap ls pool-name/image-name
```

For example:

```
cephuser@adm > rbd --pool rbd snap ls image1
cephuser@adm > rbd snap ls rbd/image1
```

20.3.2.3 Rolling back snapshots

To rollback to a snapshot with **rbd**, specify the `snap rollback` option, the pool name, the image name, and the snapshot name.

```
cephuser@adm > rbd --pool pool-name snap rollback --snap snap-name image-name
cephuser@adm > rbd snap rollback pool-name/image-name@snap-name
```

For example:

```
cephuser@adm > rbd --pool pool1 snap rollback --snap snapshot1 image1
cephuser@adm > rbd snap rollback pool1/image1@snapshot1
```



Note

Rolling back an image to a snapshot means overwriting the current version of the image with data from a snapshot. The time it takes to execute a rollback increases with the size of the image. It is *faster to clone* from a snapshot *than to rollback* an image to a snapshot, and it is the preferred method of returning to a pre-existing state.

20.3.2.4 Deleting a snapshot

To delete a snapshot with **rbd**, specify the `snap rm` option, the pool name, the image name, and the user name.

```
cephuser@adm > rbd --pool pool-name snap rm --snap snap-name image-name
cephuser@adm > rbd snap rm pool-name/image-name@snap-name
```

For example:

```
cephuser@adm > rbd --pool pool1 snap rm --snap snapshot1 image1
cephuser@adm > rbd snap rm pool1/image1@snapshot1
```



Note

Ceph OSDs delete data asynchronously, so deleting a snapshot does not free up the disk space immediately.

20.3.2.5 Purging snapshots

To delete all snapshots for an image with `rbd`, specify the `snap purge` option and the image name.

```
cephuser@adm > rbd --pool pool-name snap purge image-name
cephuser@adm > rbd snap purge pool-name/image-name
```

For example:

```
cephuser@adm > rbd --pool pool1 snap purge image1
cephuser@adm > rbd snap purge pool1/image1
```

20.3.3 Snapshot layering

Ceph supports the ability to create multiple copy-on-write (COW) clones of a block device snapshot. Snapshot layering enables Ceph block device clients to create images very quickly. For example, you might create a block device image with a Linux VM written to it, then, snapshot the image, protect the snapshot, and create as many copy-on-write clones as you like. A snapshot is read-only, so cloning a snapshot simplifies semantics—making it possible to create clones rapidly.



Note

The terms 'parent' and 'child' mentioned in the command line examples below mean a Ceph block device snapshot (parent) and the corresponding image cloned from the snapshot (child).

Each cloned image (child) stores a reference to its parent image, which enables the cloned image to open the parent snapshot and read it.

A COW clone of a snapshot behaves exactly like any other Ceph block device image. You can read to, write from, clone, and resize cloned images. There are no special restrictions with cloned images. However, the copy-on-write clone of a snapshot refers to the snapshot, so you *must* protect the snapshot before you clone it.



Note: `--image-format 1` not supported

You cannot create snapshots of images created with the deprecated `rbd create --image-format 1` option. Ceph only supports cloning of the default `format 2` images.

20.3.3.1 Getting started with layering

Ceph block device layering is a simple process. You must have an image. You must create a snapshot of the image. You must protect the snapshot. After you have performed these steps, you can begin cloning the snapshot.

The cloned image has a reference to the parent snapshot, and includes the pool ID, image ID, and snapshot ID. The inclusion of the pool ID means that you may clone snapshots from one pool to images in another pool.

- *Image Template:* A common use case for block device layering is to create a master image and a snapshot that serves as a template for clones. For example, a user may create an image for a Linux distribution (for example, SUSE Linux Enterprise Server), and create a snapshot for it. Periodically, the user may update the image and create a new snapshot (for example, `zypper ref && zypper patch` followed by `rbd snap create`). As the image matures, the user can clone any one of the snapshots.
- *Extended Template:* A more advanced use case includes extending a template image that provides more information than a base image. For example, a user may clone an image (a VM template) and install other software (for example, a database, a content management system, or an analytics system), and then snapshot the extended image, which itself may be updated in the same way as the base image.

- *Template Pool*: One way to use block device layering is to create a pool that contains master images that act as templates, and snapshots of those templates. You may then extend read-only privileges to users so that they may clone the snapshots without the ability to write or execute within the pool.
- *Image Migration/Recovery*: One way to use block device layering is to migrate or recover data from one pool into another pool.

20.3.3.2 Protecting a snapshot

Clones access the parent snapshots. All clones would break if a user inadvertently deleted the parent snapshot. To prevent data loss, you need to protect the snapshot before you can clone it.

```
cephuser@adm > rbd --pool pool-name snap protect \
--image image-name --snap snapshot-name
cephuser@adm > rbd snap protect pool-name/image-name@snapshot-name
```

For example:

```
cephuser@adm > rbd --pool pool1 snap protect --image image1 --snap snapshot1
cephuser@adm > rbd snap protect pool1/image1@snapshot1
```



Note

You cannot delete a protected snapshot.

20.3.3.3 Cloning a snapshot

To clone a snapshot, you need to specify the parent pool, image, snapshot, the child pool, and the image name. You need to protect the snapshot before you can clone it.

```
cephuser@adm > rbd clone --pool pool-name --image parent-image \
--snap snap-name --dest-pool pool-name \
--dest child-image
cephuser@adm > rbd clone pool-name/parent-image@snap-name \
pool-name/child-image-name
```

For example:

```
cephuser@adm > rbd clone pool1/image1@snapshot1 pool1/image2
```



Note

You may clone a snapshot from one pool to an image in another pool. For example, you may maintain read-only images and snapshots as templates in one pool, and writable clones in another pool.

20.3.3.4 Unprotecting a snapshot

Before you can delete a snapshot, you must unprotect it first. Additionally, you may *not* delete snapshots that have references from clones. You need to flatten each clone of a snapshot before you can delete the snapshot.

```
cephuser@adm > rbd --pool pool-name snap unprotect --image image-name \  
--snap snapshot-name  
cephuser@adm > rbd snap unprotect pool-name/image-name@snapshot-name
```

For example:

```
cephuser@adm > rbd --pool pool1 snap unprotect --image image1 --snap snapshot1  
cephuser@adm > rbd snap unprotect pool1/image1@snapshot1
```

20.3.3.5 Listing children of a snapshot

To list the children of a snapshot, execute the following:

```
cephuser@adm > rbd --pool pool-name children --image image-name --snap snap-name  
cephuser@adm > rbd children pool-name/image-name@snapshot-name
```

For example:

```
cephuser@adm > rbd --pool pool1 children --image image1 --snap snapshot1  
cephuser@adm > rbd children pool1/image1@snapshot1
```

20.3.3.6 Flattening a cloned image

Cloned images retain a reference to the parent snapshot. When you remove the reference from the child clone to the parent snapshot, you effectively 'flatten' the image by copying the information from the snapshot to the clone. The time it takes to flatten a clone increases with the size of the snapshot. To delete a snapshot, you must flatten the child images first.

```
cephuser@adm > rbd --pool pool-name flatten --image image-name
cephuser@adm > rbd flatten pool-name/image-name
```

For example:

```
cephuser@adm > rbd --pool pool1 flatten --image image1
cephuser@adm > rbd flatten pool1/image1
```



Note

Since a flattened image contains all the information from the snapshot, a flattened image will take up more storage space than a layered clone.

20.4 RBD image mirrors

RBD images can be asynchronously mirrored between two Ceph clusters. This capability is available in two modes:

Journal-based

This mode uses the RBD journaling image feature to ensure point-in-time, crash-consistent replication between clusters. Every write to the RBD image is first recorded to the associated journal before modifying the actual image. The `remote` cluster will read from the journal and replay the updates to its local copy of the image. Since each write to the RBD image will result in two writes to the Ceph cluster, expect write latencies to nearly double when using the RBD journaling image feature.

Snapshot-based

This mode uses periodically-scheduled or manually-created RBD image mirror-snapshots to replicate crash-consistent RBD images between clusters. The `remote` cluster will determine any data or metadata updates between two mirror-snapshots, and copy the deltas to its local copy of the image. With the help of the RBD fast-diff image feature, updated data blocks can be quickly computed without the need to scan the full RBD image. Since this mode is not point-in-time consistent, the full snapshot delta will need to be synchronized prior to use during a failover scenario. Any partially-applied snapshot deltas will be rolled back to the last fully synchronized snapshot prior to use.

Mirroring is configured on a per-pool basis within peer clusters. This can be configured on a specific subset of images within the pool, or configured to automatically mirror all images within a pool when using journal-based mirroring only. Mirroring is configured using the `rbd` command. The `rbd-mirror` daemon is responsible for pulling image updates from the `remote`, peer cluster and applying them to the image within the `local` cluster.

Depending on the desired needs for replication, RBD mirroring can be configured for either one- or two-way replication:

One-way Replication

When data is only mirrored from a primary cluster to a secondary cluster, the `rbd-mirror` daemon runs only on the secondary cluster.

Two-way Replication

When data is mirrored from primary images on one cluster to non-primary images on another cluster (and vice-versa), the `rbd-mirror` daemon runs on both clusters.



Important

Each instance of the `rbd-mirror` daemon needs to be able to connect to both the `local` and `remote` Ceph clusters simultaneously. For example, all monitor and OSD hosts. Additionally, the network needs to have sufficient bandwidth between the two data centers to handle mirroring workload.

20.4.1 Pool configuration

The following procedures demonstrate how to perform the basic administrative tasks to configure mirroring using the `rbd` command. Mirroring is configured on a per-pool basis within the Ceph clusters.

You need to perform the pool configuration steps on both peer clusters. These procedures assume two clusters, named `local` and `remote`, are accessible from a single host for clarity.

See the `rbd` manual page (`man 8 rbd`) for additional details on how to connect to different Ceph clusters.



Tip: Multiple clusters

The cluster name in the following examples corresponds to a Ceph configuration file of the same name `/etc/ceph/remote.conf` and Ceph keyring file of the same name `/etc/ceph/remote.client.admin.keyring`.

20.4.1.1 Enable mirroring on a pool

To enable mirroring on a pool, specify the `mirror pool enable` subcommand, the pool name, and the mirroring mode. The mirroring mode can either be pool or image:

pool

All images in the pool with the journaling feature enabled are mirrored.

image

Mirroring needs to be explicitly enabled on each image. See [Section 20.4.2.1, “Enabling image mirroring”](#) for more information.

For example:

```
cephuser@adm > rbd --cluster local mirror pool enable POOL_NAME pool
cephuser@adm > rbd --cluster remote mirror pool enable POOL_NAME pool
```

20.4.1.2 Disable mirroring

To disable mirroring on a pool, specify the `mirror pool disable` subcommand and the pool name. When mirroring is disabled on a pool in this way, mirroring will also be disabled on any images (within the pool) for which mirroring was enabled explicitly.

```
cephuser@adm > rbd --cluster local mirror pool disable POOL_NAME
cephuser@adm > rbd --cluster remote mirror pool disable POOL_NAME
```

20.4.1.3 Bootstrapping peers

In order for the `rbd-mirror` daemon to discover its peer cluster, the peer needs to be registered to the pool and a user account needs to be created. This process can be automated with `rbd` and the `mirror pool peer bootstrap create` and `mirror pool peer bootstrap import` commands.


```
EOF
```

```
cephuser@adm > rbd --cluster remote mirror pool peer bootstrap import \  
--site-name remote image-pool token
```

20.4.1.4 Adding a cluster peer manually

Alternatively to bootstrapping peers as described in [Section 20.4.1.3, “Bootstrapping peers”](#), you can specify peers manually. The remote `rbd-mirror` daemon will need access to the local cluster to perform mirroring. Create a new local Ceph user that the remote `rbd-mirror` daemon will use, for example `rbd-mirror-peer`:

```
cephuser@adm > ceph auth get-or-create client.rbd-mirror-peer \  
mon 'profile rbd' osd 'profile rbd'
```

Use the following syntax to add a mirroring peer Ceph cluster with the `rbd` command:

```
rbd mirror pool peer add POOL_NAME CLIENT_NAME@CLUSTER_NAME
```

For example:

```
cephuser@adm > rbd --cluster site-a mirror pool peer add image-pool client.rbd-mirror-  
peer@site-b  
cephuser@adm > rbd --cluster site-b mirror pool peer add image-pool client.rbd-mirror-  
peer@site-a
```

By default, the `rbd-mirror` daemon needs to have access to the Ceph configuration file located at `/etc/ceph/.CLUSTER_NAME.conf`. It provides IP addresses of the peer cluster’s MONs and a keyring for a client named `CLIENT_NAME` located in the default or custom keyring search paths, for example `/etc/ceph/CLUSTER_NAME.CLIENT_NAME.keyring`.

Alternatively, the peer cluster’s MON and/or client key can be securely stored within the local Ceph config-key store. To specify the peer cluster connection attributes when adding a mirroring peer, use the `--remote-mon-host` and `--remote-key-file` options. For example:

```
cephuser@adm > rbd --cluster site-a mirror pool peer add image-pool \  
client.rbd-mirror-peer@site-b --remote-mon-host 192.168.1.1,192.168.1.2 \  
--remote-key-file /PATH/TO/KEY_FILE  
cephuser@adm > rbd --cluster site-a mirror pool info image-pool --all  
Mode: pool  
Peers:  
  UUID          NAME    CLIENT                      MON_HOST          KEY  
  587b08db...  site-b client.rbd-mirror-peer 192.168.1.1,192.168.1.2 AQAeuZdb...
```

20.4.1.5 Remove cluster peer

To remove a mirroring peer cluster, specify the **mirror pool peer remove** subcommand, the pool name, and the peer UUID (available from the **rbd mirror pool info** command):

```
cephuser@adm > rbd --cluster local mirror pool peer remove POOL_NAME \
55672766-c02b-4729-8567-f13a66893445
cephuser@adm > rbd --cluster remote mirror pool peer remove POOL_NAME \
60c0e299-b38f-4234-91f6-eed0a367be08
```

20.4.1.6 Data pools

When creating images in the destination cluster, **rbd-mirror** selects a data pool as follows:

- If the destination cluster has a default data pool configured (with the **rbd_default_data_pool** configuration option), it will be used.
- Otherwise, if the source image uses a separate data pool, and a pool with the same name exists on the destination cluster, that pool will be used.
- If neither of the above is true, no data pool will be set.

20.4.2 RBD Image configuration

Unlike pool configuration, image configuration only needs to be performed against a single mirroring peer Ceph cluster.

Mirrored RBD images are designated as either *primary* or *non-primary*. This is a property of the image and not the pool. Images that are designated as non-primary cannot be modified.

Images are automatically promoted to primary when mirroring is first enabled on an image (either implicitly if the pool mirror mode was 'pool' and the image has the journaling image feature enabled, or explicitly (see [Section 20.4.2.1, "Enabling image mirroring"](#)) by the **rbd** command).

20.4.2.1 Enabling image mirroring

If mirroring is configured in the **image** mode, then it is necessary to explicitly enable mirroring for each image within the pool. To enable mirroring for a specific image with **rbd**, specify the **mirror image enable** subcommand along with the pool and image name:

```
cephuser@adm > rbd --cluster local mirror image enable \
```

`POOL_NAME/IMAGE_NAME`

The mirror image mode can either be `journal` or `snapshot`:

journal (default)

When configured in `journal` mode, mirroring will use the RBD journaling image feature to replicate the image contents. If the RBD journaling image feature is not yet enabled on the image, it will be automatically enabled.

snapshot

When configured in `snapshot` mode, mirroring will use RBD image mirror-snapshots to replicate the image contents. Once enabled, an initial mirror-snapshot will automatically be created. Additional RBD image mirror-snapshots can be created by the `rbd` command.

For example:

```
cephuser@adm > rbd --cluster local mirror image enable image-pool/image-1 snapshot
cephuser@adm > rbd --cluster local mirror image enable image-pool/image-2 journal
```

20.4.2.2 Enabling the image journaling feature

RBD mirroring uses the RBD journaling feature to ensure that the replicated image always remains crash-consistent. When using the `image` mirroring mode, the journaling feature will be automatically enabled if mirroring is enabled on the image. When using the `pool` mirroring mode, before an image can be mirrored to a peer cluster, the RBD image journaling feature must be enabled. The feature can be enabled at image creation time by providing the `--image-feature exclusive-lock,journaling` option to the `rbd` command.

Alternatively, the journaling feature can be dynamically enabled on pre-existing RBD images. To enable journaling, specify the `feature enable` subcommand, the pool and image name, and the feature name:

```
cephuser@adm > rbd --cluster local feature enable POOL_NAME/IMAGE_NAME exclusive-lock
cephuser@adm > rbd --cluster local feature enable POOL_NAME/IMAGE_NAME journaling
```



Note: Option dependency

The `journaling` feature is dependent on the `exclusive-lock` feature. If the `exclusive-lock` feature is not already enabled, you need to enable it prior to enabling the `journaling` feature.



Tip

You can enable journaling on all new images by default by adding `rd default features = layering,exclusive-lock,object-map,deep-flatten,journaling` to your Ceph configuration file.

20.4.2.3 Creating image mirror-snapshots

When using snapshot-based mirroring, mirror-snapshots will need to be created whenever it is desired to mirror the changed contents of the RBD image. To create a mirror-snapshot manually with `rbd`, specify the `mirror image snapshot` command along with the pool and image name:

```
cephuser@adm > rbd mirror image snapshot POOL_NAME/IMAGE_NAME
```

For example:

```
cephuser@adm > rbd --cluster local mirror image snapshot image-pool/image-1
```

By default only three mirror-snapshots will be created per image. The most recent mirror-snapshot is automatically pruned if the limit is reached. The limit can be overridden via the `rbd mirroring_max_mirroring_snapshots` configuration option if required. Additionally, mirror-snapshots are automatically deleted when the image is removed or when mirroring is disabled.

Mirror-snapshots can also be automatically created on a periodic basis if mirror-snapshot schedules are defined. The mirror-snapshot can be scheduled globally, per-pool, or per-image levels. Multiple mirror-snapshot schedules can be defined at any level, but only the most-specific snapshot schedules that match an individual mirrored image will run.

To create a mirror-snapshot schedule with `rbd`, specify the `mirror snapshot schedule add` command along with an optional pool or image name, interval, and optional start time.

The interval can be specified in days, hours, or minutes using the suffixes `d`, `h`, or `m` respectively. The optional start time can be specified using the ISO 8601 time format. For example:

```
cephuser@adm > rbd --cluster local mirror snapshot schedule add --pool image-pool 24h
14:00:00-05:00
cephuser@adm > rbd --cluster local mirror snapshot schedule add --pool image-pool --image
image1 6h
```

To remove a mirror-snapshot schedule with `rbd`, specify the `mirror snapshot schedule remove` command with options that match the corresponding add schedule command.

To list all snapshot schedules for a specific level (global, pool, or image) with **rbd**, specify the **mirror snapshot schedule ls** command along with an optional pool or image name. Additionally, the **--recursive** option can be specified to list all schedules at the specified level and below. For example:

```
cephuser@adm > rbd --cluster local mirror schedule ls --pool image-pool --recursive
POOL      NAMESPACE IMAGE  SCHEDULE
image-pool -      -      every 1d starting at 14:00:00-05:00
image-pool      image1 every 6h
```

To find out when the next snapshots will be created for snapshot-based mirroring RBD images with **rbd**, specify the **mirror snapshot schedule status** command along with an optional pool or image name. For example:

```
cephuser@adm > rbd --cluster local mirror schedule status
SCHEDULE TIME      IMAGE
2020-02-26 18:00:00 image-pool/image1
```

20.4.2.4 Disabling image mirroring

To disable mirroring for a specific image, specify the **mirror image disable** subcommand along with the pool and image name:

```
cephuser@adm > rbd --cluster local mirror image disable POOL_NAME/IMAGE_NAME
```

20.4.2.5 Promoting and demoting images

In a failover scenario where the primary designation needs to be moved to the image in the peer cluster, you need to stop access to the primary image, demote the current primary image, promote the new primary image, and resume access to the image on the alternate cluster.



Note: Forced promotion

Promotion can be forced using the **--force** option. Forced promotion is needed when the demotion cannot be propagated to the peer cluster (for example, in case of cluster failure or communication outage). This will result in a split-brain scenario between the two peers, and the image will no longer be synchronized until a **resync** subcommand is issued.

To demote a specific image to non-primary, specify the **mirror image demote** subcommand along with the pool and image name:

```
cephuser@adm > rbd --cluster local mirror image demote POOL_NAME/IMAGE_NAME
```

To demote all primary images within a pool to non-primary, specify the **mirror pool demote** subcommand along with the pool name:

```
cephuser@adm > rbd --cluster local mirror pool demote POOL_NAME
```

To promote a specific image to primary, specify the **mirror image promote** subcommand along with the pool and image name:

```
cephuser@adm > rbd --cluster remote mirror image promote POOL_NAME/IMAGE_NAME
```

To promote all non-primary images within a pool to primary, specify the **mirror pool promote** subcommand along with the pool name:

```
cephuser@adm > rbd --cluster local mirror pool promote POOL_NAME
```



Tip: Split I/O load

Since the primary or non-primary status is per-image, it is possible to have two clusters split the I/O load and stage failover or failback.

20.4.2.6 Forcing image resync

If a split-brain event is detected by the **rbd-mirror** daemon, it will not attempt to mirror the affected image until corrected. To resume mirroring for an image, first demote the image determined to be out of date and then request a resync to the primary image. To request an image resync, specify the **mirror image resync** subcommand along with the pool and image name:

```
cephuser@adm > rbd mirror image resync POOL_NAME/IMAGE_NAME
```

20.4.3 Checking the mirror status

The peer cluster replication status is stored for every primary mirrored image. This status can be retrieved using the `mirror image status` and `mirror pool status` subcommands:

To request the mirror image status, specify the `mirror image status` subcommand along with the pool and image name:

```
cephuser@adm > rbd mirror image status POOL_NAME/IMAGE_NAME
```

To request the mirror pool summary status, specify the `mirror pool status` subcommand along with the pool name:

```
cephuser@adm > rbd mirror pool status POOL_NAME
```



Tip:

Adding the `--verbose` option to the `mirror pool status` subcommand will additionally output status details for every mirroring image in the pool.

20.5 Cache settings

The user space implementation of the Ceph block device (`librbd`) cannot take advantage of the Linux page cache. Therefore, it includes its own in-memory caching. RBD caching behaves similar to hard disk caching. When the OS sends a barrier or a flush request, all 'dirty' data is written to the OSDs. This means that using write-back caching is just as safe as using a well-behaved physical hard disk with a VM that properly sends flushes. The cache uses a *Least Recently Used* (LRU) algorithm, and in write-back mode it can merge adjacent requests for better throughput. Ceph supports write-back caching for RBD. To enable it, run

```
cephuser@adm > ceph config set client rbd_cache true
```

By default, `librbd` does not perform any caching. Writes and reads go directly to the storage cluster, and writes return only when the data is on disk on all replicas. With caching enabled, writes return immediately, unless there are more unflushed bytes than set in the `rbd cache max dirty` option. In such a case, the write triggers writeback and blocks until enough bytes are flushed.

Ceph supports write-through caching for RBD. You can set the size of the cache, and you can set targets and limits to switch from write-back caching to write-through caching. To enable write-through mode, run

```
cephuser@adm > ceph config set client rbd_cache_max_dirty 0
```

This means writes return only when the data is on disk on all replicas, but reads may come from the cache. The cache is in memory on the client, and each RBD image has its own cache. Since the cache is local to the client, there is no coherency if there are others accessing the image. Running GFS or OCFS on top of RBD will not work with caching enabled.

The following parameters affect the behavior of RADOS Block Devices. To set them, use the `client` category:

```
cephuser@adm > ceph config set client PARAMETER VALUE
```

rbd cache

Enable caching for RADOS Block Device (RBD). Default is 'true'.

rbd cache size

The RBD cache size in bytes. Default is 32 MB.

rbd cache max dirty

The 'dirty' limit in bytes at which the cache triggers write-back. `rbd cache max dirty` needs to be less than `rbd cache size`. If set to 0, uses write-through caching. Default is 24 MB.

rbd cache target dirty

The 'dirty target' before the cache begins writing data to the data storage. Does not block writes to the cache. Default is 16 MB.

rbd cache max dirty age

The number of seconds dirty data is in the cache before writeback starts. Default is 1.

rbd cache writethrough until flush

Start out in write-through mode, and switch to write-back after the first flush request is received. Enabling this is a conservative but safe setting in case virtual machines running on `rbd` are too old to send flushes (for example, the virtio driver in Linux before kernel 2.6.32). Default is 'true'.

20.6 QoS settings

Generally, Quality of Service (QoS) refers to methods of traffic prioritization and resource reservation. It is particularly important for the transportation of traffic with special requirements.

Important: Not supported by iSCSI

The following QoS settings are used only by the user space RBD implementation `librbd` and *not* used by the `kRBD` implementation. Because iSCSI uses `kRBD`, it does not use the QoS settings. However, for iSCSI you can configure QoS on the kernel block device layer using standard kernel facilities.

`rbd qos iops limit`

The desired limit of I/O operations per second. Default is 0 (no limit).

`rbd qos bps limit`

The desired limit of I/O bytes per second. Default is 0 (no limit).

`rbd qos read iops limit`

The desired limit of read operations per second. Default is 0 (no limit).

`rbd qos write iops limit`

The desired limit of write operations per second. Default is 0 (no limit).

`rbd qos read bps limit`

The desired limit of read bytes per second. Default is 0 (no limit).

`rbd qos write bps limit`

The desired limit of write bytes per second. Default is 0 (no limit).

`rbd qos iops burst`

The desired burst limit of I/O operations. Default is 0 (no limit).

`rbd qos bps burst`

The desired burst limit of I/O bytes. Default is 0 (no limit).

`rbd qos read iops burst`

The desired burst limit of read operations. Default is 0 (no limit).

`rbd qos write iops burst`

The desired burst limit of write operations. Default is 0 (no limit).

rbd qos read bps burst

The desired burst limit of read bytes. Default is 0 (no limit).

rbd qos write bps burst

The desired burst limit of write bytes. Default is 0 (no limit).

rbd qos schedule tick min

The minimum schedule tick (in milliseconds) for QoS. Default is 50.

20.7 Read-ahead settings

RADOS Block Device supports read-ahead/prefetching to optimize small, sequential reads. This should normally be handled by the guest OS in the case of a virtual machine, but boot loaders may not issue efficient reads. Read-ahead is automatically disabled if caching is disabled.



Important: Not supported by iSCSI

The following read-ahead settings are used only by the user space RBD implementation librbd and *not* used by the kRBD implementation. Because iSCSI uses kRBD, it does not use the read-ahead settings. However, for iSCSI you can configure read-ahead on the kernel block device layer using standard kernel facilities.

rbd readahead trigger requests

Number of sequential read requests necessary to trigger read-ahead. Default is 10.

rbd readahead max bytes

Maximum size of a read-ahead request. If set to 0, read-ahead is disabled. Default is 512 kB.

rbd readahead disable after bytes

After this many bytes have been read from an RBD image, read-ahead is disabled for that image until it is closed. This allows the guest OS to take over read-ahead when it is booted. If set to 0, read-ahead stays enabled. Default is 50 MB.

20.8 Advanced features

RADOS Block Device supports advanced features that enhance the functionality of RBD images. You can specify the features either on the command line when creating an RBD image, or in the Ceph configuration file by using the rbd_default_features option.

You can specify the values of the `rbd_default_features` option in two ways:

- As a sum of features' internal values. Each feature has its own internal value—for example 'layering' has 1 and 'fast-diff' has 16. Therefore to activate these two feature by default, include the following:

```
rbd_default_features = 17
```

- As a comma-separated list of features. The previous example will look as follows:

```
rbd_default_features = layering,fast-diff
```



Note: Features not supported by iSCSI

RBD images with the following features will not be supported by iSCSI: `deep-flatten`, `object-map`, `journaling`, `fast-diff`, `striping`

A list of advanced RBD features follows:

layering

Layering enables you to use cloning.
Internal value is 1, default is 'yes'.

striping

Striping spreads data across multiple objects and helps with parallelism for sequential read/write workloads. It prevents single node bottlenecks for large or busy RADOS Block Devices.
Internal value is 2, default is 'yes'.

exclusive-lock

When enabled, it requires a client to get a lock on an object before making a write. Enable the exclusive lock only when a single client is accessing an image at the same time. Internal value is 4. Default is 'yes'.

object-map

Object map support depends on exclusive lock support. Block devices are thin provisioned, meaning that they only store data that actually exists. Object map support helps track which objects actually exist (have data stored on a drive). Enabling object map support speeds up I/O operations for cloning, importing and exporting a sparsely populated image, and deleting.

Internal value is 8, default is 'yes'.

fast-diff

Fast-diff support depends on object map support and exclusive lock support. It adds another property to the object map, which makes it much faster to generate diffs between snapshots of an image and the actual data usage of a snapshot.

Internal value is 16, default is 'yes'.

deep-flatten

Deep-flatten makes the **rbd flatten** (see [Section 20.3.3.6, “Flattening a cloned image”](#)) work on all the snapshots of an image, in addition to the image itself. Without it, snapshots of an image will still rely on the parent, therefore you will not be able to delete the parent image until the snapshots are deleted. Deep-flatten makes a parent independent of its clones, even if they have snapshots.

Internal value is 32, default is 'yes'.

journaling

Journaling support depends on exclusive lock support. Journaling records all modifications to an image in the order they occur. RBD mirroring (see [Section 20.4, “RBD image mirrors”](#)) uses the journal to replicate a crash consistent image to a remote cluster.

Internal value is 64, default is 'no'.

20.9 Mapping RBD using old kernel clients

Old clients (for example, SLE11 SP4) may not be able to map RBD images because a cluster deployed with SUSE Enterprise Storage 7 forces some features (both RBD image level features and RADOS level features) that these old clients do not support. When this happens, the OSD logs will show messages similar to the following:

```
2019-05-17 16:11:33.739133 7fcb83a2e700 0 -- 192.168.122.221:0/1006830 >> \  
192.168.122.152:6789/0 pipe(0x65d4e0 sd=3 :57323 s=1 pgs=0 cs=0 l=1 c=0x65d770).connect \  
protocol feature mismatch, my 2fffffffffff < peer 4010ff8ffacffff missing 401000000000000
```



Warning: Changing CRUSH Map bucket types causes massive rebalancing

If you intend to switch the CRUSH Map bucket types between 'straw' and 'straw2', do it in a planned manner. Expect a significant impact on the cluster load because changing bucket type will cause massive cluster rebalancing.

1. Disable any RBD image features that are not supported. For example:

```
cephuser@adm > rbd feature disable pool1/image1 object-map
cephuser@adm > rbd feature disable pool1/image1 exclusive-lock
```

2. Change the CRUSH Map bucket types from 'straw2' to 'straw':

- a. Save the CRUSH Map:

```
cephuser@adm > ceph osd getcrushmap -o crushmap.original
```

- b. Decompile the CRUSH Map:

```
cephuser@adm > crushtool -d crushmap.original -o crushmap.txt
```

- c. Edit the CRUSH Map and replace 'straw2' with 'straw'.

- d. Recompile the CRUSH Map:

```
cephuser@adm > crushtool -c crushmap.txt -o crushmap.new
```

- e. Set the new CRUSH Map:

```
cephuser@adm > ceph osd setcrushmap -i crushmap.new
```

20.10 Enabling block devices and Kubernetes

You can use Ceph RBD with Kubernetes v1.13 and higher through the [ceph-csi](#) driver. This driver dynamically provisions RBD images to back Kubernetes volumes, and maps these RBD images as block devices (optionally mounting a file system contained within the image) on worker nodes running pods that reference an RBD-backed volume.

To use Ceph block devices with Kubernetes, you must install and configure [ceph-csi](#) within your Kubernetes environment.

! Important

`ceph-csi` uses the RBD kernel modules by default which may not support all Ceph CRUSH tunables or RBD image features.

1. By default, Ceph block devices use the RBD pool. Create a pool for Kubernetes volume storage. Ensure your Ceph cluster is running, then create the pool:

```
cephuser@adm > ceph osd pool create kubernetes
```

2. Use the RBD tool to initialize the pool:

```
cephuser@adm > rbd pool init kubernetes
```

3. Create a new user for Kubernetes and `ceph-csi`. Execute the following and record the generated key:

```
cephuser@adm > ceph auth get-or-create client.kubernetes mon 'profile rbd' osd
'profile rbd pool=kubernetes' mgr 'profile rbd pool=kubernetes'
[client.kubernetes]
key = AqD9o0Fd6hQRChAA7fMaSZXduT3NWEqylNpmg==
```

4. `ceph-csi` requires a ConfigMap object stored in Kubernetes to define the Ceph monitor addresses for the Ceph cluster. Collect both the Ceph cluster unique fsid and the monitor addresses:

```
cephuser@adm > ceph mon dump
<...>
fsid b9127830-b0cc-4e34-aa47-9d1a2e9949a8
<...>
0: [v2:192.168.1.1:3300/0,v1:192.168.1.1:6789/0] mon.a
1: [v2:192.168.1.2:3300/0,v1:192.168.1.2:6789/0] mon.b
2: [v2:192.168.1.3:3300/0,v1:192.168.1.3:6789/0] mon.c
```

5. Generate a `csi-config-map.yaml` file similar to the example below, substituting the FSID for `clusterID`, and the monitor addresses for `monitors`:

```
kubectl@adm > cat <<EOF > csi-config-map.yaml
---
apiVersion: v1
kind: ConfigMap
data:
  config.json: |-
```

```
[
  {
    "clusterID": "b9127830-b0cc-4e34-aa47-9d1a2e9949a8",
    "monitors": [
      "192.168.1.1:6789",
      "192.168.1.2:6789",
      "192.168.1.3:6789"
    ]
  }
]
metadata:
  name: ceph-csi-config
EOF
```

- When generated, store the new ConfigMap object in Kubernetes:

```
kubectl@adm > kubectl apply -f csi-config-map.yaml
```

- ceph-csi requires the cephx credentials for communicating with the Ceph cluster. Generate a `csi-rbd-secret.yaml` file similar to the example below, using the newly-created Kubernetes user ID and cephx key:

```
kubectl@adm > cat <<EOF > csi-rbd-secret.yaml
---
apiVersion: v1
kind: Secret
metadata:
  name: csi-rbd-secret
  namespace: default
stringData:
  userID: kubernetes
  userKey: AQD9o0Fd6hQRChAAt7fMaSZXduT3NWEqylNpmg==
EOF
```

- When generated, store the new secret object in Kubernetes:

```
kubectl@adm > kubectl apply -f csi-rbd-secret.yaml
```

- Create the required ServiceAccount and RBAC ClusterRole/ClusterRoleBinding Kubernetes objects. These objects do not necessarily need to be customized for your Kubernetes environment, and therefore can be used directly from the ceph-csi deployment YAML files:

```
kubectl@adm > kubectl apply -f https://raw.githubusercontent.com/ceph/ceph-csi/master/deploy/rbd/kubernetes/csi-provisioner-rbac.yaml
```

```
kubectl@adm > kubectl apply -f https://raw.githubusercontent.com/ceph/ceph-csi/master/deploy/rbd/kubernetes/csi-nodeplugin-rbac.yaml
```

10. Create the `ceph-csi` provisioner and node plugins:

```
kubectl@adm > wget https://raw.githubusercontent.com/ceph/ceph-csi/master/deploy/rbd/kubernetes/csi-rbdplugin-provisioner.yaml
kubectl@adm > kubectl apply -f csi-rbdplugin-provisioner.yaml
kubectl@adm > wget https://raw.githubusercontent.com/ceph/ceph-csi/master/deploy/rbd/kubernetes/csi-rbdplugin.yaml
kubectl@adm > kubectl apply -f csi-rbdplugin.yaml
```



Important

By default, the provisioner and node plugin YAML files will pull the development release of the `ceph-csi` container. The YAML files should be updated to use a release version.

20.10.1 Using Ceph block devices in Kubernetes

The Kubernetes StorageClass defines a class of storage. Multiple StorageClass objects can be created to map to different quality-of-service levels and features. For example, NVMe versus HDD-based pools.

To create a `ceph-csi` StorageClass that maps to the Kubernetes pool created above, the following YAML file can be used, after ensuring that the `clusterID` property matches your Ceph cluster's FSID:

```
kubectl@adm > cat <<EOF > csi-rbd-sc.yaml
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-rbd-sc
provisioner: rbd.csi.ceph.com
parameters:
  clusterID: b9127830-b0cc-4e34-aa47-9d1a2e9949a8
  pool: kubernetes
  csi.storage.k8s.io/provisioner-secret-name: csi-rbd-secret
  csi.storage.k8s.io/provisioner-secret-namespace: default
  csi.storage.k8s.io/node-stage-secret-name: csi-rbd-secret
  csi.storage.k8s.io/node-stage-secret-namespace: default
```

```
reclaimPolicy: Delete
mountOptions:
  - discard
EOF
kubectl@adm > kubectl apply -f csi-rbd-sc.yaml
```

A PersistentVolumeClaim is a request for abstract storage resources by a user. The PersistentVolumeClaim would then be associated to a pod resource to provision a PersistentVolume, which would be backed by a Ceph block image. An optional volumeMode can be included to select between a mounted file system (default) or raw block-device-based volume.

Using ceph-csi, specifying Filesystem for volumeMode can support both ReadWriteOnce and ReadOnlyMany accessMode claims, and specifying Block for volumeMode can support ReadWriteOnce, ReadWriteMany, and ReadOnlyMany accessMode claims.

For example, to create a block-based PersistentVolumeClaim that uses the ceph-csi-based StorageClass created above, the following YAML file can be used to request raw block storage from the csi-rbd-sc StorageClass:

```
kubectl@adm > cat <<EOF > raw-block-pvc.yaml
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: raw-block-pvc
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Block
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-rbd-sc
EOF
kubectl@adm > kubectl apply -f raw-block-pvc.yaml
```

The following demonstrates an example of binding the above PersistentVolumeClaim to a pod resource as a raw block device:

```
kubectl@adm > cat <<EOF > raw-block-pod.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: pod-with-raw-block-volume
spec:
```

```

containers:
  - name: fc-container
    image: fedora:26
    command: ["/bin/sh", "-c"]
    args: ["tail -f /dev/null"]
    volumeDevices:
      - name: data
        devicePath: /dev/xvda
volumes:
  - name: data
    persistentVolumeClaim:
      claimName: raw-block-pvc
EOF
kubectl@adm > kubectl apply -f raw-block-pod.yaml

```

To create a file-system-based PersistentVolumeClaim that uses the ceph-csi-based StorageClass created above, the following YAML file can be used to request a mounted file system (backed by an RBD image) from the csi-rbd-sc StorageClass:

```

kubectl@adm > cat <<EOF > pvc.yaml
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: rbd-pvc
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-rbd-sc
EOF
kubectl@adm > kubectl apply -f pvc.yaml

```

The following demonstrates an example of binding the above PersistentVolumeClaim to a pod resource as a mounted file system:

```

kubectl@adm > cat <<EOF > pod.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: csi-rbd-demo-pod
spec:
  containers:

```

```
- name: web-server
  image: nginx
  volumeMounts:
    - name: mypvc
      mountPath: /var/lib/www/html
  volumes:
    - name: mypvc
      persistentVolumeClaim:
        claimName: rbd-pvc
        readOnly: false
EOF
kubectl@adm > kubectl apply -f pod.yaml
```

IV Accessing Cluster Data

- 21 Ceph Object Gateway **248**
- 22 Ceph iSCSI gateway **302**
- 23 Clustered file system **320**
- 24 Export Ceph data via Samba **330**
- 25 NFS Ganesha **347**

21 Ceph Object Gateway

This chapter introduces details about administration tasks related to Object Gateway, such as checking status of the service, managing accounts, multisite gateways, or LDAP authentication.

21.1 Object Gateway restrictions and naming limitations

Following is a list of important Object Gateway limits:

21.1.1 Bucket limitations

When approaching Object Gateway via the S3 API, bucket names are limited to DNS-compliant names with a dash character '-' allowed. When approaching Object Gateway via the Swift API, you may use any combination of UTF-8 supported characters except for a slash character '/'. The maximum length of a bucket name is 255 characters. Bucket names must be unique.



Tip: Use DNS-compliant bucket names

Although you may use any UTF-8 based bucket name via the Swift API, it is recommended to name buckets with regard to the S3 naming limitations to avoid problems accessing the same bucket via the S3 API.

21.1.2 Stored object limitations

Maximum number of objects per user

No restriction by default (limited by $\sim 2^{63}$).

Maximum number of objects per bucket

No restriction by default (limited by $\sim 2^{63}$).

Maximum size of an object to upload/store

Single uploads are restricted to 5 GB. Use multipart for larger object sizes. The maximum number of multipart chunks is 10000.

21.1.3 HTTP header limitations

HTTP header and request limitation depend on the Web front-end used. The default Beast restricts the size of the HTTP header to 16 kB.

21.2 Deploying the Object Gateway

The Ceph Object Gateway deployment follows the same procedure as the deployment of other Ceph services—by means of `cephadm`. For more details, refer to *Book “Deployment Guide”, Chapter 8 “Deploying the remaining core services using cephadm”, Section 8.2 “Service and placement specification”, specifically to Book “Deployment Guide”, Chapter 8 “Deploying the remaining core services using cephadm”, Section 8.3.4 “Deploying Object Gateways”.*

21.3 Operating the Object Gateway service

You can operate the Object Gateways same as other Ceph services by first identifying the service name with the `ceph orch ps` command, and running the following command for operating services, for example:

```
ceph orch daemon restart OGW_SERVICE_NAME
```

Refer to *Chapter 14, Operation of Ceph services* for complete information about operating Ceph services.

21.4 Configuration options

Refer to *Section 28.5, “Ceph Object Gateway”* for a list of Object Gateway configuration options.

21.5 Managing Object Gateway access

You can communicate with Object Gateway using either S3- or Swift-compatible interface. S3 interface is compatible with a large subset of the Amazon S3 RESTful API. Swift interface is compatible with a large subset of the OpenStack Swift API.

Both interfaces require you to create a specific user, and install the relevant client software to communicate with the gateway using the user's secret key.

21.5.1 Accessing Object Gateway

21.5.1.1 S3 interface access

To access the S3 interface, you need a REST client. `S3cmd` is a command line S3 client. You can find it in the [OpenSUSE Build Service \(https://build.opensuse.org/package/show/Cloud:Tools/s3cmd\)](https://build.opensuse.org/package/show/Cloud:Tools/s3cmd). The repository contains versions for both SUSE Linux Enterprise and openSUSE based distributions.

If you want to test your access to the S3 interface, you can also write a small Python script. The script will connect to Object Gateway, create a new bucket, and list all buckets. The values for `aws_access_key_id` and `aws_secret_access_key` are taken from the values of `access_key` and `secret_key` returned by the `radosgw_admin` command from [Section 21.5.2.1, "Adding S3 and Swift users"](#).

1. Install the `python-boto` package:

```
# zypper in python-boto
```

2. Create a new Python script called `s3test.py` with the following content:

```
import boto
import boto.s3.connection
access_key = '11BS02LGFB6AL6H1ADMW'
secret_key = 'vzCEkuryfn060dfee4fgQPqFrncKEIkh3Zcd0ANY'
conn = boto.connect_s3(
    aws_access_key_id = access_key,
    aws_secret_access_key = secret_key,
    host = 'HOSTNAME',
    is_secure=False,
    calling_format = boto.s3.connection.OrdinaryCallingFormat(),
)
bucket = conn.create_bucket('my-new-bucket')
for bucket in conn.get_all_buckets():
    print "NAME\tCREATED".format(
        name = bucket.name,
        created = bucket.creation_date,
    )
```

Replace `HOSTNAME` with the host name of the host where you configured the Object Gateway service, for example `gateway_host`.

3. Run the script:

```
python s3test.py
```

The script outputs something like the following:

```
my-new-bucket 2015-07-22T15:37:42.000Z
```

21.5.1.2 Swift interface access

To access Object Gateway via Swift interface, you need the `swift` command line client. Its manual page `man 1 swift` tells you more about its command line options.

The package is included in the 'Public Cloud' module for SUSE Linux Enterprise 12 from SP3 and SUSE Linux Enterprise 15. Before installing the package, you need to activate the module and refresh the software repository:

```
# SUSEConnect -p sle-module-public-cloud/12/SYSTEM-ARCH
sudo zypper refresh
```

Or

```
# SUSEConnect -p sle-module-public-cloud/15/SYSTEM-ARCH
# zypper refresh
```

To install the `swift` command, run the following:

```
# zypper in python-swiftclient
```

The `swift` access uses the following syntax:

```
> swift -A http://IP_ADDRESS/auth/1.0 \
-U example_user:swift -K 'SWIFT_SECRET_KEY' list
```

Replace `IP_ADDRESS` with the IP address of the gateway server, and `SWIFT_SECRET_KEY` with its value from the output of the `radosgw-admin key create` command executed for the `swift` user in [Section 21.5.2.1, "Adding S3 and Swift users"](#).

For example:

```
> swift -A http://gateway.example.com/auth/1.0 -U example_user:swift \
-K 'r5wWixj0CeE07DixD1FjTLmNYIViaC6JVhi3013h' list
```

The output is:

```
my-new-bucket
```

21.5.2 Manage S3 and Swift accounts

21.5.2.1 Adding S3 and Swift users

You need to create a user, access key and secret to enable end users to interact with the gateway. There are two types of users: a *user* and *subuser*. While *users* are used when interacting with the S3 interface, *subusers* are users of the Swift interface. Each subuser is associated to a user.

To create a Swift user, follow the steps:

1. To create a Swift user—which is a *subuser* in our terminology—you need to create the associated *user* first.

```
cephuser@adm > radosgw-admin user create --uid=USERNAME \  
--display-name="DISPLAY-NAME" --email=EMAIL
```

For example:

```
cephuser@adm > radosgw-admin user create \  
--uid=example_user \  
--display-name="Example User" \  
--email=penguin@example.com
```

2. To create a subuser (Swift interface) for the user, you must specify the user ID (`--uid= USERNAME`), a subuser ID, and the access level for the subuser.

```
cephuser@adm > radosgw-admin subuser create --uid=UID \  
--subuser=UID \  
--access=[ read | write | readwrite | full ]
```

For example:

```
cephuser@adm > radosgw-admin subuser create --uid=example_user \  
--subuser=example_user:swift --access=full
```

3. Generate a secret key for the user.

```
cephuser@adm > radosgw-admin key create \  
--gen-secret \  
--subuser=example_user:swift \  
--key-type=swift
```

- Both commands will output JSON-formatted data showing the user state. Notice the following lines, and remember the `secret_key` value:

```
"swift_keys": [
  { "user": "example_user:swift",
    "secret_key": "r5wWIXj0CeE07DixD1FjTlMNYIViaC6JVhi3013h"}],
```

When accessing Object Gateway through the S3 interface you need to create an S3 user by running:

```
cephuser@adm > radosgw-admin user create --uid=USERNAME \
--display-name="DISPLAY-NAME" --email=EMAIL
```

For example:

```
cephuser@adm > radosgw-admin user create \
--uid=example_user \
--display-name="Example User" \
--email=penguin@example.com
```

The command also creates the user's access and secret key. Check its output for `access_key` and `secret_key` keywords and their values:

```
[...]
"keys": [
  { "user": "example_user",
    "access_key": "11BS02LGFB6AL6H1ADMW",
    "secret_key": "vzCEkuryfn060dfee4fgQPqFrncKEIkh3Zcd0ANY"}],
[...]
```

21.5.2.2 Removing S3 and Swift users

The procedure for deleting users is similar for S3 and Swift users. But in case of Swift users you may need to delete the user including its subusers.

To remove a S3 or Swift user (including all its subusers), specify `user rm` and the user ID in the following command:

```
cephuser@adm > radosgw-admin user rm --uid=example_user
```

To remove a subuser, specify `subuser rm` and the subuser ID.

```
cephuser@adm > radosgw-admin subuser rm --uid=example_user:swift
```

You can make use of the following options:

--purge-data

Purges all data associated to the user ID.

--purge-keys

Purges all keys associated to the user ID.



Tip: Removing a subuser

When you remove a subuser, you are removing access to the Swift interface. The user will remain in the system.

21.5.2.3 Changing S3 and Swift user access and secret keys

The `access_key` and `secret_key` parameters identify the Object Gateway user when accessing the gateway. Changing the existing user keys is the same as creating new ones, as the old keys get overwritten.

For S3 users, run the following:

```
cephuser@adm > radosgw-admin key create --uid=EXAMPLE_USER --key-type=s3 --gen-access-key --gen-secret
```

For Swift users, run the following:

```
cephuser@adm > radosgw-admin key create --subuser=EXAMPLE_USER:swift --key-type=swift --gen-secret
```

--key-type=TYPE

Specifies the type of key. Either `swift` or `s3`.

--gen-access-key

Generates a random access key (for S3 user by default).

--gen-secret

Generates a random secret key.

--secret=KEY

Specifies a secret key, for example manually generated.

21.5.2.4 Enabling user quota management

The Ceph Object Gateway enables you to set quotas on users and buckets owned by users. Quotas include the maximum number of objects in a bucket and the maximum storage size in megabytes. Before you enable a user quota, you first need to set its parameters:

```
cephuser@adm > radosgw-admin quota set --quota-scope=user --uid=EXAMPLE_USER \  
--max-objects=1024 --max-size=1024
```

--max-objects

Specifies the maximum number of objects. A negative value disables the check.

--max-size

Specifies the maximum number of bytes. A negative value disables the check.

--quota-scope

Sets the scope for the quota. The options are bucket and user. Bucket quotas apply to buckets a user owns. User quotas apply to a user.

Once you set a user quota, you may enable it:

```
cephuser@adm > radosgw-admin quota enable --quota-scope=user --uid=EXAMPLE_USER
```

To disable a quota:

```
cephuser@adm > radosgw-admin quota disable --quota-scope=user --uid=EXAMPLE_USER
```

To list quota settings:

```
cephuser@adm > radosgw-admin user info --uid=EXAMPLE_USER
```

To update quota statistics:

```
cephuser@adm > radosgw-admin user stats --uid=EXAMPLE_USER --sync-stats
```

21.6 HTTP front-ends

The Ceph Object Gateway supports two embedded HTTP front-ends: *Beast* and *Civetweb*.

The *Beast* front-end uses the Boost.Beast library for HTTP parsing and the Boost.Asio library for asynchronous network I/O.

The *Civetweb* front-end uses the *Civetweb* HTTP library, which is a fork of *Mongoose*.

You can configure them with the `rgw_frontends` option. Refer to [Section 28.5, “Ceph Object Gateway”](#) for a list of configuration options.

21.7 Enable HTTPS/SSL for Object Gateways

To enable the Object Gateway to communicate securely using SSL, you need to either have a CA-issued certificate or create a self-signed one.

21.7.1 Creating a self-signed certificate



Tip

Skip this section if you already have a valid certificate signed by CA.

The following procedure describes how to generate a self-signed SSL certificate on the Salt Master.

1. If you need your Object Gateway to be known by additional subject identities, add them to the `subjectAltName` option in the `[v3_req]` section of the `/etc/ssl/openssl.cnf` file:

```
[...]
[ v3_req ]
subjectAltName = DNS:server1.example.com DNS:server2.example.com
[...]
```



Tip: IP addresses in `subjectAltName`

To use IP addresses instead of domain names in the `subjectAltName` option, replace the example line with the following:

```
subjectAltName = IP:10.0.0.10 IP:10.0.0.11
```

2. Create the key and the certificate using `openssl`. Enter all data you need to include in your certificate. We recommend entering the FQDN as the common name. Before signing the certificate, verify that 'X509v3 Subject Alternative Name:' is included in requested extensions, and that the resulting certificate has "X509v3 Subject Alternative Name:" set.

```
root@master # openssl req -x509 -nodes -days 1095 \  
-newkey rsa:4096 -keyout rgw.key \  
-out rgw.pem
```

3. Append the key to the certificate file:

```
root@master # cat rgw.key >> rgw.pem
```

21.7.2 Configuring Object Gateway with SSL

To configure Object Gateway to use SSL certificates, use the `rgw_frontends` option. For example:

```
cephuser@adm > ceph config set WHO rgw_frontends \  
beast ssl_port=443 ssl_certificate=config://CERT ssl_key=config://KEY
```

If you do not specify the `CERT` and `KEY` configuration keys, then the Object Gateway service will look for the SSL certificate and key under the following configuration keys:

```
rgw/cert/RGW_REALM/RGW_ZONE.key  
rgw/cert/RGW_REALM/RGW_ZONE.crt
```

If you want to override the default SSL key and certificate location, import them to the configuration database by using the following command:

```
ceph config-key set CUSTOM_CONFIG_KEY -i PATH_TO_CERT_FILE
```

Then use your custom configuration keys using the `config://` directive.

21.8 Synchronization modules

Object Gateway is deployed as a multi-site service while you can mirror data and metadata between the zones. *Synchronization modules* are built atop of the multisite framework that allows for forwarding data and metadata to a different external tier. A synchronization module allows for a set of actions to be performed whenever a change in data occurs (for example, metadata operations such as bucket or user creation). As the Object Gateway multisite changes are eventually consistent at remote sites, changes are propagated asynchronously. This covers use cases such as backing up the object storage to an external cloud cluster, a custom backup solution using tape drives, or indexing metadata in ElasticSearch.

21.8.1 Configuring synchronization modules

All synchronization modules are configured in a similar way. You need to create a new zone (refer to [Section 21.13, “Multisite Object Gateways”](#) for more details) and set its `--tier_type` option, for example `--tier-type=cloud` for the cloud synchronization module:

```
cephuser@adm > radosgw-admin zone create --rgw-zonegroup=ZONE-GROUP-NAME \  
--rgw-zone=ZONE-NAME \  
--endpoints=http://endpoint1.example.com,http://endpoint2.example.com, [...] \  
--tier-type=cloud
```

You can configure the specific tier by using the following command:

```
cephuser@adm > radosgw-admin zone modify --rgw-zonegroup=ZONE-GROUP-NAME \  
--rgw-zone=ZONE-NAME \  
--tier-config=KEY1=VALUE1,KEY2=VALUE2
```

The `KEY` in the configuration specifies the configuration variable that you want to update, and the `VALUE` specifies its new value. Nested values can be accessed using period. For example:

```
cephuser@adm > radosgw-admin zone modify --rgw-zonegroup=ZONE-GROUP-NAME \  
--rgw-zone=ZONE-NAME \  
--tier-config=connection.access_key=KEY,connection.secret=SECRET
```

You can access array entries by appending square brackets '['] with the referenced entry. You can add a new array entry by using square brackets '[']. Index value of -1 references the last entry in the array. It is not possible to create a new entry and reference it again in the same command. For example, a command to create a new profile for buckets starting with `PREFIX` follows:

```
cephuser@adm > radosgw-admin zone modify --rgw-zonegroup=ZONE-GROUP-NAME \  
--rgw-zone=ZONE-NAME \  
--tier-config=profiles[].source_bucket=PREFIX'*'  
cephuser@adm > radosgw-admin zone modify --rgw-zonegroup=ZONE-GROUP-NAME \  
--rgw-zone=ZONE-NAME \  
--tier-config=profiles[-1].connection_id=CONNECTION_ID,profiles[-1].acls_id=ACLS_ID
```



Tip: Adding and removing configuration entries

You can add a new tier configuration entry by using the `--tier-config-add=KEY=VALUE` parameter.

You can remove an existing entry by using `--tier-config-rm=KEY`.

21.8.2 Synchronizing zones

A synchronization module configuration is local to a zone. The synchronization module determines whether the zone exports data or can only consume data that was modified in another zone. As of Luminous the supported synchronization plug-ins are `ElasticSearch`, `rgw`, which is the default synchronization plug-in that synchronizes data between the zones and `log` which is a trivial synchronization plug-in that logs the metadata operation that happens in the remote zones. The following sections are written with the example of a zone using `ElasticSearch` synchronization module. The process would be similar for configuring any other synchronization plug-in.



Note: Default synchronization plug-in

`rgw` is the default synchronization plug-in and there is no need to explicitly configure this.

21.8.2.1 Requirements and assumptions

Let us assume a simple multisite configuration as described in [Section 21.13, “Multisite Object Gateways”](#) consists of 2 zones: `us-east` and `us-west`. Now we add a third zone `us-east-es` which is a zone that only processes metadata from the other sites. This zone can be in the same or a different Ceph cluster than `us-east`. This zone would only consume metadata from other zones and Object Gateways in this zone will not serve any end user requests directly.

21.8.2.2 Configuring zones

1. Create the third zone similar to the ones described in [Section 21.13, “Multisite Object Gateways”](#), for example

```
cephuser@adm > radosgw-admin zone create --rgw-zonegroup=us --rgw-zone=us-east-es \  
--access-key=SYSTEM-KEY --secret=SECRET --endpoints=http://rgw-es:80
```

2. A synchronization module can be configured for this zone via the following:

```
cephuser@adm > radosgw-admin zone modify --rgw-zone=ZONE-NAME --tier-type=TIER-TYPE \  
\   
--tier-config={set of key=value pairs}
```

3. For example in the `ElasticSearch` synchronization module

```
cephuser@adm > radosgw-admin zone modify --rgw-zone=ZONE-NAME --tier-  
type=elasticsearch \  
--tier-config=endpoint=http://localhost:9200,num_shards=10,num_replicas=1
```

For the various supported tier-config options refer to [Section 21.8.3, “ElasticSearch synchronization module”](#).

4. Finally update the period

```
cephuser@adm > radosgw-admin period update --commit
```

5. Now start the Object Gateway in the zone

```
cephuser@adm > ceph orch start rgw.REALM-NAME.ZONE-NAME
```

21.8.3 ElasticSearch synchronization module

This synchronization module writes the metadata from other zones to ElasticSearch. As of Luminous this is JSON of data fields we currently store in ElasticSearch.

```
{  
  "_index" : "rgw-gold-ee5863d6",  
  "_type" : "object",  
  "_id" : "34137443-8592-48d9-8ca7-160255d52ade.34137.1:object1:null",  
  "_score" : 1.0,  
  "_source" : {  
    "bucket" : "testbucket123",  
    "name" : "object1",  
    "instance" : "null",  
    "versioned_epoch" : 0,  
    "owner" : {  
      "id" : "user1",  
      "display_name" : "user1"  
    },  
  },  
  "permissions" : [  
    "user1"  
  ],  
  "meta" : {  
    "size" : 712354,  
    "mtime" : "2017-05-04T12:54:16.462Z",  
    "etag" : "7ac66c0f148de9519b8bd264312c4d64"  
  }  
}
```

```
}
```

21.8.3.1 Elasticsearch tier type configuration parameters

endpoint

Specifies the Elasticsearch server endpoint to access.

num_shards

(integer) The number of shards that Elasticsearch will be configured with on data synchronization initialization. Note that this cannot be changed after initialization. Any change here requires rebuild of the Elasticsearch index and reinitialization of the data synchronization process.

num_replicas

(integer) The number of replicas that Elasticsearch will be configured with on data synchronization initialization.

explicit_custom_meta

(true | false) Specifies whether all user custom metadata will be indexed, or whether user will need to configure (at the bucket level) what customer metadata entries should be indexed. This is false by default

index_buckets_list

(comma separated list of strings) If empty, all buckets will be indexed. Otherwise, only buckets specified here will be indexed. It is possible to provide bucket prefixes (for example 'foo*'), or bucket suffixes (for example '*bar').

approved_owners_list

(comma separated list of strings) If empty, buckets of all owners will be indexed (subject to other restrictions), otherwise, only buckets owned by specified owners will be indexed. Suffixes and prefixes can also be provided.

override_index_path

(string) if not empty, this string will be used as the Elasticsearch index path. Otherwise the index path will be determined and generated on synchronization initialization.

username

Specifies a user name for Elasticsearch if authentication is required.

password

Specifies a password for Elasticsearch if authentication is required.

21.8.3.2 Metadata queries

Since the ElasticSearch cluster now stores object metadata, it is important that the ElasticSearch endpoint is not exposed to the public and only accessible to the cluster administrators. For exposing metadata queries to the end user itself this poses a problem since we'd want the user to only query their metadata and not of any other users, this would require the ElasticSearch cluster to authenticate users in a way similar to RGW does which poses a problem.

As of Luminous RGW in the metadata master zone can now service end user requests. This allows for not exposing the ElasticSearch endpoint in public and also solves the authentication and authorization problem since RGW itself can authenticate the end user requests. For this purpose RGW introduces a new query in the bucket APIs that can service ElasticSearch requests. All these requests must be sent to the metadata master zone.

Get an ElasticSearch Query

```
GET /BUCKET?query=QUERY-EXPR
```

request params:

- max-keys: max number of entries to return
- marker: pagination marker

```
expression := [(]<arg> <op> <value> [)][<and|or> ...]
```

op is one of the following: <, <=, =, >=, >

For example:

```
GET /?query=name==foo
```

Will return all the indexed keys that user has read permission to, and are named 'foo'. The output will be a list of keys in XML that is similar to the S3 list buckets response.

Configure custom metadata fields

Define which custom metadata entries should be indexed (under the specified bucket), and what are the types of these keys. If explicit custom metadata indexing is configured, this is needed so that rgw will index the specified custom metadata values. Otherwise it is needed in cases where the indexed metadata keys are of a type other than string.

```
POST /BUCKET?mdsearch
x-amz-meta-search: <key [; type]> [, ...]
```

Multiple metadata fields must be comma separated, a type can be forced for a field with a `;`. The currently allowed types are string(default), integer and date, for example, if you want to index a custom object metadata x-amz-meta-year as int, x-amz-meta-date as type date and x-amz-meta-title as string, you would do

```
POST /mybooks?mdsearch
x-amz-meta-search: x-amz-meta-year;int, x-amz-meta-release-date;date, x-amz-meta-
title;string
```

Delete custom metadata configuration

Delete custom metadata bucket configuration.

```
DELETE /BUCKET?mdsearch
```

Get custom metadata configuration

Retrieve custom metadata bucket configuration.

```
GET /BUCKET?mdsearch
```

21.8.4 Cloud synchronization module

This section introduces a module that synchronizes the zone data to a remote cloud service. The synchronization is only unidirectional—the date is not synchronized back from the remote zone. The main goal of this module is to enable synchronizing data to multiple cloud service providers. Currently it supports cloud providers that are compatible with AWS (S3).

To synchronize data to a remote cloud service, you need to configure user credentials. Because many cloud services introduce limits on the number of buckets that each user can create, you can configure the mapping of source objects and buckets, different targets to different buckets and bucket prefixes. Note that source access lists (ACLs) will not be preserved. It is possible to map permissions of specific source users to specific destination users.

Because of API limitations, there is no way to preserve original object modification time and HTTP entity tag (ETag). The cloud synchronization module stores these as metadata attributes on the destination objects.

21.8.4.1 Configuring the cloud synchronization module

Following are examples of a trivial and non-trivial configuration for the cloud synchronization module. Note that the trivial configuration can collide with the non-trivial one.

EXAMPLE 21.1: TRIVIAL CONFIGURATION

```
{
  "connection": {
    "access_key": ACCESS,
    "secret": SECRET,
    "endpoint": ENDPOINT,
    "host_style": path | virtual,
  },
  "acls": [ { "type": id | email | uri,
    "source_id": SOURCE_ID,
    "dest_id": DEST_ID } ... ],
  "target_path": TARGET_PATH,
}
```

EXAMPLE 21.2: NON-TRIVIAL CONFIGURATION

```
{
  "default": {
    "connection": {
      "access_key": ACCESS,
      "secret": SECRET,
      "endpoint": ENDPOINT,
      "host_style" path | virtual,
    },
    "acls": [
      {
        "type": id | email | uri, # optional, default is id
        "source_id": ID,
        "dest_id": ID
      } ... ]
    "target_path": PATH # optional
  },
  "connections": [
    {
      "connection_id": ID,
      "access_key": ACCESS,
      "secret": SECRET,
      "endpoint": ENDPOINT,
      "host_style": path | virtual, # optional
    } ... ],
  "acl_profiles": [
    {
      "acls_id": ID, # acl mappings
      "acls": [ {
        "type": id | email | uri,
        "source_id": ID,
        "dest_id": ID
      } ... ]
    } ... ]
}
```

```

    } ... ]
  }
],
"profiles": [
{
  "source_bucket": SOURCE,
  "connection_id": CONNECTION_ID,
  "acls_id": MAPPINGS_ID,
  "target_path": DEST,          # optional
} ... ],
}

```

Explanation of used configuration terms follows:

connection

Represents a connection to the remote cloud service. Contains 'connection_id', 'access_key', 'secret', 'endpoint', and 'host_style'.

access_key

The remote cloud access key that will be used for the specific connection.

secret

The secret key for the remote cloud service.

endpoint

URL of remote cloud service endpoint.

host_style

Type of host style ('path' or 'virtual') to be used when accessing remote cloud endpoint. Default is 'path'.

acls

Array of access list mappings.

acl_mapping

Each 'acl_mapping' structure contains 'type', 'source_id', and 'dest_id'. These will define the ACL mutation for each object. An ACL mutation allows converting source user ID to a destination ID.

type

ACL type: 'id' defines user ID, 'email' defines user by email, and 'uri' defines user by uri (group).

source_id

ID of user in the source zone.

dest_id

ID of user in the destination.

target_path

A string that defines how the target path is created. The target path specifies a prefix to which the source object name is appended. The target path configurable can include any of the following variables:

SID

A unique string that represents the synchronization instance ID.

ZONEGROUP

Zonegroup name.

ZONEGROUP_ID

Zonegroup ID.

ZONE

Zone name.

ZONE_ID

Zone ID.

BUCKET

Source bucket name.

OWNER

Source bucket owner ID.

For example: `target_path = rgwx-ZONE - SID / OWNER / BUCKET`

acl_profiles

An array of access list profiles.

acl_profile

Each profile contains 'acls_id' that represents the profile, and an 'acls' array that holds a list of 'acl_mappings'.

profiles

A list of profiles. Each profile contains the following:

source_bucket

Either a bucket name, or a bucket prefix (if ends with *) that defines the source bucket(s) for this profile.

target_path

See above for the explanation.

connection_id

ID of the connection that will be used for this profile.

acls_id

ID of ACL's profile that will be used for this profile.

21.8.4.2 S3 specific configurables

The cloud synchronization module will only work with back-ends that are compatible with AWS S3. There are a few configurables that can be used to tweak its behavior when accessing S3 cloud services:

```
{
  "multipart_sync_threshold": OBJECT_SIZE,
  "multipart_min_part_size": PART_SIZE
}
```

multipart_sync_threshold

Objects whose size is equal to or larger than this value will be synchronized with the cloud service using multipart upload.

multipart_min_part_size

Minimum parts size to use when synchronizing objects using multipart upload.

21.8.5 Archive synchronization module

The *archive sync module* uses the versioning feature of S3 objects in Object Gateway. You can configure an *archive zone* that captures the different versions of S3 objects as they occur over time in other zones. The history of versions that the archive zone keeps can only be eliminated via gateways associated with the archive zone.

With such an architecture, several non-versioned zones can mirror their data and metadata via their zone gateways providing high availability to the end users, while the archive zone captures all the data updates to consolidate them as versions of S3 objects.

By including the archive zone in a multi-zone configuration, you gain the flexibility of an S3 object history in one zone while saving the space that the replicas of the versioned S3 objects would consume in the remaining zones.

21.8.5.1 Configuring the archive synchronization module



Tip: More information

Refer to [Section 21.13, “Multisite Object Gateways”](#) for details on configuring multisite gateways.

Refer to [Section 21.8, “Synchronization modules”](#) for details on configuring synchronization modules.

To use the archive sync module, you need to create a new zone whose tier type is set to `archive`:

```
cephuser@adm > radosgw-admin zone create --rgw-zonegroup=ZONE_GROUP_NAME \  
--rgw-zone=OGW_ZONE_NAME \  
--endpoints=http://OGW_ENDPOINT1_URL[,http://OGW_ENDPOINT2_URL,...] \  
--tier-type=archive
```

21.9 LDAP authentication

Apart from the default local user authentication, Object Gateway can use LDAP server services to authenticate users as well.

21.9.1 Authentication mechanism

The Object Gateway extracts the user's LDAP credentials from a token. A search filter is constructed from the user name. The Object Gateway uses the configured service account to search the directory for a matching entry. If an entry is found, the Object Gateway attempts to bind to the found distinguished name with the password from the token. If the credentials are valid, the bind will succeed, and the Object Gateway grants access.

You can limit the allowed users by setting the base for the search to a specific organizational unit or by specifying a custom search filter, for example requiring specific group membership, custom object classes, or attributes.

21.9.2 Requirements

- *LDAP or Active Directory*: A running LDAP instance accessible by the Object Gateway.
- *Service account*: LDAP credentials to be used by the Object Gateway with search permissions.
- *User account*: At least one user account in the LDAP directory.



Important: Do not overlap LDAP and local users

You should not use the same user names for local users and for users being authenticated by using LDAP. The Object Gateway cannot distinguish them and it treats them as the same user.



Tip: Sanity checks

Use the `ldapsearch` utility to verify the service account or the LDAP connection. For example:

```
> ldapsearch -x -D "uid=ceph,ou=system,dc=example,dc=com" -W \  
-H ldaps://example.com -b "ou=users,dc=example,dc=com" 'uid=*' dn
```

Make sure to use the same LDAP parameters as in the Ceph configuration file to eliminate possible problems.

21.9.3 Configuring Object Gateway to use LDAP authentication

The following parameters are related to the LDAP authentication:

`rgw_s3_auth_use_ldap`

Set this option to `true` to enable S3 authentication with LDAP.

rgw_ldap_uri

Specifies the LDAP server to use. Make sure to use the `ldaps://FQDN:PORT` parameter to avoid transmitting the plain text credentials openly.

rgw_ldap_binddn

The Distinguished Name (DN) of the service account used by the Object Gateway.

rgw_ldap_secret

The password for the service account.

rgw_ldap_searchdn

Specifies the base in the directory information tree for searching users. This might be your users organizational unit or some more specific Organizational Unit (OU).

rgw_ldap_dnattr

The attribute being used in the constructed search filter to match a user name. Depending on your Directory Information Tree (DIT) this would probably be `uid` or `cn`.

rgw_search_filter

If not specified, the Object Gateway automatically constructs the search filter with the `rgw_ldap_dnattr` setting. Use this parameter to narrow the list of allowed users in very flexible ways. Consult [Section 21.9.4, "Using a custom search filter to limit user access"](#) for details.

21.9.4 Using a custom search filter to limit user access

There are two ways you can use the `rgw_search_filter` parameter.

21.9.4.1 Partial filter to further limit the constructed search filter

An example of a partial filter:

```
"objectclass=inetorgperson"
```

The Object Gateway will generate the search filter as usual with the user name from the token and the value of `rgw_ldap_dnattr`. The constructed filter is then combined with the partial filter from the `rgw_search_filter` attribute. Depending on the user name and the settings the final search filter may become:

```
"(&(uid=hari)(objectclass=inetorgperson))"
```

In that case, user 'hari' will only be granted access if he is found in the LDAP directory, has an object class of 'inetorgperson', and did specify a valid password.

21.9.4.2 Complete filter

A complete filter must contain a `USERNAME` token which will be substituted with the user name during the authentication attempt. The `rgw_ldap_dnattr` parameter is not used anymore in this case. For example, to limit valid users to a specific group, use the following filter:

```
" (&(uid=USERNAME)(memberOf=cn=ceph-users,ou=groups,dc=mycompany,dc=com)) "
```



Note: `memberOf` attribute

Using the `memberOf` attribute in LDAP searches requires server side support from you specific LDAP server implementation.

21.9.5 Generating an access token for LDAP authentication

The `radosgw-token` utility generates the access token based on the LDAP user name and password. It outputs a base-64 encoded string which is the actual access token. Use your favorite S3 client (refer to [Section 21.5.1, "Accessing Object Gateway"](#)) and specify the token as the access key and use an empty secret key.

```
> export RGW_ACCESS_KEY_ID="USERNAME"
> export RGW_SECRET_ACCESS_KEY="PASSWORD"
cephuser@adm > radosgw-token --encode --ttype=ldap
```



Important: Clear text credentials

The access token is a base-64 encoded JSON structure and contains the LDAP credentials as a clear text.



Note: Active Directory

For Active Directory, use the `--ttype=ad` parameter.

21.10 Bucket index sharding

The Object Gateway stores bucket index data in an index pool, which defaults to `.rgw.buckets.index`. If you put too many (hundreds of thousands) objects into a single bucket and the quota for maximum number of objects per bucket (`rgw bucket default quota max objects`) is not set, the performance of the index pool may degrade. *Bucket index sharding* prevents such performance decreases and allows a high number of objects per bucket.

21.10.1 Bucket index resharding

If a bucket has grown large and its initial configuration is not sufficient anymore, the bucket's index pool needs to be resharded. You can either use automatic online bucket index resharding (refer to [Section 21.10.1.1, "Dynamic resharding"](#)), or reshard the bucket index offline manually (refer to [Section 21.10.1.2, "Resharding manually"](#)).

21.10.1.1 Dynamic resharding

From SUSE Enterprise Storage 5, we support online bucket resharding. This detects if the number of objects per bucket reaches a certain threshold, and automatically increases the number of shards used by the bucket index. This process reduces the number of entries in each bucket index shard.

The detection process runs:

- When new objects are added to the bucket.
- In a background process that periodically scans all the buckets. This is needed in order to deal with existing buckets that are not being updated.

A bucket that requires resharding is added to the `reshard_log` queue and will be scheduled to be resharded later. The reshard threads run in the background and execute the scheduled resharding, one at a time.

CONFIGURING DYNAMIC RESHARDING

`rgw_dynamic_resharding`

Enables or disables dynamic bucket index resharding. Possible values are 'true' or 'false'. Defaults to 'true'.

rgw_reshard_num_logs

Number of shards for the resharding log. Defaults to 16.

rgw_reshard_bucket_lock_duration

Duration of lock on the bucket object during resharding. Defaults to 120 seconds.

rgw_max_objs_per_shard

Maximum number of objects per bucket index shard. Defaults to 100000 objects.

rgw_reshard_thread_interval

Maximum time between rounds of reshard thread processing. Defaults to 600 seconds.

COMMANDS TO ADMINISTER THE RESHARDING PROCESS

Add a bucket to the resharding queue:

```
cephuser@adm > radosgw-admin reshard add \  
--bucket BUCKET_NAME \  
--num-shards NEW_NUMBER_OF_SHARDS
```

List resharding queue:

```
cephuser@adm > radosgw-admin reshard list
```

Process/schedule a bucket resharding:

```
cephuser@adm > radosgw-admin reshard process
```

Display the bucket resharding status:

```
cephuser@adm > radosgw-admin reshard status --bucket BUCKET_NAME
```

Cancel pending bucket resharding:

```
cephuser@adm > radosgw-admin reshard cancel --bucket BUCKET_NAME
```

21.10.1.2 Resharding manually

Dynamic resharding as mentioned in [Section 21.10.1.1, “Dynamic resharding”](#) is supported only for simple Object Gateway configurations. For multisite configurations, use manual resharding as described in this section.

To reshard the bucket index manually offline, use the following command:

```
cephuser@adm > radosgw-admin bucket reshard
```

The **bucket reshard** command performs the following:

- Creates a new set of bucket index objects for the specified object.
- Spreads all entries of these index objects.
- Creates a new bucket instance.
- Links the new bucket instance with the bucket so that all new index operations go through the new bucket indexes.
- Prints the old and the new bucket ID to the standard output.



Tip

When choosing a number of shards, note the following: aim for no more than 100000 entries per shard. Bucket index shards that are prime numbers tend to work better in evenly distributing bucket index entries across the shards. For example, 503 bucket index shards is better than 500 since the former is prime.



Warning

Multi-site configurations do not support resharding a bucket index.

For multi-site configurations, resharding a bucket index requires resynchronizing all data from the master zone to all slave zones. Depending on the bucket size, this can take a considerable amount of time and resources.

PROCEDURE 21.1: RESHARDING THE BUCKET INDEX

1. Make sure that all operations to the bucket are stopped.
2. Back up the original bucket index:

```
cephuser@adm > radosgw-admin bi list \  
  --bucket=BUCKET_NAME \  
  > BUCKET_NAME.list.backup
```

3. Reshard the bucket index:

```
cephuser@adm > radosgw-admin bucket reshard \  
--bucket=BUCKET_NAME \  
--num-shards=NEW_SHARDS_NUMBER
```



Tip: Old bucket ID

As part of its output, this command also prints the new and the old bucket ID.

21.10.2 Bucket index sharding for new buckets

There are two options that affect bucket index sharding:

- Use the `rgw_override_bucket_index_max_shards` option for simple configurations.
- Use the `bucket_index_max_shards` option for multisite configurations.

Setting the options to `0` disables bucket index sharding. A value greater than `0` enables bucket index sharding and sets the maximum number of shards.

The following formula helps you calculate the recommended number of shards:

```
number_of_objects_expected_in_a_bucket / 100000
```

Be aware that the maximum number of shards is 7877.

21.10.2.1 Multisite configurations

Multisite configurations can have a different index pool to manage failover. To configure a consistent shard count for zones in one zone group, set the `bucket_index_max_shards` option in the zone group's configuration:

1. Export the zonegroup configuration to the `zonegroup.json` file:

```
cephuser@adm > radosgw-admin zonegroup get > zonegroup.json
```

2. Edit the `zonegroup.json` file and set the `bucket_index_max_shards` option for each named zone.

3. Reset the zonegroup:

```
cephuser@adm > radosgw-admin zonegroup set < zonegroup.json
```

4. Update the period. See [Section 21.13.3.6, "Update the period"](#).

21.11 OpenStack Keystone integration

OpenStack Keystone is an identity service for the OpenStack product. You can integrate the Object Gateway with Keystone to set up a gateway that accepts a Keystone authentication token. A user authorized by Keystone to access the gateway will be verified on the Ceph Object Gateway side and automatically created if needed. The Object Gateway queries Keystone periodically for a list of revoked tokens.

21.11.1 Configuring OpenStack

Before configuring the Ceph Object Gateway, you need to configure the OpenStack Keystone to enable the Swift service and point it to the Ceph Object Gateway:

1. *Set the Swift service.* To use OpenStack to validate Swift users, first create the Swift service:

```
> openstack service create \  
  --name=swift \  
  --description="Swift Service" \  
  object-store
```

2. *Set the endpoints.* After you create the Swift service, point to the Ceph Object Gateway. Replace `REGION_NAME` with the name of the gateway's zonegroup name or region name.

```
> openstack endpoint create --region REGION_NAME \  
  --publicurl "http://radosgw.example.com:8080/swift/v1" \  
  --adminurl "http://radosgw.example.com:8080/swift/v1" \  
  --internalurl "http://radosgw.example.com:8080/swift/v1" \  
  swift
```

3. *Verify the settings.* After you create the Swift service and set the endpoints, show the endpoints to verify that all the settings are correct.

```
> openstack endpoint show object-store
```

21.11.2 Configuring the Ceph Object Gateway

21.11.2.1 Configure SSL certificates

The Ceph Object Gateway queries Keystone periodically for a list of revoked tokens. These requests are encoded and signed. Keystone may be also configured to provide self-signed tokens, which are also encoded and signed. You need to configure the gateway so that it can decode and verify these signed messages. Therefore, the OpenSSL certificates that Keystone uses to create the requests need to be converted to the 'nss db' format:

```
# mkdir /var/ceph/nss
# openssl x509 -in /etc/keystone/ssl/certs/ca.pem \
  -pubkey | certutil -d /var/ceph/nss -A -n ca -t "TCu,Cu,Tuw"
rootopenssl x509 -in /etc/keystone/ssl/certs/signing_cert.pem \
  -pubkey | certutil -A -d /var/ceph/nss -n signing_cert -t "P,P,P"
```

To allow Ceph Object Gateway to interact with OpenStack Keystone, OpenStack Keystone can use a self-signed SSL certificate. Either install Keystone's SSL certificate on the node running the Ceph Object Gateway, or alternatively set the value of the option `rgw keystone verify ssl` to 'false'. Setting `rgw keystone verify ssl` to 'false' means that the gateway will not attempt to verify the certificate.

21.11.2.2 Configure the Object Gateway's options

You can configure Keystone integration using the following options:

`rgw keystone api version`

Version of the Keystone API. Valid options are 2 or 3. Defaults to 2.

`rgw keystone url`

The URL and port number of the administrative RESTful API on the Keystone server. Follows the pattern `SERVER_URL:PORT_NUMBER`.

`rgw keystone admin token`

The token or shared secret that is configured internally in Keystone for administrative requests.

`rgw keystone accepted roles`

The roles required to serve requests. Defaults to 'Member, admin'.

rgw keystone accepted admin roles

The list of roles allowing a user to gain administrative privileges.

rgw keystone token cache size

The maximum number of entries in the Keystone token cache.

rgw keystone revocation interval

The number of seconds before checking revoked tokens. Defaults to 15 * 60.

rgw keystone implicit tenants

Create new users in their own tenants of the same name. Defaults to 'false'.

rgw s3 auth use keystone

If set to 'true', the Ceph Object Gateway will authenticate users using Keystone. Defaults to 'false'.

nss db path

The path to the NSS database.

It is also possible to configure the Keystone service tenant, user, and password for Keystone (for version 2.0 of the OpenStack Identity API), similar to the way OpenStack services tend to be configured. This way you can avoid setting the shared secret `rgw keystone admin token` in the configuration file, which should be disabled in production environments. The service tenant credentials should have admin privileges. For more details refer to the [official OpenStack Keystone documentation \(https://docs.openstack.org/keystone/latest/#setting-up-projects-users-and-roles\)](https://docs.openstack.org/keystone/latest/#setting-up-projects-users-and-roles). The related configuration options follow:

rgw keystone admin user

The Keystone administrator user name.

rgw keystone admin password

The keystone administrator user password.

rgw keystone admin tenant

The Keystone version 2.0 administrator user tenant.

A Ceph Object Gateway user is mapped to a Keystone tenant. A Keystone user has different roles assigned to it, possibly on more than one tenant. When the Ceph Object Gateway gets the ticket, it looks at the tenant and the user roles that are assigned to that ticket, and accepts or rejects the request according to the setting of the `rgw keystone accepted roles` option.



Tip: Mapping to OpenStack tenants

Although Swift tenants are mapped to the Object Gateway user by default, they can be also mapped to OpenStack tenants via the `rgw keystone implicit tenants` option. This will make containers use the tenant namespace instead of the S3 like global namespace that the Object Gateway defaults to. We recommend deciding on the mapping method at the planning stage to avoid confusion. The reason for this is that toggling the option later affects only newer requests which get mapped under a tenant, while older buckets created before still continue to be in a global namespace.

For version 3 of the OpenStack Identity API, you should replace the `rgw keystone admin tenant` option with:

```
rgw keystone admin domain
```

The Keystone administrator user domain.

```
rgw keystone admin project
```

The Keystone administrator user project.

21.12 Pool placement and storage classes

21.12.1 Displaying placement targets

Placement targets control which pools are associated with a particular bucket. A bucket's placement target is selected on creation, and cannot be modified. You can display its `placement_rule` by running the following command:

```
cephuser@adm > radosgw-admin bucket stats
```

The zonegroup configuration contains a list of placement targets with an initial target named 'default-placement'. The zone configuration then maps each zonegroup placement target name onto its local storage. This zone placement information includes the 'index_pool' name for the bucket index, the 'data_extra_pool' name for metadata about incomplete multipart uploads, and a 'data_pool' name for each storage class.

21.12.2 Storage classes

Storage classes help customizing the placement of object data. S3 Bucket Lifecycle rules can automate the transition of objects between storage classes.

Storage classes are defined in terms of placement targets. Each zonegroup placement target lists its available storage classes with an initial class named 'STANDARD'. The zone configuration is responsible for providing a 'data_pool' pool name for each of the zonegroup's storage classes.

21.12.3 Configuring zonegroups and zones

Use the `radosgw-admin` command on the zonegroups and zones to configure their placement. You can query the zonegroup placement configuration using the following command:

```
cephuser@adm > radosgw-admin zonegroup get
{
  "id": "ab01123f-e0df-4f29-9d71-b44888d67cd5",
  "name": "default",
  "api_name": "default",
  ...
  "placement_targets": [
    {
      "name": "default-placement",
      "tags": [],
      "storage_classes": [
        "STANDARD"
      ]
    }
  ],
  "default_placement": "default-placement",
  ...
}
```

To query the zone placement configuration, run:

```
cephuser@adm > radosgw-admin zone get
{
  "id": "557cdcee-3aae-4e9e-85c7-2f86f5eddb1f",
  "name": "default",
  "domain_root": "default.rgw.meta:root",
  ...
  "placement_pools": [
    {
      "key": "default-placement",
```

```

    "val": {
      "index_pool": "default.rgw.buckets.index",
      "storage_classes": {
        "STANDARD": {
          "data_pool": "default.rgw.buckets.data"
        }
      },
      "data_extra_pool": "default.rgw.buckets.non-ec",
      "index_type": 0
    }
  },
  ...
}

```



Note: No previous multisite configuration

If you have not done any previous multisite configuration, a 'default' zone and zonegroup are created for you, and changes to the zone/zonegroup will not take effect until you restart the Ceph Object Gateways. If you have created a realm for multisite, the zone/zonegroup changes will take effect after you commit the changes with the `radosgw-admin period update --commit` command.

21.12.3.1 Adding a placement target

To create a new placement target named 'temporary', start by adding it to the zonegroup:

```

cephuser@adm > radosgw-admin zonegroup placement add \
  --rgw-zonegroup default \
  --placement-id temporary

```

Then provide the zone placement info for that target:

```

cephuser@adm > radosgw-admin zone placement add \
  --rgw-zone default \
  --placement-id temporary \
  --data-pool default.rgw.temporary.data \
  --index-pool default.rgw.temporary.index \
  --data-extra-pool default.rgw.temporary.non-ec

```

21.12.3.2 Adding a storage class

To add a new storage class named 'COLD' to the 'default-placement' target, start by adding it to the zonegroup:

```
cephuser@adm > radosgw-admin zonegroup placement add \  
  --rgw-zonegroup default \  
  --placement-id default-placement \  
  --storage-class COLD
```

Then provide the zone placement info for that storage class:

```
cephuser@adm > radosgw-admin zone placement add \  
  --rgw-zone default \  
  --placement-id default-placement \  
  --storage-class COLD \  
  --data-pool default.rgw.cold.data \  
  --compression lz4
```

21.12.4 Placement customization

21.12.4.1 Editing default zonegroup placement

By default, new buckets will use the zonegroup's `default_placement` target. You can change this zonegroup setting with:

```
cephuser@adm > radosgw-admin zonegroup placement default \  
  --rgw-zonegroup default \  
  --placement-id new-placement
```

21.12.4.2 Editing default user placement

A Ceph Object Gateway user can override the zonegroup's default placement target by setting a non-empty `default_placement` field in the user info. Similarly, the `default_storage_class` can override the `STANDARD` storage class applied to objects by default.

```
cephuser@adm > radosgw-admin user info --uid testid  
{  
  ...  
  "default_placement": "",  
  "default_storage_class": "",  
  "placement_tags": [],
```

```
...  
}
```

If a zonegroup's placement target contains any tags, users will be unable to create buckets with that placement target unless their user info contains at least one matching tag in its 'placement_tags' field. This can be useful to restrict access to certain types of storage.

The `radosgw-admin` command cannot modify these fields directly, therefore you need to edit the JSON format manually:

```
cephuser@adm > radosgw-admin metadata get user:USER-ID > user.json  
> vi user.json      # edit the file as required  
cephuser@adm > radosgw-admin metadata put user:USER-ID < user.json
```

21.12.4.3 Editing the S3 default bucket placement

When creating a bucket with the S3 protocol, a placement target can be provided as part of the `LocationConstraint` to override the default placement targets from the user and zonegroup.

Normally, the `LocationConstraint` needs to match the zonegroup's `api_name`:

```
<LocationConstraint>default</LocationConstraint>
```

You can add a custom placement target to the `api_name` following a colon:

```
<LocationConstraint>default:new-placement</LocationConstraint>
```

21.12.4.4 Editing the Swift bucket placement

When creating a bucket with the Swift protocol, you can provide a placement target in the HTTP header's `X-Storage-Policy`:

```
X-Storage-Policy: NEW-PLACEMENT
```

21.12.5 Using storage classes

All placement targets have a `STANDARD` storage class which is applied to new objects by default. You can override this default with its `default_storage_class`.

To create an object in a non-default storage class, provide that storage class name in an HTTP header with the request. The S3 protocol uses the `X-Amz-Storage-Class` header, while the Swift protocol uses the `X-Object-Storage-Class` header.

You can use *S3 Object Lifecycle Management* to move object data between storage classes using Transition actions.

21.13 Multisite Object Gateways

Ceph supports several multi-site configuration options for the Ceph Object Gateway:

Multi-zone

A configuration consisting of one zonegroup and multiple zones, each zone with one or more ceph-radosgw instances. Each zone is backed by its own Ceph Storage Cluster. Multiple zones in a zone group provide disaster recovery for the zonegroup should one of the zones experience a significant failure. Each zone is active and may receive write operations. In addition to disaster recovery, multiple active zones may also serve as a foundation for content delivery networks.

Multi-zone-group

Ceph Object Gateway supports multiple zonegroups, each zonegroup with one or more zones. Objects stored to zones in one zonegroup within the same realm as another zonegroup share a global object namespace, ensuring unique object IDs across zonegroups and zones.



Note

It is important to note that zonegroups *only* sync metadata amongst themselves. Data and metadata are replicated between the zones within the zonegroup. No data or metadata is shared across a realm.

Multiple realms

Ceph Object Gateway supports the notion of realms; a globally unique namespace. Multiple realms are supported which may encompass single or multiple zonegroups.

You can configure each Object Gateway to work in an active-active zone configuration, allowing for writes to non-master zones. The multi-site configuration is stored within a container called a realm. The realm stores zonegroups, zones, and a time period with multiple epochs for tracking changes to the configuration. The rgw daemons handle the synchronization, eliminating the need for a separate synchronization agent. This approach to synchronization allows the Ceph Object Gateway to operate with an active-active configuration instead of active-passive.

21.13.1 Requirements and assumptions

A multi-site configuration requires at least two Ceph storage clusters, and at least two Ceph Object Gateway instances, one for each Ceph storage cluster. The following configuration assumes at least two Ceph storage clusters are in geographically separate locations. However, the configuration can work on the same site. For example, named `rgw1` and `rgw2`.

A multi-site configuration requires a master zonegroup and a master zone. A master zone is the source of truth with regard to all metadata operations in a multisite cluster. Additionally, each zonegroup requires a master zone. zonegroups may have one or more secondary or non-master zones. In this guide, the `rgw1` host serves as the master zone of the master zonegroup and the `rgw2` host serves as the secondary zone of the master zonegroup.

21.13.2 Limitations

Multi-site configurations do not support resharding a bucket index.

As a workaround, the bucket can be purged from the slave zones, resharded on the master zone, and then resynchronized. Depending on the contents of the bucket, this can be a time- and resource-intensive operation.

21.13.3 Configuring a master zone

All gateways in a multi-site configuration retrieve their configuration from a `ceph-radosgw` daemon on a host within the master zonegroup and master zone. To configure your gateways in a multi-site configuration, select a `ceph-radosgw` instance to configure the master zonegroup and master zone.

21.13.3.1 Creating a realm

A realm represents a globally unique namespace consisting of one or more zonegroups containing one or more zones. Zones contain buckets, which in turn contain objects. A realm enables the Ceph Object Gateway to support multiple namespaces and their configuration on the same hardware. A realm contains the notion of periods. Each period represents the state of the zonegroup and zone configuration in time. Each time you make a change to a zonegroup or zone, update the period and commit it. By default, the Ceph Object Gateway does not create a realm for backward compatibility. As a best practice, we recommend creating realms for new clusters.

Create a new realm called `gold` for the multi-site configuration by opening a command line interface on a host identified to serve in the master zonegroup and zone. Then, execute the following:

```
cephuser@adm > radosgw-admin realm create --rgw-realm=gold --default
```

If the cluster has a single realm, specify the `--default` flag. If `--default` is specified, `radosgw-admin` uses this realm by default. If `--default` is not specified, adding zone-groups and zones requires specifying either the `--rgw-realm` flag or the `--realm-id` flag to identify the realm when adding zonegroups and zones.

After creating the realm, `radosgw-admin` returns the realm configuration:

```
{
  "id": "4a367026-bd8f-40ee-b486-8212482ddcd7",
  "name": "gold",
  "current_period": "09559832-67a4-4101-8b3f-10dfcd6b2707",
  "epoch": 1
}
```



Note

Ceph generates a unique ID for the realm, which allows the renaming of a realm if the need arises.

21.13.3.2 Creating a master zonegroup

A realm must have at least one zonegroup to serve as the master zonegroup for the realm. Create a new master zonegroup for the multi-site configuration by opening a command line interface on a host identified to serve in the master zonegroup and zone. Create a master zonegroup called `us` by executing the following:

```
cephuser@adm > radosgw-admin zonegroup create --rgw-zonegroup=us \
--endpoints=http://rgw1:80 --master --default
```

If the realm only has a single zonegroup, specify the `--default` flag. If `--default` is specified, `radosgw-admin` uses this zonegroup by default when adding new zones. If `--default` is not specified, adding zones requires either the `--rgw-zonegroup` flag or the `--zonegroup-id` flag to identify the zonegroup when adding or modifying zones.

After creating the master zonegroup, `radosgw-admin` returns the zonegroup configuration. For example:

```
{
  "id": "d4018b8d-8c0d-4072-8919-608726fa369e",
  "name": "us",
  "api_name": "us",
  "is_master": "true",
  "endpoints": [
    "http://rgw1:80"
  ],
  "hostnames": [],
  "hostnames_s3website": [],
  "master_zone": "",
  "zones": [],
  "placement_targets": [],
  "default_placement": "",
  "realm_id": "4a367026-bd8f-40ee-b486-8212482ddcd7"
}
```

21.13.3.3 Creating a master zone



Important

Zones need to be created on a Ceph Object Gateway node that will be within the zone.

Create a new master zone for the multi-site configuration by opening a command line interface on a host identified to serve in the master zonegroup and zone. Execute the following:

```
cephuser@adm > radosgw-admin zone create --rgw-zonegroup=us --rgw-zone=us-east-1 \
--endpoints=http://rgw1:80 --access-key=SYSTEM_ACCESS_KEY --secret=SYSTEM_SECRET_KEY
```



Note

The `--access-key` and `--secret` options are not specified in the above example. These settings are added to the zone when the user is created in the next section.

After creating the master zone, `radosgw-admin` returns the zone configuration. For example:

```
{
  "id": "56dfabbb-2f4e-4223-925e-de3c72de3866",
```

```

"name": "us-east-1",
"domain_root": "us-east-1.rgw.meta:root",
"control_pool": "us-east-1.rgw.control",
"gc_pool": "us-east-1.rgw.log:gc",
"lc_pool": "us-east-1.rgw.log:lc",
"log_pool": "us-east-1.rgw.log",
"intent_log_pool": "us-east-1.rgw.log:intent",
"usage_log_pool": "us-east-1.rgw.log:usage",
"reshard_pool": "us-east-1.rgw.log:reshard",
"user_keys_pool": "us-east-1.rgw.meta:users.keys",
"user_email_pool": "us-east-1.rgw.meta:users.email",
"user_swift_pool": "us-east-1.rgw.meta:users.swift",
"user_uid_pool": "us-east-1.rgw.meta:users.uid",
"otp_pool": "us-east-1.rgw.otp",
"system_key": {
  "access_key": "1555b35654ad1656d804",
  "secret_key": "h7GhxuBLTrlhVUyxSPUKUV8r/2EI4ngqJxD7iBdBYLhwluN30JaT3Q=="
},
"placement_pools": [
  {
    "key": "us-east-1-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
    }
  }
],
"metadata_heap": "",
"realm_id": ""
}

```

21.13.3.4 Deleting the default zone and group

Important

The following steps assume a multi-site configuration using newly installed systems that are not storing data yet. **Do not delete** the default zone and its pools if you are already using it to store data, or the data will be deleted and unrecoverable.

The default installation of Object Gateway creates the default zonegroup called `default`. Delete the default zone if it exists. Make sure to remove it from the default zonegroup first.

```
cephuser@adm > radosgw-admin zonegroup delete --rgw-zonegroup=default
```

Delete the default pools in your Ceph storage cluster if they exist:

Important

The following step assumes a multi-site configuration using newly installed systems that are not currently storing data. **Do not delete** the default zonegroup if you are already using it to store data.

```
cephuser@adm > ceph osd pool rm default.rgw.control default.rgw.control --yes-i-really-really-mean-it
cephuser@adm > ceph osd pool rm default.rgw.data.root default.rgw.data.root --yes-i-really-really-mean-it
cephuser@adm > ceph osd pool rm default.rgw.gc default.rgw.gc --yes-i-really-really-mean-it
cephuser@adm > ceph osd pool rm default.rgw.log default.rgw.log --yes-i-really-really-mean-it
cephuser@adm > ceph osd pool rm default.rgw.meta default.rgw.meta --yes-i-really-really-mean-it
```

Warning

If you delete the default zonegroup, you are also deleting the system user. If your admin user keys are not propagated, the Object Gateway management functionality of the Ceph Dashboard will fail. Follow on to the next section to re-create your system user if you go ahead with this step.

21.13.3.5 Creating system users

The `ceph-radosgw` daemons must authenticate before pulling realm and period information. In the master zone, create a system user to simplify authentication between daemons:

```
cephuser@adm > radosgw-admin user create --uid=zone.user \  
--display-name="Zone User" --access-key=SYSTEM_ACCESS_KEY \  
--secret=SYSTEM_SECRET_KEY --system
```

Make a note of the `access_key` and `secret_key` as the secondary zones require them to authenticate with the master zone.

Add the system user to the master zone:

```
cephuser@adm > radosgw-admin zone modify --rgw-zone=us-east-1 \  
--access-key=ACCESS-KEY --secret=SECRET
```

Update the period to make the changes take effect:

```
cephuser@adm > radosgw-admin period update --commit
```

21.13.3.6 Update the period

After updating the master zone configuration, update the period:

```
cephuser@adm > radosgw-admin period update --commit
```

After updating the period, `radosgw-admin` returns the period configuration. For example:

```
{  
  "id": "09559832-67a4-4101-8b3f-10dfcd6b2707", "epoch": 1, "predecessor_uuid": "",  
  "sync_status": [], "period_map":  
  {  
    "id": "09559832-67a4-4101-8b3f-10dfcd6b2707", "zonegroups": [], "short_zone_ids": []  
  }, "master_zonegroup": "", "master_zone": "", "period_config":  
  {  
    "bucket_quota": {  
      "enabled": false, "max_size_kb": -1, "max_objects": -1  
    }, "user_quota": {  
      "enabled": false, "max_size_kb": -1, "max_objects": -1  
    }  
  }, "realm_id": "4a367026-bd8f-40ee-b486-8212482ddcd7", "realm_name": "gold",  
  "realm_epoch": 1  
}
```



Note

Updating the period changes the epoch and ensures that other zones receive the updated configuration.

21.13.3.7 Start the gateway

On the Object Gateway host, start and enable the Ceph Object Gateway service. To identify the unique FSID of the cluster, run `ceph fsid`. To identify the Object Gateway daemon name, run `ceph orch ps --hostname HOSTNAME`.

```
cephuser@ogw > systemctl start ceph-FSID@DAEMON_NAME
cephuser@ogw > systemctl enable ceph-FSID@DAEMON_NAME
```

21.13.4 Configure secondary zones

Zones within a zonegroup replicate all data to ensure that each zone has the same data. When creating the secondary zone, execute all of the following operations on a host identified to serve the secondary zone.



Note

To add a third zone, follow the same procedures as for adding the secondary zone. Use different zone name.



Important

You must execute metadata operations, such as user creation, on a host within the master zone. The master zone and the secondary zone can receive bucket operations, but the secondary zone redirects bucket operations to the master zone. If the master zone is down, bucket operations will fail.

21.13.4.1 Pulling the realm

Using the URL path, access key, and secret of the master zone in the master zonegroup, pull the realm configuration to the host. To pull a non-default realm, specify the realm using the `--rgw-realm` or `--realm-id` configuration options.

```
cephuser@adm > radosgw-admin realm pull --url=url-to-master-zone-gateway --access-key=access-key --secret=secret
```



Note

Pulling the realm also retrieves the remote's current period configuration, and makes it the current period on this host as well.

If this realm is the default realm or the only realm, make the realm the default realm.

```
cephuser@adm > radosgw-admin realm default --rgw-realm=REALM-NAME
```

21.13.4.2 Creating a secondary zone

Create a secondary zone for the multi-site configuration by opening a command line interface on a host identified to serve the secondary zone. Specify the zonegroup ID, the new zone name and an endpoint for the zone. *Do not* use the `--master` flag. All zones run in an active-active configuration by default. If the secondary zone should not accept write operations, specify the `--read-only` flag to create an active-passive configuration between the master zone and the secondary zone. Additionally, provide the `access_key` and `secret_key` of the generated system user stored in the master zone of the master zonegroup. Execute the following:

```
cephuser@adm > radosgw-admin zone create --rgw-zonegroup=ZONE-GROUP-NAME \
--rgw-zone=ZONE-NAME --endpoints=URL \
--access-key=SYSTEM-KEY --secret=SECRET \
--endpoints=http://FQDN:80 \
[--read-only]
```

For example:

```
cephuser@adm > radosgw-admin zone create --rgw-zonegroup=us --endpoints=http://rgw2:80 \
--rgw-zone=us-east-2 --access-key=SYSTEM_ACCESS_KEY --secret=SYSTEM_SECRET_KEY
{
  "id": "950c1a43-6836-41a2-a161-64777e07e8b8",
  "name": "us-east-2",
  "domain_root": "us-east-2.rgw.data.root",
```

```

"control_pool": "us-east-2.rgw.control",
"gc_pool": "us-east-2.rgw.gc",
"log_pool": "us-east-2.rgw.log",
"intent_log_pool": "us-east-2.rgw.intent-log",
"usage_log_pool": "us-east-2.rgw.usage",
"user_keys_pool": "us-east-2.rgw.users.keys",
"user_email_pool": "us-east-2.rgw.users.email",
"user_swift_pool": "us-east-2.rgw.users.swift",
"user_uid_pool": "us-east-2.rgw.users.uid",
"system_key": {
  "access_key": "1555b35654ad1656d804",
  "secret_key": "h7GhxuBLTrlhVUyxSPUKUV8r\2EI4ngqJxD7iBdBYLhwluN30JaT3Q=="
},
"placement_pools": [
  {
    "key": "default-placement",
    "val": {
      "index_pool": "us-east-2.rgw.buckets.index",
      "data_pool": "us-east-2.rgw.buckets.data",
      "data_extra_pool": "us-east-2.rgw.buckets.non-ec",
      "index_type": 0
    }
  }
],
"metadata_heap": "us-east-2.rgw.meta",
"realm_id": "815d74c2-80d6-4e63-8cfc-232037f7ff5c"
}

```

Important

The following steps assume a multi-site configuration using newly-installed systems that are not yet storing data. **Do not delete** the default zone and its pools if you are already using it to store data, or the data will be lost and unrecoverable.

Delete the default zone if needed:

```
cephuser@adm > radosgw-admin zone delete --rgw-zone=default
```

Delete the default pools in your Ceph storage cluster if needed:

```

cephuser@adm > ceph osd pool rm default.rgw.control default.rgw.control --yes-i-really-really-mean-it
cephuser@adm > ceph osd pool rm default.rgw.data.root default.rgw.data.root --yes-i-really-really-mean-it
cephuser@adm > ceph osd pool rm default.rgw.gc default.rgw.gc --yes-i-really-really-mean-it

```

```
cephuser@adm > ceph osd pool rm default.rgw.log default.rgw.log --yes-i-really-really-mean-it
cephuser@adm > ceph osd pool rm default.rgw.users.uid default.rgw.users.uid --yes-i-really-really-mean-it
```

21.13.4.3 Updating the Ceph configuration file

Update the Ceph configuration file on the secondary zone hosts by adding the `rgw_zone` configuration option and the name of the secondary zone to the instance entry.

To do so, execute the following command:

```
cephuser@adm > ceph config set SERVICE_NAME rgw_zone us-west
```

21.13.4.4 Updating the period

After updating the master zone configuration, update the period:

```
cephuser@adm > radosgw-admin period update --commit
{
  "id": "b5e4d3ec-2a62-4746-b479-4b2bc14b27d1",
  "epoch": 2,
  "predecessor_uuid": "09559832-67a4-4101-8b3f-10dfcd6b2707",
  "sync_status": [ "[...]"
],
  "period_map": {
    "id": "b5e4d3ec-2a62-4746-b479-4b2bc14b27d1",
    "zonegroups": [
      {
        "id": "d4018b8d-8c0d-4072-8919-608726fa369e",
        "name": "us",
        "api_name": "us",
        "is_master": "true",
        "endpoints": [
          "http://rgw1:80"
        ],
        "hostnames": [],
        "hostnames_s3website": [],
        "master_zone": "83859a9a-9901-4f00-aa6d-285c777e10f0",
        "zones": [
          {
            "id": "83859a9a-9901-4f00-aa6d-285c777e10f0",
            "name": "us-east-1",
            "endpoints": [
```

```

        "http:\\\\rgw1:80"
    ],
    "log_meta": "true",
    "log_data": "false",
    "bucket_index_max_shards": 0,
    "read_only": "false"
  },
  {
    "id": "950c1a43-6836-41a2-a161-64777e07e8b8",
    "name": "us-east-2",
    "endpoints": [
      "http:\\\\rgw2:80"
    ],
    "log_meta": "false",
    "log_data": "true",
    "bucket_index_max_shards": 0,
    "read_only": "false"
  }
],
"placement_targets": [
  {
    "name": "default-placement",
    "tags": []
  }
],
"default_placement": "default-placement",
"realm_id": "4a367026-bd8f-40ee-b486-8212482ddcd7"
}
],
"short_zone_ids": [
  {
    "key": "83859a9a-9901-4f00-aa6d-285c777e10f0",
    "val": 630926044
  },
  {
    "key": "950c1a43-6836-41a2-a161-64777e07e8b8",
    "val": 4276257543
  }
]
},
"master_zonegroup": "d4018b8d-8c0d-4072-8919-608726fa369e",
"master_zone": "83859a9a-9901-4f00-aa6d-285c777e10f0",
"period_config": {
  "bucket_quota": {
    "enabled": false,

```

```

        "max_size_kb": -1,
        "max_objects": -1
    },
    "user_quota": {
        "enabled": false,
        "max_size_kb": -1,
        "max_objects": -1
    }
},
"realm_id": "4a367026-bd8f-40ee-b486-8212482ddcd7",
"realm_name": "gold",
"realm_epoch": 2
}

```



Note

Updating the period changes the epoch and ensures that other zones receive the updated configuration.

21.13.4.5 Starting the Object Gateway

On the Object Gateway host, start and enable the Ceph Object Gateway service:

```
cephuser@adm > ceph orch start rgw.us-east-2
```

21.13.4.6 Checking the synchronization status

When the secondary zone is up and running, check the synchronization status. Synchronization copies users and buckets created in the master zone to the secondary zone.

```
cephuser@adm > radosgw-admin sync status
```

The output provides the status of synchronization operations. For example:

```

realm f3239bc5-e1a8-4206-a81d-e1576480804d (gold)
  zonegroup c50dbb7e-d9ce-47cc-a8bb-97d9b399d388 (us)
    zone 4c453b70-4a16-4ce8-8185-1893b05d346e (us-west)
metadata sync syncing
  full sync: 0/64 shards
  metadata is caught up with master
  incremental sync: 64/64 shards

```

```
data sync source: 1ee9da3e-114d-4ae3-a8a4-056e8a17f532 (us-east)
  syncing
  full sync: 0/128 shards
  incremental sync: 128/128 shards
  data is caught up with source
```



Note

Secondary zones accept bucket operations; however, secondary zones redirect bucket operations to the master zone and then synchronize with the master zone to receive the result of the bucket operations. If the master zone is down, bucket operations executed on the secondary zone will fail, but object operations should succeed.

21.13.4.7 Verification of an Object

By default, objects are not verified again after the synchronization of an object was successful. To enable verification, set the `rgw_sync_obj_etag_verify` option to `true`. After enabling, the optional objects will be synchronized. An additional MD5 checksum will verify that it is computed on the source and the destination. This is to ensure the integrity of the objects fetched from a remote server over HTTP including multisite sync. This option can decrease the performance of RGWs as more computation is needed.

21.13.5 General Object Gateway maintenance

21.13.5.1 Checking the synchronization status

Information about the replication status of a zone can be queried with:

```
cephuser@adm > radosgw-admin sync status
  realm b3bc1c37-9c44-4b89-a03b-04c269bea5da (gold)
  zonegroup f54f9b22-b4b6-4a0e-9211-fa6ac1693f49 (us)
  zone adcellc9-b8ed-4a90-8bc5-3fc029ff0816 (us-west)
  metadata sync syncing
    full sync: 0/64 shards
    incremental sync: 64/64 shards
    metadata is behind on 1 shards
    oldest incremental change not applied: 2017-03-22 10:20:00.0.881361s
  data sync source: 341c2d81-4574-4d08-ab0f-5a2a7b168028 (us-east)
```

```
syncing
full sync: 0/128 shards
incremental sync: 128/128 shards
data is caught up with source
source: 3b5d1a3f-3f27-4e4a-8f34-6072d4bb1275 (us-3)
syncing
full sync: 0/128 shards
incremental sync: 128/128 shards
data is caught up with source
```

The output can differ depending on the sync status. The shards are described as two different types during sync:

Behind shards

Behind shards are shards that need a full data sync and shards needing an incremental data sync because they are not up-to-date.

Recovery shards

Recovery shards are shards that encountered an error during sync and marked for retry. The error mostly occurs on minor issues like acquiring a lock on a bucket. This will typically resolve itself.

21.13.5.2 Check the logs

For multi-site only, you can check out the metadata log (`mdlog`), the bucket index log (`bilog`) and the data log (`datalog`). You can list them and also trim them. This is not needed in most cases as `rgw_sync_log_trim_interval` option is set to 20 minutes as default. If it is not manually set to 0, you will not have to trim it at any time as it could cause side effects otherwise.

21.13.5.3 Changing the metadata master zone



Important

Be careful when changing which zone is the metadata master. If a zone has not finished syncing metadata from the current master zone, it is unable to serve any remaining entries when promoted to master and those changes will be lost. For this reason, we recommend waiting for a zone's `radosgw-admin` synchronization status to catch up on metadata sync before promoting it to master. Similarly, if changes to metadata are being processed by the current master zone while another zone is being promoted to master, those changes

are likely to be lost. To avoid this, we recommend shutting down any Object Gateway instances on the previous master zone. After promoting another zone, its new period can be fetched with `radosgw-admin period pull` and the gateway(s) can be restarted.

To promote a zone (for example, zone `us-west` in zonegroup `us`) to metadata master, run the following commands on that zone:

```
cephuser@ogw > radosgw-admin zone modify --rgw-zone=us-west --master
cephuser@ogw > radosgw-admin zonegroup modify --rgw-zonegroup=us --master
cephuser@ogw > radosgw-admin period update --commit
```

This generates a new period, and the Object Gateway instance(s) in zone `us-west` sends this period to other zones.

21.13.5.4 Resharding a bucket index



Important

Resharding a bucket index in a multi-site setup requires a full resynchronization of the bucket content. Depending on the size and number of objects in the bucket, this is a time- and resource-intensive operation.

PROCEDURE 21.2: RESHARDING THE BUCKET INDEX

1. Make sure that all operations to the bucket are stopped.
2. Back up the original bucket index:

```
cephuser@adm > radosgw-admin bi list \
--bucket=BUCKET_NAME \
> BUCKET_NAME.list.backup
```

3. Disable bucket synchronization for the affected bucket:

```
cephuser@adm > radosgw-admin bucket sync disable --bucket=BUCKET_NAME
```

4. Wait for the synchronization to finish on all zones. Check on master and slave zones with the following command:

```
cephuser@adm > radosgw-admin sync status
```

5. Stop the Object Gateway instances. First on all slave zones, then on the master zone, too.

```
cephuser@ogw > systemctl stop ceph-radosgw@rgw.NODE.service
```

6. Reshard the bucket index on the master zone:

```
cephuser@adm > radosgw-admin bucket reshard \  
--bucket=BUCKET_NAME \  
--num-shards=NEW_SHARDS_NUMBER
```



Tip: Old bucket ID

As part of its output, this command also prints the new and the old bucket ID.

7. Purge the bucket on all slave zones:

```
cephuser@adm > radosgw-admin bucket rm \  
--purge-objects \  
--bucket=BUCKET_NAME \  
--yes-i-really-mean-it
```

8. Restart the Object Gateway on the master zone first, then on the slave zones as well.

```
cephuser@ogw > systemctl restart ceph-radosgw.target
```

9. On the master zone, re-enable bucket synchronization.

```
cephuser@adm > radosgw-admin bucket sync enable --bucket=BUCKET_NAME
```

21.13.6 Performing failover and disaster recovery

If the master zone should fail, failover to the secondary zone for disaster recovery.

1. Make the secondary zone the master and default zone. For example:

```
cephuser@adm > radosgw-admin zone modify --rgw-zone=ZONE-NAME --master --default
```

By default, Ceph Object Gateway runs in an active-active configuration. If the cluster was configured to run in an active-passive configuration, the secondary zone is a read-only zone. Remove the `--read-only` status to allow the zone to receive write operations. For example:

```
cephuser@adm > radosgw-admin zone modify --rgw-zone=ZONE-NAME --master --default \  
--read-only
```

```
--read-only=false
```

2. Update the period to make the changes take effect:

```
cephuser@adm > radosgw-admin period update --commit
```

3. Restart the Ceph Object Gateway:

```
cephuser@adm > ceph orch restart rgw
```

If the former master zone recovers, revert the operation.

1. From the recovered zone, pull the latest realm configuration from the current master zone.

```
cephuser@adm > radosgw-admin realm pull --url=URL-TO-MASTER-ZONE-GATEWAY \  
--access-key=ACCESS-KEY --secret=SECRET
```

2. Make the recovered zone the master and default zone:

```
cephuser@adm > radosgw-admin zone modify --rgw-zone=ZONE-NAME --master --default
```

3. Update the period to make the changes take effect:

```
cephuser@adm > radosgw-admin period update --commit
```

4. Restart the Ceph Object Gateway in the recovered zone:

```
cephuser@adm > ceph orch restart rgw@rgw
```

5. If the secondary zone needs to be a read-only configuration, update the secondary zone:

```
cephuser@adm > radosgw-admin zone modify --rgw-zone=ZONE-NAME --read-only
```

6. Update the period to make the changes take effect:

```
cephuser@adm > radosgw-admin period update --commit
```

7. Restart the Ceph Object Gateway in the secondary zone:

```
cephuser@adm > ceph orch restart@rgw
```

22 Ceph iSCSI gateway

The chapter focuses on administration tasks related to the iSCSI Gateway. For a procedure of deployment refer to *Book "Deployment Guide", Chapter 8 "Deploying the remaining core services using cephadm", Section 8.3.5 "Deploying iSCSI Gateways"*.

22.1 ceph-iscsi managed targets

This chapter describes how to connect to `ceph-iscsi` managed targets from clients running Linux, Microsoft Windows, or VMware.

22.1.1 Connecting to open-iscsi

Connecting to `ceph-iscsi` backed iSCSI targets with `open-iscsi` is a two-step process. First the initiator must discover the iSCSI targets available on the gateway host, then it must log in and map the available Logical Units (LUs).

Both steps require that the `open-iscsi` daemon is running. The way you start the `open-iscsi` daemon is dependent on your Linux distribution:

- On SUSE Linux Enterprise Server (SLES); and Red Hat Enterprise Linux (RHEL) hosts, run `systemctl start iscsid` (or `service iscsid start` if `systemctl` is not available).
- On Debian and Ubuntu hosts, run `systemctl start open-iscsi` (or `service open-iscsi start`).

If your initiator host runs SUSE Linux Enterprise Server, refer to <https://documentation.suse.com/sles/15-SP2/html/SLES-all/cha-iscsi.html#sec-iscsi-initiator> for details on how to connect to an iSCSI target.

For any other Linux distribution supporting `open-iscsi`, proceed to discover targets on your `ceph-iscsi` gateway (this example uses `iscsi1.example.com` as the portal address; for multipath access repeat these steps with `iscsi2.example.com`):

```
# iscsiadm -m discovery -t sendtargets -p iscsi1.example.com
192.168.124.104:3260,1 iqn.2003-01.org.linux-iscsi.iscsi.SYSTEM-ARCH:testvol
```

Then, log in to the portal. If the login completes successfully, any RBD-backed logical units on the portal will immediately become available on the system SCSI bus:

```
# iscsiadm -m node -p iscsi1.example.com --login
Logging in to [iface: default, target: iqn.2003-01.org.linux-iscsi.iscsi.SYSTEM-ARCH:testvol, portal: 192.168.124.104,3260] (multiple)
Login to [iface: default, target: iqn.2003-01.org.linux-iscsi.iscsi.SYSTEM-ARCH:testvol, portal: 192.168.124.104,3260] successful.
```

Repeat this process for other portal IP addresses or hosts.

If your system has the `lsscsi` utility installed, you use it to enumerate available SCSI devices on your system:

```
lsscsi
[8:0:0:0]    disk    SUSE    RBD                4.0    /dev/sde
[9:0:0:0]    disk    SUSE    RBD                4.0    /dev/sdf
```

In a multipath configuration (where two connected iSCSI devices represent one and the same LU), you can also examine the multipath device state with the `multipath` utility:

```
# multipath -ll
360014050cf9dcfcb2603933ac3298dca dm-9 SUSE,RBD
size=49G features='0' hwhandler='0' wp=rw
|+- policy='service-time 0' prio=1 status=active
| `- 8:0:0:0 sde 8:64 active ready running
`+- policy='service-time 0' prio=1 status=enabled
`- 9:0:0:0 sdf 8:80 active ready running
```

You can now use this multipath device as you would any block device. For example, you can use the device as a Physical Volume for Linux Logical Volume Management (LVM), or you can simply create a file system on it. The example below demonstrates how to create an XFS file system on the newly connected multipath iSCSI volume:

```
# mkfs -t xfs /dev/mapper/360014050cf9dcfcb2603933ac3298dca
log stripe unit (4194304 bytes) is too large (maximum is 256KiB)
log stripe unit adjusted to 32KiB
meta-data=/dev/mapper/360014050cf9dcfcb2603933ac3298dca isize=256    agcount=17,
    agsize=799744 blks
    =                       sectsz=512    attr=2, projid32bit=1
    =                       crc=0          finobt=0
data      =                       bsize=4096    blocks=12800000, imaxpct=25
    =                       sunit=1024    swidth=1024 blks
naming    =version 2              bsize=4096    ascii-ci=0 ftype=0
log       =internal log          bsize=4096    blocks=6256, version=2
```

```
= sectsz=512 sunit=8 blks, lazy-count=1
realtime =none extsz=4096 blocks=0, rtextents=0
```

Note that XFS being a non-clustered file system, you may only ever mount it on a single iSCSI initiator node at any given time.

If at any time you want to discontinue using the iSCSI LUs associated with a particular target, run the following command:

```
# iscsiadm -m node -p iscsi1.example.com --logout
Logging out of session [sid: 18, iqn.2003-01.org.linux-iscsi.iscsi.SYSTEM-ARCH:testvol,
portal: 192.168.124.104,3260]
Logout of [sid: 18, target: iqn.2003-01.org.linux-iscsi.iscsi.SYSTEM-ARCH:testvol,
portal: 192.168.124.104,3260] successful.
```

As with discovery and login, you must repeat the logout steps for all portal IP addresses or host names.

22.1.1.1 Configuring multipath

The multipath configuration is maintained on the clients or initiators and is independent of any `ceph-iscsi` configuration. Select a strategy prior to using block storage. After editing the `/etc/multipath.conf`, restart `multipathd` with

```
# systemctl restart multipathd
```

For an active-passive configuration with friendly names, add

```
defaults {
    user_friendly_names yes
}
```

to your `/etc/multipath.conf`. After connecting to your targets successfully, run

```
# multipath -ll
mpathd (36001405dbb561b2b5e439f0aed2f8e1e) dm-0 SUSE,RBD
size=2.0G features='0' hwhandler='0' wp=rw
|-+- policy='service-time 0' prio=1 status=active
|  `-- 2:0:0:3 sdl 8:176 active ready running
|-+- policy='service-time 0' prio=1 status=enabled
|  `-- 3:0:0:3 sdj 8:144 active ready running
`-+- policy='service-time 0' prio=1 status=enabled
   `-- 4:0:0:3 sdk 8:160 active ready running
```

Note the status of each link. For an active-active configuration, add

```
defaults {
  user_friendly_names yes
}

devices {
  device {
    vendor "(LIO-ORG|SUSE)"
    product "RBD"
    path_grouping_policy "multibus"
    path_checker "tur"
    features "0"
    hardware_handler "1 alua"
    prio "alua"
    failback "immediate"
    rr_weight "uniform"
    no_path_retry 12
    rr_min_io 100
  }
}
```

to your `/etc/multipath.conf`. Restart `multipathd` and run

```
# multipath -ll
mpathd (36001405dbb561b2b5e439f0aed2f8e1e) dm-3 SUSE,RBD
size=2.0G features='1 queue_if_no_path' hwhandler='1 alua' wp=rw
`-+- policy='service-time 0' prio=50 status=active
   |- 4:0:0:3 sdj 8:144 active ready running
   |- 3:0:0:3 sdk 8:160 active ready running
   `-- 2:0:0:3 sdl 8:176 active ready running
```

22.1.2 Connecting Microsoft Windows (Microsoft iSCSI initiator)

To connect to a SUSE Enterprise Storage iSCSI target from a Windows 2012 server, follow these steps:

1. Open Windows Server Manager. From the Dashboard, select *Tools > iSCSI Initiator*. The *iSCSI Initiator Properties* dialog appears. Select the *Discovery* tab:

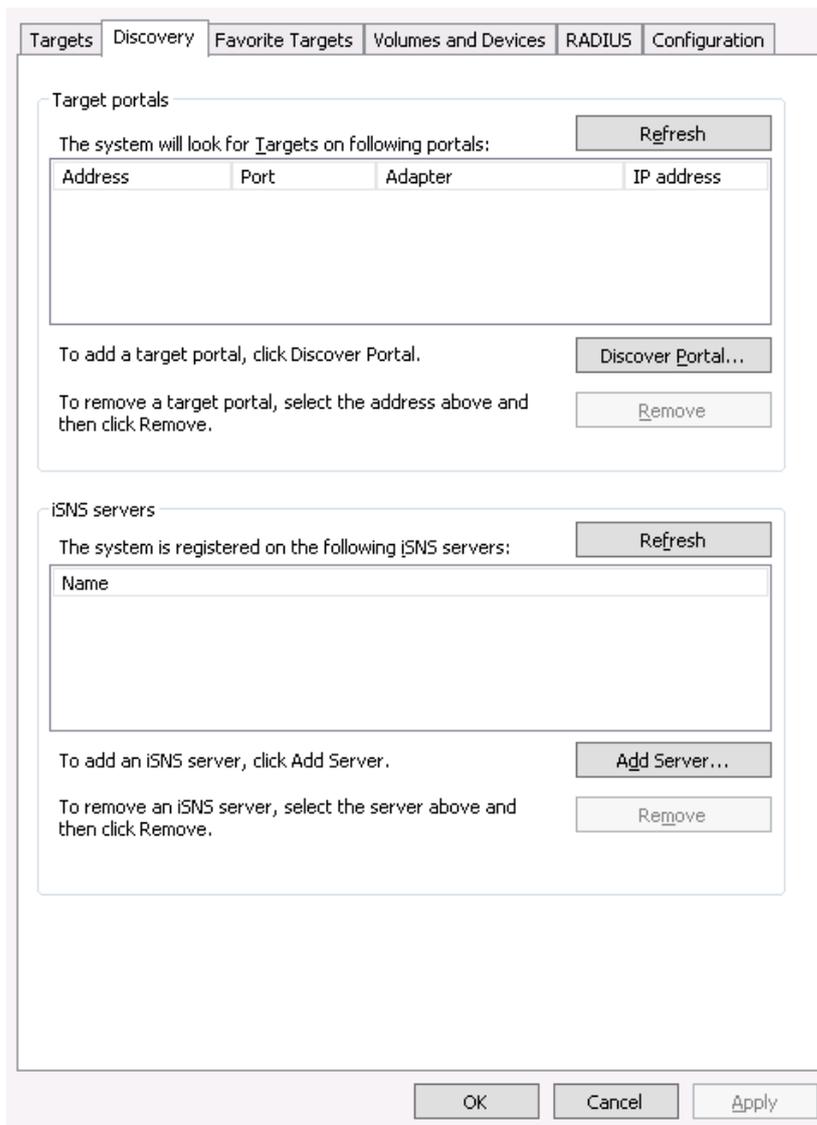


FIGURE 22.1: iSCSI INITIATOR PROPERTIES

2. In the *Discover Target Portal* dialog, enter the target's host name or IP address in the *Target* field and click *OK*:

Enter the IP address or DNS name and port number of the portal you want to add.

To change the default settings of the discovery of the target portal, click the Advanced button.

IP address or DNS name: Port: (Default is 3260.)

FIGURE 22.2: DISCOVER TARGET PORTAL

- Repeat this process for all other gateway host names or IP addresses. When completed, review the *Target Portals* list:

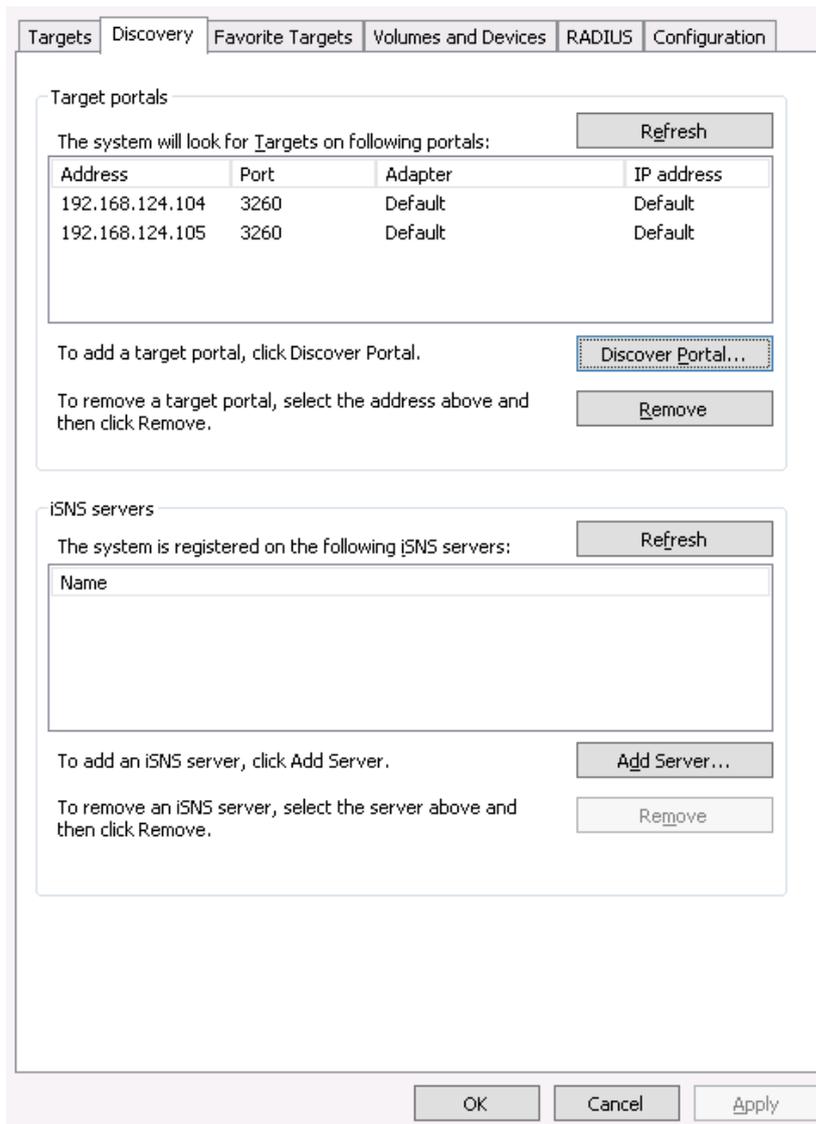


FIGURE 22.3: TARGET PORTALS

- Next, switch to the *Targets* tab and review your discovered target(s).

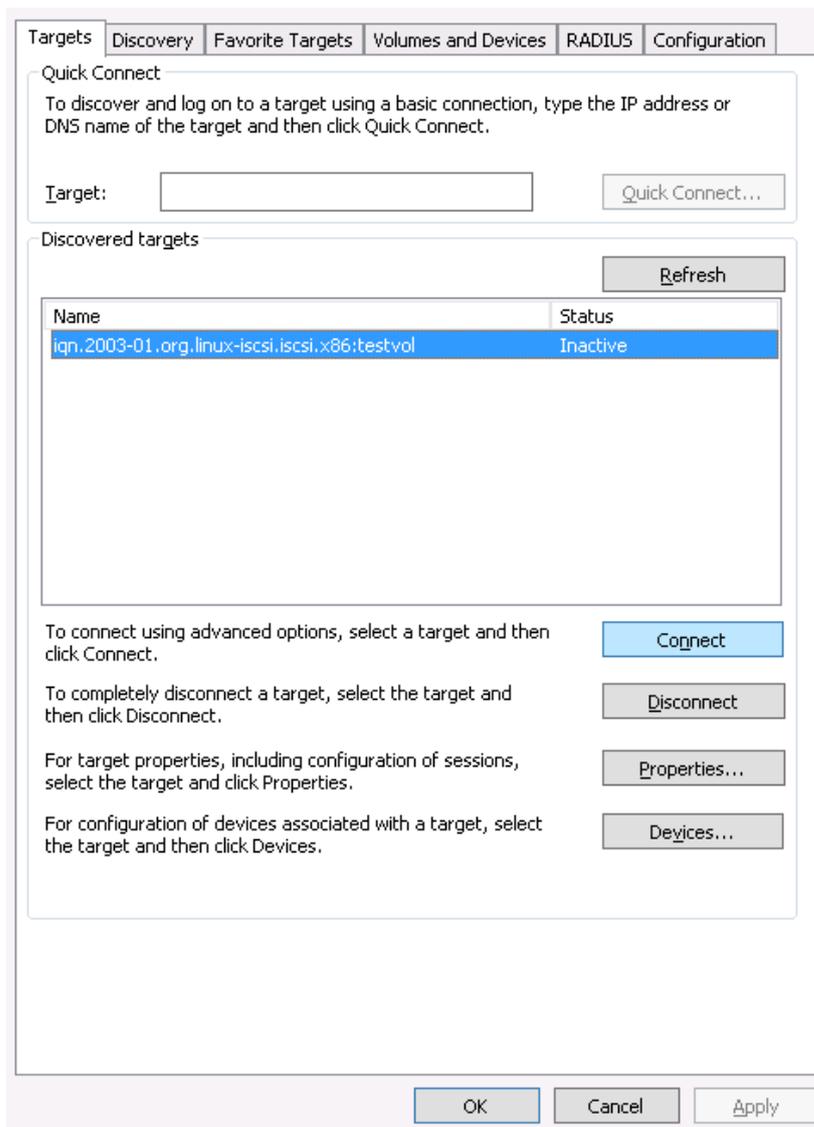


FIGURE 22.4: TARGETS

5. Click *Connect* in the *Targets* tab. The *Connect To Target* dialog appears. Select the *Enable Multi-path* check box to enable multipath I/O (MPIO), then click *OK*:

6. When the *Connect to Target* dialog closes, select *Properties* to review the target's properties:

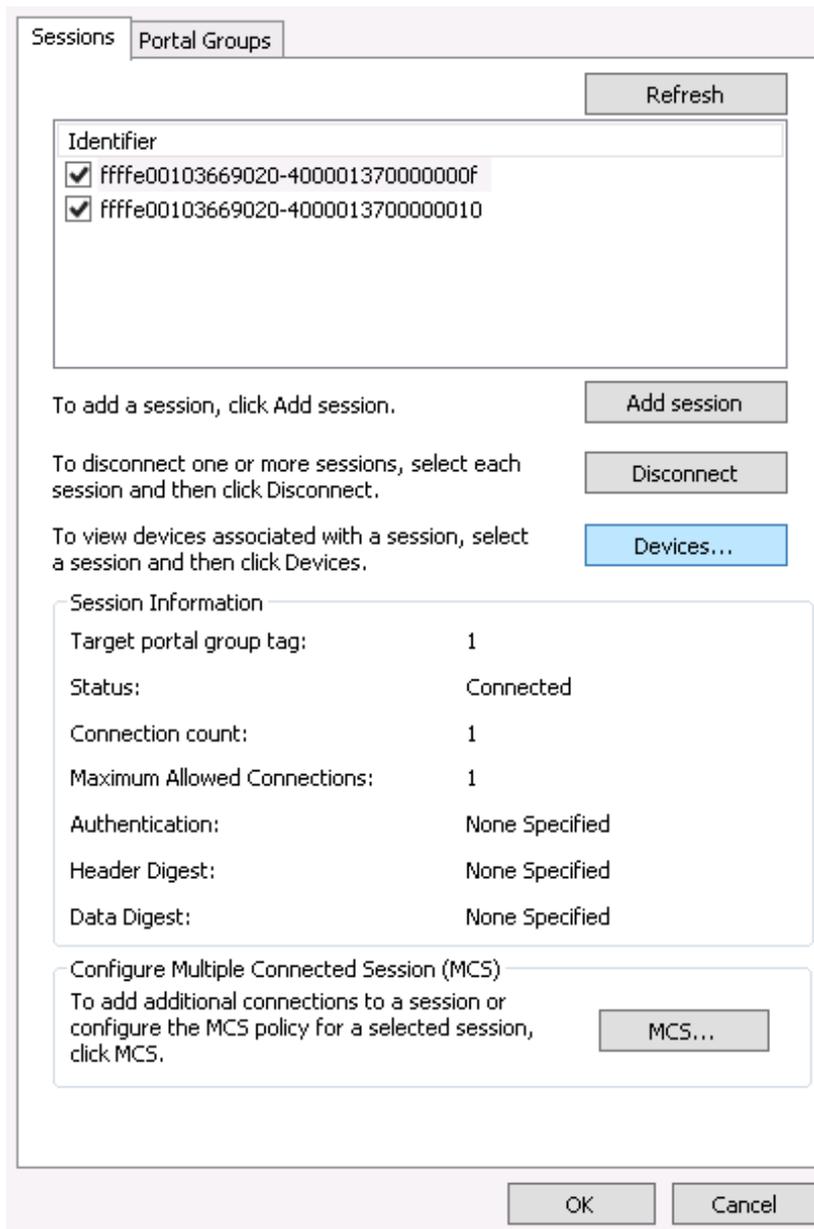


FIGURE 22.5: ICSI TARGET PROPERTIES

7. Select *Devices*, and click *MPIO* to review the multipath I/O configuration:

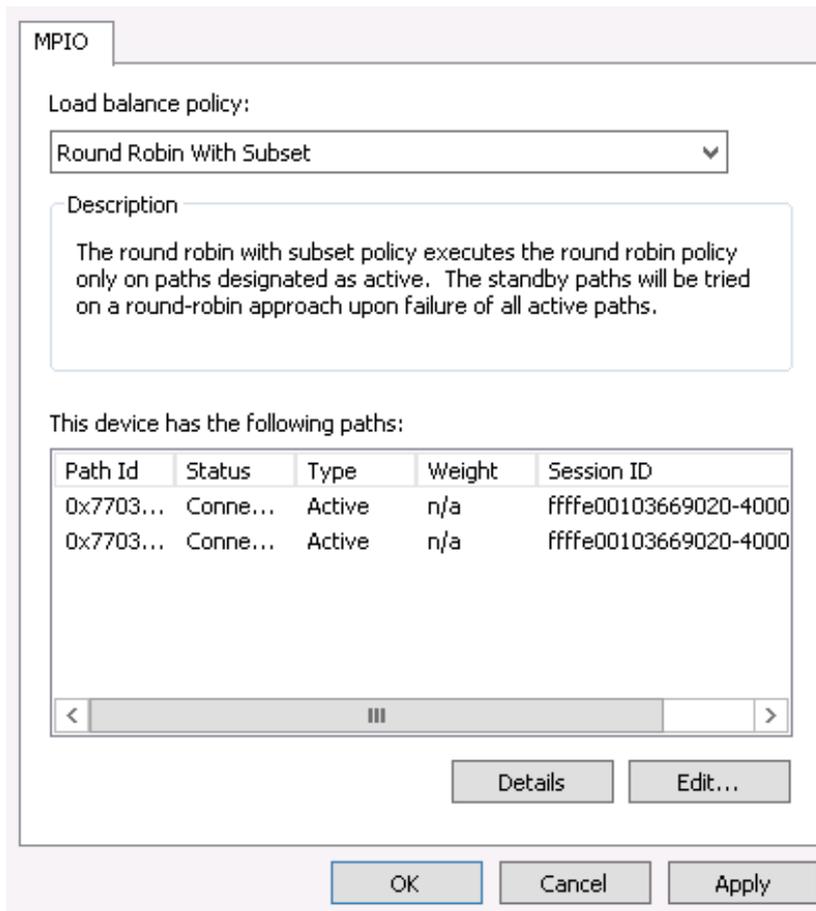


FIGURE 22.6: DEVICE DETAILS

The default *Load Balance policy* is *Round Robin With Subset*. If you prefer a pure failover configuration, change it to *Fail Over Only*.

This concludes the iSCSI initiator configuration. The iSCSI volumes are now available like any other SCSI devices, and may be initialized for use as volumes and drives. Click *OK* to close the *iSCSI Initiator Properties* dialog, and proceed with the *File and Storage Services* role from the *Server Manager* dashboard.

Observe the newly connected volume. It identifies as *SUSE RBD SCSI Multi-Path Drive* on the iSCSI bus, and is initially marked with an *Offline* status and a partition table type of *Unknown*. If the new volume does not appear immediately, select *Rescan Storage* from the *Tasks* drop-down box to rescan the iSCSI bus.

1. Right-click on the iSCSI volume and select *New Volume* from the context menu. The *New Volume Wizard* appears. Click *Next*, highlight the newly connected iSCSI volume and click *Next* to begin.

Select the server and disk

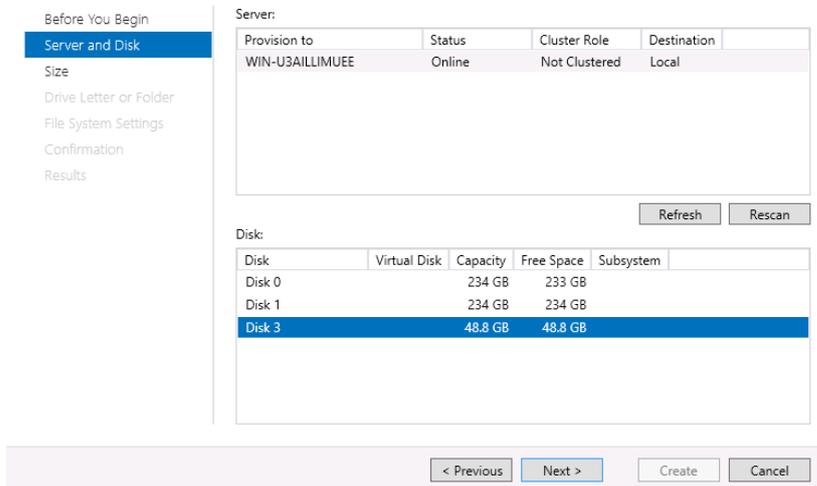


FIGURE 22.7: NEW VOLUME WIZARD

2. Initially, the device is empty and does not contain a partition table. When prompted, confirm the dialog indicating that the volume will be initialized with a GPT partition table:

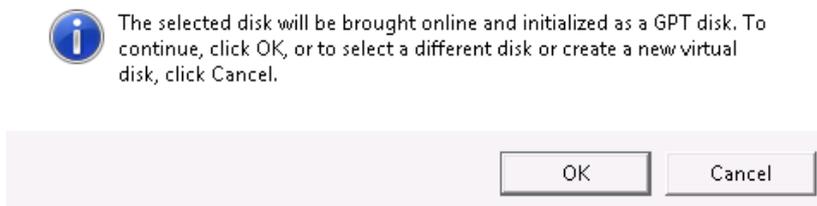


FIGURE 22.8: OFFLINE DISK PROMPT

3. Select the volume size. Typically, you would use the device's full capacity. Then assign a drive letter or directory name where the newly created volume will become available. Then select a file system to create on the new volume, and finally confirm your selections with *Create* to finish creating the volume:

Confirm selections

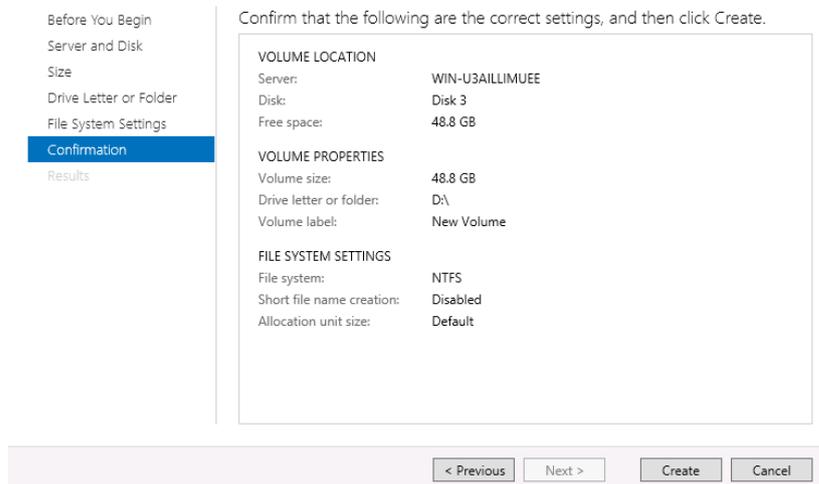


FIGURE 22.9: CONFIRM VOLUME SELECTIONS

When the process finishes, review the results, then *Close* to conclude the drive initialization. Once initialization completes, the volume (and its NTFS file system) becomes available like a newly initialized local drive.

22.1.3 Connecting VMware

1. To connect to `ceph-iscsi` managed iSCSI volumes you need a configured iSCSI software adapter. If no such adapter is available in your vSphere configuration, create one by selecting *Configuration > Storage Adapters > Add > iSCSI Software initiator*.

2. When available, select the adapter's properties by right-clicking the adapter and selecting *Properties* from the context menu:

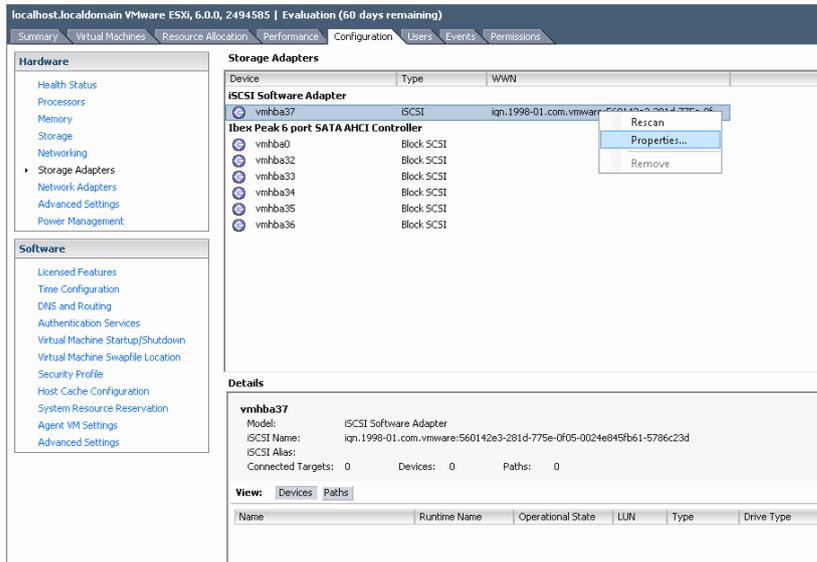


FIGURE 22.10: iSCSI INITIATOR PROPERTIES

3. In the *iSCSI Software Initiator* dialog, click the *Configure* button. Then go to the *Dynamic Discovery* tab and select *Add*.
4. Enter the IP address or host name of your `ceph-iscsi` iSCSI gateway. If you run multiple iSCSI gateways in a failover configuration, repeat this step for as many gateways as you operate.

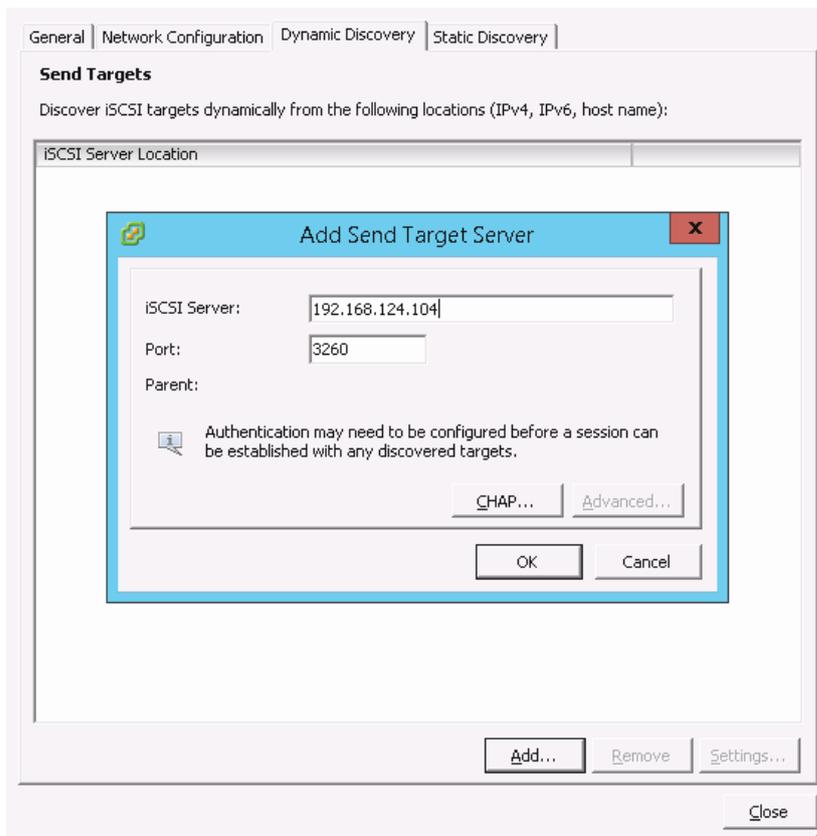


FIGURE 22.11: ADD TARGET SERVER

When you have entered all iSCSI gateways, click *OK* in the dialog to initiate a rescan of the iSCSI adapter.

- When the rescan completes, the new iSCSI device appears below the *Storage Adapters* list in the *Details* pane. For multipath devices, you can now right-click on the adapter and select *Manage Paths* from the context menu:

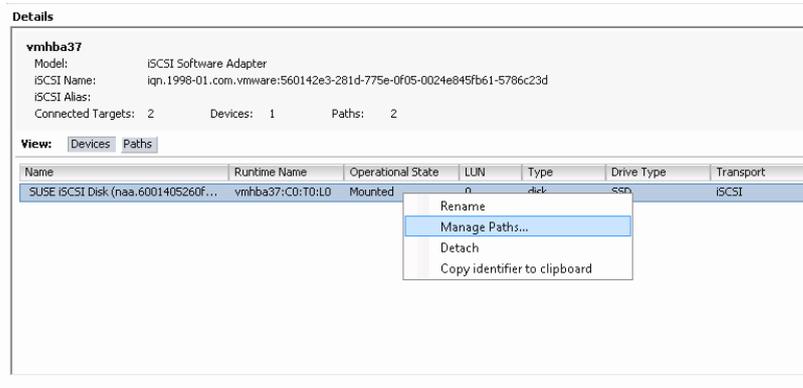


FIGURE 22.12: MANAGE MULTIPATH DEVICES

You should now see all paths with a green light under *Status*. One of your paths should be marked *Active (I/O)* and all others simply *Active*:

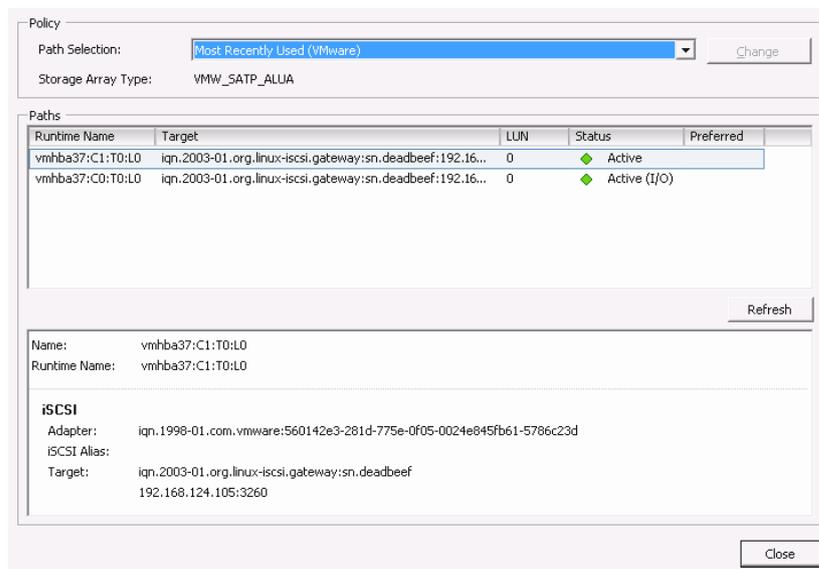


FIGURE 22.13: PATHS LISTING FOR MULTIPATH

- You can now switch from *Storage Adapters* to the item labeled *Storage*. Select *Add Storage...* in the top-right corner of the pane to bring up the *Add Storage* dialog. Then, select *Disk/LUN* and click *Next*. The newly added iSCSI device appears in the *Select Disk/LUN* list. Select it, then click *Next* to proceed:

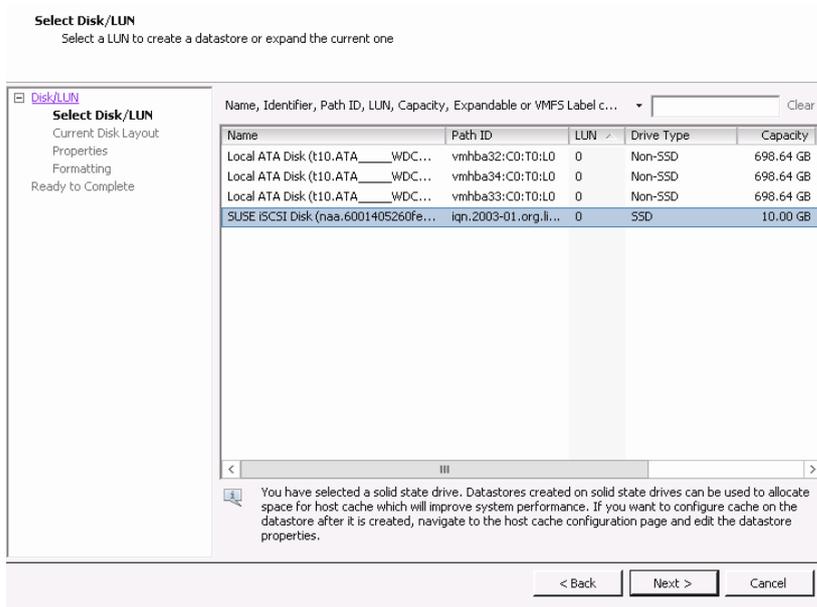


FIGURE 22.14: ADD STORAGE DIALOG

Click *Next* to accept the default disk layout.

7. In the *Properties* pane, assign a name to the new datastore, and click *Next*. Accept the default setting to use the volume's entire space for the datastore, or select *Custom Space Setting* for a smaller datastore:

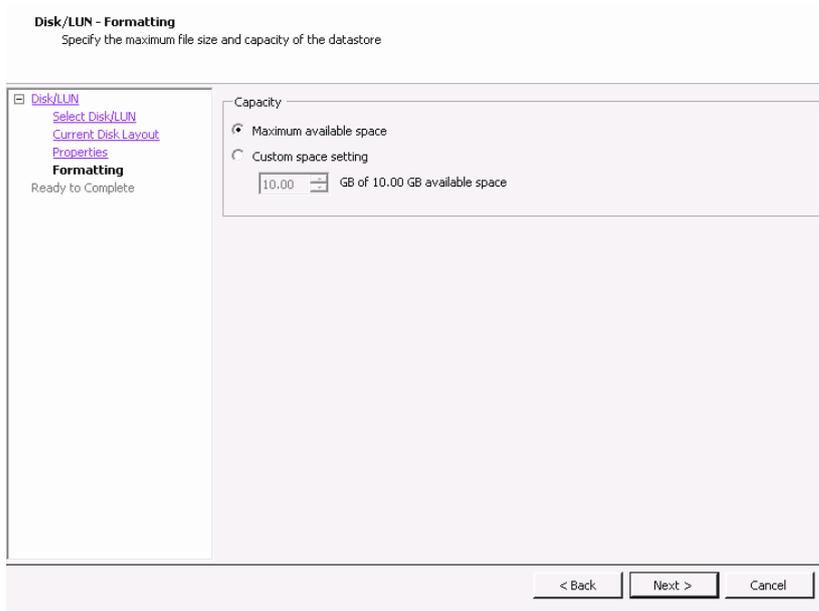


FIGURE 22.15: CUSTOM SPACE SETTING

Click *Finish* to complete the datastore creation.

The new datastore now appears in the datastore list and you can select it to retrieve details. You are now able to use the `ceph-iscsi` backed iSCSI volume like any other vSphere datastore.

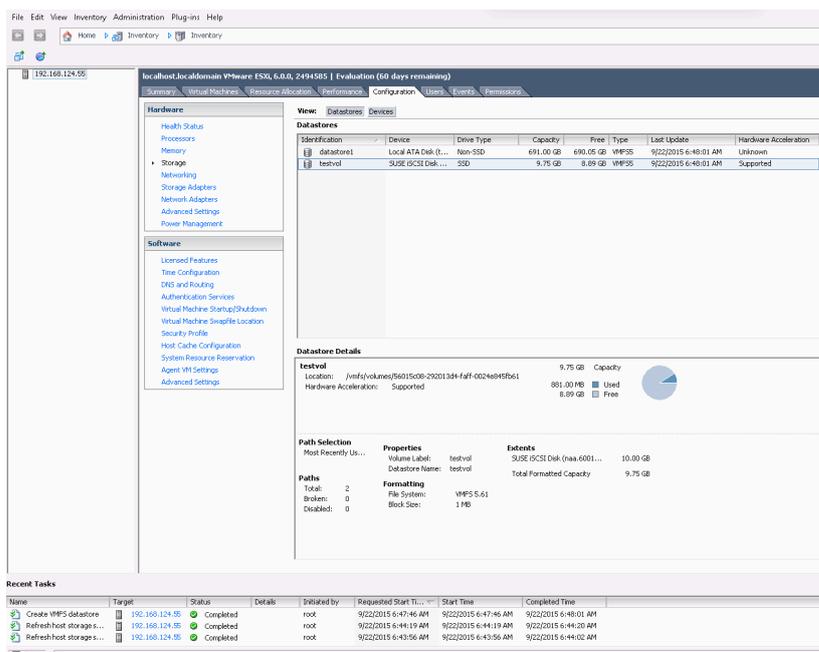


FIGURE 22.16: ISCSI DATASTORE OVERVIEW

22.2 Conclusion

`ceph-iscsi` is a key component of SUSE Enterprise Storage 7 that enables access to distributed, highly available block storage from any server or client capable of speaking the iSCSI protocol. By using `ceph-iscsi` on one or more iSCSI gateway hosts, Ceph RBD images become available as Logical Units (LUs) associated with iSCSI targets, which can be accessed in an optionally load-balanced, highly available fashion.

Since all of `ceph-iscsi` configuration is stored in the Ceph RADOS object store, `ceph-iscsi` gateway hosts are inherently without persistent state and thus can be replaced, augmented, or reduced at will. As a result, SUSE Enterprise Storage 7 enables SUSE customers to run a truly distributed, highly-available, resilient, and self-healing enterprise storage technology on commodity hardware and an entirely open source platform.

23 Clustered file system

This chapter describes administration tasks that are normally performed after the cluster is set up and CephFS exported. If you need more information on setting up CephFS, refer to *Book "Deployment Guide", Chapter 8 "Deploying the remaining core services using cephadm", Section 8.3.3 "Deploying Metadata Servers"*.

23.1 Mounting CephFS

When the file system is created and the MDS is active, you are ready to mount the file system from a client host.

23.1.1 Preparing the client

If the client host is running SUSE Linux Enterprise 12 SP2 or later, the system is ready to mount CephFS 'out of the box'.

If the client host is running SUSE Linux Enterprise 12 SP1, you need to apply all the latest patches before mounting CephFS.

In any case, everything needed to mount CephFS is included in SUSE Linux Enterprise. The SUSE Enterprise Storage 7 product is not needed.

To support the full `mount` syntax, the `ceph-common` package (which is shipped with SUSE Linux Enterprise) should be installed before trying to mount CephFS.



Important

Without the `ceph-common` package (and thus without the `mount.ceph` helper), the monitors' IPs will need to be used instead of their names. This is because the kernel client will be unable to perform name resolution.

The basic mount syntax is:

```
# mount -t ceph MON1_IP[:PORT],MON2_IP[:PORT],...:CEPHFS_MOUNT_TARGET \  
MOUNT_POINT -o name=CEPHX_USER_NAME,secret=SECRET_STRING
```

23.1.2 Creating a secret file

The Ceph cluster runs with authentication turned on by default. You should create a file that stores your secret key (not the keyring itself). To obtain the secret key for a particular user and then create the file, do the following:

PROCEDURE 23.1: CREATING A SECRET KEY

1. View the key for the particular user in a keyring file:

```
cephuser@adm > cat /etc/ceph/ceph.client.admin.keyring
```

2. Copy the key of the user who will be using the mounted Ceph FS file system. Usually, the key looks similar to the following:

```
AQCj2YpRiAe6CxAA7/ETt7Hcl9IyxyYciVs47w==
```

3. Create a file with the user name as a file name part, for example `/etc/ceph/admin.secret` for the user *admin*.
4. Paste the key value to the file created in the previous step.
5. Set proper access rights to the file. The user should be the only one who can read the file—others may not have any access rights.

23.1.3 Mounting CephFS

You can mount CephFS with the `mount` command. You need to specify the monitor host name or IP address. Because the `cephx` authentication is enabled by default in SUSE Enterprise Storage, you need to specify a user name and their related secret as well:

```
# mount -t ceph ceph_mon1:6789:/ /mnt/cephfs \  
-o name=admin,secret=AQATSKdNGBnwLhAAAnNDKnH65FmVKpXZJVasUeQ==
```

As the previous command remains in the shell history, a more secure approach is to read the secret from a file:

```
# mount -t ceph ceph_mon1:6789:/ /mnt/cephfs \  
-o name=admin,secretfile=/etc/ceph/admin.secret
```

Note that the secret file should only contain the actual keyring secret. In our example, the file will then contain only the following line:

```
AQATSKdNGBnwLhAAAnNDKnH65FmVKpXZJVasUeQ==
```



Tip: Specify multiple monitors

It is a good idea to specify multiple monitors separated by commas on the `mount` command line in case one monitor happens to be down at the time of mount. Each monitor address takes the form `host[:port]`. If the port is not specified, it defaults to 6789.

Create the mount point on the local host:

```
# mkdir /mnt/cephfs
```

Mount the CephFS:

```
# mount -t ceph ceph_mon1:6789:/ /mnt/cephfs \  
-o name=admin,secretfile=/etc/ceph/admin.secret
```

A subdirectory `subdir` may be specified if a subset of the file system is to be mounted:

```
# mount -t ceph ceph_mon1:6789:/subdir /mnt/cephfs \  
-o name=admin,secretfile=/etc/ceph/admin.secret
```

You can specify more than one monitor host in the `mount` command:

```
# mount -t ceph ceph_mon1,ceph_mon2,ceph_mon3:6789:/ /mnt/cephfs \  
-o name=admin,secretfile=/etc/ceph/admin.secret
```



Important: Read access to the root directory

If clients with path restriction are used, the MDS capabilities need to include read access to the root directory. For example, a keyring may look as follows:

```
client.bar  
key: supersecretkey  
caps: [mds] allow rw path=/barjail, allow r path=  
caps: [mon] allow r  
caps: [osd] allow rwx
```

The `allow r path=` part means that path-restricted clients are able to see the root volume, but cannot write to it. This may be an issue for use cases where complete isolation is a requirement.

23.2 Unmounting CephFS

To unmount the CephFS, use the `umount` command:

```
# umount /mnt/cephfs
```

23.3 Mounting CephFS in `/etc/fstab`

To mount CephFS automatically upon client start-up, insert the corresponding line in its file systems table `/etc/fstab`:

```
mon1:6790,mon2:/subdir /mnt/cephfs ceph name=admin,secretfile=/etc/ceph/  
secret.key,noatime,_netdev 0 2
```

23.4 Multiple active MDS daemons (active-active MDS)

CephFS is configured for a single active MDS daemon by default. To scale metadata performance for large-scale systems, you can enable multiple active MDS daemons, which will share the metadata workload with one another.

23.4.1 Using active-active MDS

Consider using multiple active MDS daemons when your metadata performance is bottlenecked on the default single MDS.

Adding more daemons does not increase performance on all workload types. For example, a single application running on a single client will not benefit from an increased number of MDS daemons unless the application is doing a lot of metadata operations in parallel.

Workloads that typically benefit from a larger number of active MDS daemons are those with many clients, perhaps working on many separate directories.

23.4.2 Increasing the MDS active cluster size

Each CephFS file system has a `max_mds` setting, which controls how many ranks will be created. The actual number of ranks in the file system will only be increased if a spare daemon is available to take on the new rank. For example, if there is only one MDS daemon running and `max_mds` is set to two, no second rank will be created.

In the following example, we set the `max_mds` option to 2 to create a new rank apart from the default one. To see the changes, run `ceph status` before and after you set `max_mds`, and watch the line containing `fsmap`:

```
cephuser@adm > ceph status
[...]
services:
  [...]
  mds: cephfs-1/1/1 up {0=node2=up:active}, 1 up:standby
  [...]
cephuser@adm > ceph fs set cephfs max_mds 2
cephuser@adm > ceph status
[...]
services:
  [...]
  mds: cephfs-2/2/2 up {0=node2=up:active,1=node1=up:active}
  [...]
```

The newly created rank (1) passes through the 'creating' state and then enter its 'active' state.



Important: Standby daemons

Even with multiple active MDS daemons, a highly available system still requires standby daemons to take over if any of the servers running an active daemon fail.

Consequently, the practical maximum of `max_mds` for highly available systems is one less than the total number of MDS servers in your system. To remain available in the event of multiple server failures, increase the number of standby daemons in the system to match the number of server failures you need to survive.

23.4.3 Decreasing the number of ranks

All ranks—including the ranks to be removed—must first be active. This means that you need to have at least `max_mds` MDS daemons available.

First, set `max_mds` to a lower number. For example, go back to having a single active MDS:

```
cephuser@adm > ceph status
[...]
services:
  [...]
  mds: cephfs-2/2/2 up {0=node2=up:active,1=node1=up:active}
  [...]
cephuser@adm > ceph fs set cephfs max_mds 1
cephuser@adm > ceph status
[...]
services:
  [...]
  mds: cephfs-1/1/1 up {0=node2=up:active}, 1 up:standby
  [...]
```

23.4.4 Manually pinning directory trees to a rank

In multiple active metadata server configurations, a balancer runs, which works to spread metadata load evenly across the cluster. This usually works well enough for most users, but sometimes it is desirable to override the dynamic balancer with explicit mappings of metadata to particular ranks. This can allow the administrator or users to evenly spread application load or limit impact of users' metadata requests on the entire cluster.

The mechanism provided for this purpose is called an 'export pin'. It is an extended attribute of directories. The name of this extended attribute is `ceph.dir.pin`. Users can set this attribute using standard commands:

```
# setfattr -n ceph.dir.pin -v 2 /path/to/dir
```

The value (`-v`) of the extended attribute is the rank to assign the directory sub-tree to. A default value of `-1` indicates that the directory is not pinned.

A directory export pin is inherited from its closest parent with a set export pin. Therefore, setting the export pin on a directory affects all of its children. However, the parent's pin can be overridden by setting the child directory export pin. For example:

```
# mkdir -p a/b # "a" and "a/b" start with no export pin set.
setfattr -n ceph.dir.pin -v 1 a/ # "a" and "b" are now pinned to rank 1.
setfattr -n ceph.dir.pin -v 0 a/b # "a/b" is now pinned to rank 0
# and "a/" and the rest of its children
# are still pinned to rank 1.
```

23.5 Managing failover

If an MDS daemon stops communicating with the monitor, the monitor will wait `mds_beacon_grace` seconds (default 15 seconds) before marking the daemon as *laggy*. You can configure one or more 'standby' daemons that will take over during the MDS daemon failover.

23.5.1 Configuring standby replay

Each CephFS file system may be configured to add standby-replay daemons. These standby daemons follow the active MDS's metadata journal to reduce failover time in the event that the active MDS becomes unavailable. Each active MDS may have only one standby-replay daemon following it.

Configure standby-replay on a file system with the following command:

```
cephuser@adm > ceph fs set FS-NAME allow_standby_replay BOOL
```

When set the monitors will assign available standby daemons to follow the active MDSs in that file system.

Once an MDS has entered the standby-replay state, it will only be used as a standby for the rank that it is following. If another rank fails, this standby-replay daemon will not be used as a replacement, even if no other standbys are available. For this reason, it is advised that if standby-replay is used then every active MDS should have a standby-replay daemon.

23.6 Setting CephFS quotas

You can set quotas on any subdirectory of the Ceph file system. The quota restricts either the number of **bytes** or **files** stored beneath the specified point in the directory hierarchy.

23.6.1 CephFS quota limitations

Using quotas with CephFS has the following limitations:

Quotas are cooperative and non-competing.

Ceph quotas rely on the client that is mounting the file system to stop writing to it when a limit is reached. The server part cannot prevent a malicious client from writing as much data as it needs. Do not use quotas to prevent filling the file system in environments where the clients are fully untrusted.

Quotas are imprecise.

Processes that are writing to the file system will be stopped shortly after the quota limit is reached. They will inevitably be allowed to write some amount of data over the configured limit. Client writers will be stopped within tenths of seconds after crossing the configured limit.

Quotas are implemented in the kernel client from version 4.17.

Quotas are supported by the user space client (libcephfs, ceph-fuse). Linux kernel clients 4.17 and higher support CephFS quotas on SUSE Enterprise Storage 7 clusters. Kernel clients (even recent versions) will fail to handle quotas on older clusters, even if they are able to set the quotas extended attributes. SLE12-SP3 (and later) kernels already include the required backports to handle quotas.

Configure quotas carefully when used with path-based mount restrictions.

The client needs to have access to the directory inode on which quotas are configured in order to enforce them. If the client has restricted access to a specific path (for example `/home/user`) based on the MDS capability, and a quota is configured on an ancestor directory they do not have access to (`/home`), the client will not enforce it. When using path-based access restrictions, be sure to configure the quota on the directory that the client can access (for example `/home/user` or `/home/user/quota_dir`).

23.6.2 Configuring CephFS quotas

You can configure CephFS quotas by using virtual extended attributes:

`ceph.quota.max_files`

Configures a *file* limit.

`ceph.quota.max_bytes`

Configures a *byte* limit.

If the attributes appear on a directory inode, a quota is configured there. If they are not present then no quota is set on that directory (although one may still be configured on a parent directory).

To set a 100 MB quota, run:

```
cephuser@mds > setfattr -n ceph.quota.max_bytes -v 100000000 /SOME/DIRECTORY
```

To set a 10,000 files quota, run:

```
cephuser@mds > setfattr -n ceph.quota.max_files -v 10000 /SOME/DIRECTORY
```

To view quota setting, run:

```
cephuser@mds > getfattr -n ceph.quota.max_bytes /SOME/DIRECTORY
```

```
cephuser@mds > getfattr -n ceph.quota.max_files /SOME/DIRECTORY
```



Note: Quota not set

If the value of the extended attribute is '0', the quota is not set.

To remove a quota, run:

```
cephuser@mds > setfattr -n ceph.quota.max_bytes -v 0 /SOME/DIRECTORY  
cephuser@mds > setfattr -n ceph.quota.max_files -v 0 /SOME/DIRECTORY
```

23.7 Managing CephFS snapshots

CephFS snapshots create a read-only view of the file system at the point in time they are taken. You can create a snapshot in any directory. The snapshot will cover all data in the file system under the specified directory. After creating a snapshot, the buffered data is flushed out asynchronously from various clients. As a result, creating a snapshot is very fast.



Important: Multiple file systems

If you have multiple CephFS file systems sharing a single pool (via name spaces), their snapshots will collide, and deleting one snapshot will result in missing file data for other snapshots sharing the same pool.

23.7.1 Creating snapshots

The CephFS snapshot feature is enabled by default on new file systems. To enable it on existing file systems, run:

```
cephuser@adm > ceph fs set CEPHFS_NAME allow_new_snaps true
```

After you enable snapshots, all directories in the CephFS will have a special `.snap` subdirectory.



Note

This is a *virtual* subdirectory. It does not appear in the directory listing of the parent directory, but the name `.snap` cannot be used as a file or directory name. To access the `.snap` directory one needs to explicitly access it, for example:

```
> ls -la /CEPHFS_MOUNT/.snap/
```



Important: Kernel clients limitation

CephFS kernel clients have a limitation: they cannot handle more than 400 snapshots in a file system. The number of snapshots should always be kept below this limit, regardless of which client you are using. If using older CephFS clients, such as SLE12-SP3, keep in mind that going above 400 snapshots is harmful to operations as the client will crash.



Tip: Custom snapshot subdirectory name

You may configure a different name for the snapshots subdirectory by setting the `client snapdir` setting.

To create a snapshot, create a subdirectory under the `.snap` directory with a custom name. For example, to create a snapshot of the directory `/CEPHFS_MOUNT/2/3/`, run:

```
> mkdir /CEPHFS_MOUNT/2/3/.snap/CUSTOM_SNAPSHOT_NAME
```

23.7.2 Deleting snapshots

To delete a snapshot, remove its subdirectory inside the `.snap` directory:

```
> rmdir /CEPHFS_MOUNT/2/3/.snap/CUSTOM_SNAPSHOT_NAME
```

24 Export Ceph data via Samba

This chapter describes how to export data stored in a Ceph cluster via a Samba/CIFS share so that you can easily access them from Windows* client machines. It also includes information that will help you configure a Ceph Samba gateway to join Active Directory in the Windows* domain to authenticate and authorize users.



Note: Samba gateway performance

Because of increased protocol overhead and additional latency caused by extra network hops between the client and the storage, accessing CephFS via a Samba Gateway may significantly reduce application performance when compared to native Ceph clients.

24.1 Export CephFS via Samba share



Warning: Cross protocol access

Native CephFS and NFS clients are not restricted by file locks obtained via Samba, and vice versa. Applications that rely on cross protocol file locking may experience data corruption if CephFS backed Samba share paths are accessed via other means.

24.1.1 Configuring and exporting Samba packages

To configure and export a Samba share, the following packages need to be installed: `samba-ceph` and `samba-winbind`. If these packages are not installed, install them:

```
cephuser@smb > zypper install samba-ceph samba-winbind
```

24.1.2 Single gateway example

In preparation for exporting a Samba share, choose an appropriate node to act as a Samba Gateway. The node needs to have access to the Ceph client network, as well as sufficient CPU, memory, and networking resources.

Failover functionality can be provided with CTDB and the SUSE Linux Enterprise High Availability Extension. Refer to [Section 24.1.3, “Configuring high availability”](#) for more information on HA setup.

1. Make sure that a working CephFS already exists in your cluster.
2. Create a Samba Gateway specific keyring on the Ceph admin node and copy it to both Samba Gateway nodes:

```
cephuser@adm > ceph auth get-or-create client.samba.gw mon 'allow r' \
  osd 'allow *' mds 'allow *' -o ceph.client.samba.gw.keyring
cephuser@adm > scp ceph.client.samba.gw.keyring SAMBA_NODE:/etc/ceph/
```

Replace `SAMBA_NODE` with the name of the Samba gateway node.

3. The following steps are executed on the Samba Gateway node. Install Samba together with the Ceph integration package:

```
cephuser@smb > sudo zypper in samba samba-ceph
```

4. Replace the default contents of the `/etc/samba/smb.conf` file with the following:

```
[global]
  netbios name = SAMBA-GW
  clustering = no
  idmap config * : backend = tdb2
  passdb backend = tdbsam
  # disable print server
  load printers = no
  smbd: backgroundqueue = no

[SHARE_NAME]
  path = CEPHFS_MOUNT
  read only = no
  oplocks = no
  kernel share modes = no
```

The `CEPHFS_MOUNT` path above must be mounted prior to starting Samba with a kernel CephFS share configuration. See [Section 23.3, “Mounting CephFS in /etc/fstab”](#).

The above share configuration uses the Linux kernel CephFS client, which is recommended for performance reasons. As an alternative, the Samba `vfs_ceph` module can also be used to communicate with the Ceph cluster. The instructions are shown below for legacy purposes and are not recommended for new Samba deployments:

```
[SHARE_NAME]
path = /
vfs objects = ceph
ceph: config_file = /etc/ceph/ceph.conf
ceph: user_id = samba.gw
read only = no
oplocks = no
kernel share modes = no
```

Tip: Oplocks and share modes

`oplocks` (also known as SMB2 + leases) allow for improved performance through aggressive client caching, but are currently unsafe when Samba is deployed together with other CephFS clients, such as kernel `mount .ceph`, FUSE, or NFS Ganesha.

If all CephFS file system path access is exclusively handled by Samba, then the `oplocks` parameter can be safely enabled.

Currently `kernel share modes` needs to be disabled in a share running with the CephFS `vfs` module for file serving to work properly.

Important: Permitting access

Samba maps SMB users and groups to local accounts. Local users can be assigned a password for Samba share access via:

```
# smbpasswd -a USERNAME
```

For successful I/O, the share path's access control list (ACL) needs to permit access to the user connected via Samba. You can modify the ACL by temporarily mounting via the CephFS kernel client and using the `chmod`, `chown`, or `setfacl` utilities against the share path. For example, to permit access for all users, run:

```
# chmod 777 MOUNTED_SHARE_PATH
```

24.1.2.1 Starting Samba services

Start or restart stand-alone Samba services using the following commands:

```
# systemctl restart smb.service
# systemctl restart nmb.service
# systemctl restart winbind.service
```

To ensure that Samba services start on boot, enable them via:

```
# systemctl enable smb.service
# systemctl enable nmb.service
# systemctl enable winbind.service
```



Tip: Optional nmb and winbind services

If you do not require network share browsing, you do not need to enable and start the `nmb` service.

The `winbind` service is only needed when configured as an Active Directory domain member. See [Section 24.2, “Joining Samba Gateway and Active Directory”](#).

24.1.3 Configuring high availability



Important: Transparent failover not supported

Although a multi-node Samba + CTDB deployment is more highly available compared to the single node (see [Chapter 24, Export Ceph data via Samba](#)), client-side transparent failover is not supported. Applications will likely experience a short outage on Samba Gateway node failure.

This section provides an example of how to set up a two-node high availability configuration of Samba servers. The setup requires the SUSE Linux Enterprise High Availability Extension. The two nodes are called `earth` (192.168.1.1) and `mars` (192.168.1.2).

For details about SUSE Linux Enterprise High Availability Extension, see <https://documentation.suse.com/sle-ha/15-SP1/>.

Additionally, two floating virtual IP addresses allow clients to connect to the service no matter which physical node it is running on. `192.168.1.10` is used for cluster administration with Hawk2 and `192.168.2.1` is used exclusively for the CIFS exports. This makes it easier to apply security restrictions later.

The following procedure describes the example installation. More details can be found at <https://documentation.suse.com/sle-ha/15-SP2/html/SLE-HA-all/art-sleha-install-quick.html>.

1. Create a Samba Gateway specific keyring on the Admin Node and copy it to both nodes:

```
cephuser@adm > ceph auth get-or-create client.samba.gw mon 'allow r' \  
    osd 'allow *' mds 'allow *' -o ceph.client.samba.gw.keyring  
cephuser@adm > scp ceph.client.samba.gw.keyring earth:/etc/ceph/  
cephuser@adm > scp ceph.client.samba.gw.keyring mars:/etc/ceph/
```

2. SLE-HA setup requires a fencing device to avoid a *split brain* situation when active cluster nodes become unsynchronized. For this purpose, you can use a Ceph RBD image with Stonith Block Device (SBD). Refer to <https://documentation.suse.com/sle-ha/15-SP2/html/SLE-HA-all/cha-ha-storage-protect.html#sec-ha-storage-protect-fencing-setup> for more details.

If it does not yet exist, create an RBD pool called `rbd` (see [Section 18.1, "Creating a pool"](#)) and associate it with `rbd` (see [Section 18.5.1, "Associating pools with an application"](#)). Then create a related RBD image called `sbd01`:

```
cephuser@adm > ceph osd pool create rbd  
cephuser@adm > ceph osd pool application enable rbd rbd  
cephuser@adm > rbd -p rbd create sbd01 --size 64M --image-shared
```

3. Prepare `earth` and `mars` to host the Samba service:
 - a. Make sure the following packages are installed before you proceed: `ctdb`, `tdb-tools`, and `samba`.

```
# zypper in ctdb tdb-tools samba samba-ceph
```

- b. Make sure the Samba and CTDB services are stopped and disabled:

```
# systemctl disable ctdb  
# systemctl disable smb  
# systemctl disable nmb  
# systemctl disable winbind  
# systemctl stop ctdb  
# systemctl stop smb
```

```
# systemctl stop nmb
# systemctl stop winbind
```

- c. Open port `4379` of your firewall on all nodes. This is needed for CTDB to communicate with other cluster nodes.
4. On `earth`, create the configuration files for Samba. They will later automatically synchronize to `mars`.
 - a. Insert a list of private IP addresses of Samba Gateway nodes in the `/etc/ctdb/nodes` file. Find more details in the ctdb manual page (`man 7 ctdb`).

```
192.168.1.1
192.168.1.2
```

- b. Configure Samba. Add the following lines in the `[global]` section of `/etc/samba/smb.conf`. Use the host name of your choice in place of `CTDB-SERVER` (all nodes in the cluster will appear as one big node with this name). Add a share definition as well, consider `SHARE_NAME` as an example:

```
[global]
netbios name = SAMBA-HA-GW
clustering = yes
idmap config * : backend = tdb2
passdb backend = tdbsam
ctdbd socket = /var/lib/ctdb/ctdb.socket
# disable print server
load printers = no
smbd: backgroundqueue = no

[SHARE_NAME]
path = /
vfs objects = ceph
ceph: config_file = /etc/ceph/ceph.conf
ceph: user_id = samba.gw
read only = no
oplocks = no
kernel share modes = no
```

Note that the `/etc/ctdb/nodes` and `/etc/samba/smb.conf` files need to match on all Samba Gateway nodes.

5. Install and bootstrap the SUSE Linux Enterprise High Availability cluster.

- a. Register the SUSE Linux Enterprise High Availability Extension on earth and mars:

```
root@earth # SUSEConnect -r ACTIVATION_CODE -e E_MAIL
```

```
root@mars # SUSEConnect -r ACTIVATION_CODE -e E_MAIL
```

- b. Install ha-cluster-bootstrap on both nodes:

```
root@earth # zypper in ha-cluster-bootstrap
```

```
root@mars # zypper in ha-cluster-bootstrap
```

- c. Map the RBD image sbd01 on both Samba Gateways via rbdmap.service.
Edit /etc/ceph/rbdmap and add an entry for the SBD image:

```
rbd/sbd01 id=samba.gw,keyring=/etc/ceph/ceph.client.samba.gw.keyring
```

Enable and start rbdmap.service:

```
root@earth # systemctl enable rbdmap.service && systemctl start rbdmap.service
root@mars # systemctl enable rbdmap.service && systemctl start rbdmap.service
```

The /dev/rbd/rbd/sbd01 device should be available on both Samba Gateways.

- d. Initialize the cluster on earth and let mars join it.

```
root@earth # ha-cluster-init
```

```
root@mars # ha-cluster-join -c earth
```

Important

During the process of initialization and joining the cluster, you will be interactively asked whether to use SBD. Confirm with y and then specify /dev/rbd/rbd/sbd01 as a path to the storage device.

6. Check the status of the cluster. You should see two nodes added in the cluster:

```
root@earth # crm status
2 nodes configured
1 resource configured
```

```
Online: [ earth mars ]
```

```
Full list of resources:
```

```
admin-ip      (ocf::heartbeat:IPAddr2):      Started earth
```

7. Execute the following commands on earth to configure the CTDB resource:

```
root@earth # crm configure
crm(live)configure# primitive ctdb ocf:heartbeat:CTDB params \
    ctdb_manages_winbind="false" \
    ctdb_manages_samba="false" \
    ctdb_recovery_lock="!/usr/lib64/ctdb/ctdb_mutex_ceph_rados_helper
    ceph client.samba.gw cephfs_metadata ctdb-mutex"
    ctdb_socket="/var/lib/ctdb/ctdb.socket" \
    op monitor interval="10" timeout="20" \
    op start interval="0" timeout="200" \
    op stop interval="0" timeout="100"
crm(live)configure# primitive smb systemd:smb \
    op start timeout="100" interval="0" \
    op stop timeout="100" interval="0" \
    op monitor interval="60" timeout="100"
crm(live)configure# primitive nmb systemd:nmb \
    op start timeout="100" interval="0" \
    op stop timeout="100" interval="0" \
    op monitor interval="60" timeout="100"
crm(live)configure# primitive winbind systemd:winbind \
    op start timeout="100" interval="0" \
    op stop timeout="100" interval="0" \
    op monitor interval="60" timeout="100"
crm(live)configure# group g-ctdb ctdb winbind nmb smb
crm(live)configure# clone cl-ctdb g-ctdb meta interleave="true"
crm(live)configure# commit
```



Tip: Optional nmb and winbind primitives

If you do not require network share browsing, you do not need to add the nmb primitive.

The winbind primitive is only needed when configured as an Active Directory domain member. See [Section 24.2, "Joining Samba Gateway and Active Directory"](#).

The binary `/usr/lib64/ctdb/ctdb_mutex_ceph_rados_helper` in the configuration option `ctdb_recovery_lock` has the parameters `CLUSTER_NAME`, `CEPHX_USER`, `RADOS_POOL`, and `RADOS_OBJECT`, in this order.

An extra lock-timeout parameter can be appended to override the default value used (10 seconds). A higher value will increase the CTDB recovery master failover time, whereas a lower value may result in the recovery master being incorrectly detected as down, triggering flapping failovers.

8. Add a clustered IP address:

```
crm(live)configure# primitive ip ocf:heartbeat:IPaddr2
  params ip=192.168.2.1 \
  unique_clone_address="true" \
  op monitor interval="60" \
  meta resource-stickiness="0"
crm(live)configure# clone cl-ip ip \
  meta interleave="true" clone-node-max="2" globally-unique="true"
crm(live)configure# colocation col-with-ctdb 0: cl-ip cl-ctdb
crm(live)configure# order o-with-ctdb 0: cl-ip cl-ctdb
crm(live)configure# commit
```

If `unique_clone_address` is set to `true`, the `IPaddr2` resource agent adds a clone ID to the specified address, leading to three different IP addresses. These are usually not needed, but help with load balancing. For further information about this topic, see <https://documentation.suse.com/sle-ha/15-SP2/html/SLE-HA-all/cha-ha-lb.html>.

9. Check the result:

```
root@earth # crm status
Clone Set: base-clone [dlm]
  Started: [ factory-1 ]
  Stopped: [ factory-0 ]
Clone Set: cl-ctdb [g-ctdb]
  Started: [ factory-1 ]
  Started: [ factory-0 ]
Clone Set: cl-ip [ip] (unique)
  ip:0      (ocf:heartbeat:IPaddr2):      Started factory-0
  ip:1      (ocf:heartbeat:IPaddr2):      Started factory-1
```

10. Test from a client machine. On a Linux client, run the following command to see if you can copy files from and to the system:

```
# smbclient //192.168.2.1/myshare
```

24.1.3.1 Restarting HA Samba resources

Following any Samba or CTDB configuration changes, HA resources may need to be restarted for the changes to take effect. This can be done by via:

```
# crm resource restart cl-ctdb
```

24.2 Joining Samba Gateway and Active Directory

You can configure the Ceph Samba gateway to become a member of Samba domain with Active Directory (AD) support. As a Samba domain member, you can use domain users and groups in local access lists (ACLs) on files and directories from the exported CephFS.

24.2.1 Preparing Samba installation

This section introduces preparatory steps that you need to take care of before configuring the Samba itself. Starting with a clean environment helps you prevent confusion and verifies that no files from the previous Samba installation are mixed with the new domain member installation.



Tip: Synchronizing clocks

All Samba Gateway nodes' clocks need to be synchronized with the Active Directory Domain controller. Clock skew may result in authentication failures.

Verify that no Samba or name caching processes are running:

```
cephuser@smb > ps ax | egrep "samba|smbd|nmbd|winbindd|nscd"
```

If the output lists any samba, smbd, nmbd, winbindd, or nscd processes, stop them.

If you have previously run a Samba installation on this host, remove the `/etc/samba/smb.conf` file. Also remove all Samba database files, such as `*.tdb` and `*.ldb` files. To list directories containing Samba databases, run:

```
cephuser@smb > smb -b | egrep "LOCKDIR|STATEDIR|CACHEDIR|PRIVATE_DIR"
```

24.2.2 Verifying DNS

Active Directory (AD) uses DNS to locate other domain controllers (DCs) and services, such as Kerberos. Therefore AD domain members and servers need to be able to resolve the AD DNS zones.

Verify that DNS is correctly configured and that both forward and reverse lookup resolve correctly, for example:

```
cephuser@adm > nslookup DC1.domain.example.com
Server:          10.99.0.1
Address:         10.99.0.1#53

Name:   DC1.domain.example.com
Address: 10.99.0.1
```

```
cephuser@adm > 10.99.0.1
Server:          10.99.0.1
Address: 10.99.0.1#53

1.0.99.10.in-addr.arpa name = DC1.domain.example.com.
```

24.2.3 Resolving SRV records

AD uses SRV records to locate services, such as Kerberos and LDAP. To verify that SRV records are resolved correctly, use the `nslookup` interactive shell, for example:

```
cephuser@adm > nslookup
Default Server:  10.99.0.1
Address:         10.99.0.1

> set type=SRV
> _ldap._tcp.domain.example.com.
Server:   UnKnown
Address:  10.99.0.1
```

```
_ldap._tcp.domain.example.com  SRV service location:
    priority      = 0
    weight       = 100
    port        = 389
    svr hostname  = dc1.domain.example.com
domain.example.com  nameserver = dc1.domain.example.com
dc1.domain.example.com  internet address = 10.99.0.1
```

24.2.4 Configuring Kerberos

Samba supports Heimdal and MIT Kerberos back-ends. To configure Kerberos on the domain member, set the following in your `/etc/krb5.conf` file:

```
[libdefaults]
default_realm = DOMAIN.EXAMPLE.COM
dns_lookup_realm = false
dns_lookup_kdc = true
```

The previous example configures Kerberos for the `DOMAIN.EXAMPLE.COM` realm. We do not recommend to set any further parameters in the `/etc/krb5.conf` file. If your `/etc/krb5.conf` contains an `include` line it will not work—you **must** remove this line.

24.2.5 Resolving localhost name

When you join a host to the domain, Samba tries to register the host name in the AD DNS zone. For this, the `net` utility needs to be able to resolve the host name using DNS or using a correct entry in the `/etc/hosts` file.

To verify that your host name resolves correctly, use the `getent hosts` command:

```
cephuser@adm > getent hosts example-host
10.99.0.5      example-host.domain.example.com  example-host
```

The host name and FQDN must not resolve to the `127.0.0.1` IP address or any IP address other than the one used on the LAN interface of the domain member. If no output is displayed or the host is resolved to the wrong IP address and you are not using DHCP, set the correct entry in the `/etc/hosts` file:

```
127.0.0.1      localhost
10.99.0.5      example-host.samdom.example.com  example-host
```



Tip: DHCP and `/etc/hosts`

If you are using DHCP, check that `/etc/hosts` only contains the '127.0.0.1' line. If you continue to have problems, contact the administrator of your DHCP server.

If you need to add aliases to the machine host name, add them to the end of the line that starts with the machine's IP address, not to the '127.0.0.1' line.

24.2.6 Configuring Samba

This section introduces information about specific configuration options that you need to include in the Samba configuration.

Active Directory domain membership is primarily configured by setting `security = ADS` alongside appropriate Kerberos realm and ID mapping parameters in the `[global]` section of `/etc/samba/smb.conf`.

```
[global]
security = ADS
workgroup = DOMAIN
realm = DOMAIN.EXAMPLE.COM
...
```

24.2.6.1 Choosing the back-end for ID mapping in `winbindd`

If you need your users to have different login shells and/or Unix home directory paths, or you want them to have the same ID everywhere, you will need to use the `winbind 'ad'` back-end and add RFC2307 attributes to AD.



Important: RFC2307 Attributes and ID Numbers

The RFC2307 attributes are not added automatically when users or groups are created.

The ID numbers found on a DC (numbers in the 3000000 range) are *not* RFC2307 attributes and will not be used on Unix Domain Members. If you need to have the same ID numbers everywhere, add `uidNumber` and `gidNumber` attributes to AD and use the `winbind 'ad'` back-end on Unix Domain Members. If you do decide to add `uidNumber` and `gidNumber` attributes to AD, do not use numbers in the 3000000 range.

If your users will only use the Samba AD DC for authentication and will not store data on it or log in to it, you can use the winbind 'rid' back-end. This calculates the user and group IDs from the Windows* RID. If you use the same `[global]` section of the `smb.conf` on every Unix domain member, you will get the same IDs. If you use the 'rid' back-end, you do not need to add anything to AD and RFC2307 attributes will be ignored. When using the 'rid' back-end, set the `template shell` and `template homedir` parameters in `smb.conf`. These settings are global and everyone gets the same login shell and Unix home directory path (unlike the RFC2307 attributes where you can set individual Unix home directory paths and shells).

There is another way of setting up Samba—when you require your users and groups to have the same ID everywhere, but only need your users to have the same login shell and use the same Unix home directory path. You can do this by using the winbind 'ad' back-end and using the template lines in `smb.conf`. This way you only need to add `uidNumber` and `gidNumber` attributes to AD.



Tip: More Information about Back-ends for ID Mapping

Find more detailed information about available ID mapping back-ends in the related manual pages: `man 8 idmap_ad`, `man 8 idmap_rid`, and `man 8 idmap_autorid`.

24.2.6.2 Setting user and group ID ranges

After you decide which winbind back-end to use, you need to specify the ranges to use with the `idmap config` option in `smb.conf`. By default, there are multiple blocks of user and group IDs reserved on a Unix domain member:

TABLE 24.1: DEFAULT USERS AND GROUP ID BLOCKS

IDs	Range
0-999	Local system users and groups.
Starting at 1000	Local Unix users and groups.
Starting at 10000	DOMAIN users and groups.

As you can see from the above ranges, you should not set either the '*' or 'DOMAIN' ranges to start at 999 or less, as they would interfere with the local system users and groups. You also should leave a space for any local Unix users and groups, so starting the `idmap config` ranges at 3000 seems to be a good compromise.

You need to decide how large your 'DOMAIN' is likely to grow and if you plan to have any trusted domains. Then you can set the `idmap config` ranges as follows:

TABLE 24.2: ID RANGES

Domain	Range
*	3000-7999
DOMAIN	10000-999999
TRUSTED	1000000-9999999

24.2.6.3 Mapping the domain administrator account to the local root user

Samba enables you to map domain accounts to a local account. Use this feature to execute file operations on the domain member's file system as a different user than the account that requested the operation on the client.



Tip: Mapping the Domain Administrator (Optional)

Mapping the domain administrator to the local `root` account is optional. Only configure the mapping if the domain administrator needs to be able to execute file operations on the domain member using `root` permissions. Be aware that mapping Administrator to the `root` account does not allow you to log in to Unix domain members as 'Administrator'.

To map the domain administrator to the local `root` account, follow these steps:

1. Add the following parameter to the `[global]` section of your `smb.conf` file:

```
username map = /etc/samba/user.map
```

2. Create the `/etc/samba/user.map` file with the following content:

```
!root = DOMAIN\Administrator
```

! Important

When using the 'ad' ID mapping back-end, do not set the `uidNumber` attribute for the domain administrator account. If the account has the attribute set, the value overrides the local UID '0' of the `root` user, and therefore the mapping fails.

For more details, see the `username map` parameter in the `smb.conf` manual page ([man 5 smb.conf](#)).

24.2.7 Joining the Active Directory domain

To join the host to an Active Directory, run:

```
cephuser@smb > net ads join -U administrator
Enter administrator's password: PASSWORD
Using short domain name -- DOMAIN
Joined EXAMPLE-HOST to dns domain 'DOMAIN.example.com'
```

24.2.8 Configuring the name service switch

To make domain users and groups available to the local system, you need to enable the name service switch (NSS) library. Append the `winbind` entry to the following databases in the `/etc/nsswitch.conf` file:

```
passwd: files winbind
group:  files winbind
```

! Important: Points to Consider

- Keep the `files` entry as the first source for both databases. This enables NSS to look up domain users and groups from the `/etc/passwd` and `/etc/group` files before querying the `winbind` service.
- Do not add the `winbind` entry to the NSS `shadow` database. This can cause the `wbinfo` utility to fail.
- Do not use the same user names in the local `/etc/passwd` file as in the domain.

24.2.9 Starting the services

Following configuration changes, restart Samba services as per [Section 24.1.2.1, "Starting Samba services"](#) or [Section 24.1.3.1, "Restarting HA Samba resources"](#).

24.2.10 Test the winbindd connectivity

24.2.10.1 Sending a winbindd ping

To verify if the `winbindd` service is able to connect to AD Domain Controllers (DC) or a primary domain controller (PDC), enter:

```
cephuser@smb > wbinfo --ping-dc
checking the NETLOGON for domain[DOMAIN] dc connection to "DC.DOMAIN.EXAMPLE.COM"
succeeded
```

If the previous command fails, verify that the `winbindd` service is running and that the `sm-b.conf` file is set up correctly.

24.2.10.2 Looking up domain users and groups

The `libnss_winbind` library enables you to look up domain users and groups. For example, to look up the domain user 'DOMAIN\demo01':

```
cephuser@smb > getent passwd DOMAIN\demo01
DOMAIN\demo01:*:10000:10000:demo01:/home/demo01:/bin/bash
```

To look up the domain group 'Domain Users':

```
cephuser@smb > getent group "DOMAIN\Domain Users"
DOMAIN\domain users:x:10000:
```

24.2.10.3 Assigning file permissions to domain users and groups

The name service switch (NSS) library enables you to use domain user accounts and groups in commands. For example to set the owner of a file to the 'demo01' domain user and the group to the 'Domain Users' domain group, enter:

```
cephuser@smb > chown "DOMAIN\demo01:DOMAIN\domain users" file.txt
```

25 NFS Ganesha

NFS Ganesha is an NFS server that runs in a user address space instead of as part of the operating system kernel. With NFS Ganesha, you can plug in your own storage mechanism—such as Ceph—and access it from any NFS client. For installation instructions, see *Book “Deployment Guide”, Chapter 8 “Deploying the remaining core services using cephadm”, Section 8.3.6 “Deploying NFS Ganesha”*.



Note: NFS Ganesha performance

Because of increased protocol overhead and additional latency caused by extra network hops between the client and the storage, accessing Ceph via an NFS Gateway may significantly reduce application performance when compared to native CephFS.

Each NFS Ganesha service consists of a configuration hierarchy that contains:

- A bootstrap `ganesha.conf`
- A per-service RADOS common configuration object
- A per export RADOS configuration object

The bootstrap configuration is the minimal configuration to start the `nfs-ganesha` daemon within a container. Each bootstrap configuration will contain a `%url` directive that includes any additional configuration from the RADOS common configuration object. The common configuration object can include additional `%url` directives for each of the NFS exports defined in the export RADOS configuration objects.

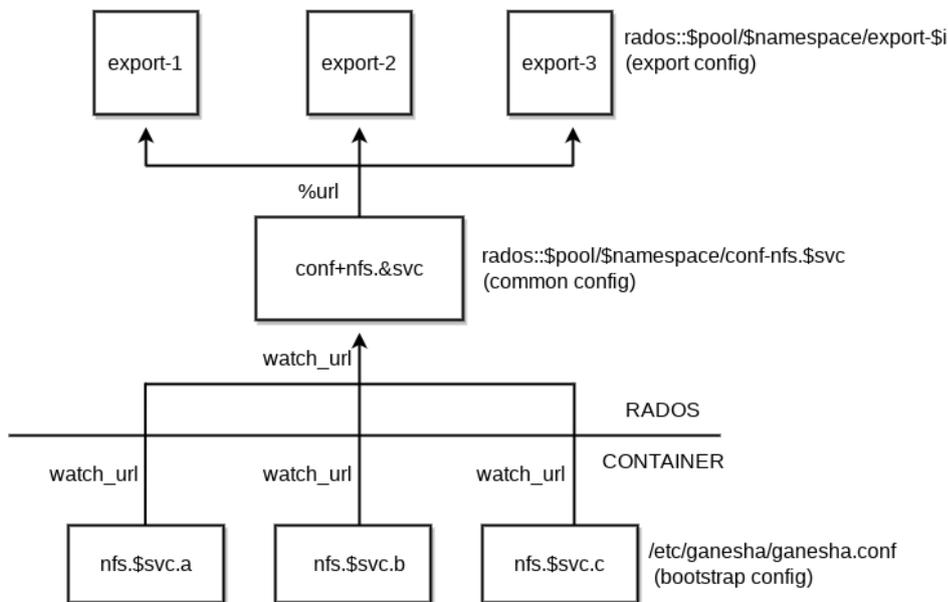


FIGURE 25.1: NFS GANESHA STRUCTURE

25.1 Creating an NFS service

The recommended way to specify the deployment of Ceph services is to create a YAML-formatted file with the specification of the services that you intend to deploy. You can create a separate specification file for each type of service, or you specify multiple (or all) services types in one file. Depending on what you have chosen to do, you will need to update or create a relevant YAML-formatted file to create a NFS Ganesha service. For more information on creating the file, see *Book "Deployment Guide", Chapter 8 "Deploying the remaining core services using cephadm", Section 8.2 "Service and placement specification"*.

Once you have updated or created the file, execute the following to create a `nfs-ganesha` service:

```
cephuser@adm > ceph orch apply -i FILE_NAME
```

25.2 Starting or Restarting NFS Ganesha

Important

Starting the NFS Ganesha service does not automatically export a CephFS file system. To export a CephFS file system, create an export configuration file. Refer to [Section 25.4, “Creating an NFS export”](#) for more details.

To start the NFS Ganesha service, run:

```
cephuser@adm > ceph orch start nfs.SERVICE_ID
```

To restart the NFS Ganesha service, run:

```
cephuser@adm > ceph orch restart nfs.SERVICE_ID
```

If you only want to restart a single NFS Ganesha daemon, run:

```
cephuser@adm > ceph orch daemon restart nfs.SERVICE_ID
```

When NFS Ganesha is started or restarted, it has a grace timeout of 90 seconds for NFS v4. During the grace period, new requests from clients are actively rejected. Hence, clients may face a slowdown of requests when NFS is in the grace period.

25.3 Listing objects in the NFS recovery pool

Execute the following to list the objects in the NFS recovery pool:

```
cephuser@adm > rados --pool POOL_NAME --namespace NAMESPACE_NAME ls
```

25.4 Creating an NFS export

You can create an NFS export either in the Ceph Dashboard, or manually on the command line. To create the export by using the Ceph Dashboard, refer to [Chapter 7, Manage NFS Ganesha](#), more specifically to [Section 7.1, “Creating NFS exports”](#).

To create an NFS export manually, create a configuration file for the export. For example, a file `/tmp/export-1` with the following content:

```
EXPORT {  
    export_id = 1;
```

```

path = "/";
pseudo = "/";
access_type = "RW";
squash = "no_root_squash";
protocols = 3, 4;
transports = "TCP", "UDP";
FSAL {
    name = "CEPH";
    user_id = "admin";
    filesystem = "a";
    secret_access_key = "SECRET_ACCESS_KEY";
}
}

```

After you have created and saved the configuration file for the new export, run the following command to create the export:

```
rados --pool POOL_NAME --namespace NAMESPACE_NAME put EXPORT_NAME EXPORT_CONFIG_FILE
```

For example:

```
cephuser@adm > rados --pool example_pool --namespace example_namespace put export-1 /tmp/export-1
```



Note

The FSAL block should be modified to include the desired `cephx` user ID and secret access key.

25.5 Verifying the NFS export

NFS v4 will build a list of exports at the root of a pseudo file system. You can verify that the NFS shares are exported by mounting `/` of the NFS Ganesha server node:

```
# mount -t nfs nfs_ganesha_server_hostname:/ /path/to/local/mountpoint
# ls /path/to/local/mountpoint cephfs
```



Note: NFS Ganesha is v4 only

By default, `cephadm` will configure an NFS v4 server. NFS v4 does not interact with `rpcbind` nor the `mountd` daemon. NFS client tools such as `showmount` will not show any configured exports.

25.6 Mounting the NFS export

To mount the exported NFS share on a client host, run:

```
# mount -t nfs nfs_ganesha_server_hostname:/ /path/to/local/mountpoint
```

25.7 Multiple NFS Ganesha clusters

Multiple NFS Ganesha clusters can be defined. This allows for:

- Separated NFS Ganesha clusters for accessing CephFS.

V Integration with Virtualization Tools

26 libvirt and Ceph 353

27 Ceph as a back-end for QEMU KVM instance 359

26 libvirt and Ceph

The `libvirt` library creates a virtual machine abstraction layer between hypervisor interfaces and the software applications that use them. With `libvirt`, developers and system administrators can focus on a common management framework, common API, and common shell interface (`virsh`) to many different hypervisors, including QEMU/KVM, Xen, LXC, or VirtualBox.

Ceph block devices support QEMU/KVM. You can use Ceph block devices with software that interfaces with `libvirt`. The cloud solution uses `libvirt` to interact with QEMU/KVM, and QEMU/KVM interacts with Ceph block devices via `librbd`.

To create VMs that use Ceph block devices, use the procedures in the following sections. In the examples, we have used `libvirt-pool` for the pool name, `client.libvirt` for the user name, and `new-libvirt-image` for the image name. You may use any value you like, but ensure you replace those values when executing commands in the subsequent procedures.

26.1 Configuring Ceph with libvirt

To configure Ceph for use with `libvirt`, perform the following steps:

1. Create a pool. The following example uses the pool name `libvirt-pool` with 128 placement groups.

```
cephuser@adm > ceph osd pool create libvirt-pool 128 128
```

Verify that the pool exists.

```
cephuser@adm > ceph osd lspools
```

2. Create a Ceph User. The following example uses the Ceph user name `client.libvirt` and references `libvirt-pool`.

```
cephuser@adm > ceph auth get-or-create client.libvirt mon 'profile rbd' osd \
'profile rbd pool=libvirt-pool'
```

Verify the name exists.

```
cephuser@adm > ceph auth list
```



Note: User name or ID

`libvirt` will access Ceph using the ID `libvirt`, not the Ceph name `client.libvirt`. See [Section 30.2.1.1, "User"](#) for a detailed explanation of the difference between ID and name.

3. Use QEMU to create an image in your RBD pool. The following example uses the image name `new-libvirt-image` and references `libvirt-pool`.



Tip: Keyring file location

The `libvirt` user key is stored in a keyring file placed in the `/etc/ceph` directory. The keyring file needs to have an appropriate name that includes the name of the Ceph cluster it belongs to. For the default cluster name 'ceph', the keyring file name is `/etc/ceph/ceph.client.libvirt.keyring`.

If the keyring does not exist, create it with:

```
cephuser@adm > ceph auth get client.libvirt > /etc/ceph/  
ceph.client.libvirt.keyring
```

```
# qemu-img create -f raw rbd:libvirt-pool/new-libvirt-image:id=libvirt 2G
```

Verify the image exists.

```
cephuser@adm > rbd -p libvirt-pool ls
```

26.2 Preparing the VM manager

You may use `libvirt` without a VM manager, but you may find it simpler to create your first domain with `virt-manager`.

1. Install a virtual machine manager.

```
# zypper in virt-manager
```

2. Prepare/download an OS image of the system you want to run virtualized.

3. Launch the virtual machine manager.

```
virt-manager
```

26.3 Creating a VM

To create a VM with `virt-manager`, perform the following steps:

1. Choose the connection from the list, right-click it, and select *New*.
2. *Import existing disk image* by providing the path to the existing storage. Specify OS type, memory settings, and *Name* the virtual machine, for example `libvirt-virtual-machine`.
3. Finish the configuration and start the VM.
4. Verify that the newly created domain exists with `sudo virsh list`. If needed, specify the connection string, such as

```
virsh -c qemu+ssh://root@vm_host_hostname/system list
Id      Name                                State
-----
[...]
 9      libvirt-virtual-machine             running
```

5. Log in to the VM and stop it before configuring it for use with Ceph.

26.4 Configuring the VM

In this chapter, we focus on configuring VMs for integration with Ceph using `virsh`. `virsh` commands often require root privileges (`sudo`) and will not return appropriate results or notify you that root privileges are required. For a reference of `virsh` commands, refer to `man 1 virsh` (requires the package `libvirt-client` to be installed).

1. Open the configuration file with `virsh edit vm-domain-name`.

```
# virsh edit libvirt-virtual-machine
```

2. Under `<devices>` there should be a `<disk>` entry.

```
<devices>
  <emulator>/usr/bin/qemu-system-SYSTEM-ARCH</emulator>
  <disk type='file' device='disk'>
    <driver name='qemu' type='raw' />
    <source file='/path/to/image/recent-linux.img' />
    <target dev='vda' bus='virtio' />
    <address type='drive' controller='0' bus='0' unit='0' />
  </disk>
```

Replace `/path/to/image/recent-linux.img` with the path to the OS image.

Important

Use `sudo virsh edit` instead of a text editor. If you edit the configuration file under `/etc/libvirt/qemu` with a text editor, `libvirt` may not recognize the change. If there is a discrepancy between the contents of the XML file under `/etc/libvirt/qemu` and the result of `sudo virsh dumpxml vm-domain-name`, then your VM may not work properly.

3. Add the Ceph RBD image you previously created as a `< disk >` entry.

```
<disk type='network' device='disk'>
  <source protocol='rbd' name='libvirt-pool/new-libvirt-image'>
    <host name='monitor-host' port='6789' />
  </source>
  <target dev='vda' bus='virtio' />
</disk>
```

Replace `monitor-host` with the name of your host, and replace the pool and/or image name as necessary. You may add multiple `< host >` entries for your Ceph monitors. The `dev` attribute is the logical device name that will appear under the `/dev` directory of your VM. The optional bus attribute indicates the type of disk device to emulate. The valid settings are driver specific (for example `ide`, `scsi`, `virtio`, `xen`, `usb` or `sata`).

4. Save the file.
5. If your Ceph cluster has authentication enabled (it does by default), you must generate a secret. Open an editor of your choice and create a file called `secret.xml` with the following content:

```
<secret ephemeral='no' private='no'>
  <usage type='ceph'>
```

```
        <name>client.libvirt secret</name>
    </usage>
</secret>
```

6. Define the secret.

```
# virsh secret-define --file secret.xml
<uuid of secret is output here>
```

7. Get the `client.libvirt` key and save the key string to a file.

```
cephuser@adm > ceph auth get-key client.libvirt | sudo tee client.libvirt.key
```

8. Set the UUID of the secret.

```
# virsh secret-set-value --secret uuid of secret \
--base64 $(cat client.libvirt.key) && rm client.libvirt.key secret.xml
```

You must also set the secret manually by adding the following `<auth>` entry to the `<disk>` element you entered earlier (replacing the uuid value with the result from the command line example above).

```
# virsh edit libvirt-virtual-machine
```

Then, add `<auth></auth>` element to the domain configuration file:

```
...
</source>
<auth username='libvirt'>
    <secret type='ceph' uuid='9ec59067-fdbc-a6c0-03ff-df165c0587b8' />
</auth>
<target ...
```



Note

The exemplary ID is `libvirt`, not the Ceph name `client.libvirt` as generated at step 2 of [Section 26.1, “Configuring Ceph with libvirt”](#). Ensure you use the ID component of the Ceph name you generated. If for some reason you need to regenerate the secret, you will need to execute `sudo virsh secret-undefine uuid` before executing `sudo virsh secret-set-value` again.

26.5 Summary

Once you have configured the VM for use with Ceph, you can start the VM. To verify that the VM and Ceph are communicating, you may perform the following procedures.

1. Check to see if Ceph is running:

```
cephuser@adm > ceph health
```

2. Check to see if the VM is running:

```
# virsh list
```

3. Check to see if the VM is communicating with Ceph. Replace vm-domain-name with the name of your VM domain:

```
# virsh qemu-monitor-command --hmp vm-domain-name 'info block'
```

4. Check to see if the device from &target dev='hdb' bus='ide' /> appears under /dev or under /proc/partitions:

```
> ls /dev  
> cat /proc/partitions
```

27 Ceph as a back-end for QEMU KVM instance

The most frequent Ceph use case involves providing block device images to virtual machines. For example, a user may create a 'golden' image with an OS and any relevant software in an ideal configuration. Then, the user takes a snapshot of the image. Finally, the user clones the snapshot (usually many times, see [Section 20.3, "Snapshots"](#) for details). The ability to make copy-on-write clones of a snapshot means that Ceph can provision block device images to virtual machines quickly, because the client does not need to download an entire image each time it spins up a new virtual machine.

Ceph block devices can integrate with the QEMU virtual machines. For more information on QEMU KVM, see <https://documentation.suse.com/sles/15-SP2/html/SLES-all/part-virt-qemu.html>.

27.1 Installing `qemu-block-rbd`

In order to use Ceph block devices, QEMU needs to have the appropriate driver installed. Check whether the `qemu-block-rbd` package is installed, and install it if needed:

```
# zypper install qemu-block-rbd
```

27.2 Using QEMU

The QEMU command line expects you to specify the pool name and image name. You may also specify a snapshot name.

```
qemu-img command options \  
rbd:pool-name/image-name@snapshot-name:option1=value1:option2=value2...
```

For example, specifying the `id` and `conf` options might look like the following:

```
qemu-img command options \  
rbd:pool_name/image_name:id=glance:conf=/etc/ceph/ceph.conf
```

27.3 Creating images with QEMU

You can create a block device image from QEMU. You must specify `rbd`, the pool name, and the name of the image you want to create. You must also specify the size of the image.

```
qemu-img create -f raw rbd:pool-name/image-name size
```

For example:

```
qemu-img create -f raw rbd:pool1/image1 10G
Formatting 'rbd:pool1/image1', fmt=raw size=10737418240 nocow=off cluster_size=0
```

Important

The raw data format is really the only sensible format option to use with RBD. Technically, you could use other QEMU-supported formats such as qcow2, but doing so would add additional overhead, and would also render the volume unsafe for virtual machine live migration when caching is enabled.

27.4 Resizing images with QEMU

You can resize a block device image from QEMU. You must specify rbd, the pool name, and the name of the image you want to resize. You must also specify the size of the image.

```
qemu-img resize rbd:pool-name/image-name size
```

For example:

```
qemu-img resize rbd:pool1/image1 9G
Image resized.
```

27.5 Retrieving image info with QEMU

You can retrieve block device image information from QEMU. You must specify rbd, the pool name, and the name of the image.

```
qemu-img info rbd:pool-name/image-name
```

For example:

```
qemu-img info rbd:pool1/image1
image: rbd:pool1/image1
file format: raw
virtual size: 9.0G (9663676416 bytes)
```

```
disk size: unavailable
cluster_size: 4194304
```

27.6 Running QEMU with RBD

QEMU can access an image as a virtual block device directly via `librbd`. This avoids an additional context switch, and can take advantage of RBD caching.

You can use `qemu-img` to convert existing virtual machine images to Ceph block device images. For example, if you have a qcow2 image, you could run:

```
qemu-img convert -f qcow2 -O raw sles12.qcow2 rbd:pool1/sles12
```

To run a virtual machine booting from that image, you could run:

```
# qemu -m 1024 -drive format=raw,file=rbd:pool1/sles12
```

RBD caching can significantly improve performance. QEMU's cache options control `librbd` caching:

```
# qemu -m 1024 -drive format=rbd,file=rbd:pool1/sles12,cache=writeback
```

For more information on RBD caching, refer to [Section 20.5, "Cache settings"](#).

27.7 Enabling discard and TRIM

Ceph block devices support the discard operation. This means that a guest can send TRIM requests to let a Ceph block device reclaim unused space. This can be enabled in the guest by mounting `XFS` with the discard option.

For this to be available to the guest, it must be explicitly enabled for the block device. To do this, you must specify a `discard_granularity` associated with the drive:

```
# qemu -m 1024 -drive format=raw,file=rbd:pool1/sles12,id=drive1,if=none \
-device driver=ide-hd,drive=drive1,discard_granularity=512
```



Note

The above example uses the IDE driver. The virtio driver does not support discard.

If using `libvirt`, edit your libvirt domain's configuration file using `virsh edit` to include the `xmlns:qemu` value. Then, add a `qemu:commandline` block as a child of that domain. The following example shows how to set two devices with `qemu id=` to different `discard_granularity` values.

```
<domain type='kvm' xmlns:qemu='http://libvirt.org/schemas/domain/qemu/1.0'>
  <qemu:commandline>
    <qemu:arg value='-set' />
    <qemu:arg value='block.scsi0-0-0.discard_granularity=4096' />
    <qemu:arg value='-set' />
    <qemu:arg value='block.scsi0-0-1.discard_granularity=65536' />
  </qemu:commandline>
</domain>
```

27.8 Setting QEMU cache options

QEMU's cache options correspond to the following Ceph RBD Cache settings.

Writeback:

```
rbd_cache = true
```

Writethrough:

```
rbd_cache = true
rbd_cache_max_dirty = 0
```

None:

```
rbd_cache = false
```

QEMU's cache settings override Ceph's default settings (settings that are not explicitly set in the Ceph configuration file). If you explicitly set RBD Cache settings in your Ceph configuration file (refer to [Section 20.5, "Cache settings"](#)), your Ceph settings override the QEMU cache settings. If you set cache settings on the QEMU command line, the QEMU command line settings override the Ceph configuration file settings.

VI Configuring a Cluster

- 28 Ceph cluster configuration **364**
- 29 Ceph Manager modules **383**
- 30 Authentication with cephx **388**

28 Ceph cluster configuration

This chapter describes how to configure the Ceph cluster by means of configuration options.

28.1 Configure the `ceph.conf` file

`cephadm` uses a basic `ceph.conf` file that only contains a minimal set of options for connecting to MONs, authenticating, and fetching configuration information. In most cases, this is limited to the `mon_host` option (although this can be avoided through the use of DNS SRV records).

Important

The `ceph.conf` file no longer serves as a central place for storing cluster configuration, in favor of the configuration database (see [Section 28.2, “Configuration database”](#)).

If you still need to change cluster configuration via the `ceph.conf` file—for example, because you use a client that does not support reading options from the configuration database—you need to run the following command, and take care of maintaining and distributing the `ceph.conf` file across the whole cluster:

```
cephuser@adm > ceph config set mgr mgr/cephadm/manage_etc_ceph_ceph_conf false
```

28.1.1 Accessing `ceph.conf` inside container images

Although Ceph daemons run inside containers, you can still access their `ceph.conf` configuration file. It is *bind-mounted* as the following file on the host system:

```
/var/lib/ceph/CLUSTER_FSID/DAEMON_NAME/config
```

Replace `CLUSTER_FSID` with the unique FSID of the running cluster as returned by the `ceph fsid` command, and `DAEMON_NAME` with the name of the specific daemon as listed by the `ceph orch ps` command. For example:

```
/var/lib/ceph/b4b30c6e-9681-11ea-ac39-525400d7702d/osd.2/config
```

To modify the configuration of a daemon, edit its `config` file and restart it:

```
# systemctl restart ceph-CLUSTER_FSID-DAEMON_NAME
```

For example:

```
# systemctl restart ceph-b4b30c6e-9681-11ea-ac39-525400d7702d-osd.2
```



Important

All custom settings will be lost after cephadm redeploys the daemon.

28.2 Configuration database

Ceph Monitors manage a central database of configuration options that affect the behavior of the whole cluster.

28.2.1 Configuring sections and masks

Configuration options stored by the MON can live in a *global* section, *daemon type* section, or a *specific daemon* section. In addition, options may also have a *mask* associated with them to further restrict to which daemons or clients the option applies. Masks have two forms:

- *TYPE:LOCATION* where *TYPE* is a CRUSH property such as *rack* or *host*, while *LOCATION* is a value for that property.
For example, *host:example_host* will limit the option only to daemons or clients running on a particular host.
- *CLASS:DEVICE_CLASS* where *DEVICE_CLASS* is the name of a CRUSH device class such as *hdd* or *ssd*. For example, *class:ssd* will limit the option only to OSDs backed by SSDs. This mask has no effect for non-OSD daemons or clients.

28.2.2 Setting and reading configuration options

Use the following commands to set or read cluster configuration options. The *WHO* parameter may be a section name, a mask, or a combination of both separated by a slash (/) character. For example, *osd/rack:foo* represents all OSD daemons in the rack called *foo*.

ceph config dump

Dumps the entire configuration database for a whole cluster.

`ceph config get WHO`

Dumps the configuration for a specific daemon or client (for example, `mds.a`), as stored in the configuration database.

`ceph config set WHO OPTION VALUE`

Sets the configuration option to the specified value in the configuration database.

`ceph config show WHO`

Shows the reported running configuration for a running daemon. These settings may differ from those stored by the monitors if there are also local configuration files in use, or options have been overridden on the command line or at runtime. The source of the option values is reported as part of the output.

`ceph config assimilate-conf -i INPUT_FILE -o OUTPUT_FILE`

Imports a configuration file specified as `INPUT_FILE` and stores any valid options into the configuration database. Any settings that are unrecognized, invalid, or cannot be controlled by the monitor will be returned in an abbreviated file stored as `OUTPUT_FILE`. This command is useful for transitioning from legacy configuration files to centralized monitor-based configuration.

28.2.3 Configuring daemons at runtime

In most cases, Ceph allows you to make changes to the configuration of a daemon at runtime. This is useful, for example, when you need to increase or decrease the amount of logging output, or when performing runtime cluster optimization.

You can update the values of configuration options with the following command:

```
cephuser@adm > ceph config set DAEMON OPTION VALUE
```

For example, to adjust the debugging log level on a specific OSD, run:

```
cephuser@adm > ceph config set osd.123 debug_ms 20
```



Note

If the same option is also customized in a local configuration file, the monitor setting will be ignored because it has lower priority than the configuration file.

28.2.3.1 Overriding values

You can temporarily modify an option value using the `tell` or `daemon` subcommands. Such modification only affect the running process and is discarded after the daemon or process restarts.

There are two ways to override values:

- Use the `tell` subcommand to send a message to a specific daemon from any cluster node:

```
cephuser@adm > ceph tell DAEMON config set OPTION VALUE
```

For example:

```
cephuser@adm > ceph tell osd.123 config set debug_osd 20
```



Tip

The `tell` subcommand accepts wild cards as daemon identifiers. For example, to adjust the debug level on all OSD daemons, run:

```
cephuser@adm > ceph tell osd.* config set debug_osd 20
```

- Use the `daemon` subcommand to connect to a specific daemon process via a socket in `/var/run/ceph` from the node where the process is running:

```
cephuser@adm > cephadm enter --name osd.ID -- ceph daemon DAEMON config  
set OPTION VALUE
```

For example:

```
cephuser@adm > cephadm enter --name osd.4 -- ceph daemon osd.4 config set debug_osd  
20
```



Tip

When viewing runtime settings with the `ceph config show` command (see [Section 28.2.3.2, “Viewing runtime settings”](#)), temporarily overridden values will be shown with a source override.

28.2.3.2 Viewing runtime settings

To view all options set for a daemon:

```
cephuser@adm > ceph config show-with-defaults osd.0
```

To view all non-default options set for a daemon:

```
cephuser@adm > ceph config show osd.0
```

To inspect a specific option:

```
cephuser@adm > ceph config show osd.0 debug_osd
```

You can also connect to a running daemon from the node where its process is running, and observe its configuration:

```
cephuser@adm > cephadm enter --name osd.0 -- ceph daemon osd.0 config show
```

To view only non-default settings:

```
cephuser@adm > cephadm enter --name osd.0 -- ceph daemon osd.0 config diff
```

To inspect a specific option:

```
cephuser@adm > cephadm enter --name osd.0 -- ceph daemon osd.0 config get debug_osd
```

28.3 config-key store

`config-key` is a general-purpose service offered by the Ceph Monitors. It simplifies managing configuration keys by storing key-value pairs persistently. `config-key` is mainly used by Ceph tools and daemons.



Tip

After you add a new key or modify an existing one, restart the affected service for the changes to take effect. Find more details about operating Ceph services in [Chapter 14, Operation of Ceph services](#).

Use the `config-key` command to operate the `config-key` store. The `config-key` command uses the following subcommands:

`ceph config-key rm KEY`

Deletes the specified key.

ceph config-key exists KEY

Checks for the existence of the specified key.

ceph config-key get KEY

Retrieves the value of the specified key.

ceph config-key ls

Lists all keys.

ceph config-key dump

Dumps all keys and their values.

ceph config-key set KEY VALUE

Stores the specified key with the given value.

28.3.1 iSCSI Gateway

The iSCSI Gateway uses the config-key store to save or read its configuration options. All iSCSI Gateway related keys are prefixed with the iscsi string, for example:

```
iscsi/trusted_ip_list
iscsi/api_port
iscsi/api_user
iscsi/api_password
iscsi/api_secure
```

If you need, for example, two sets of configuration options, extend the prefix with another descriptive keyword, for example datacenterA and datacenterB:

```
iscsi/datacenterA/trusted_ip_list
iscsi/datacenterA/api_port
[...]
iscsi/datacenterB/trusted_ip_list
iscsi/datacenterB/api_port
[...]
```

28.4 Ceph OSD and BlueStore

28.4.1 Configuring automatic cache sizing

BlueStore can be configured to automatically resize its caches when `tc_malloc` is configured as the memory allocator and the `bluestore_cache_autotune` setting is enabled. This option is currently enabled by default. BlueStore will attempt to keep OSD heap memory usage under a designated target size via the `osd_memory_target` configuration option. This is a best effort algorithm and caches will not shrink smaller than the amount specified by `osd_memory_cache_min`. Cache ratios will be chosen based on a hierarchy of priorities. If priority information is not available, the `bluestore_cache_meta_ratio` and `bluestore_cache_kv_ratio` options are used as fallbacks.

`bluestore_cache_autotune`

Automatically tunes the ratios assigned to different BlueStore caches while respecting minimum values. Default is `True`.

`osd_memory_target`

When `tc_malloc` and `bluestore_cache_autotune` are enabled, try to keep this many bytes mapped in memory.



Note

This may not exactly match the RSS memory usage of the process. While the total amount of heap memory mapped by the process should generally stay close to this target, there is no guarantee that the kernel will actually reclaim memory that has been unmapped.

`osd_memory_cache_min`

When `tc_malloc` and `bluestore_cache_autotune` are enabled, set the minimum amount of memory used for caches.



Note

Setting this value too low can result in significant cache thrashing.

28.5 Ceph Object Gateway

You can influence the Object Gateway behavior by a number of options. If an option is not specified, its default value is used. A complete list of the Object Gateway options follows:

28.5.1 General Settings

`rgw_frontends`

Configures the HTTP front-end(s). Specify multiple front-ends in a comma-delimited list. Each front-end configuration may include a list of options separated by spaces, where each option is in the form “key = value” or “key”. Default is `beast port=7480`.

`rgw_data`

Sets the location of the data files for the Object Gateway. Default is `/var/lib/ceph/radosgw/CLUSTER_ID`.

`rgw_enable_apis`

Enables the specified APIs. Default is 's3, swift, swift_auth, admin All APIs'.

`rgw_cache_enabled`

Enables or disables the Object Gateway cache. Default is `true`.

`rgw_cache_lru_size`

The number of entries in the Object Gateway cache. Default is 10000.

`rgw_socket_path`

The socket path for the domain socket. `FastCgiExternalServer` uses this socket. If you do not specify a socket path, the Object Gateway will not run as an external server. The path you specify here needs to be the same as the path specified in the `rgw.conf` file.

`rgw_fcgi_socket_backlog`

The socket backlog for fcgi. Default is 1024.

`rgw_host`

The host for the Object Gateway instance. It can be an IP address or a host name. Default is `0.0.0.0`.

`rgw_port`

The port number where the instance listens for requests. If not specified, the Object Gateway runs external FastCGI.

rgw_dns_name

The DNS name of the served domain.

rgw_script_uri

The alternative value for the SCRIPT_URI if not set in the request.

rgw_request_uri

The alternative value for the REQUEST_URI if not set in the request.

rgw_print_continue

Enable 100-continue if it is operational. Default is true.

rgw_remote_addr_param

The remote address parameter. For example, the HTTP field containing the remote address, or the X-Forwarded-For address if a reverse proxy is operational. Default is REMOTE_ADDR.

rgw_op_thread_timeout

The timeout in seconds for open threads. Default is 600.

rgw_op_thread_suicide_timeout

The time timeout in seconds before the Object Gateway process dies. Disabled if set to 0 (default).

rgw_thread_pool_size

Number of threads for the Beast server. Increase to a higher value if you need to serve more requests. Defaults to 100 threads.

rgw_num_rados_handles

The number of RADOS cluster handles for Object Gateway. Each Object Gateway worker thread now gets to pick a RADOS handle for its lifetime. This option may be deprecated and removed in future releases. Default is 1.

rgw_num_control_oids

The number of notification objects used for cache synchronization between different Object Gateway instances. Default is 8.

rgw_init_timeout

The number of seconds before the Object Gateway gives up on initialization. Default is 30.

rgw_mime_types_file

The path and location of the MIME types. Used for Swift auto-detection of object types. Default is /etc/mime.types.

rgw_gc_max_objs

The maximum number of objects that may be handled by garbage collection in one garbage collection processing cycle. Default is 32.

rgw_gc_obj_min_wait

The minimum wait time before the object may be removed and handled by garbage collection processing. Default is $2 * 3600$.

rgw_gc_processor_max_time

The maximum time between the beginning of two consecutive garbage collection processing cycles. Default is 3600.

rgw_gc_processor_period

The cycle time for garbage collection processing. Default is 3600.

rgw_s3_success_create_obj_status

The alternate success status response for `create-obj`. Default is 0.

rgw_resolve_cname

Whether the Object Gateway should use DNS CNAME record of the request host name field (if host name is not equal to the Object Gateway DNS name). Default is `false`.

rgw_obj_stripe_size

The size of an object stripe for Object Gateway objects. Default is $4 \ll 20$.

rgw_extended_http_attrs

Add a new set of attributes that can be set on an entity (for example, a user, a bucket, or an object). These extra attributes can be set through HTTP header fields when putting the entity or modifying it using the POST method. If set, these attributes will return as HTTP fields when requesting GET/HEAD on the entity. Default is `content_foo, content_bar, x-foo-bar`.

rgw_exit_timeout_secs

Number of seconds to wait for a process before exiting unconditionally. Default is 120.

rgw_get_obj_window_size

The window size in bytes for a single object request. Default is $16 \ll 20$.

rgw_get_obj_max_req_size

The maximum request size of a single GET operation sent to the Ceph Storage Cluster. Default is $4 \ll 20$.

`rgw_relaxed_s3_bucket_names`

Enables relaxed S3 bucket name rules for US region buckets. Default is `false`.

`rgw_list_buckets_max_chunk`

The maximum number of buckets to retrieve in a single operation when listing user buckets. Default is 1000.

`rgw_override_bucket_index_max_shards`

Represents the number of shards for the bucket index object. Setting 0 (default) indicates there is no sharding. It is not recommended to set a value too large (for example 1000) as it increases the cost for bucket listing. This variable should be set in the client or global sections so that it is automatically applied to `radosgw-admin` commands.

`rgw_curl_wait_timeout_ms`

The timeout in milliseconds for certain `curl` calls. Default is 1000.

`rgw_copy_obj_progress`

Enables output of object progress during long copy operations. Default is `true`.

`rgw_copy_obj_progress_every_bytes`

The minimum bytes between copy progress output. Default is $1024 * 1024$.

`rgw_admin_entry`

The entry point for an admin request URL. Default is `admin`.

`rgw_content_length_compat`

Enable compatibility handling of FCGI requests with both `CONTENT_LENGTH` AND `HTTP_CONTENT_LENGTH` set. Default is `false`.

`rgw_bucket_quota_ttl`

The amount of time in seconds for which cached quota information is trusted. After this timeout, the quota information will be re-fetched from the cluster. Default is 600.

`rgw_user_quota_bucket_sync_interval`

The amount of time in seconds for which the bucket quota information is accumulated before synchronizing to the cluster. During this time, other Object Gateway instances will not see the changes in the bucket quota stats related to operations on this instance. Default is 180.

`rgw_user_quota_sync_interval`

The amount of time in seconds for which user quota information is accumulated before synchronizing to the cluster. During this time, other Object Gateway instances will not see the changes in the user quota stats related to operations on this instance. Default is 180.

`rgw_bucket_default_quota_max_objects`

Default maximum number of objects per bucket. It is set on new users if no other quota is specified, and has no effect on existing users. This variable should be set in the client or global sections so that it is automatically applied to `radosgw-admin` commands. Default is -1.

`rgw_bucket_default_quota_max_size`

Default maximum capacity per bucket in bytes. It is set on new users if no other quota is specified, and has no effect on existing users. Default is -1.

`rgw_user_default_quota_max_objects`

Default maximum number of objects for a user. This includes all objects in all buckets owned by the user. It is set on new users if no other quota is specified, and has no effect on existing users. Default is -1.

`rgw_user_default_quota_max_size`

The value for user maximum size quota in bytes set on new users if no other quota is specified. It has no effect on existing users. Default is -1.

`rgw_verify_ssl`

Verify SSL certificates while making requests. Default is `true`.

`rgw_max_chunk_size`

Maximum size of a chunk of data that will be read in a single operation. Increasing the value to 4 MB (4194304) will provide better performance when processing large objects. Default is 128 kB (131072).

MULTISITE SETTINGS

`rgw_zone`

The name of the zone for the gateway instance. If no zone is set, a cluster-wide default can be configured with the `radosgw-admin zone default` command.

`rgw_zonegroup`

The name of the zonegroup for the gateway instance. If no zonegroup is set, a cluster-wide default can be configured with the `radosgw-admin zonegroup default` command.

`rgw_realm`

The name of the realm for the gateway instance. If no realm is set, a cluster-wide default can be configured with the `radosgw-admin realm default` command.

`rgw_run_sync_thread`

If there are other zones in the realm to synchronize from, spawn threads to handle the synchronization of data and metadata. Default is `true`.

`rgw_data_log_window`

The data log entries window in seconds. Default is 30.

`rgw_data_log_changes_size`

The number of in-memory entries to hold for the data changes log. Default is 1000.

`rgw_data_log_obj_prefix`

The object name prefix for the data log. Default is 'data_log'.

`rgw_data_log_num_shards`

The number of shards (objects) on which to keep the data changes log. Default is 128.

`rgw_md_log_max_shards`

The maximum number of shards for the metadata log. Default is 64.

SWIFT SETTINGS

`rgw_enforce_swift_acls`

Enforces the Swift Access Control List (ACL) settings. Default is `true`.

`rgw_swift_token_expiration`

The time in seconds for expiring a Swift token. Default is $24 * 3600$.

`rgw_swift_url`

The URL for the Ceph Object Gateway Swift API.

`rgw_swift_url_prefix`

The URL prefix for the Swift StorageURL that goes in front of the '/v1' part. This allows to run several Gateway instances on the same host. For compatibility, setting this configuration variable to empty causes the default '/swift' to be used. Use explicit prefix '/' to start StorageURL at the root.



Warning

Setting this option to `'/'` will not work if S3 API is enabled. Keep in mind that disabling S3 will make it impossible to deploy the Object Gateway in the multisite configuration.

`rgw_swift_auth_url`

Default URL for verifying v1 authentication tokens when the internal Swift authentication is not used.

`rgw_swift_auth_entry`

The entry point for a Swift authentication URL. Default is `auth`.

`rgw_swift_versioning_enabled`

Enables the Object Versioning of OpenStack Object Storage API. This allows clients to put the `X-Versions-Location` attribute on containers that should be versioned. The attribute specifies the name of container storing archived versions. It must be owned by the same user as the versioned container for reasons of access control verification—ACLs are *not* taken into consideration. Those containers cannot be versioned by the S3 object versioning mechanism. Default is `false`.

LOGGING SETTINGS

`rgw_log_nonexistent_bucket`

Enables the Object Gateway to log a request for a non-existent bucket. Default is `false`.

`rgw_log_object_name`

The logging format for an object name. See the manual page `man 1 date` for details about format specifiers. Default is `%Y-%m-%d-%H-%i-%n`.

`rgw_log_object_name_utc`

Whether a logged object name includes a UTC time. If set to `false` (default), it uses the local time.

`rgw_usage_max_shards`

The maximum number of shards for usage logging. Default is 32.

`rgw_usage_max_user_shards`

The maximum number of shards used for a single user's usage logging. Default is 1.

`rgw_enable_ops_log`

Enable logging for each successful Object Gateway operation. Default is `false`.

`rgw_enable_usage_log`

Enable the usage log. Default is `false`.

`rgw_ops_log_rados`

Whether the operations log should be written to the Ceph Storage Cluster back-end. Default is `true`.

`rgw_ops_log_socket_path`

The Unix domain socket for writing operations logs.

`rgw_ops_log_data_backlog`

The maximum data backlog data size for operations logs written to a Unix domain socket. Default is 5 << 20.

`rgw_usage_log_flush_threshold`

The number of dirty merged entries in the usage log before flushing synchronously. Default is 1024.

`rgw_usage_log_tick_interval`

Flush pending usage log data every 'n' seconds. Default is 30.

`rgw_log_http_headers`

Comma-delimited list of HTTP headers to include in log entries. Header names are case-insensitive, and use the full header name with words separated by underscores. For example, 'http_x_forwarded_for', 'http_x_special_k'.

`rgw_intent_log_object_name`

The logging format for the intent log object name. See the manual page [man 1 date](#) for details about format specifiers. Default is '%Y-%m-%d-%i-%n'.

`rgw_intent_log_object_name_utc`

Whether the intent log object name includes a UTC time. If set to `false` (default), it uses the local time.

KEYSTONE SETTINGS

`rgw_keystone_url`

The URL for the Keystone server.

`rgw_keystone_api_version`

The version (2 or 3) of OpenStack Identity API that should be used for communication with the Keystone server. Default is 2.

`rgw_keystone_admin_domain`

The name of the OpenStack domain with the administrator privilege when using OpenStack Identity API v3.

`rgw_keystone_admin_project`

The name of the OpenStack project with the administrator privilege when using OpenStack Identity API v3. If not set, the value of the `rgw_keystone_admin_tenant` will be used instead.

`rgw_keystone_admin_token`

The Keystone administrator token (shared secret). In the Object Gateway, authentication with the administrator token has priority over authentication with the administrator credentials (options `rgw_keystone_admin_user`, `rgw_keystone_admin_password`, `rgw_keystone_admin_tenant`, `rgw_keystone_admin_project`, and `rgw_keystone_admin_domain`). The administrator token feature is considered as deprecated.

`rgw_keystone_admin_tenant`

The name of the OpenStack tenant with the administrator privilege (Service Tenant) when using OpenStack Identity API v2.

`rgw_keystone_admin_user`

The name of the OpenStack user with the administrator privilege for Keystone authentication (Service User) when using OpenStack Identity API v2.

`rgw_keystone_admin_password`

The password for the OpenStack administrator user when using OpenStack Identity API v2.

`rgw_keystone_accepted_roles`

The roles required to serve requests. Default is 'Member, admin'.

`rgw_keystone_token_cache_size`

The maximum number of entries in each Keystone token cache. Default is 10000.

`rgw_keystone_revocation_interval`

The number of seconds between token revocation checks. Default is 15 * 60.

`rgw_keystone_verify_ssl`

Verify SSL certificates while making token requests to Keystone. Default is `true`.

28.5.1.1 Additional notes

rgw_dns_name

Allows clients to use vhost -style buckets.

vhost -style access refers to the use of *bucketname.s3-endpoint/object-path*. This is in comparison to path -style access: *s3-endpoint/bucket/object*

If the rgw dns name is set, verify that the S3 client is configured to direct requests to the endpoint specified by rgw dns name.

28.5.2 Configuring HTTP front-ends

28.5.2.1 Beast

port, ssl_port

IPv4 & IPv6 listening port numbers. You can specify multiple port numbers:

```
port=80 port=8000 ssl_port=8080
```

Default is 80.

endpoint, ssl_endpoint

The listening addresses in the form 'address[:port]', where the address is an IPv4 address string in dotted decimal form, or an IPv6 address in hexadecimal notation surrounded by square brackets. Specifying an IPv6 endpoint would listen to IPv6 only. The optional port number defaults to 80 for endpoint and 443 for ssl_endpoint. You can specify multiple addresses:

```
endpoint=[::1] endpoint=192.168.0.100:8000 ssl_endpoint=192.168.0.100:8080
```

ssl_private_key

Optional path to the private key file used for SSL-enabled endpoints. If not specified, the ssl_certificate file is used as a private key.

tcp_nodelay

If specified, the socket option will disable Nagle's algorithm on the connection. It means that packets will be sent as soon as possible instead of waiting for a full buffer or timeout to occur.

'1' disables Nagle's algorithm for all sockets.

'0' keeps Nagle's algorithm enabled (default).

EXAMPLE 28.1: EXAMPLE BEAST CONFIGURATION

```
cephuser@adm > ceph config set rgw.myrealm.myzone.ses-min1.kwwazo \  
rgw_frontends beast port=8000 ssl_port=443 \  
ssl_certificate=/etc/ssl/ssl.crt \  
error_log_file=/var/log/radosgw/beast.error.log
```

28.5.2.2 CivetWeb

port

The listening port number. For SSL-enabled ports, add an 's' suffix (for example, '443s'). To bind a specific IPv4 or IPv6 address, use the form 'address:port'. You can specify multiple endpoints either by joining them with '+' or by providing multiple options:

```
port=127.0.0.1:8000+443s  
port=8000 port=443s
```

Default is 7480.

num_threads

The number of threads spawned by Civetweb to handle incoming HTTP connections. This effectively limits the number of concurrent connections that the front-end can service.

Default is the value specified by the `rgw_thread_pool_size` option.

request_timeout_ms

The amount of time in milliseconds that Civetweb will wait for more incoming data before giving up.

Default is 30000 milliseconds.

access_log_file

Path to the access log file. You can specify either a full path, or a path relative to the current working directory. If not specified (default), then accesses are not logged.

error_log_file

Path to the error log file. You can specify either a full path, or a path relative to the current working directory. If not specified (default), then errors are not logged.

EXAMPLE 28.2: EXAMPLE CIVETWEB CONFIGURATION IN `/etc/ceph/ceph.conf`

```
cephuser@adm > ceph config set rgw.myrealm.myzone.ses-min2.ingabw \  
  rgw_frontends civetweb port=8000+443s request_timeout_ms=30000 \  
  error_log_file=/var/log/radosgw/civetweb.error.log
```

28.5.2.3 Common Options

ssl_certificate

Path to the SSL certificate file used for SSL-enabled endpoints.

prefix

A prefix string that is inserted into the URI of all requests. For example, a Swift-only front-end could supply a URI prefix of /swift.

29 Ceph Manager modules

The architecture of the Ceph Manager (refer to *Book "Deployment Guide", Chapter 1 "SES and Ceph", Section 1.2.3 "Ceph nodes and daemons"* for a brief introduction) allows extending its functionality via *modules*, such as 'dashboard' (see *Part I, "Ceph Dashboard"*), 'prometheus' (see *Chapter 16, Monitoring and alerting*), or 'balancer'.

To list all available modules, run:

```
cephuser@adm > ceph mgr module ls
{
  "enabled_modules": [
    "restful",
    "status"
  ],
  "disabled_modules": [
    "dashboard"
  ]
}
```

To enable or disable a specific module, run:

```
cephuser@adm > ceph mgr module enable MODULE-NAME
```

For example:

```
cephuser@adm > ceph mgr module disable dashboard
```

To list the services that the enabled modules provide, run:

```
cephuser@adm > ceph mgr services
{
  "dashboard": "http://myserver.com:7789/",
  "restful": "https://myserver.com:8789/"
}
```

29.1 Balancer

The balancer module optimizes the placement group (PG) distribution across OSDs for a more balanced deployment. Although the module is activated by default, it is inactive. It supports the following two modes: `crush-compat` and `upmap`.



Tip: Current Balancer Status and Configuration

To view the current balancer status and configuration information, run:

```
cephuser@adm > ceph balancer status
```

29.1.1 The 'crush-compat' mode

In 'crush-compat' mode, the balancer adjusts the OSDs' reweight-sets to achieve improved distribution of the data. It moves PGs between OSDs, temporarily causing a `HEALTH_WARN` cluster state resulting from misplaced PGs.



Tip: Mode Activation

Although 'crush-compat' is the default mode, we recommend activating it explicitly:

```
cephuser@adm > ceph balancer mode crush-compat
```

29.1.2 Planning and executing of data balancing

Using the balancer module, you can create a plan for data balancing. You can then execute the plan manually, or let the balancer balance PGs continuously.

The decision whether to run the balancer in manual or automatic mode depends on several factors, such as the current data imbalance, cluster size, PG count, or I/O activity. We recommend creating an initial plan and executing it at a time of low I/O load in the cluster. The reason for this is that the initial imbalance will probably be considerable and it is a good practice to keep the impact on clients low. After an initial manual run, consider activating the automatic mode and monitor the rebalance traffic under normal I/O load. The improvements in PG distribution need to be weighed against the rebalance traffic caused by the balancer.



Tip: Movable Fraction of Placement Groups (PGs)

During the process of balancing, the balancer module throttles PG movements so that only a configurable fraction of PGs is moved. The default is 5% and you can adjust the fraction, to 9% for example, by running the following command:

```
cephuser@adm > ceph config set mgr target_max_misplaced_ratio .09
```

To create and execute a balancing plan, follow these steps:

1. Check the current cluster score:

```
cephuser@adm > ceph balancer eval
```

2. Create a plan. For example, 'great_plan':

```
cephuser@adm > ceph balancer optimize great_plan
```

3. See what changes the 'great_plan' will entail:

```
cephuser@adm > ceph balancer show great_plan
```

4. Check the potential cluster score if you decide to apply the 'great_plan':

```
cephuser@adm > ceph balancer eval great_plan
```

5. Execute the 'great_plan' for one time only:

```
cephuser@adm > ceph balancer execute great_plan
```

6. Observe the cluster balancing with the `ceph -s` command. If you are satisfied with the result, activate automatic balancing:

```
cephuser@adm > ceph balancer on
```

If you later decide to deactivate automatic balancing, run:

```
cephuser@adm > ceph balancer off
```



Tip: Automatic Balancing without Initial Plan

You can activate automatic balancing without executing an initial plan. In such case, expect a potentially long running rebalancing of placement groups.

29.2 Enabling the telemetry module

The telemetry plugin sends the Ceph project anonymous data about the cluster in which the plugin is running.

This (opt-in) component contains counters and statistics on how the cluster has been deployed, the version of Ceph, the distribution of the hosts and other parameters which help the project to gain a better understanding of the way Ceph is used. It does not contain any sensitive data like pool names, object names, object contents, or host names.

The purpose of the telemetry module is to provide an automated feedback loop for the developers to help quantify adoption rates, tracking, or point out things that need to be better explained or validated during configuration to prevent undesirable outcomes.



Note

The telemetry module requires the Ceph Manager nodes to have the ability to push data over HTTPS to the upstream servers. Ensure your corporate firewalls permit this action.

1. To enable the telemetry module:

```
cephuser@adm > ceph mgr module enable telemetry
```



Note

This command only enables you to view your data locally. This command does not share your data with the Ceph community.

2. To allow the telemetry module to start sharing data:

```
cephuser@adm > ceph telemetry on
```

3. To disable telemetry data sharing:

```
cephuser@adm > ceph telemetry off
```

4. To generate a JSON report that can be printed:

```
cephuser@adm > ceph telemetry show
```

5. To add a contact and description to the report:

```
cephuser@adm > ceph config set mgr mgr/telemetry/contact John Doe  
john.doe@example.com  
cephuser@adm > ceph config set mgr mgr/telemetry/description 'My first Ceph cluster'
```

6. The module compiles and sends a new report every 24 hours by default. To adjust this interval:

```
cephuser@adm > ceph config set mgr mgr/telemetry/interval HOURS
```

30 Authentication with cephx

To identify clients and protect against man-in-the-middle attacks, Ceph provides its `cephx` authentication system. *Clients* in this context are either human users—such as the admin user—or Ceph-related services/daemons, for example OSDs, monitors, or Object Gateways.



Note

The `cephx` protocol does not address data encryption in transport, such as TLS/SSL.

30.1 Authentication architecture

`cephx` uses shared secret keys for authentication, meaning both the client and Ceph Monitors have a copy of the client's secret key. The authentication protocol enables both parties to prove to each other that they have a copy of the key without actually revealing it. This provides mutual authentication, which means the cluster is sure the user possesses the secret key, and the user is sure that the cluster has a copy of the secret key as well.

A key scalability feature of Ceph is to avoid a centralized interface to the Ceph object store. This means that Ceph clients can interact with OSDs directly. To protect data, Ceph provides its `cephx` authentication system, which authenticates Ceph clients.

Each monitor can authenticate clients and distribute keys, so there is no single point of failure or bottleneck when using `cephx`. The monitor returns an authentication data structure that contains a session key for use in obtaining Ceph services. This session key is itself encrypted with the client's permanent secret key, so that only the client can request services from the Ceph monitors. The client then uses the session key to request its desired services from the monitor, and the monitor provides the client with a ticket that will authenticate the client to the OSDs that actually handle data. Ceph monitors and OSDs share a secret, so the client can use the ticket provided by the monitor with any OSD or metadata server in the cluster. `cephx` tickets expire, so an attacker cannot use an expired ticket or session key obtained wrongfully.

To use `cephx`, an administrator must setup clients/users first. In the following diagram, the `client.admin` user invokes `ceph auth get-or-create-key` from the command line to generate a user name and secret key. Ceph's `auth` subsystem generates the user name and key, stores a copy with the monitor(s) and transmits the user's secret back to the `client.admin` user. This means that the client and the monitor share a secret key.

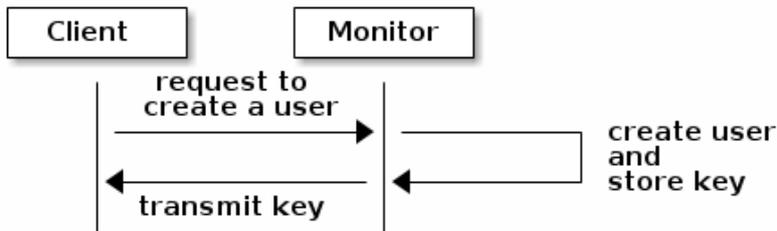


FIGURE 30.1: BASIC cephx AUTHENTICATION

To authenticate with the monitor, the client passes the user name to the monitor. The monitor generates a session key and encrypts it with the secret key associated with the user name and transmits the encrypted ticket back to the client. The client then decrypts the data with the shared secret key to retrieve the session key. The session key identifies the user for the current session. The client then requests a ticket related to the user, which is signed by the session key. The monitor generates a ticket, encrypts it with the user’s secret key and transmits it back to the client. The client decrypts the ticket and uses it to sign requests to OSDs and metadata servers throughout the cluster.

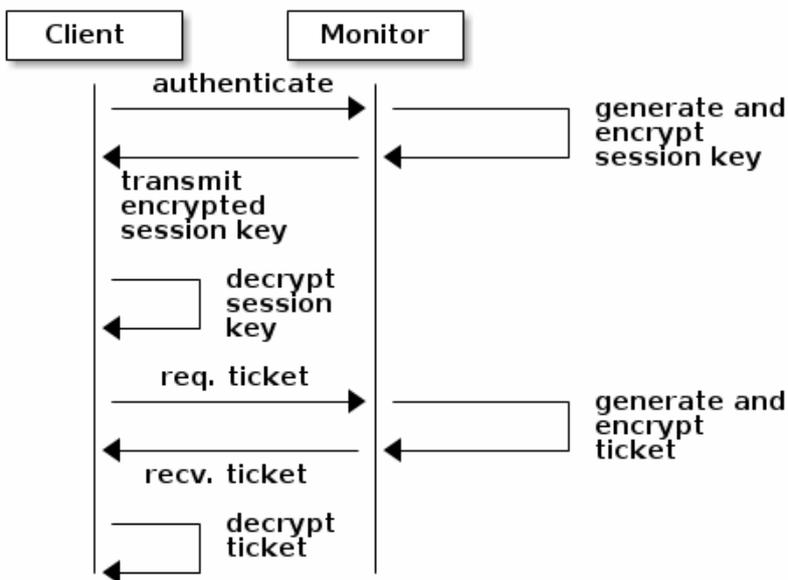


FIGURE 30.2: cephx AUTHENTICATION

The `cephx` protocol authenticates ongoing communications between the client machine and the Ceph servers. Each message sent between a client and a server after the initial authentication is signed using a ticket that the monitors, OSDs, and metadata servers can verify with their shared secret.

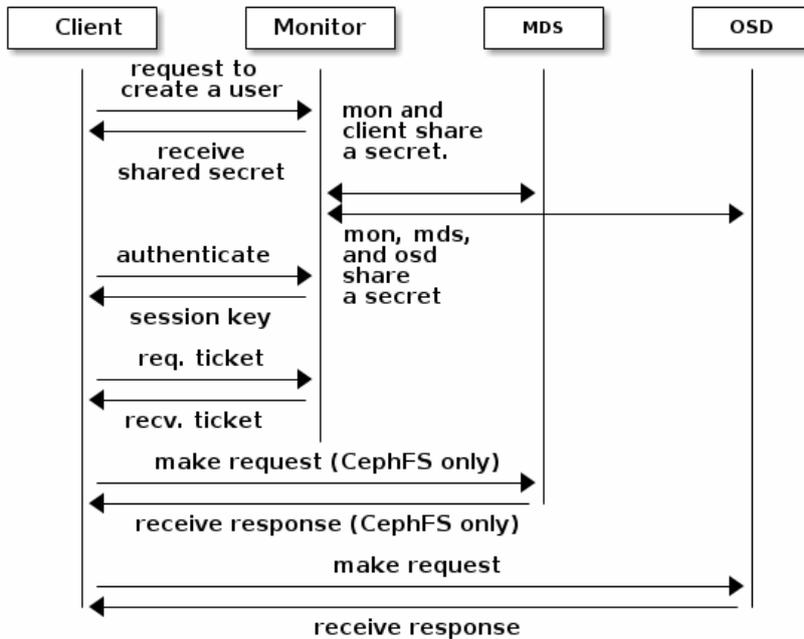


FIGURE 30.3: `cephx` AUTHENTICATION - MDS AND OSD

! Important

The protection offered by this authentication is between the Ceph client and the Ceph cluster hosts. The authentication is not extended beyond the Ceph client. If the user accesses the Ceph client from a remote host, Ceph authentication is not applied to the connection between the user’s host and the client host.

30.2 Key management

This section describes Ceph client users and their authentication and authorization with the Ceph storage cluster. *Users* are either individuals or system actors such as applications, which use Ceph clients to interact with the Ceph storage cluster daemons.

When Ceph runs with authentication and authorization enabled (enabled by default), you must specify a user name and a keyring containing the secret key of the specified user (usually via the command line). If you do not specify a user name, Ceph will use `client.admin` as the default user name. If you do not specify a keyring, Ceph will look for a keyring via the keyring setting in the Ceph configuration file. For example, if you execute the `ceph health` command without specifying a user name or keyring, Ceph interprets the command like this:

```
cephuser@adm > ceph -n client.admin --keyring=/etc/ceph/ceph.client.admin.keyring health
```

Alternatively, you may use the `CEPH_ARGS` environment variable to avoid re-entering the user name and secret.

30.2.1 Background information

Regardless of the type of Ceph client (for example, block device, object storage, file system, native API), Ceph stores all data as objects within *pools*. Ceph users need to have access to pools in order to read and write data. Additionally, Ceph users must have execute permissions to use Ceph's administrative commands. The following concepts will help you understand Ceph user management.

30.2.1.1 User

A user is either an individual or a system actor such as an application. Creating users allows you to control who (or what) can access your Ceph storage cluster, its pools, and the data within pools.

Ceph uses *types* of users. For the purposes of user management, the type will always be `client`. Ceph identifies users in period (.) delimited form, consisting of the user type and the user ID. For example, `TYPE.ID`, `client.admin`, or `client.user1`. The reason for user typing is that Ceph monitors, OSDs, and metadata servers also use the cephx protocol, but they are not clients. Distinguishing the user type helps to distinguish between client users and other users, streamlining access control, user monitoring, and traceability.

Sometimes Ceph's user type may seem confusing, because the Ceph command line allows you to specify a user with or without the type, depending upon your command line usage. If you specify `--user` or `--id`, you can omit the type. So `client.user1` can be entered simply as `user1`. If you specify `--name` or `-n`, you must specify the type and name, such as `client.user1`. We recommend using the type and name as a best practice wherever possible.



Note

A Ceph storage cluster user is not the same as a Ceph object storage user or a Ceph file system user. The Ceph Object Gateway uses a Ceph storage cluster user to communicate between the gateway daemon and the storage cluster, but the gateway has its own user management functionality for end users. The Ceph file system uses POSIX semantics. The user space associated with it is not the same as a Ceph storage cluster user.

30.2.1.2 Authorization and capabilities

Ceph uses the term 'capabilities' (caps) to describe authorizing an authenticated user to exercise the functionality of the monitors, OSDs, and metadata servers. Capabilities can also restrict access to data within a pool or pool namespace. A Ceph administrative user sets a user's capabilities when creating or updating a user.

Capability syntax follows the form:

```
daemon-type 'allow capability' [...]
```

Following is a list of capabilities for each service type:

Monitor capabilities

include r, w, x and allow profile cap.

```
mon 'allow rwx'
mon 'allow profile osd'
```

OSD capabilities

include r, w, x, class-read, class-write and profile osd. Additionally, OSD capabilities also allow for pool and namespace settings.

```
osd 'allow capability' [pool=poolname] [namespace=namespace-name]
```

MDS capability

simply requires allow, or blank.

```
mds 'allow'
```

The following entries describe each capability:

allow

Precedes access settings for a daemon. Implies rw for MDS only.

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 (both read and write) and to conduct auth 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/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 mds

Gives a user permissions to connect as an MDS to other MDSs or monitors.

profile bootstrap-osd

Gives a user permissions to bootstrap an OSD. Delegated to deployment tools so that they have permissions to add keys when bootstrapping an OSD.

profile bootstrap-mds

Gives a user permissions to bootstrap a metadata server. Delegated to deployment tools so they have permissions to add keys when bootstrapping a metadata server.

30.2.1.3 Pools

A pool is a logical partition where users store data. In Ceph deployments, it is common to create a pool as a logical partition for similar types of data. For example, when deploying Ceph as a back-end for OpenStack, a typical deployment would have pools for volumes, images, backups and virtual machines, and users such as client.glance or client.cinder.

30.2.2 Managing users

User management functionality provides Ceph cluster administrators with the ability to create, update, and delete users directly in the Ceph cluster.

When you create or delete users in the Ceph cluster, you may need to distribute keys to clients so that they can be added to keyrings. See [Section 30.2.3, “Managing keyrings”](#) for details.

30.2.2.1 Listing users

To list the users in your cluster, execute the following:

```
cephuser@adm > ceph auth list
```

Ceph will list all users in your cluster. For example, in a cluster with two nodes, `ceph auth list` output looks similar to this:

```
installed auth entries:

osd.0
  key: AQCvCbtToC6MDhAATtuT70Sl+DymPCfDSsyV4w==
  caps: [mon] allow profile osd
  caps: [osd] allow *

osd.1
  key: AQC4CbtTCFJBChAAVq5spj0ff4eHZICxIOVZeA==
  caps: [mon] allow profile osd
  caps: [osd] allow *

client.admin
  key: AQBHCbtT6APDHhAA5W00cBchwKQjh3dkKsyPjw==
  caps: [mds] allow
  caps: [mon] allow *
  caps: [osd] allow *

client.bootstrap-mds
  key: AQBICbtT0K9uGBAAdbe5zcIGHZL3T/u2g6EBww==
  caps: [mon] allow profile bootstrap-mds

client.bootstrap-osd
  key: AQBHCbtT4Gxq0RAADE5u7RkpCN/oo4e5W0uBtw==
  caps: [mon] allow profile bootstrap-osd
```



Note: TYPE.ID notation

Note that the `TYPE.ID` notation for users applies such that `osd.0` specifies a user of type `osd` and its ID is `0`. `client.admin` is a user of type `client` and its ID is `admin`. Note also that each entry has a `key: value` entry, and one or more `caps: entries`.

You may use the `-o filename` option with `ceph auth list` to save the output to a file.

30.2.2.2 Getting information about users

To retrieve a specific user, key, and capabilities, execute the following:

```
cephuser@adm > ceph auth get TYPE.ID
```

For example:

```
cephuser@adm > ceph auth get client.admin
exported keyring for client.admin
[client.admin]
key = AQA19uZUqIwkHxAAFuUwvq0eJD4S173oFRxe0g==
caps mds = "allow"
caps mon = "allow *"
caps osd = "allow *"
```

Developers may also execute the following:

```
cephuser@adm > ceph auth export TYPE.ID
```

The `auth export` command is identical to `auth get`, but also prints the internal authentication ID.

30.2.2.3 Adding users

Adding a user creates a user name (`TYPE.ID`), a secret key, and any capabilities included in the command you use to create the user.

A user's key enables the user to authenticate with the Ceph storage cluster. The user's capabilities authorize the user to read, write, or execute on Ceph monitors (`mon`), Ceph OSDs (`osd`), or Ceph metadata servers (`mds`).

There are a few commands available to add a user:

ceph auth add

This command is the canonical way to add a user. It will create the user, generate a key, and add any specified capabilities.

ceph auth get-or-create

This command is often the most convenient way to create a user, because it returns a keyfile format with the user name (in brackets) and the key. If the user already exists, this command simply returns the user name and key in the keyfile format. You may use the `-o filename` option to save the output to a file.

ceph auth get-or-create-key

This command is a convenient way to create a user and return the user's key (only). This is useful for clients that need the key only (for example `libvirt`). If the user already exists, this command simply returns the key. You may use the `-o filename` option to save the output to a file.

When creating client users, you may create a user with no capabilities. A user with no capabilities can authenticate but nothing more. Such client cannot retrieve the cluster map from the monitor. However, you can create a user with no capabilities if you want to defer adding capabilities later using the `ceph auth caps` command.

A typical user has at least read capabilities on the Ceph monitor and read and write capabilities on Ceph OSDs. Additionally, a user's OSD permissions are often restricted to accessing a particular pool.

```
cephuser@adm > ceph auth add client.john mon 'allow r' osd \  
'allow rw pool=liverpool'  
cephuser@adm > ceph auth get-or-create client.paul mon 'allow r' osd \  
'allow rw pool=liverpool'  
cephuser@adm > ceph auth get-or-create client.george mon 'allow r' osd \  
'allow rw pool=liverpool' -o george.keyring  
cephuser@adm > ceph auth get-or-create-key client.ringo mon 'allow r' osd \  
'allow rw pool=liverpool' -o ringo.key
```

Important

If you provide a user with capabilities to OSDs, but you *do not* restrict access to particular pools, the user will have access to *all* pools in the cluster.

30.2.2.4 Modifying user capabilities

The **ceph auth caps** command allows you to specify a user and change the user's capabilities. Setting new capabilities will overwrite current ones. To view current capabilities run **ceph auth get *USERTYPE.USERID***. To add capabilities, you also need to specify the existing capabilities when using the following form:

```
cephuser@adm > ceph auth caps USERTYPE.USERID daemon 'allow [r|w|x|*|...] \
  [pool=pool-name] [namespace=namespace-name]' [daemon 'allow [r|w|x|*|...] \
  [pool=pool-name] [namespace=namespace-name']
```

For example:

```
cephuser@adm > ceph auth get client.john
cephuser@adm > ceph auth caps client.john mon 'allow r' osd 'allow rw pool=prague'
cephuser@adm > ceph auth caps client.paul mon 'allow rw' osd 'allow r pool=prague'
cephuser@adm > ceph auth caps client.brian-manager mon 'allow *' osd 'allow *'
```

To remove a capability, you may reset the capability. If you want the user to have no access to a particular daemon that was previously set, specify an empty string:

```
cephuser@adm > ceph auth caps client.ringo mon ' ' osd ' '
```

30.2.2.5 Deleting users

To delete a user, use **ceph auth del**:

```
cephuser@adm > ceph auth del TYPE.ID
```

where *TYPE* is one of *client*, *osd*, *mon*, or *mds*, and *ID* is the user name or ID of the daemon. If you created users with permissions strictly for a pool that no longer exists, you should consider deleting those users too.

30.2.2.6 Printing a user's key

To print a user's authentication key to standard output, execute the following:

```
cephuser@adm > ceph auth print-key TYPE.ID
```

where *TYPE* is one of *client*, *osd*, *mon*, or *mds*, and *ID* is the user name or ID of the daemon.

Printing a user's key is useful when you need to populate client software with a user's key (such as `libvirt`), as in the following example:

```
# mount -t ceph host:/ mount_point \  
-o name=client.user,secret=`ceph auth print-key client.user`
```

30.2.2.7 Importing users

To import one or more users, use `ceph auth import` and specify a keyring:

```
cephuser@adm > ceph auth import -i /etc/ceph/ceph.keyring
```



Note

The Ceph storage cluster will add new users, their keys and their capabilities and will update existing users, their keys and their capabilities.

30.2.3 Managing keyrings

When you access Ceph via a Ceph client, the client will look for a local keyring. Ceph presets the keyring setting with the following four keyring names by default so you do not need to set them in your Ceph configuration file unless you want to override the defaults:

```
/etc/ceph/cluster.name.keyring  
/etc/ceph/cluster.keyring  
/etc/ceph/keyring  
/etc/ceph/keyring.bin
```

The `cluster` metavariable is your Ceph cluster name as defined by the name of the Ceph configuration file. `ceph.conf` means that the cluster name is `ceph`, thus `ceph.keyring`. The `name` metavariable is the user type and user ID, for example `client.admin`, thus `ceph.client.admin.keyring`.

After you create a user (for example `client.ringo`), you must get the key and add it to a keyring on a Ceph client so that the user can access the Ceph storage cluster.

[Section 30.2, “Key management”](#) details how to list, get, add, modify and delete users directly in the Ceph storage cluster. However, Ceph also provides the `ceph-authtool` utility to allow you to manage keyrings from a Ceph client.

30.2.3.1 Creating a keyring

When you use the procedures in [Section 30.2, “Key management”](#) to create users, you need to provide user keys to the Ceph client(s) so that the client can retrieve the key for the specified user and authenticate with the Ceph storage cluster. Ceph clients access keyrings to look up a user name and retrieve the user's key:

```
cephuser@adm > ceph-authtool --create-keyring /path/to/keyring
```

When creating a keyring with multiple users, we recommend using the cluster name (for example `cluster.keyring`) for the keyring file name and saving it in the `/etc/ceph` directory so that the keyring configuration default setting will pick up the file name without requiring you to specify it in the local copy of your Ceph configuration file. For example, create `ceph.keyring` by executing the following:

```
cephuser@adm > ceph-authtool -C /etc/ceph/ceph.keyring
```

When creating a keyring with a single user, we recommend using the cluster name, the user type and the user name and saving it in the `/etc/ceph` directory. For example, `ceph.client.admin.keyring` for the `client.admin` user.

30.2.3.2 Adding a user to a keyring

When you add a user to the Ceph storage cluster (see [Section 30.2.2.3, “Adding users”](#)), you can retrieve the user, key and capabilities, and save the user to a keyring.

If you only want to use one user per keyring, the `ceph auth get` command with the `-o` option will save the output in the keyring file format. For example, to create a keyring for the `client.admin` user, execute the following:

```
cephuser@adm > ceph auth get client.admin -o /etc/ceph/ceph.client.admin.keyring
```

When you want to import users to a keyring, you can use `ceph-authtool` to specify the destination keyring and the source keyring:

```
cephuser@adm > ceph-authtool /etc/ceph/ceph.keyring \  
--import-keyring /etc/ceph/ceph.client.admin.keyring
```



Important

If your keyring is compromised, delete your key from the `/etc/ceph` directory and re-create a new key using the same instructions from [Section 30.2.3.1, “Creating a keyring”](#).

30.2.3.3 Creating a user

Ceph provides the `ceph auth add` command to create a user directly in the Ceph storage cluster. However, you can also create a user, keys and capabilities directly on a Ceph client keyring. Then, you can import the user to the Ceph storage cluster:

```
cephuser@adm > ceph-authtool -n client.ringo --cap osd 'allow rwx' \  
--cap mon 'allow rwx' /etc/ceph/ceph.keyring
```

You can also create a keyring and add a new user to the keyring simultaneously:

```
cephuser@adm > ceph-authtool -C /etc/ceph/ceph.keyring -n client.ringo \  
--cap osd 'allow rwx' --cap mon 'allow rwx' --gen-key
```

In the previous scenarios, the new user `client.ringo` is only in the keyring. To add the new user to the Ceph storage cluster, you must still add the new user to the cluster:

```
cephuser@adm > ceph auth add client.ringo -i /etc/ceph/ceph.keyring
```

30.2.3.4 Modifying users

To modify the capabilities of a user record in a keyring, specify the keyring and the user followed by the capabilities:

```
cephuser@adm > ceph-authtool /etc/ceph/ceph.keyring -n client.ringo \  
--cap osd 'allow rwx' --cap mon 'allow rwx'
```

To update the modified user within the Ceph cluster environment, you must import the changes from the keyring to the user entry in the Ceph cluster:

```
cephuser@adm > ceph auth import -i /etc/ceph/ceph.keyring
```

See [Section 30.2.2.7, "Importing users"](#) for details on updating a Ceph storage cluster user from a keyring.

30.2.4 Commandline usage

The **ceph** command supports the following options related to the user name and secret manipulation:

--id or --user

Ceph identifies users with a type and an ID (*TYPE.ID*, such as `client.admin` or `client.user1`). The `id`, `name` and `-n` options enable you to specify the ID portion of the user name (for example `admin` or `user1`). You can specify the user with the `--id` and omit the type. For example, to specify user `client.foo` enter the following:

```
cephuser@adm > ceph --id foo --keyring /path/to/keyring health
cephuser@adm > ceph --user foo --keyring /path/to/keyring health
```

--name or -n

Ceph identifies users with a type and an ID (*TYPE.ID*, such as `client.admin` or `client.user1`). The `--name` and `-n` options enable you to specify the fully qualified user name. You must specify the user type (typically `client`) with the user ID:

```
cephuser@adm > ceph --name client.foo --keyring /path/to/keyring health
cephuser@adm > ceph -n client.foo --keyring /path/to/keyring health
```

--keyring

The path to the keyring containing one or more user name and secret. The `--secret` option provides the same functionality, but it does not work with Object Gateway, which uses `--secret` for another purpose. You may retrieve a keyring with **ceph auth get-or-create** and store it locally. This is a preferred approach, because you can switch user names without switching the keyring path:

```
cephuser@adm > rbd map --id foo --keyring /path/to/keyring mypool/myimage
```

A Ceph maintenance updates based on upstream 'Octopus' point releases

Several key packages in SUSE Enterprise Storage 7 are based on the Octopus release series of Ceph. When the Ceph project (<https://github.com/ceph/ceph>) publishes new point releases in the Octopus series, SUSE Enterprise Storage 7 is updated to ensure that the product benefits from the latest upstream bug fixes and feature backports.

This chapter contains summaries of notable changes contained in each upstream point release that has been—or is planned to be—included in the product.

Octopus 15.2.11 Point Release

This release includes a security fix that ensures the `global_id` value (a numeric value that should be unique for every authenticated client or daemon in the cluster) is reclaimed after a network disconnect or ticket renewal in a secure fashion. Two new health alerts may appear during the upgrade indicating that there are clients or daemons that are not yet patched with the appropriate fix.

To temporarily mute the health alerts around insecure clients for the duration of the upgrade, you may want to run:

```
cephuser@adm > ceph health mute AUTH_INSECURE_GLOBAL_ID_RECLAIM 1h
cephuser@adm > ceph health mute AUTH_INSECURE_GLOBAL_ID_RECLAIM_ALLOWED 1h
```

When all clients are updated, enable the new secure behavior, not allowing old insecure clients to join the cluster:

```
cephuser@adm > ceph config set mon auth_allow_insecure_global_id_reclaim false
```

For more details, refer to <https://docs.ceph.com/en/latest/security/CVE-2021-20288/>.

Octopus 15.2.10 Point Release

This backport release includes the following fixes:

- The containers include an updated `tcmmalloc` that avoids crashes seen on 15.2.9.
- RADOS: BlueStore handling of huge (> 4GB) writes from RocksDB to BlueFS has been fixed.
- When upgrading from a previous `cephadm` release, `systemctl` may hang when trying to start or restart the monitoring containers. This is caused by a change in the `systemd` unit to use `type=forking`.) After the upgrade, please run:

```
cephuser@adm > ceph orch redeploy nfs
cephuser@adm > ceph orch redeploy iscsi
cephuser@adm > ceph orch redeploy node-exporter
cephuser@adm > ceph orch redeploy prometheus
cephuser@adm > ceph orch redeploy grafana
cephuser@adm > ceph orch redeploy alertmanager
```

Octopus 15.2.9 Point Release

This backport release includes the following fixes:

- MGR: progress module can now be turned on/off, using the commands: `ceph progress on` and `ceph progress off`.
- OSD: PG removal has been optimized in this release.

Octopus 15.2.8 Point Release

This release fixes a security flaw in CephFS and includes a number of bug fixes:

- OpenStack Manila use of `ceph_volume_client.py` library allowed tenant access to any Ceph credential's secret.
- **ceph-volume**: The `lvm batch` subcommand received a major rewrite. This closed a number of bugs and improves usability in terms of size specification and calculation, as well as idempotency behaviour and disk replacement process. Please refer to <https://docs.ceph.com/en/latest/ceph-volume/lvm/batch/> for more detailed information.

- **MON:** The cluster log now logs health detail every `mon_health_to_clog_interval`, which has been changed from 1hr to 10min. Logging of health detail will be skipped if there is no change in health summary since last known.
- The `ceph df` command now lists the number of PGs in each pool.
- The `bluefs_preextend_wal_files` option has been removed.
- It is now possible to specify the initial monitor to contact for Ceph tools and daemons using the `mon_host_override` config option or `--mon-host-override` command line switch. This generally should only be used for debugging and only affects initial communication with Ceph's monitor cluster.

Octopus 15.2.7 Point Release

This release fixes a serious bug in RGW that has been shown to cause data loss when a read of a large RGW object (for example, one with at least one tail segment) takes longer than one half the time specified in the configuration option `rgw_gc_obj_min_wait`. The bug causes the tail segments of that read object to be added to the RGW garbage collection queue, which will in turn cause them to be deleted after a period of time.

Octopus 15.2.6 Point Release

This releases fixes a security flaw affecting Messenger V2 for Octopus and Nautilus.

Octopus 15.2.5 Point Release

The Octopus point release 15.2.5 brought the following fixes and other changes:

- **CephFS:** Automatic static sub-tree partitioning policies may now be configured using the new distributed and random ephemeral pinning extended attributes on directories. See the following documentation for more information: <https://docs.ceph.com/docs/master/cephfs/multimds/>
- Monitors now have a configuration option `mon_osd_warn_num_repaired`, which is set to 10 by default. If any OSD has repaired more than this many I/O errors in stored data a `OSD_T00_MANY_REPAIRS` health warning is generated.

- Now, when no scrub and/or no deep-scrub flags are set globally or per pool, scheduled scrubs of the type disabled will be aborted. All user initiated scrubs are NOT interrupted.
- Fixed an issue with osdm maps not being trimmed in a healthy cluster.

Octopus 15.2.4 Point Release

The Octopus point release 15.2.4 brought the following fixes and other changes:

- CVE-2020-10753: rgw: sanitize newlines in s3 CORSConfiguration's ExposeHeader
- Object Gateway: The radosgw-admin sub-commands dealing with orphans—radosgw-admin orphans find, radosgw-admin orphans finish, and radosgw-admin orphans list-jobs—have been deprecated. They had not been actively maintained, and since they store intermediate results on the cluster, they could potentially fill a nearly-full cluster. They have been replaced by a tool, rgw-orphan-list, which is currently considered experimental.
- RBD: The name of the RBD pool object that is used to store RBD trash purge schedule is changed from rbd_trash_trash_purge_schedule to rbd_trash_purge_schedule. Users that have already started using RBD trash purge schedule functionality and have per pool or name space schedules configured should copy the rbd_trash_trash_purge_schedule object to rbd_trash_purge_schedule before the upgrade and remove rbd_trash_purge_schedule using the following commands in every RBD pool and name space where a trash purge schedule was previously configured:

```
rados -p pool-name [-N namespace] cp rbd_trash_trash_purge_schedule
  rbd_trash_purge_schedule
rados -p pool-name [-N namespace] rm rbd_trash_trash_purge_schedule
```

Alternatively, use any other convenient way to restore the schedule after the upgrade.

Octopus 15.2.3 Point Release

- The Octopus point release 15.2.3 was a hot-fix release to address an issue where WAL corruption was seen when bluefs_preextend_wal_files and bluefs_buffered_io were enabled at the same time. The fix in 15.2.3 is only a temporary measure (changing the

default value of `bluefs_preextend_wal_files` to `false`). The permanent fix will be to remove the `bluefs_preextend_wal_files` option completely: this fix will most likely arrive in the 15.2.6 point release.

Octopus 15.2.2 Point Release

The Octopus point release 15.2.2 patched one security vulnerability:

- CVE-2020-10736: Fixed an authorization bypass in MONs and MGRs

Octopus 15.2.1 Point Release

The Octopus point release 15.2.1 fixed an issue where upgrading quickly from Luminous (SES5.5) to Nautilus (SES6) to Octopus (SES7) caused OSDs to crash. In addition, it patched two security vulnerabilities that were present in the initial Octopus (15.2.0) release:

- CVE-2020-1759: Fixed nonce reuse in msgr V2 secure mode
- CVE-2020-1760: Fixed XSS because of RGW GetObject header-splitting

Glossary

General

Admin node

The host from which you run the Ceph-related commands to administer cluster hosts.

Alertmanager

A single binary which handles alerts sent by the Prometheus server and notifies the end user.

archive sync module

Module that enables creating an Object Gateway zone for keeping the history of S3 object versions.

Bucket

A point that aggregates other nodes into a hierarchy of physical locations.

Ceph Client

The collection of Ceph components which can access a Ceph Storage Cluster. These include the Object Gateway, the Ceph Block Device, the CephFS, and their corresponding libraries, kernel modules, and FUSE clients.

Ceph Dashboard

A built-in Web-based Ceph management and monitoring application to administer various aspects and objects of the cluster. The dashboard is implemented as a Ceph Manager module.

Ceph Manager

Ceph Manager or MGR is the Ceph manager software, which collects all the state from the whole cluster in one place.

Ceph Monitor

Ceph Monitor or MON is the Ceph monitor software.

Ceph Object Storage

The object storage "product", service or capabilities, which consists of a Ceph Storage Cluster and a Ceph Object Gateway.

Ceph OSD Daemon

The **ceph-osd** daemon is the component of Ceph that is responsible for storing objects on a local file system and providing access to them over the network.

Ceph Storage Cluster

The core set of storage software which stores the user's data. Such a set consists of Ceph monitors and OSDs.

ceph-salt

Provides tooling for deploying Ceph clusters managed by cephadm using Salt.

cephadm

cephadm deploys and manages a Ceph cluster by connecting to hosts from the manager daemon via SSH to add, remove, or update Ceph daemon containers.

CephFS

The Ceph file system.

CephX

The Ceph authentication protocol. Cephx operates like Kerberos, but it has no single point of failure.

CRUSH rule

The CRUSH data placement rule that applies to a particular pool or pools.

CRUSH, CRUSH Map

Controlled Replication Under Scalable Hashing: An algorithm that determines how to store and retrieve data by computing data storage locations. CRUSH requires a map of the cluster to pseudo-randomly store and retrieve data in OSDs with a uniform distribution of data across the cluster.

DriveGroups

DriveGroups are a declaration of one or more OSD layouts that can be mapped to physical drives. An OSD layout defines how Ceph physically allocates OSD storage on the media matching the specified criteria.

Grafana

Database analytics and monitoring solution.

Metadata Server

Metadata Server or MDS is the Ceph metadata software.

Multi-zone

Node

Any single machine or server in a Ceph cluster.

Object Gateway

The S3/Swift gateway component for Ceph Object Store. Also known as the RADOS Gateway (RGW).

OSD

Object Storage Device: A physical or logical storage unit.

OSD node

A cluster node that stores data, handles data replication, recovery, backfilling, rebalancing, and provides some monitoring information to Ceph monitors by checking other Ceph OSD daemons.

PG

Placement Group: a sub-division of a *pool*, used for performance tuning.

Point Release

Any ad-hoc release that includes only bug or security fixes.

Pool

Logical partitions for storing objects such as disk images.

Prometheus

Systems monitoring and alerting toolkit.

RADOS Block Device (RBD)

The block storage component of Ceph. Also known as the Ceph block device.

Reliable Autonomic Distributed Object Store (RADOS)

The core set of storage software which stores the user's data (MON + OSD).

Routing tree

A term given to any diagram that shows the various routes a receiver can run.

Rule Set

Rules to determine data placement for a pool.

Samba

Windows integration software.

Samba Gateway

The Samba Gateway joins the Active Directory in the Windows domain to authenticate and authorize users.

zonegroup