



# Red Hat Ceph Storage 5

## Administration Guide

Administration of Red Hat Ceph Storage



# Red Hat Ceph Storage 5 Administration Guide

---

Administration of Red Hat Ceph Storage

## Legal Notice

Copyright © 2021 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

## Abstract

This document describes how to manage processes, monitor cluster states, manage users, and add and remove daemons for Red Hat Ceph Storage. Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

## Table of Contents

<b>CHAPTER 1. CEPH ADMINISTRATION</b> .....	<b>5</b>
<b>CHAPTER 2. UNDERSTANDING PROCESS MANAGEMENT FOR CEPH</b> .....	<b>6</b>
2.1. PREREQUISITES	6
2.2. CEPH PROCESS MANAGEMENT	6
2.3. STARTING, STOPPING, AND RESTARTING ALL CEPH DAEMONS	6
2.4. STARTING, STOPPING, AND RESTARTING ALL CEPH SERVICES	7
2.5. VIEWING LOG FILES OF CEPH DAEMONS THAT RUN IN CONTAINERS	9
2.6. POWERING DOWN AND REBOOTING RED HAT CEPH STORAGE CLUSTER	10
<b>CHAPTER 3. MONITORING A CEPH STORAGE CLUSTER</b> .....	<b>13</b>
3.1. PREREQUISITES	13
3.2. HIGH-LEVEL MONITORING OF A CEPH STORAGE CLUSTER	13
3.2.1. Prerequisites	13
3.2.2. Using the Ceph command interface interactively	13
3.2.3. Checking the storage cluster health	14
3.2.4. Watching storage cluster events	15
3.2.5. How Ceph calculates data usage	16
3.2.6. Understanding the storage clusters usage stats	16
3.2.7. Understanding the OSD usage stats	18
3.2.8. Checking the Red Hat Ceph Storage cluster status	19
3.2.9. Checking the Ceph Monitor status	21
3.2.10. Using the Ceph administration socket	24
3.2.11. Understanding the Ceph OSD status	29
3.3. LOW-LEVEL MONITORING OF A CEPH STORAGE CLUSTER	30
3.3.1. Prerequisites	31
3.3.2. Monitoring Placement Group Sets	31
3.3.3. Ceph OSD peering	32
3.3.4. Placement Group States	32
3.3.5. Placement Group creating state	35
3.3.6. Placement group peering state	36
3.3.7. Placement group active state	36
3.3.8. Placement Group clean state	36
3.3.9. Placement Group degraded state	36
3.3.10. Placement Group recovering state	37
3.3.11. Back fill state	37
3.3.12. Placement Group remapped state	38
3.3.13. Placement Group stale state	38
3.3.14. Placement Group misplaced state	38
3.3.15. Placement Group incomplete state	39
3.3.16. Identifying stuck Placement Groups	39
3.3.17. Finding an object's location	39
<b>CHAPTER 4. OVERRIDE CEPH BEHAVIOR</b> .....	<b>41</b>
4.1. PREREQUISITES	41
4.2. SETTING AND UNSETTING CEPH OVERRIDE OPTIONS	41
4.3. CEPH OVERRIDE USE CASES	42
<b>CHAPTER 5. CEPH USER MANAGEMENT</b> .....	<b>44</b>
5.1. PREREQUISITES	44
5.2. CEPH USER MANAGEMENT BACKGROUND	44
5.3. MANAGING CEPH USERS	47

5.3.1. Prerequisites	47
5.3.2. Listing Ceph users	47
5.3.3. Display Ceph user information	49
5.3.4. Add a new Ceph user	50
5.3.5. Modifying a Ceph User	51
5.3.6. Deleting a Ceph user	51
5.3.7. Print a Ceph user key	52
<b>CHAPTER 6. CEPH PERFORMANCE BENCHMARK</b>	<b>53</b>
6.1. PREREQUISITES	53
6.2. PERFORMANCE BASELINE	53
6.3. BENCHMARKING CEPH PERFORMANCE	53
6.4. BENCHMARKING CEPH BLOCK PERFORMANCE	56
<b>CHAPTER 7. CEPH PERFORMANCE COUNTERS</b>	<b>58</b>
7.1. PREREQUISITES	58
7.2. ACCESS TO CEPH PERFORMANCE COUNTERS	58
7.3. DISPLAY THE CEPH PERFORMANCE COUNTERS	59
7.4. DUMP THE CEPH PERFORMANCE COUNTERS	60
7.5. AVERAGE COUNT AND SUM	60
7.6. CEPH MONITOR METRICS	61
7.7. CEPH OSD METRICS	66
7.8. CEPH OBJECT GATEWAY METRICS	75
<b>CHAPTER 8. BLUESTORE</b>	<b>81</b>
8.1. CEPH BLUESTORE	81
8.2. CEPH BLUESTORE DEVICES	81
8.3. CEPH BLUESTORE CACHING	82
8.4. SIZING CONSIDERATIONS FOR CEPH BLUESTORE	83
8.5. TUNING CEPH BLUESTORE USING BLUESTORE_MIN_ALLOC_SIZE PARAMETER	83
8.6. RESHARDING THE ROCKSDB DATABASE USING THE BLUESTORE ADMIN TOOL	84
8.7. THE BLUESTORE FRAGMENTATION TOOL	91
8.7.1. Prerequisites	91
8.7.2. What is the BlueStore fragmentation tool?	91
8.7.3. Checking for fragmentation	91
<b>CHAPTER 9. CEPHADM TROUBLESHOOTING</b>	<b>94</b>
9.1. PREREQUISITES	94
9.2. PAUSE OR DISABLE CEPHADM	94
9.3. PER SERVICE AND PER DAEMON EVENT	94
9.4. CHECK CEPHADM LOGS	95
9.5. GATHER LOG FILES	95
9.6. COLLECT SYSTEMD STATUS	96
9.7. LIST ALL DOWNLOADED CONTAINER IMAGES	97
9.8. MANUALLY RUN CONTAINERS	97
9.9. CIDR NETWORK ERROR	98
9.10. ACCESS THE ADMIN SOCKET	98
9.11. MANUALLY DEPLOYING A MGR DAEMON	99
<b>CHAPTER 10. CEPHADM OPERATIONS</b>	<b>101</b>
10.1. PREREQUISITES	101
10.2. MONITOR CEPHADM LOG MESSAGES	101
10.3. CEPH DAEMON LOGS	102
10.4. DATA LOCATION	103

10.5. CEPHADM HEALTH CHECKS	103
10.5.1. Prerequisites	103
10.5.2. Cephadm operations health checks	103
10.5.3. Cephadm configuration health checks	104



# CHAPTER 1. CEPH ADMINISTRATION

A Red Hat Ceph Storage cluster is the foundation for all Ceph deployments. After deploying a Red Hat Ceph Storage cluster, there are administrative operations for keeping a Red Hat Ceph Storage cluster healthy and performing optimally.

The Red Hat Ceph Storage Administration Guide helps storage administrators to perform such tasks as:

- How do I check the health of my Red Hat Ceph Storage cluster?
- How do I start and stop the Red Hat Ceph Storage cluster services?
- How do I add or remove an OSD from a running Red Hat Ceph Storage cluster?
- How do I manage user authentication and access controls to the objects stored in a Red Hat Ceph Storage cluster?
- I want to understand how to use overrides with a Red Hat Ceph Storage cluster.
- I want to monitor the performance of the Red Hat Ceph Storage cluster.

A basic Ceph storage cluster consist of two types of daemons:

- A Ceph Object Storage Device (OSD) stores data as objects within placement groups assigned to the OSD
- A Ceph Monitor maintains a master copy of the cluster map

A production system will have three or more Ceph Monitors for high availability and typically a minimum of 50 OSDs for acceptable load balancing, data re-balancing and data recovery.

## Additional Resources

- [Red Hat Ceph Storage Installation Guide](#)

## CHAPTER 2. UNDERSTANDING PROCESS MANAGEMENT FOR CEPH

As a storage administrator, you can manipulate the various Ceph daemons by type or instance in a Red Hat Ceph Storage cluster. Manipulating these daemons allows you to start, stop and restart all of the Ceph services as needed.

### 2.1. PREREQUISITES

- Installation of the Red Hat Ceph Storage software.

### 2.2. CEPH PROCESS MANAGEMENT

In Red Hat Ceph Storage, all process management is done through the Systemd service. Each time you want to **start**, **restart**, and **stop** the Ceph daemons, you must specify the daemon type or the daemon instance.

#### Additional Resources

- For more information about using Systemd, see the chapter *Managing services with systemd* in the Red Hat Enterprise Linux [System Administrator's Guide](#).

### 2.3. STARTING, STOPPING, AND RESTARTING ALL CEPH DAEMONS

You can start, stop, and restart all Ceph daemons as the root user from the host where you want to stop the Ceph daemons.

#### Prerequisites

- A running Red Hat Ceph Storage cluster.
- Having **root** access to the node.

#### Procedure

1. On the host where you want to start, stop, and restart the daemons, run the `systemctl` service to get the `SERVICE_ID` of the service.

#### Example

```
[root@host01 ~]# systemctl --type=service  
ceph-499829b4-832f-11eb-8d6d-001a4a000635@mon.host01.service
```

2. Starting all Ceph daemons:

#### Syntax

```
systemctl start SERVICE_ID
```

#### Example

```
[root@host01 ~]# systemctl start ceph-499829b4-832f-11eb-8d6d-001a4a000635@mon.host01.service
```

3. Stopping all Ceph daemons:

### Syntax

```
systemctl stop SERVICE_ID
```

### Example

```
[root@host01 ~]# systemctl stop ceph-499829b4-832f-11eb-8d6d-001a4a000635@mon.host01.service
```

4. Restarting all Ceph daemons:

### Syntax

```
systemctl restart SERVICE_ID
```

### Example

```
[root@host01 ~]# systemctl restart ceph-499829b4-832f-11eb-8d6d-001a4a000635@mon.host01.service
```

## 2.4. STARTING, STOPPING, AND RESTARTING ALL CEPH SERVICES

Ceph services are logical groups of Ceph daemons of the same type, configured to run in the same Red Hat Ceph Storage cluster. The orchestration layer in Ceph allows the user to manage these services in a centralized way, making it easy to execute operations that affect all the Ceph daemons that belong to the same logical service. The Ceph daemons running in each host are managed through the Systemd service. You can start, stop, and restart all Ceph services from the host where you want to manage the Ceph services.

### IMPORTANT

If you want to start, stop, or restart a specific Ceph daemon in a specific host, you need to use the SystemD service. To obtain a list of the SystemD services running in a specific host, connect to the host, and run the following command:

### Example

```
[root@host01 ~]# systemctl list-units "ceph*"
```

The output will give you a list of the service names that you can use, to manage each Ceph daemon.

### Prerequisites

- A running Red Hat Ceph Storage cluster.

- Having **root** access to the node.

## Procedure

1. Log into the Cephadm shell:

### Example

```
[root@host01 ~]# cephadm shell
```

2. Run the **ceph orch ls** command to get a list of Ceph services configured in the Red Hat Ceph Storage cluster and to get the specific service ID.

### Example

```
[ceph: root@host01 /]# ceph orch ls
NAME                RUNNING REFRESHED AGE PLACEMENT IMAGE NAME
IMAGE ID
alertmanager        1/1 4m ago 4M count:1 registry.redhat.io/openshift4/ose-
prometheus-alertmanager:v4.5 b7bae610cd46
crash                3/3 4m ago 4M * registry.redhat.io/rhceph-alpha/rhceph-5-
rhel8:latest        c88a5d60f510
grafana              1/1 4m ago 4M count:1 registry.redhat.io/rhceph-alpha/rhceph-5-
dashboard-rhel8:latest bd3d7748747b
mgr                  2/2 4m ago 4M count:2 registry.redhat.io/rhceph-alpha/rhceph-5-
rhel8:latest        c88a5d60f510
mon                  2/2 4m ago 10w count:2 registry.redhat.io/rhceph-alpha/rhceph-5-
rhel8:latest        c88a5d60f510
nfs.foo              0/1 - - count:1 <unknown>
<unknown>
node-exporter        1/3 4m ago 4M * registry.redhat.io/openshift4/ose-
prometheus-node-exporter:v4.5 mix
osd.all-available-devices 5/5 4m ago 3M * registry.redhat.io/rhceph-
alpha/rhceph-5-rhel8:latest c88a5d60f510
prometheus           1/1 4m ago 4M count:1 registry.redhat.io/openshift4/ose-
prometheus:v4.6      bebb0ddef7f0
rgw.test_realm.test_zone 2/2 4m ago 3M count:2 registry.redhat.io/rhceph-
alpha/rhceph-5-rhel8:latest c88a5d60f510
```

3. To start a specific service, run the following command:

### Syntax

```
ceph orch start SERVICE_ID
```

### Example

```
[ceph: root@host01 /]# ceph orch start node-exporter
```

4. To stop a specific service, run the following command:



## IMPORTANT

The **ceph orch stop SERVICE\_ID** command results in the Red Hat Ceph Storage cluster being inaccessible, only for the MON and MGR service. It is recommended to use the **systemctl stop SERVICE\_ID** command to stop a specific daemon in the host.

### Syntax

```
ceph orch stop SERVICE_ID
```

### Example

```
[ceph: root@host01 /]# ceph orch stop node-exporter
```

In the example the **ceph orch stop node-exporter** command removes all the daemons of the **node exporter** service.

- To restart a specific service, run the following command:

### Syntax

```
ceph orch restart SERVICE_ID
```

### Example

```
[ceph: root@host01 /]# ceph orch restart node-exporter
```

## 2.5. VIEWING LOG FILES OF CEPH DAEMONS THAT RUN IN CONTAINERS

Use the **journald** daemon from the container host to view a log file of a Ceph daemon from a container.

### Prerequisites

- Installation of the Red Hat Ceph Storage software.
- Root-level access to the node.

### Procedure

- To view the entire Ceph log file, run a **journalctl** command as **root** composed in the following format:

### Syntax

```
journalctl -u ceph SERVICE_ID
```

```
[root@host01 ~]# journalctl -u ceph-499829b4-832f-11eb-8d6d-001a4a000635@osd.8.service
```

In the above example, you can view the entire log for the OSD with ID **osd.8**.

- To show only the recent journal entries, use the **-f** option.

### Syntax

```
journalctl -fu SERVICE_ID
```

### Example

```
[root@host01 ~]# journalctl -fu ceph-499829b4-832f-11eb-8d6d-001a4a000635@osd.8.service
```



### NOTE

You can also use the **sosreport** utility to view the **journald** logs. For more details about SOS reports, see the [What is an sosreport and how to create one in Red Hat Enterprise Linux?](#) solution on the Red Hat Customer Portal.

### Additional Resources

- The **journalctl** manual page.

## 2.6. POWERING DOWN AND REBOOTING RED HAT CEPH STORAGE CLUSTER

Follow the below procedure for powering down and rebooting the Ceph cluster.

### Prerequisites

- A running Red Hat Ceph Storage cluster.
- Having **root** access.

### Procedure

#### Powering down the Red Hat Ceph Storage cluster

- Stop the clients from using the RBD images and RADOS Gateway on this cluster and any other clients.
- The cluster must be in healthy state (**Health\_OK** and all PGs **active+clean**) before proceeding. Run **ceph status** on the host with the client keyrings, for example, the Ceph Monitor or OpenStack controller nodes, to ensure the cluster is healthy.
- If you use the Ceph File System (**CephFS**), the **CephFS** cluster must be brought down.

### Syntax

```
ceph fs set FS_NAME max_mds 1
ceph fs fail FS_NAME
ceph status
ceph fs set FS_NAME joinable false
ceph mds fail FS_NAME:_N_
```

- 

### Example

```
[ceph: root@host01 /]# ceph fs set cephfs max_mds 1
[ceph: root@host01 /]# ceph fs fail cephfs
[ceph: root@host01 /]# ceph status
[ceph: root@host01 /]# ceph fs set cephfs joinable false
[ceph: root@host01 /]# ceph mds fail cephfs:1
```

4. Set the **noout**, **norecover**, **norebalance**, **nobackfill**, **nodown** and **pause** flags. Run the following on a node with the client keyrings. For example, the Ceph Monitor or OpenStack controller node:

### Example

```
[ceph: root@host01 /]# ceph osd set noout
[ceph: root@host01 /]# ceph osd set norecover
[ceph: root@host01 /]# ceph osd set norebalance
[ceph: root@host01 /]# ceph osd set nobackfill
[ceph: root@host01 /]# ceph osd set nodown
[ceph: root@host01 /]# ceph osd set pause
```

5. Shut down the OSD nodes one by one:

### Example

```
[root@host01 ~]# systemctl stop ceph-499829b4-832f-11eb-8d6d-001a4a000635@osd.2.service
```

6. Shut down the monitor nodes one by one:

### Example

```
[root@host01 ~]# systemctl stop ceph-499829b4-832f-11eb-8d6d-001a4a000635@mon.host01.service
```

## Rebooting the Red Hat Ceph Storage cluster

1. Power on the administration node.
2. Power on the monitor nodes:

### Example

```
[root@host01 ~]# systemctl start ceph-499829b4-832f-11eb-8d6d-001a4a000635@mon.host01.service
```

3. Power on the OSD nodes:

### Example

```
[root@host01 ~]# systemctl start ceph-499829b4-832f-11eb-8d6d-001a4a000635@osd.2.service
```

4. Wait for all the nodes to come up. Verify all the services are up and the connectivity is fine between the nodes.
5. Unset the **noout**, **norecover**, **norebalance**, **nobackfill**, **nodown** and **pause** flags. Run the following on a node with the client keyrings. For example, the Ceph Monitor or OpenStack controller node:

### Example

```
[ceph: root@host01 /]# ceph osd unset noout
[ceph: root@host01 /]# ceph osd unset norecover
[ceph: root@host01 /]# ceph osd unset norebalance
[ceph: root@host01 /]# ceph osd unset nobackfill
[ceph: root@host01 /]# ceph osd unset nodown
[ceph: root@host01 /]# ceph osd unset pause
```

6. If you use the Ceph File System (**CephFS**), the **CephFS** cluster must be brought back up by setting the **joinable** flag to **true**:

### Syntax

```
ceph fs set FS_NAME joinable true
```

### Example

```
[ceph: root@host01 /]# ceph fs set cephfs joinable true
```

7. Verify the cluster is in healthy state (**Health\_OK** and all PGs **active+clean**). Run **ceph status** on a node with the client keyrings. For example, the Ceph Monitor or OpenStack controller nodes, to ensure the cluster is healthy.

### Additional Resources

- For more information on installing Ceph see the [Red Hat Ceph Storage Installation Guide](#)

## CHAPTER 3. MONITORING A CEPH STORAGE CLUSTER

As a storage administrator, you can monitor the overall health of the Red Hat Ceph Storage cluster, along with monitoring the health of the individual components of Ceph.

Once you have a running Red Hat Ceph Storage cluster, you might begin monitoring the storage cluster to ensure that the Ceph Monitor and Ceph OSD daemons are running, at a high-level. Ceph storage cluster clients connect to a Ceph Monitor and receive the latest version of the storage cluster map before they can read and write data to the Ceph pools within the storage cluster. So the monitor cluster must have agreement on the state of the cluster before Ceph clients can read and write data.

Ceph OSDs must peer the placement groups on the primary OSD with the copies of the placement groups on secondary OSDs. If faults arise, peering will reflect something other than the **active + clean** state.

### 3.1. PREREQUISITES

- A running Red Hat Ceph Storage cluster.

### 3.2. HIGH-LEVEL MONITORING OF A CEPH STORAGE CLUSTER

As a storage administrator, you can monitor the health of the Ceph daemons to ensure that they are up and running. High level monitoring also involves checking the storage cluster capacity to ensure that the storage cluster does not exceed its **full ratio**. The [Red Hat Ceph Storage Dashboard](#) is the most common way to conduct high-level monitoring. However, you can also use the command-line interface, the Ceph admin socket or the Ceph API to monitor the storage cluster.

#### 3.2.1. Prerequisites

- A running Red Hat Ceph Storage cluster.

#### 3.2.2. Using the Ceph command interface interactively

You can interactively interface with the Ceph storage cluster by using the **ceph** command-line utility.

##### Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the node.

##### Procedure

1. To run the **ceph** utility in interactive mode.

##### Syntax

```
podman exec -it ceph-mon-MONITOR_NAME /bin/bash
```

##### Replace

- *MONITOR\_NAME* with the name of the Ceph Monitor container, found by running the **podman ps** command.

## Example

```
[root@host01 ~]# podman exec -it ceph-499829b4-832f-11eb-8d6d-001a4a000635-  
mon.host01 /bin/bash
```

This example opens an interactive terminal session on **mon.host01**, where you can start the Ceph interactive shell.

### 3.2.3. Checking the storage cluster health

After you start the Ceph storage cluster, and before you start reading or writing data, check the storage cluster's health first.

#### Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the node.

#### Procedure

1. Log into the Cephadm shell:

#### Example

```
root@host01 ~]# cephadm shell
```

2. You can check on the health of the Ceph storage cluster with the following:

#### Example

```
[ceph: root@host01 /]# ceph health
```

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.

Upon starting the Ceph cluster, you will likely encounter a health warning such as **HEALTH\_WARN XXX num placement groups stale**. Wait a few moments and check it again. When the storage cluster is ready, **ceph health** should return a message such as **HEALTH\_OK**. At that point, it is okay to begin using the cluster.

### 3.2.4. Watching storage cluster events

You can watch events that are happening with the Ceph storage cluster using the command-line interface.

#### Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the node.

#### Procedure

1. Log into the Cephadm shell:

#### Example

```
root@host01 ~]# cephadm shell
```

2. To watch the cluster's ongoing events, run the following command:

#### Example

```
[ceph: root@host01 /]# ceph -w
cluster:
  id: 8c9b0072-67ca-11eb-af06-001a4a0002a0
  health: HEALTH_OK

services:
  mon: 2 daemons, quorum Ceph5-2,Ceph5-adm (age 3d)
  mgr: Ceph5-1.nqikfh(active, since 3w), standbys: Ceph5-adm.meckej
  osd: 5 osds: 5 up (since 2d), 5 in (since 8w)
  rgw: 2 daemons active (test_realm.test_zone.Ceph5-2.bfdwcn,
test_realm.test_zone.Ceph5-adm.acndrh)

data:
  pools: 11 pools, 273 pgs
  objects: 459 objects, 32 KiB
  usage: 2.6 GiB used, 72 GiB / 75 GiB avail
  pgs: 273 active+clean

io:
  client: 170 B/s rd, 730 KiB/s wr, 0 op/s rd, 729 op/s wr

2021-06-02 15:45:21.655871 osd.0 [INF] 17.71 deep-scrub ok
2021-06-02 15:45:47.880608 osd.1 [INF] 1.0 scrub ok
2021-06-02 15:45:48.865375 osd.1 [INF] 1.3 scrub ok
2021-06-02 15:45:50.866479 osd.1 [INF] 1.4 scrub ok
2021-06-02 15:45:01.345821 mon.0 [INF] pgmap v41339: 952 pgs: 952 active+clean; 17130
MB data, 115 GB used, 167 GB / 297 GB avail
```

```

2021-06-02 15:45:05.718640 mon.0 [INF] pgmap v41340: 952 pgs: 1
active+clean+scrubbing+deep, 951 active+clean; 17130 MB data, 115 GB used, 167 GB /
297 GB avail
2021-06-02 15:45:53.997726 osd.1 [INF] 1.5 scrub ok
2021-06-02 15:45:06.734270 mon.0 [INF] pgmap v41341: 952 pgs: 1
active+clean+scrubbing+deep, 951 active+clean; 17130 MB data, 115 GB used, 167 GB /
297 GB avail
2021-06-02 15:45:15.722456 mon.0 [INF] pgmap v41342: 952 pgs: 952 active+clean; 17130
MB data, 115 GB used, 167 GB / 297 GB avail
2021-06-02 15:46:06.836430 osd.0 [INF] 17.75 deep-scrub ok
2021-06-02 15:45:55.720929 mon.0 [INF] pgmap v41343: 952 pgs: 1
active+clean+scrubbing+deep, 951 active+clean; 17130 MB data, 115 GB used, 167 GB /
297 GB avail

```

### 3.2.5. 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 of the two numbers, of the overall storage capacity of the cluster. The notional number reflects the size of the stored data before it is replicated, cloned or snapshotted. 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.

### 3.2.6. Understanding the storage clusters usage stats

To check a cluster's data usage and data distribution among pools, use the **df** option. It is similar to the Linux **df** command. You can run either the **ceph df** command or **ceph df detail** command.

#### Example

```

[ceph: root@host01 /]# ceph df
RAW STORAGE:
  CLASS  SIZE  AVAIL  USED  RAW USED  %RAW USED
  hdd    90 GiB  84 GiB  100 MiB  6.1 GiB    6.78
  TOTAL  90 GiB  84 GiB  100 MiB  6.1 GiB    6.78

POOLS:
  POOL          ID  STORED  OBJECTS  USED  %USED  MAX AVAIL
  .rgw.root          1  1.3 KiB    4  768 KiB    0  26 GiB
  default.rgw.control  2    0 B      8    0 B    0  26 GiB
  default.rgw.meta    3  2.5 KiB   12  2.1 MiB    0  26 GiB
  default.rgw.log     4  3.5 KiB  208  6.2 MiB    0  26 GiB
  default.rgw.buckets.index  5  2.4 KiB   33  2.4 KiB    0  26 GiB
  default.rgw.buckets.data  6  9.6 KiB   15  1.7 MiB    0  26 GiB
  testpool          10  231 B     5  384 KiB    0  40 GiB

```

The **ceph df detail** command gives more details about other pool statistics such as quota objects, quota bytes, used compression, and under compression.

#### Example

```

[ceph: root@host01 /]# ceph df detail
RAW STORAGE:
  CLASS  SIZE  AVAIL  USED  RAW USED  %RAW USED
  hdd    90 GiB  84 GiB  100 MiB  6.1 GiB    6.78

```

TOTAL	90 GiB	84 GiB	100 MiB	6.1 GiB	6.78				
POOLS:									
POOL	QUOTA	OBJECTS	ID	STORED	OBJECTS	USED	%USED	MAX AVAIL	
			QUOTA	BYTES	DIRTY	USED	UNDER	COMPR	
.rgw.root			1	1.3 KiB	4	768 KiB	0	26 GiB	N/A
4	0 B	0 B							
default.rgw.control			2	0 B	8	0 B	0	26 GiB	N/A
8	0 B	0 B							
default.rgw.meta			3	2.5 KiB	12	2.1 MiB	0	26 GiB	N/A
12	0 B	0 B							
default.rgw.log			4	3.5 KiB	208	6.2 MiB	0	26 GiB	N/A
208	0 B	0 B							
default.rgw.buckets.index			5	2.4 KiB	33	2.4 KiB	0	26 GiB	N/A
33	0 B	0 B							
default.rgw.buckets.data			6	9.6 KiB	15	1.7 MiB	0	26 GiB	N/A
15	0 B	0 B							
testpool			10	231 B	5	384 KiB	0	40 GiB	N/A
5	0 B	0 B							

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

- **CLASS:** The type of devices used.
- **SIZE:** The overall storage capacity managed by the storage cluster.  
In the above example, if the **SIZE** is 90 GiB, it is the total size without the replication factor, which is three by default. The total available capacity with the replication factor is  $90 \text{ GiB} / 3 = 30 \text{ GiB}$ . Based on the full ratio, which is 0.85% by default, the maximum available space is  $30 \text{ GiB} * 0.85 = 25.5 \text{ GiB}$
- **AVAIL:** The amount of free space available in the storage cluster.  
In the above example, if the **SIZE** is 90 GiB and the **USED** space is 6 GiB, then the **AVAIL** space is 84 GiB. The total available space with the replication factor, which is three by default, is  $84 \text{ GiB} / 3 = 28 \text{ GiB}$
- **USED:** The amount of used space in the storage cluster consumed by user data, internal overhead, or reserved capacity.  
In the above example, 100 MiB is the total space available after considering the replication factor. The actual available size is 33 MiB.
- **RAW USED:** The sum of **USED** space and the space allocated the **db** and **wal** BlueStore partitions.
- **% RAW USED:** The percentage of of **RAW USED**. Use this number in conjunction with the **full ratio** and **near full ratio** to ensure that you are not reaching the storage cluster's capacity.

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 1 MB of data, the notional usage will be 1 MB, but the actual usage may be 3 MB or more depending on the number of replicas for example, **size = 3**, clones and snapshots.

- **POOL:** The name of the pool.
- **ID:** The pool ID.

- **STORED:** The actual amount of data stored by the user in the pool.
- **OBJECTS:** The notional number of objects stored per pool. It is **STORED** size \* replication factor.
- **USED:** The notional amount of data stored in kilobytes, unless the number appends **M** for megabytes or **G** for gigabytes.
- **%USED:** The notional percentage of storage used per pool.
- **MAX AVAIL:** An estimate of the notional amount of data that can be written to this pool. It is the amount of data that can be used before the first OSD becomes full. It considers the projected distribution of data across disks from the CRUSH map and uses the first OSD to fill up as the target.

In the above example, **MAX AVAIL** is 153.85 MB without considering the replication factor, which is three by default.

See the Red Hat Knowledgebase article titled [ceph df MAX AVAIL is incorrect for simple replicated pool](#) to calculate the value of **MAX AVAIL**.

- **QUOTA OBJECTS:** The number of quota objects.
- **QUOTA BYTES:** The number of bytes in the quota objects.
- **USED COMPR:** The amount of space allocated for compressed data including his includes compressed data, allocation, replication and erasure coding overhead.
- **UNDER COMPR:** The amount of data passed through compression and beneficial enough to be stored in a compressed form.



#### 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 **GLOBAL** section of the output.



#### NOTE

The **MAX AVAIL** value is a complicated function of the replication or erasure code used, the CRUSH rule that maps storage to devices, the utilization of those devices, and the configured **mon\_osd\_full\_ratio**.

#### Additional Resources

- See [How Ceph calculates data usage](#) for details.
- See [Understanding the OSD usage stats](#) for details.

### 3.2.7. Understanding the OSD usage stats

Use the **ceph osd df** command to view OSD utilization stats.

#### Example

```
[ceph: root@host01 /]# ceph osd df
ID CLASS WEIGHT REWEIGHT SIZE USE DATA OMAP META AVAIL %USE VAR
PGS
3 hdd 0.90959 1.00000 931GiB 70.1GiB 69.1GiB 0B 1GiB 861GiB 7.53 2.93 66
4 hdd 0.90959 1.00000 931GiB 1.30GiB 308MiB 0B 1GiB 930GiB 0.14 0.05 59
0 hdd 0.90959 1.00000 931GiB 18.1GiB 17.1GiB 0B 1GiB 913GiB 1.94 0.76 57
MIN/MAX VAR: 0.02/2.98 STDDEV: 2.91
```

- **ID:** The name of the OSD.
- **CLASS:** The type of devices the OSD uses.
- **WEIGHT:** The weight of the OSD in the CRUSH map.
- **REWEIGHT:** The default reweight value.
- **SIZE:** The overall storage capacity of the OSD.
- **USE:** The OSD capacity.
- **DATA:** The amount of OSD capacity that is used by user data.
- **OMAP:** An estimate value of the **bluefs** storage that is being used to store object map ( **omap**) data (key value pairs stored in **rocksdb**).
- **META:** The **bluefs** space allocated, or the value set in the **bluestore\_bluefs\_min** parameter, whichever is larger, for internal metadata which is calculated as the total space allocated in **bluefs** minus the estimated **omap** data size.
- **AVAIL:** The amount of free space available on the OSD.
- **%USE:** The notional percentage of storage used by the OSD
- **VAR:** The variation above or below average utilization.
- **PGS:** The number of placement groups in the OSD.
- **MIN/MAX VAR:** The minimum and maximum variation across all OSDs.

### Additional Resources

- See [How Ceph calculates data usage](#) for details.
- See [Understanding the OSD usage stats](#) for details.
- See [CRUSH Weights](#) in *Red Hat Ceph Storage Storage Strategies Guide* for details.

### 3.2.8. Checking the Red Hat Ceph Storage cluster status

You can check the status of the Red Hat Ceph Storage cluster from the command-line interface. The **status** sub command or the **-s** argument will display the current status of the storage cluster.

### Prerequisites

- A running Red Hat Ceph Storage cluster.

- Root-level access to the node.

## Procedure

1. Log into the Cephadm shell:

### Example

```
root@host01 ~]# cephadm shell
```

2. To check a storage cluster's status, execute the following:

### Example

```
[ceph: root@host01 /]# ceph status
```

Or

### Example

```
[ceph: root@host01 /]# ceph -s
```

3. In interactive mode, type **ceph** and press **Enter**:

### Example

```
[ceph: root@host01 /]# ceph
ceph> status
cluster:
  id: 499829b4-832f-11eb-8d6d-001a4a000635
  health: HEALTH_WARN
        1 stray daemon(s) not managed by cephadm
        1/3 mons down, quorum host03,host02
        too many PGs per OSD (261 > max 250)

services:
  mon: 3 daemons, quorum host03,host02 (age 3d), out of quorum: host01
  mgr: host01.hdhzwn(active, since 9d), standbys: host05.eobuuv, host06.wquwpj
  osd: 12 osds: 11 up (since 2w), 11 in (since 5w)
  rgw: 2 daemons active (test_realm.test_zone.host04.hgbvng,
test_realm.test_zone.host05.yqqilm)
  rgw-nfs: 1 daemon active (nfs.foo.host06-rgw)

data:
  pools: 8 pools, 960 pgs
  objects: 414 objects, 1.0 MiB
  usage: 5.7 GiB used, 214 GiB / 220 GiB avail
  pgs: 960 active+clean

io:
  client: 41 KiB/s rd, 0 B/s wr, 41 op/s rd, 27 op/s wr

ceph> health
HEALTH_WARN 1 stray daemon(s) not managed by cephadm; 1/3 mons down, quorum
```

```
host03,host02; too many PGs per OSD (261 > max 250)
```

```
ceph> mon stat
e3: 3 mons at {host01=[v2:10.74.255.0:3300/0,v1:10.74.255.0:6789/0],host02=
[v2:10.74.249.253:3300/0,v1:10.74.249.253:6789/0],host03=
[v2:10.74.251.164:3300/0,v1:10.74.251.164:6789/0]}, election epoch 6688, leader 1 host03,
quorum 1,2 host03,host02
```

### 3.2.9. Checking the Ceph Monitor status

If the storage cluster has multiple Ceph Monitors, which is a requirement for a production Red Hat Ceph Storage cluster, then you can check the Ceph Monitor quorum status after starting the storage cluster, and before doing any reading or writing of data.

A quorum must be present when multiple monitors are running.

Check Ceph Monitor status periodically to ensure that they are running. If there is a problem with the Ceph Monitor, that prevents an agreement on the state of the storage cluster, the fault may prevent Ceph clients from reading and writing data.

#### Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the node.

#### Procedure

1. Log into the Cephadm shell:

#### Example

```
root@host01 ~]# cephadm shell
```

2. To display the monitor map, execute the following:

#### Example

```
[ceph: root@host01 /]# ceph mon stat
```

or

#### Example

```
[ceph: root@host01 /]# ceph mon dump
```

3. To check the quorum status for the storage cluster, execute the following:

```
[ceph: root@host01 /]# ceph quorum_status -f json-pretty
```

Ceph will return the quorum status.

#### Example

```

{
  "election_epoch": 6686,
  "quorum": [
    0,
    1,
    2
  ],
  "quorum_names": [
    "host01",
    "host03",
    "host02"
  ],
  "quorum_leader_name": "host01",
  "quorum_age": 424884,
  "features": {
    "quorum_con": "4540138297136906239",
    "quorum_mon": [
      "kraken",
      "luminous",
      "mimic",
      "osdmap-prune",
      "nautilus",
      "octopus",
      "pacific",
      "elector-pinging"
    ]
  },
  "monmap": {
    "epoch": 3,
    "fsid": "499829b4-832f-11eb-8d6d-001a4a000635",
    "modified": "2021-03-15T04:51:38.621737Z",
    "created": "2021-03-12T12:35:16.911339Z",
    "min_mon_release": 16,
    "min_mon_release_name": "pacific",
    "election_strategy": 1,
    "disallowed_leaders": "",
    "stretch_mode": false,
    "features": {
      "persistent": [
        "kraken",
        "luminous",
        "mimic",
        "osdmap-prune",
        "nautilus",
        "octopus",
        "pacific",
        "elector-pinging"
      ],
      "optional": []
    },
  },
  "mons": [
    {
      "rank": 0,
      "name": "host01",
      "public_addrs": {
        "addrvec": [

```

```

        {
            "type": "v2",
            "addr": "10.74.255.0:3300",
            "nonce": 0
        },
        {
            "type": "v1",
            "addr": "10.74.255.0:6789",
            "nonce": 0
        }
    ]
},
"addr": "10.74.255.0:6789/0",
"public_addr": "10.74.255.0:6789/0",
"priority": 0,
"weight": 0,
"crush_location": "{}"
},
{
    "rank": 1,
    "name": "host03",
    "public_addrs": {
        "addrvec": [
            {
                "type": "v2",
                "addr": "10.74.251.164:3300",
                "nonce": 0
            },
            {
                "type": "v1",
                "addr": "10.74.251.164:6789",
                "nonce": 0
            }
        ]
    },
    "addr": "10.74.251.164:6789/0",
    "public_addr": "10.74.251.164:6789/0",
    "priority": 0,
    "weight": 0,
    "crush_location": "{}"
},
{
    "rank": 2,
    "name": "host02",
    "public_addrs": {
        "addrvec": [
            {
                "type": "v2",
                "addr": "10.74.249.253:3300",
                "nonce": 0
            },
            {
                "type": "v1",
                "addr": "10.74.249.253:6789",
                "nonce": 0
            }
        ]
    }
}

```

```

    ]
  },
  "addr": "10.74.249.253:6789/0",
  "public_addr": "10.74.249.253:6789/0",
  "priority": 0,
  "weight": 0,
  "crush_location": "{}"
}
]
}
}

```

### 3.2.10. Using the Ceph administration socket

Use the administration socket to interact with a given daemon directly by using a UNIX socket file. For example, the socket enables you to:

- List the Ceph configuration at runtime
- Set configuration values at runtime directly without relying on Monitors. This is useful when Monitors are **down**.
- Dump historic operations
- Dump the operation priority queue state
- Dump operations without rebooting
- Dump performance counters

In addition, using the socket is helpful when troubleshooting problems related to Monitors or OSDs.



#### IMPORTANT

The administration socket is only available while a daemon is running. When you shut down the daemon properly, the administration socket is removed. However, if the daemon terminates unexpectedly, the administration socket might persist.

Regardless, if the daemon is not running, a following error is returned when attempting to use the administration socket:

```
Error 111: Connection Refused
```

#### Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the node.

#### Procedure

1. Log into the Cephadm shell:

#### Example

```
root@host01 ~]# cephadm shell
```

2. To use the socket:

## Syntax

```
ceph daemon MONITOR_ID COMMAND
```

Replace:

- ***MONITOR\_ID*** of the daemon
- ***COMMAND*** with the command to run. Use **help** to list the available commands for a given daemon.

## Example

To view a Monitor status of a Ceph Monitor:

```
[ceph: root@host01 /]# ceph daemon mon.host01 help
{
  "add_bootstrap_peer_hint": "add peer address as potential bootstrap peer for cluster bringup",
  "add_bootstrap_peer_hintv": "add peer address vector as potential bootstrap peer for cluster bringup",
  "compact": "cause compaction of monitor's leveldb/rocksdb storage",
  "config diff": "dump diff of current config and default config",
  "config diff get": "dump diff get <field>: dump diff of current and default config setting <field>",
  "config get": "config get <field>: get the config value",
  "config help": "get config setting schema and descriptions",
  "config set": "config set <field> <val> [<val> ...]: set a config variable",
  "config show": "dump current config settings",
  "config unset": "config unset <field>: unset a config variable",
  "connection scores dump": "show the scores used in connectivity-based elections",
  "connection scores reset": "reset the scores used in connectivity-based elections",
  "dump_historic_ops": "dump_historic_ops",
  "dump_mempools": "get mempool stats",
  "get_command_descriptions": "list available commands",
  "git_version": "get git sha1",
  "heap": "show heap usage info (available only if compiled with tcmalloc)",
  "help": "list available commands",
  "injectargs": "inject configuration arguments into running daemon",
  "log dump": "dump recent log entries to log file",
  "log flush": "flush log entries to log file",
  "log reopen": "reopen log file",
  "mon_status": "report status of monitors",
  "ops": "show the ops currently in flight",
  "perf dump": "dump perfcounters value",
  "perf histogram dump": "dump perf histogram values",
  "perf histogram schema": "dump perf histogram schema",
  "perf reset": "perf reset <name>: perf reset all or one perfcounter name",
  "perf schema": "dump perfcounters schema",
  "quorum enter": "force monitor back into quorum",
  "quorum exit": "force monitor out of the quorum",
```

```

"sessions": "list existing sessions",
"smart": "Query health metrics for underlying device",
"sync_force": "force sync of and clear monitor store",
"version": "get ceph version"
}

[ceph: root@host01 /]# ceph daemon mon.host01 mon_status

{
  "name": "host01",
  "rank": 0,
  "state": "leader",
  "election_epoch": 120,
  "quorum": [
    0,
    1,
    2
  ],
  "quorum_age": 206358,
  "features": {
    "required_con": "2449958747317026820",
    "required_mon": [
      "kraken",
      "luminous",
      "mimic",
      "osdmap-prune",
      "nautilus",
      "octopus",
      "pacific",
      "elector-pinging"
    ],
    "quorum_con": "4540138297136906239",
    "quorum_mon": [
      "kraken",
      "luminous",
      "mimic",
      "osdmap-prune",
      "nautilus",
      "octopus",
      "pacific",
      "elector-pinging"
    ]
  },
  "outside_quorum": [],
  "extra_probe_peers": [],
  "sync_provider": [],
  "monmap": {
    "epoch": 3,
    "fsid": "81a4597a-b711-11eb-8cb8-001a4a000740",
    "modified": "2021-05-18T05:50:17.782128Z",
    "created": "2021-05-17T13:13:13.383313Z",
    "min_mon_release": 16,
    "min_mon_release_name": "pacific",
    "election_strategy": 1,
    "disallowed_leaders": "",
    "stretch_mode": false,

```

```

"features": {
  "persistent": [
    "kraken",
    "luminous",
    "mimic",
    "osdmap-prune",
    "nautilus",
    "octopus",
    "pacific",
    "elector-pinging"
  ],
  "optional": []
},
"mons": [
  {
    "rank": 0,
    "name": "host01",
    "public_addrs": {
      "addrvec": [
        {
          "type": "v2",
          "addr": "10.74.249.41:3300",
          "nonce": 0
        },
        {
          "type": "v1",
          "addr": "10.74.249.41:6789",
          "nonce": 0
        }
      ]
    },
    "addr": "10.74.249.41:6789/0",
    "public_addr": "10.74.249.41:6789/0",
    "priority": 0,
    "weight": 0,
    "crush_location": "{}"
  },
  {
    "rank": 1,
    "name": "host02",
    "public_addrs": {
      "addrvec": [
        {
          "type": "v2",
          "addr": "10.74.249.55:3300",
          "nonce": 0
        },
        {
          "type": "v1",
          "addr": "10.74.249.55:6789",
          "nonce": 0
        }
      ]
    },
    "addr": "10.74.249.55:6789/0",
    "public_addr": "10.74.249.55:6789/0",

```

```

    "priority": 0,
    "weight": 0,
    "crush_location": "{}"
  },
  {
    "rank": 2,
    "name": "host03",
    "public_addrs": {
      "addrvec": [
        {
          "type": "v2",
          "addr": "10.74.249.49:3300",
          "nonce": 0
        },
        {
          "type": "v1",
          "addr": "10.74.249.49:6789",
          "nonce": 0
        }
      ]
    },
    "addr": "10.74.249.49:6789/0",
    "public_addr": "10.74.249.49:6789/0",
    "priority": 0,
    "weight": 0,
    "crush_location": "{}"
  }
]
},
"feature_map": {
  "mon": [
    {
      "features": "0x3f01cfb9ffdf",
      "release": "luminous",
      "num": 1
    }
  ],
  "osd": [
    {
      "features": "0x3f01cfb9ffdf",
      "release": "luminous",
      "num": 3
    }
  ]
},
"stretch_mode": false
}

```

- Alternatively, specify the Ceph daemon by using its socket file:

```
ceph daemon /var/run/ceph/SOCKET_FILE COMMAND
```

- To view the status of an Ceph OSD named **osd.2**:

```
[ceph: root@host01 /]# ceph daemon /var/run/ceph/ceph-osd.2.asok status
```

- To list all socket files for the Ceph processes:

```
[ceph: root@host01 /]# ls /var/run/ceph
```

### Additional Resources

- See the [Red Hat Ceph Storage Troubleshooting Guide](#) for more information.

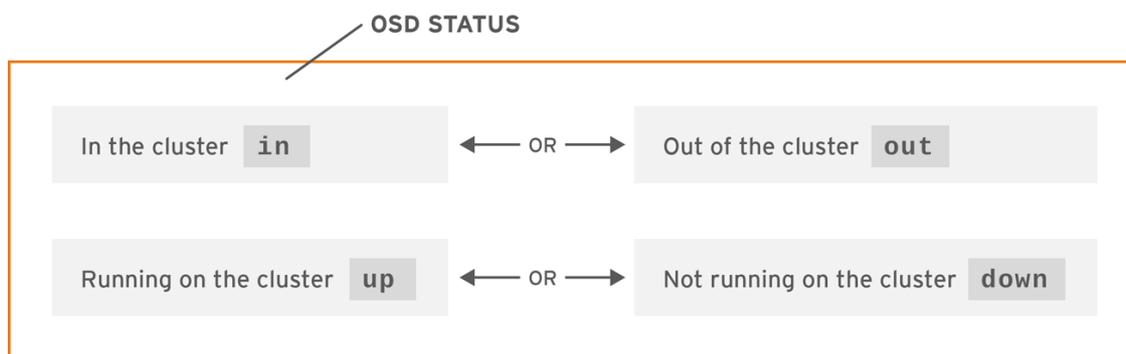
### 3.2.11. Understanding the Ceph OSD status

An OSD's status is either in the cluster, **in**, or out of the cluster, **out**. It is either up and running, **up**, or it is down and not running, or **down**. If an OSD is **up**, it may be either **in** the storage cluster, where data can be read and written, or it is **out** of the storage 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 the OSD. If an OSD is **down**, it should also be **out**.



#### NOTE

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



CEPH\_459704\_1017

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**. Don't panic. With respect to OSDs, you should expect that the cluster will **NOT** echo **HEALTH OK** in a few expected circumstances:

- You haven't started the cluster yet, it won't respond.
- You have just started or restarted the cluster and it's not ready yet, because the placement groups are getting created and the OSDs are in the process of peering.
- You just added or removed an OSD.
- You just have modified the cluster map.

An important aspect of monitoring OSDs is to ensure that when the cluster is up and running that all OSDs that are **in** the cluster are **up** and running, too.

To see if all OSDs are running, execute:

#### Example

```
[ceph: root@host01 /]# ceph osd stat
```

or

### Example

```
[ceph: root@host01 /]# ceph osd dump
```

The result should tell you the map epoch, **eNNNN**, the total number of OSDs, **x**, how many, **y**, are **up**, and how many, **z**, are **in**:

```
eNNNN: x osds: y up, z in
```

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 aren't running:

### Example

```
[ceph: root@host01 /]# ceph osd tree

# id  weight type name  up/down reweight
-1 3  pool default
-3 3  rack mainrack
-2 3  host osd-host
0 1  osd.0 up 1
1 1  osd.1 up 1
2 1  osd.2 up 1
```

### TIP

The ability to search through a well-designed CRUSH hierarchy may help you troubleshoot the storage cluster by identifying the physical locations faster.

If an OSD is **down**, connect to the node and start it. You can use Red Hat Storage Console to restart the OSD node, or you can use the command line.

### Syntax

```
systemctl start SERVICE_ID_OF_OSD_
```

### Example

```
[root@host01 ~]# systemctl start ceph-499829b4-832f-11eb-8d6d-001a4a000635@osd.6.service
```

### Additional Resources

- See the [Red Hat Ceph Storage Dashboard Guide](#) for more details.

## 3.3. LOW-LEVEL MONITORING OF A CEPH STORAGE CLUSTER

As a storage administrator, you can monitor the health of a Red Hat Ceph Storage cluster from a low-level perspective. Low-level monitoring typically involves ensuring that Ceph OSDs are peering properly. When peering faults occur, placement groups operate in a degraded state. This degraded state

can be the result of many different things, such as hardware failure, a hung or crashed Ceph daemon, network latency, or a complete site outage.

### 3.3.1. Prerequisites

- A running Red Hat Ceph Storage cluster.

### 3.3.2. Monitoring 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 the 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, don't panic. Common examples include:

- You added or removed an OSD. Then, CRUSH reassigned the placement group to other OSDs—thereby changing the composition of the Acting Set and spawning 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, that is, Ceph usually echoes a **HEALTH WARN** state with a "stuck stale" message in such scenarios.

#### Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the node.

#### Procedure

1. Log into the Cephadm shell:

##### Example

```
root@host01 ~]# cephadm shell
```

2. To retrieve a list of placement groups:

##### Example

```
[ceph: root@host01 /]# ceph pg dump
```

3. View which OSDs are in the Acting Set or in the Up Set for a given placement group.

## Syntax

```
ceph pg map PG_NUM
```

## Example

```
[ceph: root@host01 /]# ceph pg map 128
```



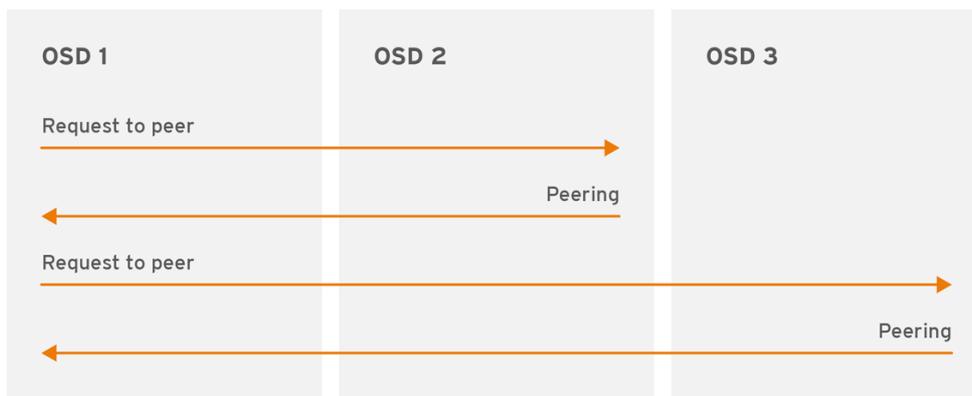
### NOTE

If the Up Set and Acting Set do not match, this may be an indicator that the cluster is rebalancing itself or of a potential problem with the cluster.

### 3.3.3. Ceph OSD 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 that is, 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 3.1. Peering



CEPH\_459704\_1017

### 3.3.4. 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 **HEALTH OK**. After you check to see if the OSDs are running, you should also check placement group states. You should expect that the cluster will **NOT** echo **HEALTH OK** in a number of placement group peering-related circumstances:

- You have just created a pool and placement groups haven't peered yet.
- The placement groups are recovering.
- You have just added an OSD to or removed an OSD from the cluster.
- You have just modified the CRUSH map and the placement groups are migrating.
- There is inconsistent data in different replicas of a placement group.
- Ceph is scrubbing a placement group's replicas.

- Ceph doesn't have enough storage capacity to complete backfilling operations.

If one of the foregoing circumstances causes Ceph to echo **HEALTH WARN**, don't 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 that all placement groups are **active**, and preferably in the **clean** state.

To see the status of all placement groups, execute:

### Example

```
[ceph: root@host01 /]# ceph pg stat
```

The result should tell you the placement group map version, **vNNNNNN**, the total number of placement groups, **x**, and how many placement groups, **y**, are in a particular state such as **active+clean**:

```
vNNNNNN: x pgs: y active+clean; z bytes data, aa MB used, bb GB / cc GB avail
```



### NOTE

It is common for Ceph to report multiple states for placement groups.

### Snapshot Trimming PG States

When snapshots exist, two additional PG states will be reported.

- **snaptrim** : The PGs are currently being trimmed
- **snaptrim\_wait** : The PGs are waiting to be trimmed

Example Output:

```
244 active+clean+snaptrim_wait
32 active+clean+snaptrim
```

In addition to the placement group states, Ceph will also echo back the amount of data 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 the **near full ratio** or **full ratio**.
- Your data isn't getting distributed across the cluster due to an error in the CRUSH configuration.

### Placement Group IDs

Placement group IDs consist of the pool number, and not the 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**. The default pool names **data**, **metadata** and **rbd** correspond to pool numbers **0**, **1** and **2** respectively. A fully qualified placement group ID has the following form:

### Syntax

```
POOL_NUM.PG_ID
```

Example output:

```
0.1f
```

- To retrieve a list of placement groups:

### Example

```
[ceph: root@host01 /]# ceph pg dump
```

- To format the output in JSON format and save it to a file:

### Syntax

```
ceph pg dump -o FILE_NAME --format=json
```

### Example

```
[ceph: root@host01 /]# ceph pg dump -o test --format=json
```

- Query a particular placement group:

### Syntax

```
ceph pg POOL_NUM.PG_ID query
```

### Example

```
[ceph: root@host01 /]# ceph pg 5.fe query
{
  "snap_trimq": "[]",
  "snap_trimq_len": 0,
  "state": "active+clean",
  "epoch": 2449,
  "up": [
    3,
    8,
    10
  ],
  "acting": [
    3,
    8,
    10
  ],
  "acting_recovery_backfill": [
    "3",
    "8",
    "10"
  ],
  "info": {
    "pgid": "5.ff",
    "last_update": "0'0",
    "last_complete": "0'0",
```

```

"log_tail": "0'0",
"last_user_version": 0,
"last_backfill": "MAX",
"purged_snaps": [],
"history": {
  "epoch_created": 114,
  "epoch_pool_created": 82,
  "last_epoch_started": 2402,
  "last_interval_started": 2401,
  "last_epoch_clean": 2402,
  "last_interval_clean": 2401,
  "last_epoch_split": 114,
  "last_epoch_marked_full": 0,
  "same_up_since": 2401,
  "same_interval_since": 2401,
  "same_primary_since": 2086,
  "last_scrub": "0'0",
  "last_scrub_stamp": "2021-06-17T01:32:03.763988+0000",
  "last_deep_scrub": "0'0",
  "last_deep_scrub_stamp": "2021-06-17T01:32:03.763988+0000",
  "last_clean_scrub_stamp": "2021-06-17T01:32:03.763988+0000",
  "prior_readable_until_ub": 0
},
"stats": {
  "version": "0'0",
  "reported_seq": "2989",
  "reported_epoch": "2449",
  "state": "active+clean",
  "last_fresh": "2021-06-18T05:16:59.401080+0000",
  "last_change": "2021-06-17T01:32:03.764162+0000",
  "last_active": "2021-06-18T05:16:59.401080+0000",
  ....

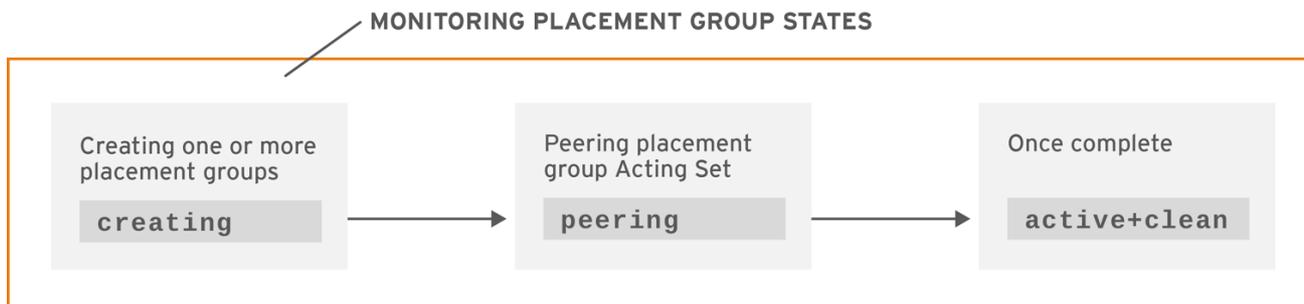
```

### Additional Resources

- See the chapter *Object Storage Daemon (OSD) configuration options* in the [OSD Object storage daemon configuratopn options](#) section in *Red Hat Ceph Storage Configuration Guide* for more details on the snapshot trimming settings.

### 3.3.5. Placement Group creating state

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. Once they are created, the OSDs that are part of a placement group's Acting Set will peer. Once peering is complete, the placement group status should be **active+clean**, which means a Ceph client can begin writing to the placement group.



CEPH\_459704\_1017

### 3.3.6. Placement group peering state

When Ceph is Peering a placement group, Ceph 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.

#### 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 disseminate 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.

### 3.3.7. Placement group active state

Once 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.

### 3.3.8. Placement Group clean state

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.

### 3.3.9. Placement Group degraded state

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 doesn't 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 online. 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 `mon_osd_down_out_interval`, which is set to **600** seconds by default.

A placement group can also be **degraded**, because Ceph cannot find one or more objects that Ceph thinks 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.

Let's say there are nine OSDs in a three way replica pool. If OSD number 9 goes down, the PGs assigned to OSD 9 go in a degraded state. If OSD 9 doesn't recover, it goes out of the cluster and the cluster rebalances. In that scenario, the PGs are degraded and then recover to an active state.

### 3.3.10. Placement Group recovering state

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 isn't always trivial, because a hardware failure might 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 one of the OSDs must recover once 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_threads` setting limits the number of threads for the recovery process, by default one thread. 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 entertain 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.

### 3.3.11. Back fill state

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. Once 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 isn't underway yet \* `backfill` indicates that a backfill operation is underway \* `backfill_too_full` indicates that a backfill operation was requested, but couldn't be completed due to 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 spike 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 `osd_backfill_full_ratio` enables an OSD to refuse a backfill request if the OSD is approaching its full ratio, by default 85%. If an OSD refuses a backfill request, the `osd_backfill_retry_interval` enables an OSD to retry the request, by default after 10 seconds. OSDs can also set `osd_backfill_scan_min` and `osd_backfill_scan_max` to manage scan intervals, by default 64 and 512.

For some workloads, it is beneficial to avoid regular recovery entirely and use backfill instead. Since backfilling occurs in the background, this allows I/O to proceed on the objects in the OSD. To force backfill rather than recovery, set **osd\_min\_pg\_log\_entries** to **1**, and set **osd\_max\_pg\_log\_entries** to **2**. Contact your Red Hat Support account team for details on when this situation is appropriate for your workload.

### 3.3.12. Placement Group remapped state

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. Once data migration completes, the mapping uses the primary OSD of the new acting set.

### 3.3.13. Placement Group stale state

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 aren't reporting statistics in a timely manner. For example, a temporary network fault. By default, OSD daemons report their placement group, up thru, boot and failure statistics every half second, that is, **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 **down**, the monitors will mark the placement group **stale**.

When you start the storage cluster, it is common to see the **stale** state until the peering process completes. After the storage cluster has been running for awhile, 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.

### 3.3.14. Placement Group misplaced state

There are some temporary backfilling scenarios where a PG gets mapped temporarily to an OSD. When that **temporary** situation should no longer be the case, the PGs might still reside in the temporary location and not in the proper location. In which case, they are said to be **misplaced**. That's because the correct number of extra copies actually exist, but one or more copies is in the wrong place.

For example, there are 3 OSDs: 0,1,2 and all PGs map to some permutation of those three. If you add another OSD (OSD 3), some PGs will now map to OSD 3 instead of one of the others. However, until OSD 3 is backfilled, the PG will have a temporary mapping allowing it to continue to serve I/O from the old mapping. During that time, the PG is **misplaced**, because it has a temporary mapping, but not **degraded**, since there are 3 copies.

#### Example

```
pg 1.5: up=acting: [0,1,2]
ADD_OSD_3
pg 1.5: up: [0,3,1] acting: [0,1,2]
```

[0,1,2] is a temporary mapping, so the **up** set is not equal to the **acting** set and the PG is **misplaced** but not **degraded** since [0,1,2] is still three copies.

#### Example

```
pg 1.5: up=acting: [0,3,1]
```

OSD 3 is now backfilled and the temporary mapping is removed, not degraded and not misplaced.

### 3.3.15. Placement Group incomplete state

A PG goes into a **incomplete** state when there is incomplete content and peering fails, that is, when there are no complete OSDs which are current enough to perform recovery.

Lets say OSD 1, 2, and 3 are the acting OSD set and it switches to OSD 1, 4, and 3, then **osd.1** will request a temporary acting set of OSD 1, 2, and 3 while backfilling 4. During this time, if OSD 1, 2, and 3 all go down, **osd.4** will be the only one left which might not have fully backfilled all the data. At this time, the PG will go **incomplete** indicating that there are no complete OSDs which are current enough to perform recovery.

Alternately, if **osd.4** is not involved and the acting set is simply OSD 1, 2, and 3 when OSD 1, 2, and 3 go down, the PG would likely go **stale** indicating that the mons have not heard anything on that PG since the acting set changed. The reason being there are no OSDs left to notify the new OSDs.

### 3.3.16. Identifying stuck Placement Groups

As previously noted, a placement group isn't necessarily problematic just because its state isn't **active+clean**. Generally, Ceph's ability to self repair may not be working when placement groups get stuck. The stuck states include:

- **Unclean:** Placement groups contain objects that are not replicated the desired number of times. They should be recovering.
- **Inactive:** Placement groups cannot process reads or writes because they are waiting for an OSD with the most up-to-date data to come back **up**.
- **Stale:** Placement groups are in an unknown state, because the OSDs that host them have not reported to the monitor cluster in a while, and can be configured with the **mon osd report timeout** setting.

#### Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the node.

#### Procedure

1. To identify stuck placement groups, execute the following:

#### Syntax

```
ceph pg dump_stuck {inactive|unclean|stale|undersized|degraded
[inactive|unclean|stale|undersized|degraded...]} {<int>}
```

#### Example

```
[ceph: root@host01 /]# ceph pg dump_stuck stale
OK
```

### 3.3.17. Finding an object's location

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.

### Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the node.

### Procedure

1. To find the object location, all you need is the object name and the pool name:

#### Syntax

```
ceph osd map POOL_NAME OBJECT_NAME
```

#### Example

```
[ceph: root@host01 /]# ceph osd map mypool myobject
```

## CHAPTER 4. OVERRIDE CEPH BEHAVIOR

As a storage administrator, you need to understand how to use overrides for the Red Hat Ceph Storage cluster to change Ceph options during runtime.

### 4.1. PREREQUISITES

- A running Red Hat Ceph Storage cluster.

### 4.2. SETTING AND UNSETTING CEPH OVERRIDE OPTIONS

You can set and unset Ceph options to override Ceph’s default behavior.

#### Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the node.

#### Procedure

1. To override Ceph’s default behavior, use the **ceph osd set** command and the behavior you wish to override:

#### Syntax

```
ceph osd set FLAG
```

Once you set the behavior, **ceph health** will reflect the override(s) that you have set for the cluster.

#### Example

```
[ceph: root@host01 /]# ceph osd set noout
```

2. To cease overriding Ceph’s default behavior, use the **ceph osd unset** command and the override you wish to cease.

#### Syntax

```
ceph osd unset FLAG
```

#### Example

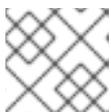
```
[ceph: root@host01 /]# ceph osd unset noout
```

Flag	Description
<b>noin</b>	Prevents OSDs from being treated as <b>in</b> the cluster.

Flag	Description
<b>noout</b>	Prevents OSDs from being treated as <b>out</b> of the cluster.
<b>noup</b>	Prevents OSDs from being treated as <b>up</b> and running.
<b>nodown</b>	Prevents OSDs from being treated as <b>down</b> .
<b>full</b>	Makes a cluster appear to have reached its <b>full_ratio</b> , and thereby prevents write operations.
<b>pause</b>	Ceph will stop processing read and write operations, but will not affect OSD <b>in, out, up</b> or <b>down</b> statuses.
<b>nobackfill</b>	Ceph will prevent new backfill operations.
<b>norebalance</b>	Ceph will prevent new rebalancing operations.
<b>norecover</b>	Ceph will prevent new recovery operations.
<b>noscrub</b>	Ceph will prevent new scrubbing operations.
<b>nodeep-scrub</b>	Ceph will prevent new deep scrubbing operations.
<b>notieragent</b>	Ceph will disable the process that is looking for cold/dirty objects to flush and evict.

### 4.3. CEPH OVERRIDE USE CASES

- **noin**: Commonly used with **noout** to address flapping OSDs.
- **noout**: If the **mon osd report timeout** is exceeded and an OSD has not reported to the monitor, the OSD will get marked **out**. If this happens erroneously, you can set **noout** to prevent the OSD(s) from getting marked **out** while you troubleshoot the issue.
- **noup**: Commonly used with **nodown** to address flapping OSDs.
- **nodown**: Networking issues may interrupt Ceph 'heartbeat' processes, and an OSD may be **up** but still get marked down. You can set **nodown** to prevent OSDs from getting marked down while troubleshooting the issue.
- **full**: If a cluster is reaching its **full\_ratio**, you can pre-emptively set the cluster to **full** and expand capacity.



#### NOTE

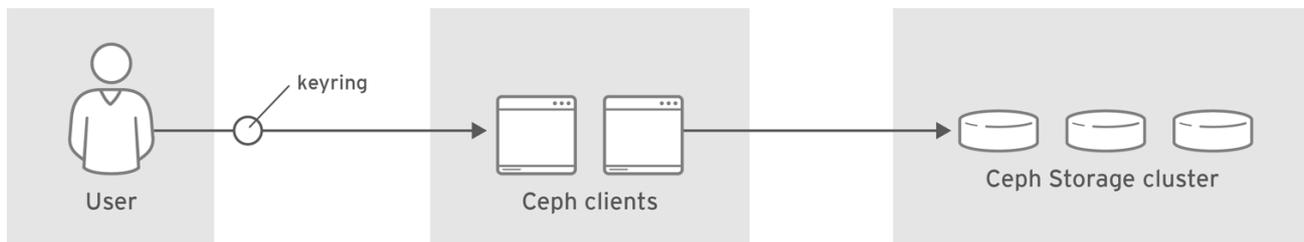
Setting the cluster to **full** will prevent write operations.

- **pause**: If you need to troubleshoot a running Ceph cluster without clients reading and writing data, you can set the cluster to **pause** to prevent client operations.

- **nobackfill**: If you need to take an OSD or node **down** temporarily, for example, upgrading daemons, you can set **nobackfill** so that Ceph will not backfill while the OSDs is **down**.
- **norecover**: If you need to replace an OSD disk and don't want the PGs to recover to another OSD while you are hotswapping disks, you can set **norecover** to prevent the other OSDs from copying a new set of PGs to other OSDs.
- **noscrub** and **nodeep-scrubb**: If you want to prevent scrubbing for example, to reduce overhead during high loads, recovery, backfilling, and rebalancing you can set **noscrub** and/or **nodeep-scrub** to prevent the cluster from scrubbing OSDs.
- **notieragent**: If you want to stop the tier agent process from finding cold objects to flush to the backing storage tier, you may set **notieragent**.

## CHAPTER 5. CEPH USER MANAGEMENT

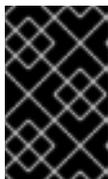
As a storage administrator, you can manage the Ceph user base by providing authentication, and access control to objects in the Red Hat Ceph Storage cluster.



CEPH\_459704\_1017

### 5.1. PREREQUISITES

- A running Red Hat Ceph Storage cluster.
- Access to a Ceph Monitor or Ceph client node.



#### IMPORTANT

Cephadm manages the client keyrings for the Red Hat Ceph Storage cluster as long as the clients are within the scope of Cephadm. Users should not modify the keyrings that are managed by Cephadm, unless there is troubleshooting.

### 5.2. CEPH USER MANAGEMENT BACKGROUND

When Ceph runs with authentication and authorization enabled, you must specify a user name. If you do not specify a user name, Ceph will use the **client.admin** administrative user as the default user name.

Alternatively, you may use the **CEPH\_ARGS** environment variable to avoid re-entry of the user name and secret.

Irrespective of the type of Ceph client, for example, block device, object store, file system, native API, or the Ceph command line, Ceph stores all data as objects within pools. Ceph users must have access to pools in order to read and write data. Additionally, administrative Ceph users must have permissions to execute Ceph's administrative commands.

The following concepts can help you understand Ceph user management.

#### Storage Cluster Users

A user of the Red Hat Ceph Storage cluster is either an individual or as an application. Creating users allows you to control who can access the storage cluster, its pools, and the data within those pools.

Ceph has the notion of a **type** of user. 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, and OSDs 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 the 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`. Red Hat recommends using the type and name as a best practice wherever possible.



## NOTE

A Red Hat Ceph Storage cluster user is not the same as a Ceph Object Gateway user. The object gateway uses a Red Hat Ceph Storage cluster user to communicate between the gateway daemon and the storage cluster, but the gateway has its own user management functionality for its end users.

## Authorization capabilities

Ceph uses the term "capabilities" (caps) to describe authorizing an authenticated user to exercise the functionality of the Ceph Monitors and OSDs. Capabilities can also restrict access to data within a pool or a namespace within a pool. A Ceph administrative user sets a user's capabilities when creating or updating a user. Capability syntax follows the form:

### Syntax

```
DAEMON_TYPE 'allow CAPABILITY' [DAEMON_TYPE 'allow CAPABILITY']
```

- **Monitor Caps:** Monitor capabilities include `r`, `w`, `x`, **allow profile CAP**, and **profile rbd**.

### Example

```
mon 'allow rwx`
mon 'allow profile osd'
```

- **OSD Caps:** OSD capabilities include `r`, `w`, `x`, **class-read**, **class-write**, **profile osd**, **profile rbd**, and **profile rbd-read-only**. Additionally, OSD capabilities also allow for pool and namespace settings. :

### Syntax

```
osd 'allow CAPABILITY' [pool=POOL_NAME] [namespace=NAMESPACE_NAME]
```



## NOTE

The Ceph Object Gateway daemon (**radosgw**) is a client of the Ceph storage cluster, so it isn't represented as a Ceph storage cluster daemon type.

The following entries describe each capability.

<b>allow</b>	Precedes access settings for a daemon.
<b>r</b>	Gives the user read access. Required with monitors to retrieve the CRUSH map.
<b>w</b>	Gives the user write access to objects.

<b>x</b>	Gives the user the capability to call class methods (that is, both read and write) and to conduct <b>auth</b> operations on monitors.
<b>class-read</b>	Gives the user the capability to call class read methods. Subset of <b>x</b> .
<b>class-write</b>	Gives the user the capability to call class write methods. Subset of <b>x</b> .
*	Gives the user read, write and execute permissions for a particular daemon or pool, and the ability to execute admin commands.
<b>profile osd</b>	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.
<b>profile bootstrap-osd</b>	Gives a user permissions to bootstrap an OSD, so that they have permissions to add keys when bootstrapping an OSD.
<b>profile rbd</b>	Gives a user read-write access to the Ceph Block Devices.
<b>profile rbd-read-only</b>	Gives a user read-only access to the Ceph Block Devices.

## Pool

A pool defines a storage strategy for Ceph clients, and acts as a logical partition for that strategy.

In Ceph deployments, it is common to create a pool to support different types of use cases. For example, cloud volumes or images, object storage, hot storage, cold storage, and so on. 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**, **client.cinder**, and so on.

## Namespace

Objects within a pool can be associated to a namespace—a logical group of objects within the pool. A user's access to a pool can be associated with a namespace such that reads and writes by the user take place only within the namespace. Objects written to a namespace within the pool can only be accessed by users who have access to the namespace.



### NOTE

Currently, namespaces are only useful for applications written on top of **librados**. Ceph clients such as block device and object storage do not currently support this feature.

The rationale for namespaces is that pools can be a computationally expensive method of segregating data by use case, because each pool creates a set of placement groups that get mapped to OSDs. If multiple pools use the same CRUSH hierarchy and ruleset, OSD performance may degrade as load increases.

For example, a pool should have approximately 100 placement groups per OSD. So an exemplary cluster with 1000 OSDs would have 100,000 placement groups for one pool. Each pool mapped to the same CRUSH hierarchy and ruleset would create another 100,000 placement groups in the exemplary cluster.

By contrast, writing an object to a namespace simply associates the namespace to the object name without the computational overhead of a separate pool. Rather than creating a separate pool for a user or set of users, you may use a namespace.



## NOTE

Only available using **librados** at this time.

### Additional Resources

- See the [Red Hat Ceph Storage Configuration Guide](#) for details on configuring the use of authentication.

## 5.3. MANAGING CEPH USERS

As a storage administrator, you can manage Ceph users by creating, modifying, deleting, and importing users. A Ceph client user can be either individuals or applications, which use Ceph clients to interact with the Red Hat Ceph Storage cluster daemons.

### 5.3.1. Prerequisites

- A running Red Hat Ceph Storage cluster.
- Access to a Ceph Monitor or Ceph client node.

### 5.3.2. Listing Ceph users

You can list the users in the storage cluster using the command-line interface.

#### Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the node.

#### Procedure

1. To list the users in the storage cluster, execute the following:

#### Example

```
[ceph: root@host01 /]# ceph auth list
installed auth entries:

osd.10
key: AQBW7U5gqOsEExAAg/CxSwZ/gSh8iOsDV3iQOA==
caps: [mgr] allow profile osd
caps: [mon] allow profile osd
caps: [osd] allow *
osd.11
key: AQBX7U5gtj/JlhAAPsLBNG+SfC2eMVEFkI3vfA==
caps: [mgr] allow profile osd
caps: [mon] allow profile osd
caps: [osd] allow *
```

```

osd.9
key: AQBv7U5g1XDULhAAKo2tw6ZhH1jki5aVui2v7g==
caps: [mgr] allow profile osd
caps: [mon] allow profile osd
caps: [osd] allow *
client.admin
key: AQADYEtgFfD3ExAAwH+C1qO7MSLE4TWRfD2g6g==
caps: [mds] allow *
caps: [mgr] allow *
caps: [mon] allow *
caps: [osd] allow *
client.bootstrap-mds
key: AQAHYEtgpbkANBAANqoFlvzEXFwD8oB0w3TF4Q==
caps: [mon] allow profile bootstrap-mds
client.bootstrap-mgr
key: AQAHYEtg3dcANBAAVQf6brq3sxTSrCrPe0pKVQ==
caps: [mon] allow profile bootstrap-mgr
client.bootstrap-osd
key: AQAHYEtgD/QANBAATS9DuP3DbxEI86MTyKEmdw==
caps: [mon] allow profile bootstrap-osd
client.bootstrap-rbd
key: AQAHYEtgjxEbnBAANho25V9tWNNvIKnHknW59A==
caps: [mon] allow profile bootstrap-rbd
client.bootstrap-rbd-mirror
key: AQAHYEtgDE8BNBAAr6rLYxZci0b2holgH9GXYw==
caps: [mon] allow profile bootstrap-rbd-mirror
client.bootstrap-rgw
key: AQAHYEtgwGkBNBAAuRzI4WSrnowBhZxr2XtTFg==
caps: [mon] allow profile bootstrap-rgw
client.crash.host04
key: AQCQYEtgz8lGGhAAy5bJS8VH9fMdxuAZ3CqX5Q==
caps: [mgr] profile crash
caps: [mon] profile crash
client.crash.host02
key: AQDuYUtggqfdOhAAsyX+Mo35M+HFpURGad7nJA==
caps: [mgr] profile crash
caps: [mon] profile crash
client.crash.host03
key: AQB98E5g5jHZAxAAkiWSvmDsh2JaL5G7FvMrrA==
caps: [mgr] profile crash
caps: [mon] profile crash
client.nfs.foo.host03
key: AQCgTk9gm+HvMxAAHbjG+XpdwL6prM/uMcdPdQ==
caps: [mon] allow r
caps: [osd] allow rw pool=nfs-ganesha namespace=foo
client.nfs.foo.host03-rgw
key: AQCgTk9g8sJQNhAAPykcoYUuPc7ljubaFx09HQ==
caps: [mon] allow r
caps: [osd] allow rwx tag rgw *=*
client.rgw.test_realm.test_zone.host01.hgbvng
key: AQD5RE9gAQKdCRAAJzxDwD/dJObbInp9J95sXw==
caps: [mgr] allow rw
caps: [mon] allow *
caps: [osd] allow rwx tag rgw *=*
client.rgw.test_realm.test_zone.host02.yqqilm
key: AQD0RE9gkxA4ExAAFxp3pLJWdlhsyTe2ZR6llw==

```

```

caps: [mgr] allow rw
caps: [mon] allow *
caps: [osd] allow rwx tag rgw *=*
mgr.host01.hdhzwn
key: AQAIEYtg3IhIBxAAMHodolpdxK0IIWF80ItQ==
caps: [mds] allow *
caps: [mon] profile mgr
caps: [osd] allow *
mgr.host02.eobuuv
key: AQA6U5gzUuiABAA2Fed+jPM1xwb4XDYtrQxaQ==
caps: [mds] allow *
caps: [mon] profile mgr
caps: [osd] allow *
mgr.host03.wquwpj
key: AQA6U5glzWsLBAAbOKUKZIUcAVe9kBLfajMKw==
caps: [mds] allow *
caps: [mon] profile mgr
caps: [osd] allow *

```



#### NOTE

The **TYPE.ID** notation for users applies such that **osd.0** is a user of type **osd** and its ID is **0**, **client.admin** is a user of type **client** and its ID is **admin**, that is, the default **client.admin** user. Note also that each entry has a **key: VALUE** entry, and one or more **caps:** entries.

You may use the **-o FILE\_NAME** option with **ceph auth list** to save the output to a file.

### 5.3.3. Display Ceph user information

You can display a Ceph's user information using the command-line interface.

#### Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the node.

#### Procedure

1. To retrieve a specific user, key and capabilities, execute the following:

#### Syntax

```
ceph auth export TYPE.ID
```

#### Example

```
[ceph: root@host01 /]# ceph auth export mgr.host02.eobuuv
```

2. You can also use the **-o FILE\_NAME** option.

#### Syntax

```
ceph auth export TYPE.ID -o FILE_NAME
```

### Example

```
[ceph: root@host01 /]# ceph auth export osd.9 -o filename
export auth(key=AQBV7U5g1XDULhAAKo2tw6ZhH1jki5aVui2v7g==)
```

The **auth export** command is identical to **auth get**, but also prints out the internal **audit**, which isn't relevant to end users.

### 5.3.4. Add a new Ceph user

Adding a user creates a username, that is, **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 ways 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 FILE\_NAME** 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 FILE\_NAME** 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 is useless beyond mere authentication, because the client cannot retrieve the cluster map from the monitor. However, you can create a user with no capabilities if you wish 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 capability on Ceph OSDs. Additionally, a user's OSD permissions are often restricted to accessing a particular pool. :

```
[ceph: root@host01 /]# ceph auth add client.john mon 'allow r' osd 'allow rw pool=mypool'
[ceph: root@host01 /]# ceph auth get-or-create client.paul mon 'allow r' osd 'allow rw pool=mypool'
[ceph: root@host01 /]# ceph auth get-or-create client.george mon 'allow r' osd 'allow rw pool=mypool'
-o george.keyring
[ceph: root@host01 /]# ceph auth get-or-create-key client.ringo mon 'allow r' osd 'allow rw
pool=mypool' -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!

### 5.3.5. Modifying a Ceph User

The **ceph auth caps** command allows you to specify a user and change the user's capabilities.

#### Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the node.

#### Procedure

1. To add capabilities, use the form:

#### Syntax

```
ceph auth caps USERTYPE.USERID DAEMON 'allow [r|w|x|*|...] [pool=POOL_NAME]  
[namespace=NAMESPACE_NAME]
```

#### Example

```
[ceph: root@host01 /]# ceph auth caps client.john mon 'allow r' osd 'allow rw pool=mypool'  
[ceph: root@host01 /]# ceph auth caps client.paul mon 'allow rw' osd 'allow rwx pool=mypool'  
[ceph: root@host01 /]# ceph auth caps client.brian-manager mon 'allow *' osd 'allow *'
```

2. 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:

#### Example

```
[ceph: root@host01 /]# ceph auth caps client.ringo mon '' osd ''
```

#### Additional Resources

- See [Authorization capabilities](#) for additional details on capabilities.

### 5.3.6. Deleting a Ceph user

You can delete a user from the Ceph storage cluster using the command-line interface.

#### Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the node.

#### Procedure

1. To delete a user, use **ceph auth del**:

#### Syntax

```
ceph auth del TYPE.ID
```

-

### Example

```
[ceph: root@host01 /]# ceph auth del osd.6
```

## 5.3.7. Print a Ceph user key

You can display a Ceph user's key information using the command-line interface.

### Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the node.

### Procedure

1. To print a user's authentication key to standard output, execute the following:

#### Syntax

```
ceph auth print-key TYPE.ID
```

#### Example

```
[ceph: root@host01 /]# ceph auth print-key osd.6  
AQBQ7U5gAry3JRAA3NoPrqBBThpFMcRL6Sr+5w==[ceph: root@host01 /]#
```

2. Printing a user's key is useful when you need to populate client software with a user's key, for example, **libvirt**.

#### Syntax

```
mount -t ceph HOSTNAME:/MOUNT_POINT -o name=client.user,secret=ceph auth print-key  
client.user
```

#### Example

```
[ceph: root@host01 /]# mount -t ceph host02:/ceph -o name=client.user,secret=`ceph auth  
print-key client.user`
```

## CHAPTER 6. CEPH PERFORMANCE BENCHMARK

As a storage administrator, you can benchmark performance of the Red Hat Ceph Storage cluster. The purpose of this section is to give Ceph administrators a basic understanding of Ceph’s native benchmarking tools. These tools will provide some insight into how the Ceph storage cluster is performing. This is not the definitive guide to Ceph performance benchmarking, nor is it a guide on how to tune Ceph accordingly.

### 6.1. PREREQUISITES

- A running Red Hat Ceph Storage cluster.

### 6.2. PERFORMANCE BASELINE

The OSD, including the journal, disks and the network throughput should each have a performance baseline to compare against. You can identify potential tuning opportunities by comparing the baseline performance data with the data from Ceph’s native tools. Red Hat Enterprise Linux has many built-in tools, along with a plethora of open source community tools, available to help accomplish these tasks.

#### Additional Resources

- For more details about some of the available tools, see this Knowledgebase [article](#).

### 6.3. BENCHMARKING CEPH PERFORMANCE

Ceph includes the **rados bench** command to do performance benchmarking on a RADOS storage cluster. The command will execute a write test and two types of read tests. The **--no-cleanup** option is important to use when testing both read and write performance. By default the **rados bench** command will delete the objects it has written to the storage pool. Leaving behind these objects allows the two read tests to measure sequential and random read performance.



#### NOTE

Before running these performance tests, drop all the file system caches by running the following:

#### Example

```
[ceph: root@host01 /]# echo 3 | sudo tee /proc/sys/vm/drop_caches && sudo sync
```

#### Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the node.

#### Procedure

1. Create a new storage pool:

#### Example

```
[ceph: root@host01 /]# ceph osd pool create testbench 100 100
```

- Execute a write test for 10 seconds to the newly created storage pool:

### Example

```
[ceph: root@host01 /]# rados bench -p testbench 10 write --no-cleanup
```

Maintaining 16 concurrent writes of 4194304 bytes for up to 10 seconds or 0 objects

Object prefix: benchmark\_data\_cephn1.home.network\_10510

sec	Cur ops	started	finished	avg MB/s	cur MB/s	last lat	avg lat
0	0	0	0	0	-	0	
1	16	16	0	0	-	0	
2	16	16	0	0	-	0	
3	16	16	0	0	-	0	
4	16	17	1	0.998879	1	3.19824	3.19824
5	16	18	2	1.59849	4	4.56163	3.87993
6	16	18	2	1.33222	0	-	3.87993
7	16	19	3	1.71239	2	6.90712	4.889
8	16	25	9	4.49551	24	7.75362	6.71216
9	16	25	9	3.99636	0	-	6.71216
10	16	27	11	4.39632	4	9.65085	7.18999
11	16	27	11	3.99685	0	-	7.18999
12	16	27	11	3.66397	0	-	7.18999
13	16	28	12	3.68975	1.33333	12.8124	7.65853
14	16	28	12	3.42617	0	-	7.65853
15	16	28	12	3.19785	0	-	7.65853
16	11	28	17	4.24726	6.66667	12.5302	9.27548
17	11	28	17	3.99751	0	-	9.27548
18	11	28	17	3.77546	0	-	9.27548
19	11	28	17	3.57683	0	-	9.27548

Total time run: 19.505620

Total writes made: 28

Write size: 4194304

Bandwidth (MB/sec): 5.742

Stddev Bandwidth: 5.4617

Max bandwidth (MB/sec): 24

Min bandwidth (MB/sec): 0

Average Latency: 10.4064

Stddev Latency: 3.80038

Max latency: 19.503

Min latency: 3.19824

- Execute a sequential read test for 10 seconds to the storage pool:

### Example

```
[ceph: root@host01 /]# rados bench -p testbench 10 seq
```

sec	Cur ops	started	finished	avg MB/s	cur MB/s	last lat	avg lat
0	0	0	0	0	-	0	

Total time run: 0.804869

Total reads made: 28

Read size: 4194304

```
Bandwidth (MB/sec): 139.153
```

```
Average Latency: 0.420841
```

```
Max latency: 0.706133
```

```
Min latency: 0.0816332
```

- Execute a random read test for 10 seconds to the storage pool:

### Example

```
[ceph: root@host01 /]# rados bench -p testbench 10 rand
```

```
sec Cur ops  started finished avg MB/s cur MB/s last lat  avg lat
0  0  0  0  0  0  -  0
1  16  46  30  119.801  120 0.440184 0.388125
2  16  81  65  129.408  140 0.577359 0.417461
3  16  120  104  138.175  156 0.597435 0.409318
4  15  157  142  141.485  152 0.683111 0.419964
5  16  206  190  151.553  192 0.310578 0.408343
6  16  253  237  157.608  188 0.0745175 0.387207
7  16  287  271  154.412  136 0.792774 0.39043
8  16  325  309  154.044  152 0.314254 0.39876
9  16  362  346  153.245  148 0.355576 0.406032
10 16  405  389  155.092  172 0.64734 0.398372
Total time run: 10.302229
Total reads made: 405
Read size: 4194304
Bandwidth (MB/sec): 157.248

Average Latency: 0.405976
Max latency: 1.00869
Min latency: 0.0378431
```

- To increase the number of concurrent reads and writes, use the **-t** option, which the default is 16 threads. Also, the **-b** parameter can adjust the size of the object being written. The default object size is 4 MB. A safe maximum object size is 16 MB. Red Hat recommends running multiple copies of these benchmark tests to different pools. Doing this shows the changes in performance from multiple clients.

Add the **--run-name LABEL** option to control the names of the objects that get written during the benchmark test. Multiple **rados bench** commands might be ran simultaneously by changing the **--run-name** label for each running command instance. This prevents potential I/O errors that can occur when multiple clients are trying to access the same object and allows for different clients to access different objects. The **--run-name** option is also useful when trying to simulate a real world workload.

### Example

```
[ceph: root@host01 /]# rados bench -p testbench 10 write -t 4 --run-name client1
```

Maintaining 4 concurrent writes of 4194304 bytes for up to 10 seconds or 0 objects

Object prefix: benchmark\_data\_node1\_12631

```
sec Cur ops  started finished avg MB/s cur MB/s last lat  avg lat
0  0  0  0  0  0  -  0
1  4  4  0  0  0  -  0
2  4  6  2  3.99099  4  1.94755  1.93361
```

```

3  4  8  4  5.32498  8  2.978  2.44034
4  4  8  4  3.99504  0  -  2.44034
5  4  10  6  4.79504  4  2.92419  2.4629
6  3  10  7  4.64471  4  3.02498  2.5432
7  4  12  8  4.55287  4  3.12204  2.61555
8  4  14  10  4.9821  8  2.55901  2.68396
9  4  16  12  5.31621  8  2.68769  2.68081
10 4  17  13  5.18488  4  2.11937  2.63763
11 4  17  13  4.71431  0  -  2.63763
12 4  18  14  4.65486  2  2.4836  2.62662
13 4  18  14  4.29757  0  -  2.62662

Total time run:      13.123548
Total writes made:   18
Write size:          4194304
Bandwidth (MB/sec):  5.486

Stddev Bandwidth:    3.0991
Max bandwidth (MB/sec): 8
Min bandwidth (MB/sec): 0
Average Latency:     2.91578
Stddev Latency:      0.956993
Max latency:         5.72685
Min latency:         1.91967

```

- Remove the data created by the **rados bench** command:

### Example

```
[ceph: root@host01 /]# rados -p testbench cleanup
```

## 6.4. BENCHMARKING CEPH BLOCK PERFORMANCE

Ceph includes the **rbd bench-write** command to test sequential writes to the block device measuring throughput and latency. The default byte size is 4096, the default number of I/O threads is 16, and the default total number of bytes to write is 1 GB. These defaults can be modified by the **--io-size**, **--io-threads** and **--io-total** options respectively.

### Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the node.

### Procedure

- Load the **rbd** kernel module, if not already loaded:

### Example

```
[root@host01 ~]# modprobe rbd
```

- Create a 1 GB **rbd** image file in the **testbench** pool:

### Example

```
[root@host01 ~]# rbd create image01 --size 1024 --pool testbench
```

3. Map the image file to a device file:

#### Example

```
[root@host01 ~]# rbd map image01 --pool testbench --name client.admin
```

4. Create an **ext4** file system on the block device:

#### Example

```
[root@host01 ~]# mkfs.ext4 /dev/rbd/testbench/image01
```

5. Create a new directory:

#### Example

```
[root@host01 ~]# mkdir /mnt/ceph-block-device
```

6. Mount the block device under **/mnt/ceph-block-device/**:

#### Example

```
[root@host01 ~]# mount /dev/rbd/testbench/image01 /mnt/ceph-block-device
```

7. Execute the write performance test against the block device

#### Example

```
[root@host01 ~]# rbd bench --io-type write image01 --pool=testbench

bench-write io_size 4096 io_threads 16 bytes 1073741824 pattern seq
SEC   OPS  OPS/SEC  BYTES/SEC
 2   11127  5479.59 22444382.79
 3   11692  3901.91 15982220.33
 4   12372  2953.34 12096895.42
 5   12580  2300.05 9421008.60
 6   13141  2101.80 8608975.15
 7   13195   356.07 1458459.94
 8   13820   390.35 1598876.60
 9   14124   325.46 1333066.62
..
```

### Additional Resources

- See the [Ceph Block Device Commands](#) section in the *Red Hat Ceph Storage Block Device Guide* for more information on the **rbd** command.

## CHAPTER 7. CEPH PERFORMANCE COUNTERS

As a storage administrator, you can gather performance metrics of the Red Hat Ceph Storage cluster. The Ceph performance counters are a collection of internal infrastructure metrics. The collection, aggregation, and graphing of this metric data can be done by an assortment of tools and can be useful for performance analytics.

### 7.1. PREREQUISITES

- A running Red Hat Ceph Storage cluster.

### 7.2. ACCESS TO CEPH PERFORMANCE COUNTERS

The performance counters are available through a socket interface for the Ceph Monitors and the OSDs. The socket file for each respective daemon is located under `/var/run/ceph`, by default. The performance counters are grouped together into collection names. These collection names represent a subsystem or an instance of a subsystem.

Here is the full list of the Monitor and the OSD collection name categories with a brief description for each :

#### Monitor Collection Name Categories

- Cluster Metrics - Displays information about the storage cluster: Monitors, OSDs, Pools, and PGs
- Level Database Metrics - Displays information about the back-end **KeyValueStore** database
- Monitor Metrics - Displays general monitor information
- Paxos Metrics - Displays information on cluster quorum management
- Throttle Metrics - Displays the statistics on how the monitor is throttling

#### OSD Collection Name Categories

- Write Back Throttle Metrics - Displays the statistics on how the write back throttle is tracking unflushed IO
- Level Database Metrics - Displays information about the back-end **KeyValueStore** database
- Objecter Metrics - Displays information on various object-based operations
- Read and Write Operations Metrics - Displays information on various read and write operations
- Recovery State Metrics - Displays - Displays latencies on various recovery states
- OSD Throttle Metrics - Display the statistics on how the OSD is throttling

#### RADOS Gateway Collection Name Categories

- Object Gateway Client Metrics - Displays statistics on GET and PUT requests
- Objecter Metrics - Displays information on various object-based operations

- Object Gateway Throttle Metrics - Display the statistics on how the OSD is throttling

## 7.3. DISPLAY THE CEPH PERFORMANCE COUNTERS

The **ceph daemon *DAEMON\_NAME* perf schema** command outputs the available metrics. Each metric has an associated bit field value type.

### Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the node.

### Procedure

1. To view the metric's schema:

#### Syntax

```
ceph daemon DAEMON_NAME perf schema
```



#### NOTE

You must run the **ceph daemon** command from the node running the daemon.

2. Executing **ceph daemon *DAEMON\_NAME* perf schema** command from the monitor node:

#### Example

```
[ceph: root@host01 /]# ceph daemon mon.host01 perf schema
```

3. Executing the **ceph daemon *DAEMON\_NAME* perf schema** command from the OSD node:

#### Example

```
[ceph: root@host01 /]# ceph daemon osd.11 perf schema
```

Table 7.1. The bit field value definitions

Bit	Meaning
1	Floating point value
2	Unsigned 64-bit integer value
4	Average (Sum + Count)
8	Counter

Each value will have bit 1 or 2 set to indicate the type, either a floating point or an integer value. When bit

4 is set, there will be two values to read, a sum and a count. When bit 8 is set, the average for the previous interval would be the sum delta, since the previous read, divided by the count delta. Alternatively, dividing the values outright would provide the lifetime average value. Typically these are used to measure latencies, the number of requests and a sum of request latencies. Some bit values are combined, for example 5, 6 and 10. A bit value of 5 is a combination of bit 1 and bit 4. This means the average will be a floating point value. A bit value of 6 is a combination of bit 2 and bit 4. This means the average value will be an integer. A bit value of 10 is a combination of bit 2 and bit 8. This means the counter value will be an integer value.

### Additional Resources

- See [Average count and sum](#) section in the Red Hat Ceph Storage Administration Guide\_ for more details.

## 7.4. DUMP THE CEPH PERFORMANCE COUNTERS

The **ceph daemon .. perf dump** command outputs the current values and groups the metrics under the collection name for each subsystem.

### Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the node.

### Procedure

1. To view the current metric data:

#### Syntax

```
ceph daemon DAEMON_NAME perf dump
```



#### NOTE

You must run the **ceph daemon** command from the node running the daemon.

2. Executing **ceph daemon .. perf dump** command from the Monitor node:

```
[ceph: root@host01 /]# ceph daemon mon.host01 perf dump
```

3. Executing the **ceph daemon .. perf dump** command from the OSD node:

```
[ceph: root@host01 /]# ceph daemon osd.11 perf dump
```

### Additional Resources

- To view a short description of each Monitor metric available, please see the [Ceph monitor metrics table](#).

## 7.5. AVERAGE COUNT AND SUM

All latency numbers have a bit field value of 5. This field contains floating point values for the average count and sum. The **avgcount** is the number of operations within this range and the **sum** is the total latency in seconds. When dividing the **sum** by the **avgcount** this will provide you with an idea of the latency per operation.

### Additional Resources

- To view a short description of each OSD metric available, please see the [Ceph OSD table](#).

## 7.6. CEPH MONITOR METRICS

Table 7.2. Cluster Metrics Table

Collection Name	Metric Name	Bit Field Value	Short Description
<b>cluster</b>	<b>num_mon</b>	2	Number of monitors
	<b>num_mon_quorum</b>	2	Number of monitors in quorum
	<b>num_osd</b>	2	Total number of OSD
	<b>num_osd_up</b>	2	Number of OSDs that are up
	<b>num_osd_in</b>	2	Number of OSDs that are in cluster
	<b>osd_epoch</b>	2	Current epoch of OSD map
	<b>osd_bytes</b>	2	Total capacity of cluster in bytes
	<b>osd_bytes_used</b>	2	Number of used bytes on cluster
	<b>osd_bytes_avail</b>	2	Number of available bytes on cluster
	<b>num_pool</b>	2	Number of pools
	<b>num_pg</b>	2	Total number of placement groups
	<b>num_pg_active_clean</b>	2	Number of placement groups in active+clean state
	<b>num_pg_active</b>	2	Number of placement groups in active state

Collection Name	Metric Name	Bit Field Value	Short Description
	<b>num_pg_peering</b>	2	Number of placement groups in peering state
	<b>num_object</b>	2	Total number of objects on cluster
	<b>num_object_degraded</b>	2	Number of degraded (missing replicas) objects
	<b>num_object_misplaced</b>	2	Number of misplaced (wrong location in the cluster) objects
	<b>num_object_unfound</b>	2	Number of unfound objects
	<b>num_bytes</b>	2	Total number of bytes of all objects
	<b>num_mds_up</b>	2	Number of MDSs that are up
	<b>num_mds_in</b>	2	Number of MDS that are in cluster
	<b>num_mds_failed</b>	2	Number of failed MDS
	<b>mds_epoch</b>	2	Current epoch of MDS map

Table 7.3. Level Database Metrics Table

Collection Name	Metric Name	Bit Field Value	Short Description
<b>leveldb</b>	<b>leveldb_get</b>	10	Gets
	<b>leveldb_transaction</b>	10	Transactions
	<b>leveldb_compact</b>	10	Compactions
	<b>leveldb_compact_range</b>	10	Compactions by range
	<b>leveldb_compact_queue_merge</b>	10	Mergings of ranges in compaction queue
	<b>leveldb_compact_queue_len</b>	2	Length of compaction queue

Table 7.4. General Monitor Metrics Table

Collection Name	Metric Name	Bit Field Value	Short Description
<b>mon</b>	<b>num_sessions</b>	2	Current number of opened monitor sessions
	<b>session_add</b>	10	Number of created monitor sessions
	<b>session_rm</b>	10	Number of remove_session calls in monitor
	<b>session_trim</b>	10	Number of trimmed monitor sessions
	<b>num_elections</b>	10	Number of elections monitor took part in
	<b>election_call</b>	10	Number of elections started by monitor
	<b>election_win</b>	10	Number of elections won by monitor
	<b>election_lose</b>	10	Number of elections lost by monitor

Table 7.5. Paxos Metrics Table

Collection Name	Metric Name	Bit Field Value	Short Description
<b>paxos</b>	<b>start_leader</b>	10	Starts in leader role
	<b>start_peon</b>	10	Starts in peon role
	<b>restart</b>	10	Restarts
	<b>refresh</b>	10	Refreshes
	<b>refresh_latency</b>	5	Refresh latency
	<b>begin</b>	10	Started and handled begins
	<b>begin_keys</b>	6	Keys in transaction on begin
	<b>begin_bytes</b>	6	Data in transaction on begin

Collection Name	Metric Name	Bit Field Value	Short Description
	<b>begin_latency</b>	5	Latency of begin operation
	<b>commit</b>	10	Commits
	<b>commit_keys</b>	6	Keys in transaction on commit
	<b>commit_bytes</b>	6	Data in transaction on commit
	<b>commit_latency</b>	5	Commit latency
	<b>collect</b>	10	Peon collects
	<b>collect_keys</b>	6	Keys in transaction on peon collect
	<b>collect_bytes</b>	6	Data in transaction on peon collect
	<b>collect_latency</b>	5	Peon collect latency
	<b>collect_uncommitted</b>	10	Uncommitted values in started and handled collects
	<b>collect_timeout</b>	10	Collect timeouts
	<b>accept_timeout</b>	10	Accept timeouts
	<b>lease_ack_timeout</b>	10	Lease acknowledgement timeouts
	<b>lease_timeout</b>	10	Lease timeouts
	<b>store_state</b>	10	Store a shared state on disk
	<b>store_state_keys</b>	6	Keys in transaction in stored state
	<b>store_state_bytes</b>	6	Data in transaction in stored state
	<b>store_state_latency</b>	5	Storing state latency
	<b>share_state</b>	10	Sharings of state
	<b>share_state_keys</b>	6	Keys in shared state

Collection Name	Metric Name	Bit Field Value	Short Description
	<b>share_state_bytes</b>	6	Data in shared state
	<b>new_pn</b>	10	New proposal number queries
	<b>new_pn_latency</b>	5	New proposal number getting latency

Table 7.6. Throttle Metrics Table

Collection Name	Metric Name	Bit Field Value	Short Description
<b>throttle-*</b>	<b>val</b>	10	Currently available throttle
	<b>max</b>	10	Max value for throttle
	<b>get</b>	10	Gets
	<b>get_sum</b>	10	Got data
	<b>get_or_fail_fail</b>	10	Get blocked during get_or_fail
	<b>get_or_fail_success</b>	10	Successful get during get_or_fail
	<b>take</b>	10	Takes
	<b>take_sum</b>	10	Taken data
	<b>put</b>	10	Puts
	<b>put_sum</b>	10	Put data
	<b>wait</b>	5	Waiting latency

#### Additional Resources

- [Cluster Metrics Table](#)
- [Level Database Metrics Table](#)
- [General Monitor Metrics Table](#)
- [Paxos Metrics Table](#)
- [Throttle Metrics Table](#)

## 7.7. CEPH OSD METRICS

Table 7.7. Write Back Throttle Metrics Table

Collection Name	Metric Name	Bit Field Value	Short Description
<b>WBThrottle</b>	<b>bytes_dirtied</b>	2	Dirty data
	<b>bytes_wb</b>	2	Written data
	<b>ios_dirtied</b>	2	Dirty operations
	<b>ios_wb</b>	2	Written operations
	<b>inodes_dirtied</b>	2	Entries waiting for write
	<b>inodes_wb</b>	2	Written entries

Table 7.8. Level Database Metrics Table

Collection Name	Metric Name	Bit Field Value	Short Description
<b>leveldb</b>	<b>leveldb_get</b>	10	Gets
	<b>leveldb_transaction</b>	10	Transactions
	<b>leveldb_compact</b>	10	Compactions
	<b>leveldb_compact_range</b>	10	Compactions by range
	<b>leveldb_compact_queue_merge</b>	10	Mergings of ranges in compaction queue
	<b>leveldb_compact_queue_len</b>	2	Length of compaction queue

Table 7.9. Objecter Metrics Table

Collection Name	Metric Name	Bit Field Value	Short Description
<b>objecter</b>	<b>op_active</b>	2	Active operations
	<b>op_laggy</b>	2	Laggy operations
	<b>op_send</b>	10	Sent operations

Collection Name	Metric Name	Bit Field Value	Short Description
	<b>op_send_bytes</b>	10	Sent data
	<b>op_resend</b>	10	Resent operations
	<b>op_ack</b>	10	Commit callbacks
	<b>op_commit</b>	10	Operation commits
	<b>op</b>	10	Operation
	<b>op_r</b>	10	Read operations
	<b>op_w</b>	10	Write operations
	<b>op_rmw</b>	10	Read-modify-write operations
	<b>op_pg</b>	10	PG operation
	<b>osdop_stat</b>	10	Stat operations
	<b>osdop_create</b>	10	Create object operations
	<b>osdop_read</b>	10	Read operations
	<b>osdop_write</b>	10	Write operations
	<b>osdop_writefull</b>	10	Write full object operations
	<b>osdop_append</b>	10	Append operation
	<b>osdop_zero</b>	10	Set object to zero operations
	<b>osdop_truncate</b>	10	Truncate object operations
	<b>osdop_delete</b>	10	Delete object operations
	<b>osdop_mapext</b>	10	Map extent operations
	<b>osdop_sparse_read</b>	10	Sparse read operations
	<b>osdop_clonerange</b>	10	Clone range operations
	<b>osdop_getxattr</b>	10	Get xattr operations
	<b>osdop_setxattr</b>	10	Set xattr operations

Collection Name	Metric Name	Bit Field Value	Short Description
	<b>osdop_cmpxattr</b>	10	Xattr comparison operations
	<b>osdop_rmxattr</b>	10	Remove xattr operations
	<b>osdop_resetxattrs</b>	10	Reset xattr operations
	<b>osdop_tmap_up</b>	10	TMAP update operations
	<b>osdop_tmap_put</b>	10	TMAP put operations
	<b>osdop_tmap_get</b>	10	TMAP get operations
	<b>osdop_call</b>	10	Call (execute) operations
	<b>osdop_watch</b>	10	Watch by object operations
	<b>osdop_notify</b>	10	Notify about object operations
	<b>osdop_src_cmpxattr</b>	10	Extended attribute comparison in multi operations
	<b>osdop_other</b>	10	Other operations
	<b>linger_active</b>	2	Active lingering operations
	<b>linger_send</b>	10	Sent lingering operations
	<b>linger_resend</b>	10	Resent lingering operations
	<b>linger_ping</b>	10	Sent pings to lingering operations
	<b>poolop_active</b>	2	Active pool operations
	<b>poolop_send</b>	10	Sent pool operations
	<b>poolop_resend</b>	10	Resent pool operations
	<b>poolstat_active</b>	2	Active get pool stat operations
	<b>poolstat_send</b>	10	Pool stat operations sent
	<b>poolstat_resend</b>	10	Resent pool stats
	<b>statfs_active</b>	2	Statfs operations

Collection Name	Metric Name	Bit Field Value	Short Description
	<b>statsfs_send</b>	10	Sent FS stats
	<b>statsfs_resend</b>	10	Resent FS stats
	<b>command_active</b>	2	Active commands
	<b>command_send</b>	10	Sent commands
	<b>command_resend</b>	10	Resent commands
	<b>map_epoch</b>	2	OSD map epoch
	<b>map_full</b>	10	Full OSD maps received
	<b>map_inc</b>	10	Incremental OSD maps received
	<b>osd_sessions</b>	2	Open sessions
	<b>osd_session_open</b>	10	Sessions opened
	<b>osd_session_close</b>	10	Sessions closed
	<b>osd_laggy</b>	2	Laggy OSD sessions

Table 7.10. Read and Write Operations Metrics Table

Collection Name	Metric Name	Bit Field Value	Short Description
<b>osd</b>	<b>op_wip</b>	2	Replication operations currently being processed (primary)
	<b>op_in_bytes</b>	10	Client operations total write size
	<b>op_out_bytes</b>	10	Client operations total read size
	<b>op_latency</b>	5	Latency of client operations (including queue time)
	<b>op_process_latency</b>	5	Latency of client operations (excluding queue time)

Collection Name	Metric Name	Bit Field Value	Short Description
	<b>op_r</b>	10	Client read operations
	<b>op_r_out_bytes</b>	10	Client data read
	<b>op_r_latency</b>	5	Latency of read operation (including queue time)
	<b>op_r_process_latency</b>	5	Latency of read operation (excluding queue time)
	<b>op_w</b>	10	Client write operations
	<b>op_w_in_bytes</b>	10	Client data written
	<b>op_w_rlat</b>	5	Client write operation readable/applied latency
	<b>op_w_latency</b>	5	Latency of write operation (including queue time)
	<b>op_w_process_latency</b>	5	Latency of write operation (excluding queue time)
	<b>op_rw</b>	10	Client read-modify-write operations
	<b>op_rw_in_bytes</b>	10	Client read-modify-write operations write in
	<b>op_rw_out_bytes</b>	10	Client read-modify-write operations read out
	<b>op_rw_rlat</b>	5	Client read-modify-write operation readable/applied latency
	<b>op_rw_latency</b>	5	Latency of read-modify-write operation (including queue time)
	<b>op_rw_process_latency</b>	5	Latency of read-modify-write operation (excluding queue time)
	<b>subop</b>	10	Suboperations
	<b>subop_in_bytes</b>	10	Suboperations total size

Collection Name	Metric Name	Bit Field Value	Short Description
	<b>subop_latency</b>	5	Suboperations latency
	<b>subop_w</b>	10	Replicated writes
	<b>subop_w_in_bytes</b>	10	Replicated written data size
	<b>subop_w_latency</b>	5	Replicated writes latency
	<b>subop_pull</b>	10	Suboperations pull requests
	<b>subop_pull_latency</b>	5	Suboperations pull latency
	<b>subop_push</b>	10	Suboperations push messages
	<b>subop_push_in_bytes</b>	10	Suboperations pushed size
	<b>subop_push_latency</b>	5	Suboperations push latency
	<b>pull</b>	10	Pull requests sent
	<b>push</b>	10	Push messages sent
	<b>push_out_bytes</b>	10	Pushed size
	<b>push_in</b>	10	Inbound push messages
	<b>push_in_bytes</b>	10	Inbound pushed size
	<b>recovery_ops</b>	10	Started recovery operations
	<b>loadavg</b>	2	CPU load
	<b>buffer_bytes</b>	2	Total allocated buffer size
	<b>numpg</b>	2	Placement groups
	<b>numpg_primary</b>	2	Placement groups for which this osd is primary
	<b>numpg_replica</b>	2	Placement groups for which this osd is replica
	<b>numpg_stray</b>	2	Placement groups ready to be deleted from this osd

Collection Name	Metric Name	Bit Field Value	Short Description
	<b>heartbeat_to_peers</b>	2	Heartbeat (ping) peers we send to
	<b>heartbeat_from_peers</b>	2	Heartbeat (ping) peers we recv from
	<b>map_messages</b>	10	OSD map messages
	<b>map_message_epochs</b>	10	OSD map epochs
	<b>map_message_epoch_dups</b>	10	OSD map duplicates
	<b>stat_bytes</b>	2	OSD size
	<b>stat_bytes_used</b>	2	Used space
	<b>stat_bytes_avail</b>	2	Available space
	<b>copyfrom</b>	10	Rados 'copy-from' operations
	<b>tier_promote</b>	10	Tier promotions
	<b>tier_flush</b>	10	Tier flushes
	<b>tier_flush_fail</b>	10	Failed tier flushes
	<b>tier_try_flush</b>	10	Tier flush attempts
	<b>tier_try_flush_fail</b>	10	Failed tier flush attempts
	<b>tier_evict</b>	10	Tier evictions
	<b>tier_whiteout</b>	10	Tier whiteouts
	<b>tier_dirty</b>	10	Dirty tier flag set
	<b>tier_clean</b>	10	Dirty tier flag cleaned
	<b>tier_delay</b>	10	Tier delays (agent waiting)
	<b>tier_proxy_read</b>	10	Tier proxy reads
	<b>agent_wake</b>	10	Tiering agent wake up

Collection Name	Metric Name	Bit Field Value	Short Description
	<b>agent_skip</b>	10	Objects skipped by agent
	<b>agent_flush</b>	10	Tiering agent flushes
	<b>agent_evict</b>	10	Tiering agent evictions
	<b>object_ctx_cache_hit</b>	10	Object context cache hits
	<b>object_ctx_cache_total</b>	10	Object context cache lookups

Table 7.11. Recovery State Metrics Table

Collection Name	Metric Name	Bit Field Value	Short Description
<b>recoverystate_perf</b>	<b>initial_latency</b>	5	Initial recovery state latency
	<b>started_latency</b>	5	Started recovery state latency
	<b>reset_latency</b>	5	Reset recovery state latency
	<b>start_latency</b>	5	Start recovery state latency
	<b>primary_latency</b>	5	Primary recovery state latency
	<b>peering_latency</b>	5	Peering recovery state latency
	<b>backfilling_latency</b>	5	Backfilling recovery state latency
	<b>waitremotebackfillreserved_latency</b>	5	Wait remote backfill reserved recovery state latency
	<b>waitlocalbackfillreserved_latency</b>	5	Wait local backfill reserved recovery state latency
	<b>notbackfilling_latency</b>	5	Notbackfilling recovery state latency
	<b>repnotrecovering_latency</b>	5	Repnotrecovering recovery state latency
	<b>repwaitrecoveryreserved_latency</b>	5	Rep wait recovery reserved recovery state latency

Collection Name	Metric Name	Bit Field Value	Short Description
	<b>repwaitbackfillreserved_latency</b>	5	Rep wait backfill reserved recovery state latency
	<b>RepRecovering_latency</b>	5	RepRecovering recovery state latency
	<b>activating_latency</b>	5	Activating recovery state latency
	<b>waitlocalrecoveryreserved_latency</b>	5	Wait local recovery reserved recovery state latency
	<b>waitremoterecoveryreserved_latency</b>	5	Wait remote recovery reserved recovery state latency
	<b>recovering_latency</b>	5	Recovering recovery state latency
	<b>recovered_latency</b>	5	Recovered recovery state latency
	<b>clean_latency</b>	5	Clean recovery state latency
	<b>active_latency</b>	5	Active recovery state latency
	<b>replicaactive_latency</b>	5	Replicaactive recovery state latency
	<b>stray_latency</b>	5	Stray recovery state latency
	<b>getinfo_latency</b>	5	Getinfo recovery state latency
	<b>getlog_latency</b>	5	Getlog recovery state latency
	<b>waitactingchange_latency</b>	5	Waitactingchange recovery state latency
	<b>incomplete_latency</b>	5	Incomplete recovery state latency
	<b>getmissing_latency</b>	5	Getmissing recovery state latency
	<b>waitupthru_latency</b>	5	Waitupthru recovery state latency

Table 7.12. OSD Throttle Metrics Table

Collection Name	Metric Name	Bit Field Value	Short Description
<b>throttle-*</b>	<b>val</b>	10	Currently available throttle
	<b>max</b>	10	Max value for throttle
	<b>get</b>	10	Gets
	<b>get_sum</b>	10	Got data
	<b>get_or_fail_fail</b>	10	Get blocked during get_or_fail
	<b>get_or_fail_success</b>	10	Successful get during get_or_fail
	<b>take</b>	10	Takes
	<b>take_sum</b>	10	Taken data
	<b>put</b>	10	Puts
	<b>put_sum</b>	10	Put data
	<b>wait</b>	5	Waiting latency

#### Additional Resources

- [Write Back Throttle Metrics Table](#)
- [Level Database Metrics Table](#)
- [Objecter Metrics Table](#)
- [Read and Write Operations Metrics Table](#)
- [Recovery State Metrics Table](#)
- [OSD Throttle Metrics Table](#)

## 7.8. CEPH OBJECT GATEWAY METRICS

Table 7.13. RADOS Client Metrics Table

Collection Name	Metric Name	Bit Field Value	Short Description
<b>client.rgw. &lt;rgw_node_name&gt;</b>	<b>req</b>	10	Requests

Collection Name	Metric Name	Bit Field Value	Short Description
	<b>failed_req</b>	10	Aborted requests
	<b>get</b>	10	Gets
	<b>get_b</b>	10	Size of gets
	<b>get_initial_lat</b>	5	Get latency
	<b>put</b>	10	Puts
	<b>put_b</b>	10	Size of puts
	<b>put_initial_lat</b>	5	Put latency
	<b>qlen</b>	2	Queue length
	<b>qactive</b>	2	Active requests queue
	<b>cache_hit</b>	10	Cache hits
	<b>cache_miss</b>	10	Cache miss
	<b>keystone_token_cache_hit</b>	10	Keystone token cache hits
	<b>keystone_token_cache_miss</b>	10	Keystone token cache miss

Table 7.14. Objecter Metrics Table

Collection Name	Metric Name	Bit Field Value	Short Description
<b>objecter</b>	<b>op_active</b>	2	Active operations
	<b>op_laggy</b>	2	Laggy operations
	<b>op_send</b>	10	Sent operations
	<b>op_send_bytes</b>	10	Sent data
	<b>op_resend</b>	10	Resent operations
	<b>op_ack</b>	10	Commit callbacks
	<b>op_commit</b>	10	Operation commits

Collection Name	Metric Name	Bit Field Value	Short Description
	<b>op</b>	10	Operation
	<b>op_r</b>	10	Read operations
	<b>op_w</b>	10	Write operations
	<b>op_rmw</b>	10	Read-modify-write operations
	<b>op_pg</b>	10	PG operation
	<b>osdop_stat</b>	10	Stat operations
	<b>osdop_create</b>	10	Create object operations
	<b>osdop_read</b>	10	Read operations
	<b>osdop_write</b>	10	Write operations
	<b>osdop_writefull</b>	10	Write full object operations
	<b>osdop_append</b>	10	Append operation
	<b>osdop_zero</b>	10	Set object to zero operations
	<b>osdop_truncate</b>	10	Truncate object operations
	<b>osdop_delete</b>	10	Delete object operations
	<b>osdop_mapext</b>	10	Map extent operations
	<b>osdop_sparse_read</b>	10	Sparse read operations
	<b>osdop_clonerange</b>	10	Clone range operations
	<b>osdop_getxattr</b>	10	Get xattr operations
	<b>osdop_setxattr</b>	10	Set xattr operations
	<b>osdop_cmpxattr</b>	10	Xattr comparison operations
	<b>osdop_rmxattr</b>	10	Remove xattr operations
	<b>osdop_resetxattrs</b>	10	Reset xattr operations
	<b>osdop_tmap_up</b>	10	TMAP update operations

Collection Name	Metric Name	Bit Field Value	Short Description
	<b>osdop_tmap_put</b>	10	TMAP put operations
	<b>osdop_tmap_get</b>	10	TMAP get operations
	<b>osdop_call</b>	10	Call (execute) operations
	<b>osdop_watch</b>	10	Watch by object operations
	<b>osdop_notify</b>	10	Notify about object operations
	<b>osdop_src_cmpxattr</b>	10	Extended attribute comparison in multi operations
	<b>osdop_other</b>	10	Other operations
	<b>linger_active</b>	2	Active lingering operations
	<b>linger_send</b>	10	Sent lingering operations
	<b>linger_resend</b>	10	Resent lingering operations
	<b>linger_ping</b>	10	Sent pings to lingering operations
	<b>poolop_active</b>	2	Active pool operations
	<b>poolop_send</b>	10	Sent pool operations
	<b>poolop_resend</b>	10	Resent pool operations
	<b>poolstat_active</b>	2	Active get pool stat operations
	<b>poolstat_send</b>	10	Pool stat operations sent
	<b>poolstat_resend</b>	10	Resent pool stats
	<b>statfs_active</b>	2	Statfs operations
	<b>statfs_send</b>	10	Sent FS stats
	<b>statfs_resend</b>	10	Resent FS stats
	<b>command_active</b>	2	Active commands
	<b>command_send</b>	10	Sent commands

Collection Name	Metric Name	Bit Field Value	Short Description
	<b>command_resend</b>	10	Resent commands
	<b>map_epoch</b>	2	OSD map epoch
	<b>map_full</b>	10	Full OSD maps received
	<b>map_inc</b>	10	Incremental OSD maps received
	<b>osd_sessions</b>	2	Open sessions
	<b>osd_session_open</b>	10	Sessions opened
	<b>osd_session_close</b>	10	Sessions closed
	<b>osd_laggy</b>	2	Laggy OSD sessions

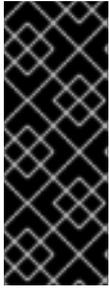
Table 7.15. RADOS Gateway Throttle Metrics Table

Collection Name	Metric Name	Bit Field Value	Short Description
<b>throttle-*</b>	<b>val</b>	10	Currently available throttle
	<b>max</b>	10	Max value for throttle
	<b>get</b>	10	Gets
	<b>get_sum</b>	10	Got data
	<b>get_or_fail_fail</b>	10	Get blocked during get_or_fail
	<b>get_or_fail_success</b>	10	Successful get during get_or_fail
	<b>take</b>	10	Takes
	<b>take_sum</b>	10	Taken data
	<b>put</b>	10	Puts
	<b>put_sum</b>	10	Put data
	<b>wait</b>	5	Waiting latency

- [RADOS Gateway Client Table](#)
- [Objecter Metrics Table](#)
- [RADOS Gateway Throttle Metrics Table](#)

## CHAPTER 8. BLUESTORE

BlueStore is the back-end object store for the OSD daemons and puts objects directly on the block device.



### IMPORTANT

BlueStore provides a high-performance backend for OSD daemons in a production environment. By default, BlueStore is configured to be self-tuning. If you determine that your environment performs better with BlueStore tuned manually, please contact [Red Hat support](#) and share the details of your configuration to help us improve the auto-tuning capability. Red Hat looks forward to your feedback and appreciates your recommendations.

## 8.1. CEPH BLUESTORE

The following are some of the main features of using BlueStore:

### Direct management of storage devices

BlueStore consumes raw block devices or partitions. This avoids any intervening layers of abstraction, such as local file systems like XFS, that might limit performance or add complexity.

### Metadata management with RocksDB

BlueStore uses the RocksDB' key-value database to manage internal metadata, such as the mapping from object names to block locations on a disk.

### Full data and metadata checksumming

By default all data and metadata written to BlueStore is protected by one or more checksums. No data or metadata are read from disk or returned to the user without verification.

### Efficient copy-on-write

The Ceph Block Device and Ceph File System snapshots rely on a copy-on-write clone mechanism that is implemented efficiently in BlueStore. This results in efficient I/O both for regular snapshots and for erasure coded pools which rely on cloning to implement efficient two-phase commits.

### No large double-writes

BlueStore first writes any new data to unallocated space on a block device, and then commits a RocksDB transaction that updates the object metadata to reference the new region of the disk. Only when the write operation is below a configurable size threshold, it falls back to a write-ahead journaling scheme.

### Multi-device support

BlueStore can use multiple block devices for storing different data. For example: Hard Disk Drive (HDD) for the data, Solid-state Drive (SSD) for metadata, Non-volatile Memory (NVM) or Non-volatile random-access memory (NVRAM) or persistent memory for the RocksDB write-ahead log (WAL). See [Ceph BlueStore devices](#) for details.

### Efficient block device usage

Because BlueStore does not use any file system, it minimizes the need to clear the storage device cache.

## 8.2. CEPH BLUESTORE DEVICES

This section explains what block devices the BlueStore back end uses.

BlueStore manages either one, two, or three storage devices.

- Primary
- WAL
- DB

In the simplest case, BlueStore consumes a single (primary) storage device. The storage device is partitioned into two parts that contain:

- **OSD metadata:** A small partition formatted with XFS that contains basic metadata for the OSD. This data directory includes information about the OSD, such as its identifier, which cluster it belongs to, and its private keyring.
- **Data:** A large partition occupying the rest of the device that is managed directly by BlueStore and that contains all of the OSD data. This primary device is identified by a block symbolic link in the data directory.

You can also use two additional devices:

- **A WAL (write-ahead-log) device:** A device that stores BlueStore internal journal or write-ahead log. It is identified by the **block.wal** symbolic link in the data directory. Consider using a WAL device only if the device is faster than the primary device. For example, when the WAL device uses an SSD disk and the primary devices uses an HDD disk.
- **A DB device:** A device that stores BlueStore internal metadata. The embedded RocksDB database puts as much metadata as it can on the DB device instead of on the primary device to improve performance. If the DB device is full, it starts adding metadata to the primary device. Consider using a DB device only if the device is faster than the primary device.



#### WARNING

If you have only a less than a gigabyte storage available on fast devices. Red Hat recommends using it as a WAL device. If you have more fast devices available, consider using it as a DB device. The BlueStore journal is always placed on the fastest device, so using a DB device provides the same benefit that the WAL device while also allows for storing additional metadata.

## 8.3. CEPH BLUESTORE CACHING

The BlueStore cache is a collection of buffers that, depending on configuration, can be populated with data as the OSD daemon does reading from or writing to the disk. By default in Red Hat Ceph Storage, BlueStore will cache on reads, but not writes. This is because the **bluestore\_default\_buffered\_write** option is set to **false** to avoid potential overhead associated with cache eviction.

If the **bluestore\_default\_buffered\_write** option is set to **true**, data is written to the buffer first, and then committed to disk. Afterwards, a write acknowledgement is sent to the client, allowing subsequent reads faster access to the data already in cache, until that data is evicted.

Read-heavy workloads will not see an immediate benefit from BlueStore caching. As more reading is done, the cache will grow over time and subsequent reads will see an improvement in performance. How fast the cache populates depends on the BlueStore block and database disk type, and the client's

workload requirements.



### IMPORTANT

Please contact [Red Hat support](#) before enabling the `bluestore_default_buffered_write` option.

## 8.4. SIZING CONSIDERATIONS FOR CEPH BLUESTORE

When mixing traditional and solid state drives using BlueStore OSDs, it is important to size the RocksDB logical volume (**block.db**) appropriately. Red Hat recommends that the RocksDB logical volume be no less than 4% of the block size with object, file and mixed workloads. Red Hat supports 1% of the BlueStore block size with RocksDB and OpenStack block workloads. For example, if the block size is 1 TB for an object workload, then at a minimum, create a 40 GB RocksDB logical volume.

When not mixing drive types, there is no requirement to have a separate RocksDB logical volume. BlueStore will automatically manage the sizing of RocksDB.

BlueStore's cache memory is used for the key-value pair metadata for RocksDB, BlueStore metadata and object data.



### NOTE

The BlueStore cache memory values are in addition to the memory footprint already being consumed by the OSD.

## 8.5. TUNING CEPH BLUESTORE USING `BLUESTORE_MIN_ALLOC_SIZE` PARAMETER

In BlueStore, the raw partition is allocated and managed in chunks of **bluestore\_min\_alloc\_size**. By default, **bluestore\_min\_alloc\_size** is **4096**, equivalent to 4 KiB for HDDs and SSDs. The unwritten area in each chunk is filled with zeroes when it is written to the raw partition. This can lead to wasted unused space when not properly sized for your workload, for example when writing small objects.

It is best practice to set **bluestore\_min\_alloc\_size** to match the smallest write so this can write amplification penalty can be avoided.



### IMPORTANT

Changing the value of **bluestore\_min\_alloc\_size** is not recommended. For any assistance, contact [Red Hat support](#).



### NOTE

The settings **bluestore\_min\_alloc\_size\_ssd** and **bluestore\_min\_alloc\_size\_hdd** are specific to SSDs and HDDs, respectively, but setting them is not necessary because setting **bluestore\_min\_alloc\_size** overrides them.

### Prerequisites

- A running Red Hat Ceph Storage cluster.
- Ceph monitors and managers are deployed in the cluster.

- Servers or nodes that can be freshly provisioned as OSD nodes
- The admin keyring for the Ceph Monitor node, if you are redeploying an existing Ceph OSD node.

### Procedure

1. On the bootstrapped node, change the value of **bluestore\_min\_alloc\_size** parameter:

#### Syntax

```
ceph config set osd.OSD_ID bluestore_min_alloc_size_DEVICE_NAME_ VALUE
```

#### Example

```
[ceph: root@host01 /]# ceph config set osd.4 bluestore_min_alloc_size_hdd 6144
```

You can see **bluestore\_min\_alloc\_size** is set to 6144 bytes, which is equivalent to 6 KiB.

2. Restart the OSD's service.

#### Syntax

```
systemctl restart SERVICE_ID
```

#### Example

```
[ceph: root@host01 /]# systemctl restart ceph-499829b4-832f-11eb-8d6d-001a4a000635@osd.4.service
```

### Verification

- Verify the setting using the **ceph daemon** command:

#### Syntax

```
ceph daemon osd.OSD_ID config get bluestore_min_alloc_size_DEVICE_
```

#### Example

```
[ceph: root@host01 /]# ceph daemon osd.4 config get bluestore_min_alloc_size_hdd  
  
ceph daemon osd.4 config get bluestore_min_alloc_size  
{  
  "bluestore_min_alloc_size": "6144"  
}
```

## 8.6. RESHARDING THE ROCKSDB DATABASE USING THE BLUESTORE ADMIN TOOL

You can reshard the database with the BlueStore admin tool. It transforms BlueStore's RocksDB

database from one shape to another into several column families without redeploying the OSDs. Column families have the same features as the whole database, but allows users to operate on smaller data sets and apply different options. It leverages the different expected lifetime of keys stored. The keys are moved during the transformation without creating new keys or deleting existing keys.

## Prerequisites

- A running Red Hat Ceph Storage cluster.
- The object store configured as BlueStore.
- A containerized deployment of Red Hat Ceph Storage.
- OSD nodes deployed on the nodes.
- Root level access to the nodes.

## Procedure

1. Download the package:

### Example

```
[root@host01 ~]# yum config-manager --add-repo=http://download.eng.bos.redhat.com/rhel-8/composes/auto/ceph-5.0-rhel-8/latest-RHCEPH-5-RHEL-8/compose/OSD/x86_64/os/
```

2. Install the **ceph-osd** package. This package includes **ceph-bluestore-tool**, used for resharding.

### Example

```
[root@host01 ~]# dnf install -y ceph-osd
```

3. To stop the OSD service, you have to run **systemctl** command to get the *SERVICE\_ID\_OF\_OSD*.
  - a. Run the **systemctl** service to get the *SERVICE\_ID\_OF\_OSD* of the OSD that has to be stopped.

### Example

```
[root@host01 ~]# systemctl --type=service
ceph-4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service
```

- b. Stop the OSD service:

### Syntax

```
systemctl stop SERVICE_ID_OF_OSD
```

### Example

```
[root@host01 ~]# systemctl stop ceph-4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service
```

- 4. Navigate to the directory where the OSDs are deployed:

### Syntax

```
cd /var/lib/ceph/OSD_DIRECTORY/OSD_ID/
```

### Example

```
[root@host01 ~]# cd /var/lib/ceph/4709608e-9089-11eb-bb5a-001a4a000740/osd.6/
```

- 5. Take a back-up of the **unit.run** file in a directory before making any changes:

### Example

```
[root@host01 osd.6]# cp /var/lib/ceph/4709608e-9089-11eb-bb5a-001a4a000740/osd.6/unit.run /var/lib/ceph/4709608e-9089-11eb-bb5a-001a4a000740/osd.6/backup_osd.6
```

- 6. Edit the **unit.run** file as follows:

- a. Check the **unit.run** file:

### Example

```
[root@host01 osd.3]# cat unit.run

/bin/podman run --rm --ipc=host --authfile=/etc/ceph/podman-auth.json --net=host --
entrypoint /usr/bin/ceph-osd --privileged --group-add=disk --name ceph-4709608e-9089-
11eb-bb5a-001a4a000740-osd.6 -d --log-driver journald --conmon-pidfile /run/ceph-
4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service-pid --cidfile /run/ceph-
4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service-cid -e
CONTAINER_IMAGE=registry.redhat.io/rhceph-alpha/rhceph-5-
rhel8@sha256:9aaea414e2c263216f3cdbc7a096f57c3adf6125ec9f4b0f5f65fa8c43987155
-e NODE_NAME=host01 -v /var/run/ceph/4709608e-9089-11eb-bb5a-
001a4a000740:/var/run/ceph:z -v /var/log/ceph/4709608e-9089-11eb-bb5a-
001a4a000740:/var/log/ceph:z -v /var/lib/ceph/4709608e-9089-11eb-bb5a-
001a4a000740/crash:/var/lib/ceph/crash:z -v /var/lib/ceph/4709608e-9089-11eb-bb5a-
001a4a000740/osd.6:/var/lib/ceph/osd/ceph-6:z -v /var/lib/ceph/4709608e-9089-11eb-
bb5a-001a4a000740/osd.6/config:/etc/ceph/ceph.conf:z -v /dev:/dev -v
/run/udev:/run/udev -v /sys:/sys -v /var/lib/ceph/4709608e-9089-11eb-bb5a-
001a4a000740/selinux:/sys/fs/selinux:ro -v /run/lvm:/run/lvm -v /run/lock/lvm:/run/lock/lvm
registry.redhat.io/rhceph-alpha/rhceph-5-
rhel8@sha256:9aaea414e2c263216f3cdbc7a096f57c3adf6125ec9f4b0f5f65fa8c43987155
-n osd.6 -f --setuser ceph --setgroup ceph --default-log-to-file=false --default-log-to-
stderr=true --default-log-stderr-prefix="debug"
```

- b. Edit the file:

### Example

```
[root@host01 osd.3]# vi unit.run
```

- i. Replace `/bin/podman run --rm --ipc=host --authfile=/etc/ceph/podman-auth.json --net=host --entrypoint /usr/bin/ceph-osd`` with ``/bin/podman run -it --entrypoint /bin/bash`
- ii. Remove the contents after the registry details. For example, in the above file, you can remove all contents after `registry.redhat.io/rhceph-alpha/rhceph-5-rhel8@sha256:9aaea414e2c263216f3cdcb7a096f57c3adf6125ec9f4b0f5f65fa8c43987155`. In this example, the following content is removed:

### Example

```
-n osd.6 -f --setuser ceph --setgroup ceph --default-log-to-file=false --default-log-to-stderr=true --default-log-stderr-prefix="debug"
```

- iii. Save the file.
- c. Check the modified `unit.run` file.

### Example

```
[root@host01 osd.6]# cat unit.run

/bin/podman run -it --entrypoint /bin/bash --privileged --group-add=disk --name ceph-4709608e-9089-11eb-bb5a-001a4a000740-osd.6 -d --log-driver journald --conmon-pidfile /run/ceph-4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service-pid --cidfile /run/ceph-4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service-cid -e CONTAINER_IMAGE=registry.redhat.io/rhceph-alpha/rhceph-5-rhel8@sha256:9aaea414e2c263216f3cdcb7a096f57c3adf6125ec9f4b0f5f65fa8c43987155 -e NODE_NAME=host01 -v /var/run/ceph/4709608e-9089-11eb-bb5a-001a4a000740:/var/run/ceph:z -v /var/log/ceph/4709608e-9089-11eb-bb5a-001a4a000740:/var/log/ceph:z -v /var/lib/ceph/4709608e-9089-11eb-bb5a-001a4a000740/crash:/var/lib/ceph/crash:z -v /var/lib/ceph/4709608e-9089-11eb-bb5a-001a4a000740/osd.6:/var/lib/ceph/osd/ceph-6:z -v /var/lib/ceph/4709608e-9089-11eb-bb5a-001a4a000740/osd.6/config:/etc/ceph/ceph.conf:z -v /dev:/dev -v /run/udev:/run/udev -v /sys:/sys -v /var/lib/ceph/4709608e-9089-11eb-bb5a-001a4a000740/selinux:/sys/fs/selinux:ro -v /run/lvm:/run/lvm -v /run/lock/lvm:/run/lock/lvm registry.redhat.io/rhceph-alpha/rhceph-5-rhel8@sha256:9aaea414e2c263216f3cdcb7a096f57c3adf6125ec9f4b0f5f65fa8c43987155
```

7. Start the OSD service:

### Syntax

```
systemctl start SERVICE_ID_OF_OSD
```

### Example

```
[root@host01 osd.6]# systemctl start ceph-4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service
```

8. Check the status of the OSD service:

### Syntax

```
systemctl status SERVICE_ID_OF_OSD
```

### Example

```
[root@host01 osd.6]# systemctl status ceph-4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service
```

- ceph-4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service - Ceph osd.6 for 4709608e-9089-11eb-bb5a-001a4a000740  
Loaded: loaded (/etc/systemd/system/ceph-4709608e-9089-11eb-bb5a-001a4a000740@.service; enabled; vendor preset: disabled)  
Active: active (running) since Mon 2021-04-05 11:05:40 IST; 44s ago

- Run the **podman** service to get the *CONTAINER\_ID* of the OSD:

```
[root@host01 osd.6]# podman ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
0ca11f9df3a5	registry.redhat.io/rhceph-alpha/rhceph-5-rhel8@sha256:9aaea414e2c263216f3cdbc7a096f57c3adf6125ec9f4b0f5f65fa8c43987155		6 minutes ago	Up 6 minutes ago		ceph-4709608e-9089-11eb-bb5a-001a4a000740-osd.6

- To reshard the RocksDB database, run these steps:

- Enter the container of the OSD:

#### Syntax

```
podman exec -it CONTAINER_ID /bin/bash
```

#### Example

```
[root@host01 osd.6]# podman exec -it 0ca11f9df3a5 /bin/bash
```

- To check file system consistency, run the **fsck** command:

#### Syntax

```
ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-OSD_ID/ fsck
```

#### Example

```
[root@0ca11f9df3a5 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-6/ fsck
```

```
fsck success
```

- To check the sharding status of the OSD node, run the **show-sharding** command:

#### Syntax

```
ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-OSD_ID/ show-sharding
```

### Example

```
[root@0ca11f9df3a5 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-6/ show-sharding
```

```
m(3) p(3,0-12) O(3,0-13)=block_cache={type=binned_lru} L P
```

- d. Run the **ceph-bluestore-tool** command to reshard. Red Hat recommends to use the parameters as given in the command:

### Syntax

```
ceph-bluestore-tool --log-level 10 -l log.txt --path /var/lib/ceph/osd/ceph-OSD_ID/ --sharding="m(3) p(3,0-12) O(3,0-13) L P" reshard
```

### Example

```
root@0ca11f9df3a5 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-6/ --sharding="m(3) p(3,0-12) O(3,0-13) L P" reshard
```

- e. To check the sharding status of the OSD node, run the **show-sharding** command:

### Syntax

```
ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-OSD_ID/ show-sharding
```

### Example

```
[root@0ca11f9df3a5 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-6/ show-sharding
```

```
m(3) p(3,0-12) O(3,0-13) L P
```

- f. To check file system consistency, run the **fsck** command:

### Syntax

```
ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-OSD_ID/ fsck
```

### Example

```
[root@0ca11f9df3a5 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-6/ fsck
```

```
fsck success
```

11. To restart the OSD, first exit the container and then run these steps:

- a. Stop the OSD service:

## Syntax

```
systemctl stop SERVICE_ID_OF_OSD
```

## Example

```
[root@host01 ~]# systemctl stop ceph-4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service
```

- b. To restore the OSD service, replace the **unit.run** file with the backed-up file.

## Example

```
[root@host01 osd.6]# cp /var/lib/ceph/4709608e-9089-11eb-bb5a-001a4a000740/osd.6/backup_osd.6 /var/lib/ceph/4709608e-9089-11eb-bb5a-001a4a000740/osd.6/unit.run
```

- c. Restart the OSD service:

## Syntax

```
systemctl start SERVICE_ID_OF_OSD
```

## Example

```
[root@host01 osd.6]# systemctl start ceph-4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service
```

## Verification

- Check the status of the OSD service:

## Syntax

```
systemctl status SERVICE_ID_OF_OSD
```

## Example

```
[root@host01 osd.6]# systemctl status ceph-4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service
```

```
• ceph-4709608e-9089-11eb-bb5a-001a4a000740@osd.6.service - Ceph osd.6 for 4709608e-9089-11eb-bb5a-001a4a000740
  Loaded: loaded (/etc/systemd/system/ceph-4709608e-9089-11eb-bb5a-001a4a000740@.service; enabled; vendor preset: disabled)
  Active: active (running) since Mon 2021-04-05 11:28:21 IST; 34min ago
```

## Additional Resources

- See the [Red Hat Ceph Storage Installation Guide](#) for more information.

## 8.7. THE BLUESTORE FRAGMENTATION TOOL

As a storage administrator, you will want to periodically check the fragmentation level of your BlueStore OSDs. You can check fragmentation levels with one simple command for offline or online OSDs.

### 8.7.1. Prerequisites

- A running Red Hat Ceph Storage cluster.
- BlueStore OSDs.

### 8.7.2. What is the BlueStore fragmentation tool?

For BlueStore OSDs, the free space gets fragmented over time on the underlying storage device. Some fragmentation is normal, but when there is excessive fragmentation this causes poor performance.

The BlueStore fragmentation tool generates a score on the fragmentation level of the BlueStore OSD. This fragmentation score is given as a range, 0 through 1. A score of 0 means no fragmentation, and a score of 1 means severe fragmentation.

**Table 8.1. Fragmentation scores' meaning**

Score	Fragmentation Amount
0.0 - 0.4	None to tiny fragmentation.
0.4 - 0.7	Small and acceptable fragmentation.
0.7 - 0.9	Considerable, but safe fragmentation.
0.9 - 1.0	Severe fragmentation and that causes performance issues.



#### IMPORTANT

If you have severe fragmentation, and need some help in resolving the issue, contact [Red Hat Support](#).

### 8.7.3. Checking for fragmentation

Checking the fragmentation level of BlueStore OSDs can be done either online or offline.

#### Prerequisites

- A running Red Hat Ceph Storage cluster.
- BlueStore OSDs.

#### Online BlueStore fragmentation score

1. Inspect a running BlueStore OSD process:

- a. Simple report:

**Syntax**

```
ceph daemon OSD_ID bluestore allocator score block
```

**Example**

```
[ceph: root@host01 /]# ceph daemon osd.123 bluestore allocator score block
```

- b. A more detailed report:

**Syntax**

```
ceph daemon OSD_ID bluestore allocator dump block
```

**Example**

```
[ceph: root@host01 /]# ceph daemon osd.123 bluestore allocator dump block
```

**Offline BlueStore fragmentation score**

1. Follow the steps for resharding for checking the offline fragmentation score.

Example

```
[root@host01 ~]# podman exec -it 7fbd6c6293c0 /bin/bash
```

1. Inspect a non-running BlueStore OSD process:

- a. Simple report:

**Syntax**

```
ceph-bluestore-tool --path PATH_TO_OSD_DATA_DIRECTORY --allocator block free-score
```

**Example**

```
[root@7fbd6c6293c0 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-123 --allocator block free-score
```

- b. A more detailed report:

**Syntax**

```
ceph-bluestore-tool --path PATH_TO_OSD_DATA_DIRECTORY --allocator block free-dump block:  
{  
  "fragmentation_rating": 0.018290238194701977  
}
```

## Example

```
[root@7fbd6c6293c0 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-123 --allocator
block free-dump
block:
{
  "capacity": 21470642176,
  "alloc_unit": 4096,
  "alloc_type": "hybrid",
  "alloc_name": "block",
  "extents": [
    {
      "offset": "0x370000",
      "length": "0x20000"
    },
    {
      "offset": "0x3a0000",
      "length": "0x10000"
    },
    {
      "offset": "0x3f0000",
      "length": "0x20000"
    },
    {
      "offset": "0x460000",
      "length": "0x10000"
    }
  ],
}
```

## Additional Resources

- See the [BlueStore Fragmentation Tool](#) for details on the fragmentation score.
- See the [Resharding the RocksDB database using the BlueStore admin tool](#) for details on resharding.

## CHAPTER 9. CEPHADM TROUBLESHOOTING

As a storage administrator, you can troubleshoot the Red Hat Ceph Storage cluster. Sometimes there is a need to investigate why a Cephadm command failed or why a specific service does not run properly.

### 9.1. PREREQUISITES

- A running Red Hat Ceph Storage cluster.

### 9.2. PAUSE OR DISABLE CEPHADM

If Cephadm does not behave as expected, you can pause most of the background activity with the following command:

#### Example

```
[ceph: root@host01 /]# ceph orch pause
```

This stops any changes, but Cephadm periodically checks hosts to refresh its inventory of daemons and devices.

If you want to disable Cephadm completely, run the following command:

#### Example

```
[ceph: root@host01 /]# ceph orch backend  
ceph mgr module disable cephadm
```

Note that previously deployed daemon containers continue to exist and start as they did before.

### 9.3. PER SERVICE AND PER DAEMON EVENT

Cephadm stores events per service and per daemon in order to aid in debugging failed daemon deployments. These events often contain relevant information:

#### Per service

##### Syntax

```
ceph orch ls --service_name SERVICE_NAME --format yaml
```

#### Example

```
[ceph: root@host01 /]# ceph orch ls --service_name alertmanager --format yaml  
service_type: alertmanager  
service_name: alertmanager  
placement:  
  hosts:  
  - unknown_host  
status:  
  ...  
  running: 1
```

```

size: 1
events:
- 2021-02-01T08:58:02.741162 service:alertmanager [INFO] "service was created"
- '2021-02-01T12:09:25.264584 service:alertmanager [ERROR] "Failed to apply: Cannot
  place <AlertManagerSpec for service_name=alertmanager> on unknown_host: Unknown hosts"'

```

## Per daemon

### Syntax

```
ceph orch ps --service-name SERVICE_NAME --daemon-id DAEMON_ID --format yaml
```

### Example

```

[ceph: root@host01 /]# ceph orch ps --service-name mds --daemon-id cephfs.hostname.ppdhsz --
format yaml
daemon_type: mds
daemon_id: cephfs.hostname.ppdhsz
hostname: hostname
status_desc: running
...
events:
- 2021-02-01T08:59:43.845866 daemon:mds.cephfs.hostname.ppdhsz [INFO] "Reconfigured
  mds.cephfs.hostname.ppdhsz on host 'hostname'"

```

## 9.4. CHECK CEPHADM LOGS

You can monitor the Cephadm log in real time with the following command:

### Example

```
[ceph: root@host01 /]# ceph -W cephadm
```

You can see the last few messages with the following command:

### Example

```
[ceph: root@host01 /]# ceph log last cephadm
```

If you have enabled logging to files, you can see a Cephadm log file called **ceph.cephadm.log** on the monitor hosts.

## 9.5. GATHER LOG FILES

You can use the **journalctl** command, to gather the log files for all the daemons.



### NOTE

You have to run all these commands outside the **cephadm** shell.

**NOTE**

By default, Cephadm stores logs in journald which means that daemon logs are no longer available in **/var/log/ceph**.

- To read the log file of a specific daemon, run the following command:

**Syntax**

```
cephadm logs --name DAEMON_NAME
```

**Example**

```
[root@host01 ~]# cephadm logs --name cephfs.hostname.ppdhsz
```

**NOTE**

This command works when run on the same hosts where the daemon is running.

- To read the log file of a specific daemon running on a different host, run the following command:

**Syntax**

```
cephadm logs --fsid FSID --name DAEMON_NAME
```

**Example**

```
[root@host01 ~]# cephadm logs --fsid 2d2fd136-6df1-11ea-ae74-002590e526e8 --name cephfs.hostname.ppdhsz
```

where **fsid** is the cluster ID provided by the **ceph status** command.

- To fetch all log files of all the daemons on a given host, run the following command:

**Syntax**

```
for name in $(cephadm ls | python3 -c "import sys, json; [print(i['name']) for i in json.load(sys.stdin)]"); do cephadm logs --fsid FSID_OF_CLUSTER --name "$name" > $name; done
```

**Example**

```
[root@host01 ~]# for name in $(cephadm ls | python3 -c "import sys, json; [print(i['name']) for i in json.load(sys.stdin)]"); do cephadm logs --fsid 57bddb48-ee04-11eb-9962-001a4a000672 --name "$name" > $name; done
```

## 9.6. COLLECT SYSTEMD STATUS

- To print the state of a systemd unit, run the following command:

```
- .
```

### Example

```
[root@host01 ~]$ systemctl status ceph-a538d494-fb2a-48e4-82c8-
b91c37bb0684@mon.host01.service
```

## 9.7. LIST ALL DOWNLOADED CONTAINER IMAGES

To list all the container images that are downloaded on a host, run the following command:

### Example

```
[ceph: root@host01 /]# podman ps -a --format json | jq '[]|.Image'
"docker.io/library/rhel8"
"registry.redhat.io/rhceph-alpha/rhceph-5-
rhel8@sha256:9aaea414e2c263216f3cdbc7a096f57c3adf6125ec9f4b0f5f65fa8c43987155"
```

## 9.8. MANUALLY RUN CONTAINERS

Cephadm writes small wrappers that runs a container. Refer to `/var/lib/ceph/CLUSTER_FSID/SERVICE_NAME/unit` to run the container execution command.

### Analysing SSH errors

If you get the following error:

### Example

```
execnet.gateway_bootstrap.HostNotFound: -F /tmp/cephadm-conf-73z09u6g -i /tmp/cephadm-
identity-ky7ahp_5 root@10.10.1.2
...
raise OrchestratorError(msg) from e
orchestrator._interface.OrchestratorError: Failed to connect to 10.10.1.2 (10.10.1.2).
Please make sure that the host is reachable and accepts connections using the cephadm SSH key
```

Try the following options to troubleshoot the issue:

- To ensure Cephadm has a SSH identity key, run the following command:

### Example

```
[ceph: root@host01 /]# ceph config-key get mgr/cephadm/ssh_identity_key >
~/cephadm_private_key
INFO:cephadm:Inferring fsid f8edc08a-7f17-11ea-8707-000c2915dd98
INFO:cephadm:Using recent ceph image docker.io/ceph/ceph:v15 obtained
'mgr/cephadm/ssh_identity_key'
[root@mon1 ~] # chmod 0600 ~/cephadm_private_key
```

If the above command fails, Cephadm does not have a key. To generate a SSH key, run the following command:

### Example

```
[ceph: root@host01 /]# chmod 0600 ~/cephadm_private_key
```

Or

### Example

```
[ceph: root@host01 /]# cat ~/cephadm_private_key | ceph cephadm set-ssk-key -i-
```

- To ensure that the SSH configuration is correct, run the following command:

### Example

```
[ceph: root@host01 /]# ceph cephadm get-ssh-config
```

- To verify the connection to the host, run the following command:

### Example

```
[ceph: root@host01 /]# ssh -F config -i ~/cephadm_private_key root@host01
```

## Verify public key is in `authorized_keys`.

To verify that the public key is in the `authorized_keys` file, run the following commands:

### Example

```
[ceph: root@host01 /]# ceph cephadm get-pub-key  
[ceph: root@host01 /]# grep "cat ~/ceph.pub`" /root/.ssh/authorized_keys
```

## 9.9. CIDR NETWORK ERROR

Classless inter domain routing (CIDR) also known as supernetting, is a method of assigning Internet Protocol (IP) addresses. The Cephadm log entries shows the current state that improves the efficiency of address distribution and replaces the previous system based on Class A, Class B and Class C networks. If you see one of the following errors:

**ERROR: Failed to infer CIDR network for mon ip \*; pass `--skip-mon-network` to configure it later**

Or

**Must set `public_network` config option or specify a CIDR network, `ceph addrvec`, or plain IP**

You need to run the following command:

### Example

```
[ceph: root@host01 /]# ceph config set host public_network hostnetwork
```

## 9.10. ACCESS THE ADMIN SOCKET

Each Ceph daemon provides an admin socket that bypasses the MONs.

To access the admin socket, enter the daemon container on the host:

### Example

```
[ceph: root@host01 /]# cephadm enter --name cephfs.hostname.ppdhsz
[ceph: root@mon1 /]# ceph --admin-daemon /var/run/ceph/ceph-cephfs.hostname.ppdhsz.asok
config show
```

## 9.11. MANUALLY DEPLOYING A MGR DAEMON

Cephadm requires a **mgr** daemon in order to manage the Red Hat Ceph Storage cluster. In case the last **mgr** daemon of a Red Hat Ceph Storage cluster was removed, you can manually deploy a **mgr** daemon, on a random host of the Red Hat Ceph Storage cluster.

### Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to all the nodes.
- Hosts are added to the cluster.

### Procedure

1. Log into the Cephadm shell:

#### Example

```
[root@host01 ~]# cephadm shell
```

2. Disable the Cephadm scheduler to prevent Cephadm from removing the new MGR daemon, with the following command:

#### Example

```
[ceph: root@host01 /]# ceph config-key set mgr/cephadm/pause true
```

3. Get or create the **auth** entry for the new MGR daemon:

#### Example

```
[ceph: root@host01 /]# ceph auth get-or-create mgr.host01.smfvfd1 mon "profile mgr" osd
"allow *" mds "allow *"
[mgr.host01.smfvfd1]
key = AQDhcORgW8toCRAAIMzIqWXnh3cGRjqYEa9ikw==
```

4. Open **ceph.conf** file:

#### Example

```
[ceph: root@host01 /]# ceph config generate-minimal-conf
# minimal ceph.conf for 8c9b0072-67ca-11eb-af06-001a4a0002a0
[global]
fsid = 8c9b0072-67ca-11eb-af06-001a4a0002a0
mon_host = [v2:10.10.200.10:3300/0,v1:10.10.200.10:6789/0]
[v2:10.10.10.100:3300/0,v1:10.10.200.100:6789/0]
```

5. Get the container image:

### Example

```
[ceph: root@host01 /]# ceph config get "mgr.host01.smfvfd1" container_image
```

6. Create a **config-json.json** file and add the following:



### NOTE

Use the values from the output of the **ceph config generate-minimal-conf** command.

### Example

```
{
  {
    "config": "# minimal ceph.conf for 8c9b0072-67ca-11eb-af06-001a4a0002a0\n[global]\n\tfsid
= 8c9b0072-67ca-11eb-af06-001a4a0002a0\n\tmon_host =
[v2:10.10.200.10:3300/0,v1:10.10.200.10:6789/0]
[v2:10.10.10.100:3300/0,v1:10.10.200.100:6789/0]\n",
    "keyring": "[mgr.Ceph5-2.smfvfd1]\n\tkey =
AQDhcORgW8toCRAAIMzLqWXnh3cGRjqYEa9ikw==\n"
  }
}
```

7. Exit from the Cephadm shell:

### Example

```
[ceph: root@host01 /]# exit
```

8. Deploy the MGR daemon:

### Example

```
[root@host01 ~]# cephadm --image registry.redhat.io/rhceph-alpha/rhceph-5-rhel8:latest
deploy --fsid 8c9b0072-67ca-11eb-af06-001a4a0002a0 --name mgr.host01.smfvfd1 --config-
json config-json.json
```

## Verification

In the Cephadm shell, run the following command:

### Example

```
[ceph: root@host01 /]# ceph -s
```

You can see a new **mgr** daemon has been added.

## CHAPTER 10. CEPHADM OPERATIONS

As a storage administrator, you can carry out Cephadm operations in the Red Hat Ceph Storage cluster.

### 10.1. PREREQUISITES

- A running Red Hat Ceph Storage cluster.

### 10.2. MONITOR CEPHADM LOG MESSAGES

Cephadm logs to the cephadm cluster log channel so you can monitor progress in real time.

- To monitor progress in realtime, run the following command:

#### Example

```
[ceph: root@host01 /]# ceph -W cephadm
```

By default, the logs display information level events and more.

#### Example

```
2021-06-24T17:51:36.335728+0000 mgr.Ceph5-1.nqikfh [INF] refreshing Ceph5-adm facts
2021-06-24T17:51:37.170982+0000 mgr.Ceph5-1.nqikfh [INF] deploying 1 monitor(s) instead
of 2 so monitors may achieve consensus
2021-06-24T17:51:37.173487+0000 mgr.Ceph5-1.nqikfh [ERR] It is NOT safe to stop
['mon.Ceph5-adm']: not enough monitors would be available (Ceph5-2) after stopping mons
[Ceph5-adm]
2021-06-24T17:51:37.174415+0000 mgr.Ceph5-1.nqikfh [INF] Checking pool "nfs-ganesha"
exists for service nfs.foo
2021-06-24T17:51:37.176389+0000 mgr.Ceph5-1.nqikfh [ERR] Failed to apply nfs.foo spec
NFSServiceSpec({'placement': PlacementSpec(count=1), 'service_type': 'nfs', 'service_id':
'foo', 'unmanaged': False, 'preview_only': False, 'pool': 'nfs-ganesha', 'namespace': 'nfs-ns'}):
Cannot find pool "nfs-ganesha" for service nfs.foo
Traceback (most recent call last):
  File "/usr/share/ceph/mgr/cephadm/serve.py", line 408, in _apply_all_services
    if self._apply_service(spec):
  File "/usr/share/ceph/mgr/cephadm/serve.py", line 509, in _apply_service
    config_func(spec)
  File "/usr/share/ceph/mgr/cephadm/services/nfs.py", line 23, in config
    self.mgr._check_pool_exists(spec.pool, spec.service_name())
  File "/usr/share/ceph/mgr/cephadm/module.py", line 1840, in _check_pool_exists
    raise OrchestratorError(f'Cannot find pool "{pool}" for '
orchestrator._interface.OrchestratorError: Cannot find pool "nfs-ganesha" for service nfs.foo
2021-06-24T17:51:37.179658+0000 mgr.Ceph5-1.nqikfh [INF] Found osd claims -> {}
2021-06-24T17:51:37.180116+0000 mgr.Ceph5-1.nqikfh [INF] Found osd claims for
drivegroup all-available-devices -> {}
2021-06-24T17:51:37.182138+0000 mgr.Ceph5-1.nqikfh [INF] Applying all-available-devices
on host Ceph5-adm...
2021-06-24T17:51:37.182987+0000 mgr.Ceph5-1.nqikfh [INF] Applying all-available-devices
on host Ceph5-1...
2021-06-24T17:51:37.183395+0000 mgr.Ceph5-1.nqikfh [INF] Applying all-available-devices
on host Ceph5-2...
2021-06-24T17:51:43.373570+0000 mgr.Ceph5-1.nqikfh [INF] Reconfiguring node-
```

```
exporter.Ceph5-1 (unknown last config time)...
2021-06-24T17:51:43.373840+0000 mgr.Ceph5-1.nqikfh [INF] Reconfiguring daemon node-
exporter.Ceph5-1 on Ceph5-1
```

- To see the debug-level messages, run the following commands:

#### Example

```
[ceph: root@host01 /]# ceph config set mgr mgr/cephadm/log_to_cluster_level debug
[ceph: root@host01 /]# ceph -W cephadm --watch-debug
```

- To see the recent events, run the following command:

#### Example

```
[ceph: root@host01 /]# ceph log last cephadm
```

These events are also logged to **ceph.cephadm.log** file on the monitor hosts and to the monitor daemon's **stderr**

## 10.3. CEPH DAEMON LOGS

You can view the Ceph daemon logs through **stderr** or files. Logging to **stdout**

Traditionally, Ceph daemons have logged to **/var/log/ceph**. By default, Cephadm daemons log to **stderr** and the logs are captured by the container runtime environment. For most systems, by default, these logs are sent to **journald** and accessible through the **journalctl** command.

- For example, to view the logs for the daemon on **host01** for a storage cluster with ID **5c5a50ae-272a-455d-99e9-32c6a013e694**:

#### Example

```
[ceph: root@host01 /]# journalctl -u ceph-5c5a50ae-272a-455d-99e9-
32c6a013e694@host01
```

This works well for normal Cephadm operations when logging levels are low.

- To disable logging to **stderr**, set the following values:

#### Example

```
[ceph: root@host01 /]# ceph config set global log_to_stderr false
[ceph: root@host01 /]# ceph config set global mon_cluster_log_to_stderr false
```

### Logging to files

You can also configure Ceph daemons to log to files instead of **stderr**. When logging to files, Ceph logs are located in **/var/log/ceph/CLUSTER\_FSID**.

- To enable logging to files, set the following values:

#### Example

```
■
```

```
[ceph: root@host01 /]# ceph config set global log_to_file true
[ceph: root@host01 /]# ceph config set global mon_cluster_log_to_file true
```



### NOTE

Red Hat recommends disabling logging to **stderr** to avoid double logs.



### IMPORTANT

Currently log rotation to a non-default path is not supported.

By default, Cephadm sets up log rotation on each host to rotate these files. You can configure the logging retention schedule by modifying `/etc/logrotate.d/ceph.CLUSTER_FSID`.

## 10.4. DATA LOCATION

Cephadm daemon data and logs are located in slightly different locations than the older versions of Ceph:

- `/var/log/ceph/CLUSTER_FSID` contains all the storage cluster logs. Note that by default Cephadm logs through **stderr** and the container runtime, so these logs are usually not present.
- `/var/lib/ceph/CLUSTER_FSID` contains all the cluster daemon data, besides logs.
- `var/lib/ceph/CLUSTER_FSID/DAEMON_NAME` contains all the data for an specific daemon.
- `/var/lib/ceph/CLUSTER_FSID/crash` contains the crash reports for the storage cluster.
- `/var/lib/ceph/CLUSTER_FSID/removed` contains old daemon data directories for the stateful daemons, for example monitor or Prometheus, that have been removed by Cephadm.

### Disk usage

A few Ceph daemons may store a significant amount of data in `/var/lib/ceph`, notably the monitors and Prometheus daemon, hence Red Hat recommends moving this directory to its own disk, partition, or logical volume so that the root file system is not filled up.

## 10.5. CEPHADM HEALTH CHECKS

As a storage administrator, you can monitor the Red Hat Ceph Storage cluster with the additional healthchecks provided by the Cephadm module. This is supplementary to the default healthchecks provided by the storage cluster.

### 10.5.1. Prerequisites

- A running Red Hat Ceph Storage cluster.

### 10.5.2. Cephadm operations health checks

Healthchecks are executed when the Cephadm module is active. You can get the following health warnings:

#### CEPHADM\_PAUSED

Cephadm background work is paused with the **ceph orch pause** command. Cephadm continues to perform passive monitoring activities such as checking the host and daemon status, but it does not make any changes like deploying or removing daemons. You can resume Cephadm work with the **ceph orch resume** command.

### CEPHADM\_STRAY\_HOST

One or more hosts have running Ceph daemons but are not registered as hosts managed by the Cephadm module. This means that those services are not currently managed by Cephadm, for example, a restart and upgrade that is included in the **ceph orch ps** command. You can manage the host(s) with the **ceph orch host add *HOST\_NAME*** command but ensure that SSH access to the remote hosts is configured. Alternatively, you can manually connect to the host and ensure that services on that host are removed or migrated to a host that is managed by Cephadm. You can also disable this warning with the setting **ceph config set mgr mgr/cephadm/warn\_on\_stray\_hosts false**

### CEPHADM\_STRAY\_DAEMON

One or more Ceph daemons are running but are not managed by the Cephadm module. This might be because they were deployed using a different tool, or because they were started manually. Those services are not currently managed by Cephadm, for example, a restart and upgrade that is included in the **ceph orch ps** command.

If the daemon is a stateful one that is a monitor or OSD daeon, these daemons should be adopted by Cephadm. For stateless daemons, you can provision a new daemon with the **ceph orch apply** command and then stop the unmanaged daemon.

You can disable this health warning with the setting **ceph config set mgr mgr/cephadm/warn\_on\_stray\_daemons false**.

### CEPHADM\_HOST\_CHECK\_FAILED

One or more hosts have failed the basic Cephadm host check, which verifies that:name: value

- The host is reachable and you can execute Cephadm.
- The host meets the basic prerequisites, like a working container runtime that is Podman , and working time synchronization. If this test fails, Cephadm wont be able to manage the services on that host.

You can manually run this check with the **ceph cephadm check-host *HOST\_NAME*** command. You can remove a broken host from management with the **ceph orch host rm *HOST\_NAME*** command. You can disable this health warning with the setting **ceph config set mgr mgr/cephadm/warn\_on\_failed\_host\_check false**.

## 10.5.3. Cephadm configuration health checks

Cephadm periodically scans each of the hosts in the storage cluster, to understand the state of the OS, disks, and NICs . These facts are analyzed for consistency across the hosts in the storage cluster to identify any configuration anomalies. The configuration checks are an optional feature.

- You can enable this feature with the following command:

#### Example

```
[ceph: root@host01 /]# ceph config set mgr mgr/cephadm/config_checks_enabled true
```

The configuration checks are triggered after each host scan, which is for a duration of one minute.

- The Cephadm log entries shows the current state and outcome of the configuration checks as follows:

#### Disabled state

#### Example

```
[ceph: root@host01 /]# ceph config set mgr mgr/cephadm/config_checks_enabled false
LL cephadm checks are disabled, use 'ceph config set mgr
mgr/cephadm/config_checks_enabled true' to enable
```

#### Enabled stat

#### Example

```
[ceph: root@host01 /]# ceph config set mgr mgr/cephadm/config_checks_enabled true
CEPHADM 8/8 checks enabled and executed (0 bypassed, 0 disabled). No issues detected
```

The configuration checks themselves are managed through several cephadm sub-commands.

- To determine whether the configuration checks are enabled, run the following command:

#### Example

```
[ceph: root@host01 /]# ceph cephadm config-check status
```

This command returns the status of the configuration checker as either **Enabled** or **Disabled**.

- To list all the configuration checks and their current state, run the following command:

#### Example

```
[ceph: root@host01 /]# ceph cephadm config-check ls
NAME          HEALTHCHECK          STATUS DESCRIPTION
kernel_security CEPHADM_CHECK_KERNEL_LSM      enabled checks
SELINUX/Apparmor profiles are consistent across cluster hosts
os_subscription CEPHADM_CHECK_SUBSCRIPTION    enabled checks subscription
states are consistent for all cluster hosts
public_network CEPHADM_CHECK_PUBLIC_MEMBERSHIP enabled check that all hosts
have a NIC on the Ceph public_netork
osd_mtu_size   CEPHADM_CHECK_MTU           enabled check that OSD hosts share a
common MTU setting
osd_linkspeed CEPHADM_CHECK_LINKSPEED      enabled check that OSD hosts
share a common linkspeed
network_missing CEPHADM_CHECK_NETWORK_MISSING enabled checks that the
cluster/public networks defined exist on the Ceph hosts
ceph_release   CEPHADM_CHECK_CEPH_RELEASE    enabled check for Ceph version
consistency - ceph daemons should be on the same release (unless upgrade is active)
kernel_version CEPHADM_CHECK_KERNEL_VERSION  enabled checks that the
MAJ.MIN of the kernel on Ceph hosts is consistent
```

Each configuration check is described as follows:

#### CEPHADM\_CHECK\_KERNEL\_LSM

Each host within the storage cluster is expected to operate within the same Linux Security Module (LSM) state. For example, if the majority of the hosts are running with SELINUX in **enforcing** mode, any host not running in this mode would be flagged as an anomaly and a healthcheck with a warning state is raised.

### **CEPHADM\_CHECK\_SUBSCRIPTION**

This check relates to the status of the vendor subscription. This check is only performed for hosts using Red Hat Enterprise Linux, but helps to confirm that all the hosts are covered by an active subscription so that patches and updates are available.

### **CEPHADM\_CHECK\_PUBLIC\_MEMBERSHIP**

All members of the cluster should have NICs configured on at least one of the public network subnets. Hosts that are not on the public network will rely on routing which may affect performance.

### **CEPHADM\_CHECK\_MTU**

The maximum transmission unit (MTU) of the NICs on OSDs can be a key factor in consistent performance. This check examines hosts that are running OSD services to ensure that the MTU is configured consistently within the cluster. This is determined by establishing the MTU setting that the majority of hosts are using, with any anomalies resulting in a Ceph healthcheck.

### **CEPHADM\_CHECK\_LINKSPEED**

Similar to the MTU check, linkspeed consistency is also a factor in consistent cluster performance. This check determines the linkspeed shared by the majority of the OSD hosts, resulting in a healthcheck for any hosts that are set at a lower linkspeed rate.

### **CEPHADM\_CHECK\_NETWORK\_MISSING**

The **public\_network** and **cluster\_network** settings support subnet definitions for IPv4 and IPv6. If these settings are not found on any host in the storage cluster a healthcheck is raised.

### **CEPHADM\_CHECK\_CEPH\_RELEASE**

Under normal operations, the Ceph cluster should be running daemons under the same Ceph release, for example all Red Hat Ceph Storage cluster 5 releases. This check looks at the active release for each daemon, and reports any anomalies as a healthcheck. This check is bypassed if an upgrade process is active within the cluster.

### **CEPHADM\_CHECK\_KERNEL\_VERSION**

The OS kernel version is checked for consistency across the hosts. Once again, the majority of the hosts is used as the basis of identifying anomalies.