Administering Red Hat Ceph Storage

Learn how to properly administer and operate Red Hat Ceph Storage.

Administration

Learn how to manage processes, monitor cluster states, manage users, and add and remove daemons for Red Hat Ceph Storage.

Ceph administration

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

This section helps storage administrators to perform such tasks as:

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

Reference

- For more information, see“Initial installation” on page 0.

Understanding process management

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.

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.

For more information on using systemd, see the **Introduction to systemd** and **Managing system services with systemctl** chapters within the *Configuring basic system settings* guide within the Product Documentation for Red Hat Enterprise Linux for your OS version, on the Managing system services with systemctl.

Starting, stopping, and restarting all Ceph services

You can start, stop, and restart all Ceph services from the host where you want to manage the Ceph services.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- Have root-node access
- Log into the `cephadm` shell.
  For example,

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

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 run 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.

> **Important:** 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:
>
> ```
> [root@host01 ~]# systemctl list-units “ceph*”
> ```
>
> The output provides a list of the service names that you can use to manage each Ceph daemon.

Get a list of Ceph services

- Get a list of Ceph services configured in the Red Hat Ceph Storage cluster and to get the specific service ID.

```
ceph orch ls
```

  For 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:v4.15   b7bae610cd46
crash                          3/3  4m ago     4M   *          registry.redhat.io/
rhceph-alpha/rhceph-9-rhel9:latest             c88a5d60f510
grafana                        1/1  4m ago     4M   count:1    registry.redhat.io/
rhceph-alpha/rhceph-9-dashboard-rhel9:latest   bd3d7748747b
mgr                            2/2  4m ago     4M   count:2    registry.redhat.io/
rhceph-alpha/rhceph-9-rhel9:latest             c88a5d60f510
mon                            2/2  4m ago     10w  count:2    registry.redhat.io/
rhceph-alpha/rhceph-9-rhel9: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-9-rhel9:latest             c88a5d60f510
prometheus                     1/1  4m ago     4M   count:1    registry.redhat.io/
openshift4/ose-prometheus:v4.15                bebb0ddef7f0
rgw.test_realm.test_zone       2/2  4m ago     3M   count:2    registry.redhat.io/
rhceph-alpha/rhceph-9-rhel9:latest             c88a5d60f510
```

Start a service

- Start a specific service by running the **`ceph orch start SERVICE_ID`** command.
  For example,

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

Stop a service

- Stop a specific service by running 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.
For more information, see Starting, stopping, and restarting all Ceph daemons using the systemctl command.

```
ceph orch stop SERVICE_ID
```

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

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

Restart a service

- Restart a specific service by running the following command.

  ```
  ceph orch restart SERVICE_ID
  ```

  For example,

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

Viewing log files of Ceph daemons

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

Prerequisites

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

Procedure

1. To view the entire Ceph log file, run a `journalctl` command as `root` composed in the following format:

   Syntax

   ```
   journalctl -u ceph SERVICE_ID
   ```

   Example

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

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

2. 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: The `sosreport` utility can be used 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.

Reference

- The `journalctl` manual page.

Powering down and rebooting the cluster

Use either the Ceph Orchestrator or `systemctl` commands to power down and restart the Red Hat Ceph Storage cluster.

Before you begin powering down Red Hat Ceph Storage, be sure that you have root-level access.

Powering down and rebooting with Ceph Orchestrator

Use the Ceph Orchestrator to shut down and restart the Red Hat Ceph Storage cluster. In most cases, logging in to a single system is sufficient to power off the cluster.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

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

The Ceph Orchestrator supports `start`, `stop`, and `restart` operations for powering down or rebooting the cluster.

Alternatively, use `systemctl` commands for powering down and rebooting the Ceph cluster. For more information, see “Powering down and rebooting by using systemctl commands” on page 7.

Use the following basic flow to power down a Red Hat Ceph Storage cluster. The procedure provides full in-depth information and commands for each step.

1. Stop all clients from using the Block Device image and Ceph Object Gateway on this cluster and on any other clients.
2. Stop the Ceph File System (CephFS) cluster and the MDS service.
3. Stop all component gateways related to block and file.
4. Stop all ingress services and RADOS gateways.
5. If mirror dameons are used, stop all mirror daemons.
6. Stop the monitoring service components.
7. Set the **noout** flag.
8. Shut down all nodes.

Powering down the Red Hat Ceph Storage cluster

1. Stop all clients from using user Block Device images, CephFS volumes, and Ceph Object Gateways.
2. Log in to the cephadm shell.
   For example,

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

   Before proceeding, the cluster must be in healthy state (`Health_OK` and all PGs `active+clean`). View the cluster state by using the **ceph -s** command.
3. When using CephFS, bring down the CephFS cluster.

```
ceph fs fail FS_NAME
 ceph status
 ceph fs set FS_NAME joinable false
 ceph mds fail FS_NAME:N
```

For example,

```
[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. Stop the MDS service.
   a. Get the MDS service name.

      ```
      ceph orch ls --service-type mds
      ```

      For example,

      ```
      [ceph: root@host01 /]# ceph orch ls --service-type mds
      ```

   b. Stop the MDS service that uses the name output in the previous step.

      ```
      ceph orch stop SERVICE_NAME
      ```

5. Stop all component gateways that are related to block and file systems.
   a. Find all relevant services that are running by using the **ceph orch ls** command.
   b. Stop each block and file system service.

      ```
      ceph orch stop SERVICE_NAME
      ```

6. Stop the Ceph Object Gateway services.
   Repeat for each deployed service.
   a. Find the Ceph Object Gateway service name.

      ```
      ceph orch ls --service-type SERVICE_TYPE
      ```

   b. Stop the Ceph Object Gateway service that uses the fetched service name.

      ```
      ceph orch stop SERVICE_NAME
      ```

7. Stop the Ceph Object Gateway ingress services.
   Repeat for each deployed service.
   a. Fetch the deployed ingress services.

      ```
      ceph orch ls --service-type ingress
      ```

   b. Stop each deployed ingress service.

      ```
      ceph orch stop SERVICE_NAME
      ```

8. Stop the monitoring service components.
   a. Stop the Alertmanager service.

      ```
      ceph orch stop alertmanager
      ```

   b. Stop the node-exporter service, which is part of the monitoring stack.

      ```
      ceph orch stop node-exporter
      ```

   c. Stop the Prometheus service.

```
ceph orch stop prometheus
```

d. Stop the Grafana dashboard service.

```
ceph orch stop grafana
```

e. Stop the crash service.

```
ceph orch stop crash
```

9. Set the **noout** flag.
Set the **noout** flag during maintenance to prevent CRUSH from automatically rebalancing the cluster.

> **Note:** Use the **pause** flag to block any forgotten client I/Os.

```
ceph osd set noout
```

For example,

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

10. Stop the OSD services from the `cephadm` node, one by one.
If there are large number of OSD daemons, you may proceed to shutdown the OSD nodes directly.

a. Fetch the OSD IDs.

```
ceph orch ps
```

b. Stop the OSD daemons.

```
ceph orch daemon stop OSD_ID
```

For example,

```
[ceph: root@host01 /]# ceph orch daemon stop osd.1
Scheduled to stop osd.1 on host 'host02'
```

c. Identify the OSD hosts.

```
ceph orch ls osd
```

d. Shut down each of the OSD hosts from the hosts listed in step “10.a” on page 6.

11. Stop the monitors.

a. Identify the hosts with monitors.

```
ceph orch ls mon
```

Note the listed monitor hosts.

b. Shut down the monitor hosts that are listed in the previous step.
Shut down the monitor hosts one by one.

```
systemtl stop SERVICE_NAME
```

12. Shut down any remaining standalone hosts.
Shut down hosts with services, such as Ceph Object Gateways that are still running.

Rebooting the Red Hat Ceph Storage cluster

PREREQUISITE

Before you begin rebooting the cluster, power on and stabilize any network equipment before powering on the Ceph hosts and nodes.

1. Power on all the Ceph hosts.
2. Log in to the administration node from the `cephadm` shell.
   For example,

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

3. Verify that all services are in the `running` state.

   ```
   ceph orch ls
   ```

4. Ensure that the cluster health is in the `Health_OK` status, by running the **ceph -s** command.
5. Unset the **noout** flag that was set in "Powering down the Red Hat Ceph Storage cluster" on page 4.

   ```
   ceph osd unset noout
   ```

   For example,

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

6. When using Ceph File System (CephFS), make the CephFS cluster available by setting the `joinable` flag to`true`.

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

   For example,

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

AFTER COMPLETING THE TASK

Verify that the cluster is in healthy state (`Health_OK` and all PGs `active+clean`). Run the **ceph -s** status command on a node with the client keyrings, for example, the Ceph Monitor or OpenStack controller nodes, to ensure that the cluster is healthy.

Powering down and rebooting by using `systemctl` commands

Use `systemctl` commands to shut down and restart the Red Hat Ceph Storage cluster. This method aligns with standard Linux service management practices.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

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

Alternatively, use the Ceph Orchestrator to power down and reboot the Ceph cluster. For more information, see "Powering down and rebooting with Ceph Orchestrator" on page 4.

Powering down the Red Hat Ceph Storage cluster

Repeat these steps for each node in the cluster.

1. Stop all clients from using user Block Device Images, CephFS volumes, and Ceph Object Gateways.
2. Log in to the `cephadm` shell.
   For example,

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

   Before proceeding, the cluster must be in healthy state (`Health_OK` and all PGs `active+clean`). View the cluster state by using the **ceph -s** command.
3. When using CephFS, bring down the CephFS cluster.

```
ceph fs fail FS_NAME
 ceph status
 ceph fs set FS_NAME joinable false
 ceph mds fail FS_NAME:N
```

For example,

```
[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. If the MDS and Ceph Object Gateway nodes are on their own dedicated nodes, stop the MDS service.
   a. Get the MDS service name.

      ```
      ceph orch ls --service-type mds
      ```

      For example,

      ```
      [ceph: root@host01 /]# ceph orch ls --service-type mds
      ```

   b. Stop the MDS service that uses the name output in the previous step.

      ```
      ceph orch stop SERVICE_NAME
      ```

5. Set the **noout** flag.
   Set the **noout** flag during maintenance to prevent CRUSH from automatically rebalancing the cluster.

   > **Note:** Use the **pause** flag to block any forgotten client I/Os.

   ```
   ceph osd set noout
   ```

   For example,

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

6. Get the systemd target of the daemons.

   ```
   systemctl list-units --type target | grep ceph
   ```

   For example,

   ```
   [root@host01 ~]# systemctl list-units --type target | grep ceph
   ceph-0b007564-ec48-11ee-b736-525400fd02f8.target loaded active active Ceph cluster
   0b007564-ec48-11ee-b736-525400fd02f8
   ceph.target                                      loaded active active All Ceph
   clusters and services
   ```

7. Disable the target that includes the cluster FSID.

   ```
   systemctl disable CLUSTER_FSID.target
   ```

   For example,

   ```
   [root@host01 ~]# systemctl disable ceph-0b007564-ec48-11ee-b736-525400fd02f8.target
   Removed "/etc/systemd/system/multi-user.target.wants/ceph-0b007564-ec48-11ee-
   b736-525400fd02f8.target".
   Removed "/etc/systemd/system/ceph.target.wants/ceph-0b007564-ec48-11ee-
   b736-525400fd02f8.target".
   ```

8. Stop all daemons on the host.

   ```
   systemctl stop CLUSTER_FSID.target
   ```

For example,

```
[root@host01 ~]# systemctl stop ceph-0b007564-ec48-11ee-b736-525400fd02f8.target
```

9. Shut down the node by using the **shutdown** command.
For example,

```
[root@host01 ~]# shutdown
Shutdown scheduled for Wed 2024-03-27 11:47:19 EDT, use 'shutdown -c' to cancel.
```

AFTER COMPLETING THE TASK

Repeat the shutdown steps for all nodes in the cluster.

Rebooting the Red Hat Ceph Storage cluster

1. Enable the systemd target to run all daemons.

```
systemctl enable CLUSTER_FSID.target
```

For example,

```
[root@host01 ~]# systemctl enable ceph-0b007564-ec48-11ee-b736-525400fd02f8.target
Created symlink /etc/systemd/system/multi-user.target.wants/ceph-0b007564-ec48-11ee-
b736-525400fd02f8.target → /etc/systemd/system/ceph-0b007564-ec48-11ee-
b736-525400fd02f8.target.
Created symlink /etc/systemd/system/ceph.target.wants/ceph-0b007564-ec48-11ee-
b736-525400fd02f8.target → /etc/systemd/system/ceph-0b007564-ec48-11ee-
b736-525400fd02f8.target.
```

2. Start the systemd target.

```
systemctl start CLUSTER_FSID.target
```

For example,

```
[root@host01 ~]# systemctl start ceph-0b007564-ec48-11ee-b736-525400fd02f8.target
```

Wait for all the nodes to come up.

3. Log in to the `cephadm` shell.

```
cephadm shell
```

4. Verify that all services are up and there are no connectivity issues between the nodes.

```
ceph orch ls
```

5. Unset the **noout** flags.
Run the following commands on a node with the client keyrings, for example, the Ceph Monitor or OpenStack controller node.

```
ceph osd unset noout
```

> **Note:** If the **pause** flag was set, unset the pause flag.

For example,

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

6. When using Ceph File System (CephFS), bring the CephFS cluster back up by setting the `joinable` flag to `true`.

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

For example,

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

AFTER COMPLETING THE TASK

Verify that the cluster is in healthy state (`Health_OK` and all PGs `active+clean`). Run the **`ceph -s`** status command on a node with the client keyrings, for example, the Ceph Monitor or OpenStack controller nodes, to ensure that the cluster is healthy.

Powering down and rebooting by using `cephadm` cluster commands

Power down and start the Red Hat Ceph Storage cluster by using automated `cephadm` cluster shutdown and startup commands.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

1. Ensure that the cluster is in a healthy state (`HEALTH_OK`) and that all placement groups are `active+clean`.
2. Ensure that all clients are stopped
   - Unmount Ceph File System (CephFS).
   - Disconnect Ceph Block Device (RBD) clients.
   - Stop Ceph Object Gateway (RGW) clients.
3. Log in to the administration host with root-level access.
4. If `cephadm` cannot automatically determine the cluster FSID, specify it by using the **`--fsid`** option.

Use `cephadm` commands to perform a coordinated shutdown and startup of all cluster services and hosts from the administration host.

> **Important:** Use the `--force` option only at the explicit direction of IBM Support. Using this option on an unhealthy cluster can result in data loss.

If you prefer manual control, see the procedures for powering down and rebooting by using Ceph Orchestrator or systemctl commands. For more information, see “Powering down and rebooting with Ceph Orchestrator” on page 4 and “Powering down and rebooting by using systemctl commands” on page 7.

1. Preview the shutdown operation.

   ```
   cephadm cluster-shutdown --dry-run
   ```

2. Shut down the cluster.

   ```
   cephadm cluster-shutdown --yes-i-really-mean-it
   ```

   The system performs a coordinated shutdown of cluster services and hosts.

3. Start the cluster.

   ```
   cephadm cluster-start [--parallel MAX_NUMBER]
   ```

   Use the --parallel option to set the maximum number of host clusters to start in parallel. If not set, the default number is 5.

   ```
   [ceph: root@host01 /]# cephadm cluster-start --parallel 10
   ```

4. Check the cluster status.

   ```
   cephadm cluster-status
   ```

Result

The cluster is shut down and restarted by using an automated workflow. Verify that the cluster returns to a healthy state.

- The shutdown process creates a state file:

  ```
  /var/lib/ceph/<fsid>/cluster-shutdown-state.json
  ```

- `cephadm` temporarily stores an SSH private key on the administration host to enable startup operations. The key is removed after startup completes.

Related information

Troubleshooting powering down and starting the cluster

Monitor Red Hat Ceph Storage clusters

You can monitor the health, performance, and status of Red Hat Ceph Storage clusters by using the Ceph Dashboard or the command-line interface (CLI). These tools provide visibility into cluster health, capacity usage, and component status at different levels of detail.

Checking Ceph version information

View version information for your Ceph cluster by using the dashboard or the command line interface.

PREREQUISITE

Before you begin, make sure that you have access to a node with connectivity to the Ceph cluster and appropriate user permissions to run Ceph commands.

The Ceph version information helps you to:

- Verify your deployment version after installation or upgrade.
- Confirm version consistency across cluster components.
- Gather version information for support cases.
- Document your environment for compliance purposes.

The Ceph Dashboard shows the product release, Ceph version, and Ceph Manager version while the command-line provides detailed version information for all daemon types.

> **Note:** There might be version mismatches across components during upgrades. In a stable cluster, all daemons run the same version. If there are mismatches after an upgrade, contact Red Hat Support.

Viewing version information from the dashboard

1. Log in to the Ceph Dashboard and navigate to the **Overview** page.
   The Ceph version is displayed prominently below the cluster ID, showing the complete version string including the Ceph version number, build number, commit hash, codename, and build type.
2. Alternatively, you can also click the **Question** (**?**) icon in the navigation bar and then click **About**, to view the version details that includes product release and Ceph Manager version.
   The dialog also displays information about the current user, user role, browser, and operating system.

Viewing version information from the CLI

1. Log in to a node with access to the Ceph cluster.
2. Run the **`ceph version`** command to view the overall cluster version.

> **Note:** This command requires Ceph admin privileges. If you encounter permission errors, contact your system administrator.

The following example shows typical output for Red Hat Ceph Storage 9.1.

```
ceph version 20.2.1-268.el9cp (7fbeb00b438402bdc9aff7f2829e682e67a9cb88) tentacle
(stable - RelWithDebInfo) release 9.1.0
```

3. To view detailed version information for each daemon type, run the **`ceph versions`** command.

> **Important:** The **`ceph versions`** command displays the Storage Package Version (for example, 20.2.1-268.el9cp) and IBM Storage Ceph version (for example, 9.9.1.0), but does not distinguish between standard releases and CVE-specific releases that share the same version numbers. To identify the exact release including CVE releases, refer to the IBM Storage Ceph release notes or contact Red Hat Support.

The following example shows typical output for Red Hat Ceph Storage 9.1.

```
{
    "mon": {
        "ceph version 20.2.1-268.el9cp (7fbeb00b438402bdc9aff7f2829e682e67a9cb88)
tentacle (stable - RelWithDebInfo) release 9.1.0": 3
    },
    "mgr": {
        "ceph version 20.2.1-268.el9cp (7fbeb00b438402bdc9aff7f2829e682e67a9cb88)
tentacle (stable - RelWithDebInfo) release 9.1.0": 2
    },
    "osd": {
        "ceph version 20.2.1-268.el9cp (7fbeb00b438402bdc9aff7f2829e682e67a9cb88)
tentacle (stable - RelWithDebInfo) release 9.1.0": 16
    },
    "mds": {
        "ceph version 20.2.1-268.el9cp (7fbeb00b438402bdc9aff7f2829e682e67a9cb88)
tentacle (stable - RelWithDebInfo) release 9.1.0": 3
    },
    "overall": {
        "ceph version 20.2.1-268.el9cp (7fbeb00b438402bdc9aff7f2829e682e67a9cb88)
tentacle (stable - RelWithDebInfo) release 9.1.0": 24
    }
}
```

The output shows version information for each daemon type (mon, mgr, osd, mds) with the number of daemons running each version. This is particularly useful for identifying version mismatches during or after upgrades.

Understanding the version string components:

- **Storage Package Version**: The version string before 'release' (for example, 20.2.1-268.el9cp) represents the Storage Package Version.
- **Ceph version**: The version string after 'release' represents the Storage Ceph version.
- **Daemon count**: The number after each version string indicates how many daemons of that type are running that specific version.

4. Review the output to verify component versions and your product release.

Monitor a Ceph cluster with the CLI

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 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.

High-level monitoring

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.

Checking 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 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 command:

   Example

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

3. You can check the status of the Ceph storage cluster by running `ceph status` command:

   Example

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

   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 Ceph 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.
   - The IO client operations.
   - An update on the upgrade process if the cluster is upgrading.

     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.

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

   2022-11-02 15:45:21.655871 osd.0 [INF] 17.71 deep-scrub ok
   2022-11-02 15:45:47.880608 osd.1 [INF] 1.0 scrub ok
   2022-11-02 15:45:48.865375 osd.1 [INF] 1.3 scrub ok
   2022-11-02 15:45:50.866479 osd.1 [INF] 1.4 scrub ok
   2022-11-02 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
   2022-11-02 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
   2022-11-02 15:45:53.997726 osd.1 [INF] 1.5 scrub ok
   2022-11-02 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
   2022-11-02 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
   2022-11-02 15:46:06.836430 osd.0 [INF] 17.75 deep-scrub ok
   2022-11-02 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
   ```

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.

Understanding 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.

The `SIZE/AVAIL/RAW USED` in the `ceph df` and **ceph status** command output are different if some OSDs are marked `OUT` of the cluster compared to when all OSDs are `IN`. The `SIZE/AVAIL/RAW USED` is calculated from sum of `SIZE` (osd disk size), `RAW USE` (total used space on disk), and `AVAIL` of all OSDs which are in `IN` state. You can see the total of `SIZE/AVAIL/RAW USED` for all OSDs in **ceph osd df tree** command output.

Example

```
[ceph: root@host01 /]#ceph df
--- RAW STORAGE ---
CLASS   SIZE    AVAIL     USED  RAW USED  %RAW USED
hdd    5 TiB  2.9 TiB  2.1 TiB   2.1 TiB      42.98

TOTAL  5 TiB  2.9 TiB  2.1 TiB   2.1 TiB      42.98


--- POOLS ---
POOL                         ID  PGS   STORED  OBJECTS     USED  %USED  MAX AVAIL
.mgr                          1    1  5.3 MiB        3   16 MiB      0    629 GiB
.rgw.root                     2   32  1.3 KiB        4   48 KiB      0    629 GiB
default.rgw.log               3   32  3.6 KiB      209  408 KiB      0    629 GiB
default.rgw.control           4   32      0 B        8      0 B      0    629 GiB
default.rgw.meta              5   32  1.7 KiB       10   96 KiB      0    629 GiB
default.rgw.buckets.index     7   32  5.5 MiB       22   17 MiB      0    629 GiB
default.rgw.buckets.data      8   32  807 KiB        3  2.4 MiB      0    629 GiB
default.rgw.buckets.non-ec    9   32  1.0 MiB        1  3.1 MiB      0    629 GiB
source-ecpool-86             11   32  1.2 TiB  391.13k  2.1 TiB  53.49    1.1 TiB
```

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

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

- `CLASS`: The class of OSD device.
- `SIZE`: The amount of storage capacity managed by the storage cluster.

  In the 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 GiB/3 = 30 GiB. Based on the full ratio, which is 0.85% by default, the maximum available space is 30 GiB * 0.85 = 25.5 GiB
- `AVAIL`: The amount of free space available in the storage cluster.

  In the 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 GiB/3 = 28 GiB
- `USED`: The amount of raw storage consumed by user data.

  In the example, 100 MiB is the total space available after considering the replication factor. The actual available size is 33 MiB.
- `RAW USED`: The amount of raw storage consumed by user data, internal overhead, or reserved capacity.
- `% RAW USED`: The percentage 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. This value changes based on the raw usage data based on (k+M)/K values, number of object copies, and the number of objects degraded at the time of pool stats calculation.
- **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 example, `MAX AVAIL` is 153.85 MB without considering the replication factor, which is three by default.

  See the 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.
- 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`.
- For more information, see "How Ceph calculates data usage" on page 14 and "Understanding OSD usage stats" on page 18.

Checking 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.

PREREQUISITE

Before you begin, be sure you have root-level access to the node.
A quorum must be present when multiple Ceph Monitors are running.

Check the 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 can prevent Ceph clients from reading and writing data.

- Log into the `cephadm` shell.
  For example,

  ```
  [root@host01 ~]# cephadm shell
  ```
- Display the Ceph Monitor map.
  Display the map either by running the **`ceph mon stat`** or **`ceph mon dump`** command.
  For example,

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

  or

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

– Check the quorum status for the storage cluster.

```
ceph quorum_status -f json-pretty
```

For example,

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

Understanding 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.

Reference

For more information, see:

- "How Ceph calculates data usage" on page 14
- "CRUSH weights" on page 0

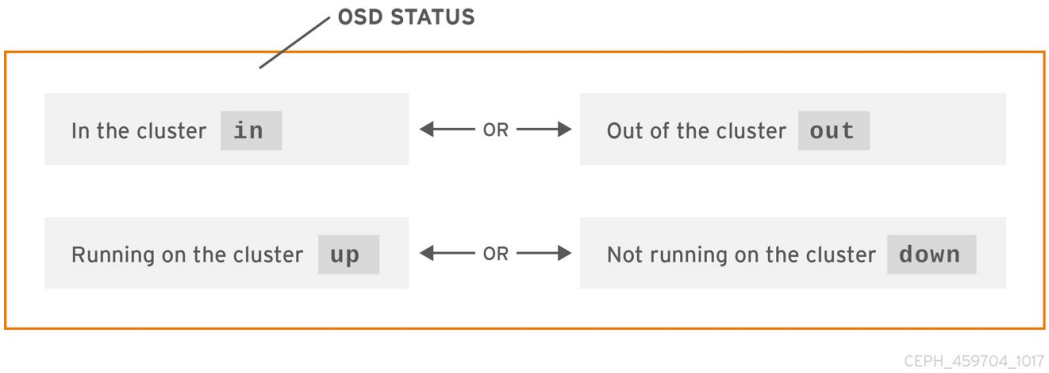
Understanding Ceph OSD status

Understand the different Ceph OSD states for the storage cluster.

A Ceph OSD's status is either `in` the storage cluster, or `out` of the storage cluster. It is either `up` and running, or it is `down` and not running. If a Ceph OSD is in an `up` state, it can 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 storage cluster and recently moved `out` of the storage cluster, Ceph starts migrating placement groups to other Ceph OSDs. If a Ceph OSD is `out` of the storage cluster, CRUSH does not assign placement groups to the Ceph OSD. If a Ceph OSD is in a `down` state, it is also `out`.

> **Note:** If a Ceph OSD is in the `down` and `in` state there is a problem and the storage cluster will not be in a healthy state.

Figure: OSD states



If you run a command such as **`ceph health`**, **`ceph -s`** or **`ceph -w`**, the storage cluster does not always echo back HEALTH OK. Not displaying the HEALTH OK output is expected in the following circumstances:

- You have not started the storage cluster yet, and it is not responding.
- You have just started or restarted the storage cluster, and it is not ready yet, because the placement groups are getting created and the Ceph OSDs are in the process of peering.
- You just added or removed a Ceph OSD.
- You just modified the storage cluster map.

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

To see whether all OSDs are running, run one of the following commands:

- `ceph osd stat`

  For example,

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

- `ceph osd dump`

  For example,

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

The command results shown as

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

where,

- *eNNNN* is the map epoch
- *x* is the total number of OSDs
- *y* is the number of OSDs that are in an `up` state
- *z* is the number OSDs that are in an `in` state

If the number of Ceph OSDs that are `in` the storage cluster are more than the number of Ceph OSDs that are `up`. Run the **`ceph osd tree`** command to identify the `ceph-osd` daemons that are not running.

```
[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 can help you troubleshoot the storage cluster by identifying the physical locations faster.

If a Ceph OSD is `down`, connect to the node and start it. Use either the Red Hat Ceph Storage Console or the command line to restart the Ceph OSD daemon.

From the command line:

```
systemctl start CEPH_OSD_SERVICE_ID
```

For example,

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

Related information

Dashboard

Checking 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.

PREREQUISITE

Before you begin, be sure you have root-level access to the node.
A quorum must be present when multiple Ceph Monitors are running.

Check the 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 can prevent Ceph clients from reading and writing data.

- Log into the `cephadm` shell.
  For example,

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

- Display the Ceph Monitor map.
  Display the map either by running the **`ceph mon stat`** or **`ceph mon dump`** command.
  For example,

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

  or

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

- Check the quorum status for the storage cluster.

  ```
  ceph quorum_status -f json-pretty
  ```

  For example,

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

Using 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 Ceph Monitors or OSDs.

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

```
Error 111: Connection Refused
```

> **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.

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.

     To view the status of a Ceph Monitor:

     Example

     ```
     [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",
    "counter dump": "dump all labeled and non-labeled counters and their values",
    "counter schema": "dump all labeled and non-labeled counters schemas",
    "dump_historic_ops": "show recent ops",
    "dump_historic_slow_ops": "show recent slow 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 non-labeled counters and their values",
    "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 non-labeled counters schemas",
    "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"
}
```

Example

```
[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": "0x3f01cfb9fffdffff",
                "release": "luminous",
                "num": 1
            }
        ],
        "osd": [
            {
                "features": "0x3f01cfb9fffdffff",
                "release": "luminous",
                "num": 3
            }
        ]
    },
    "stretch_mode": false
}
```

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

   Syntax

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

4. To view the status of a Ceph OSD named `osd.0` on the specific host:

   Example

   ```
   [ceph: root@host01 /]# ceph daemon /var/run/ceph/ceph-osd.0.asok status
   {
       "cluster_fsid": "9029b252-1668-11ee-9399-001a4a000429",
       "osd_fsid": "1de9b064-b7a5-4c54-9395-02ccda637d21",
       "whoami": 0,
       "state": "active",
       "oldest_map": 1,
       "newest_map": 58,
       "num_pgs": 33
   }
   ```

   Note: You can use `help` instead of `status` for the various options that are available for the specific daemon.

5. List all socket files for the Ceph processes:

   Example

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

Reference

- For more information, see “Identifying problems” on page 0.

Track the data availability score of a cluster (Technology Preview)

Ceph tracks the data availability of each pool and calculates a score that represents long-term data accessibility for operational assessment.

Important: Technology Preview features are not supported with production service level agreements (SLAs), might not be functionally complete, and does not recommend using them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

Ceph measures pool reliability over time by tracking when a pool is available or unavailable. From these measurements, Ceph calculates a pool's availability score, which reflects how consistently the pool has remained accessible.

The score is based on the following formula:

```
availability_score = MTBF / (MTBF + MTTR)
```

A score of 1 indicates continuous data availability, while lower values indicate interruptions.

Use data availability scoring to:

- Understand long-term reliability trends for each pool.
- Correlate incidents with data availability impact for more accurate root-cause analysis.
- Control noise in monitoring by adjusting update frequency or pausing tracking during planned work.
- Reset baselines after structural changes to a pool (for example, new CRUSH mappings).

How Ceph determines pool availability

Available

All placement groups (PGs) in the pool are active and no unfound objects exist.

Unavailable

At least one PG becomes inactive or an unfound object appears.

How uptime and downtime update

Ceph tracks how long each pool remains available or unavailable. At each interval, the system compares the pool's previous state to its current state and updates timers accordingly.

Ceph recalculates MTBF, MTTR, and the data availability score using the updated values, as detailed in "Pool state transition rules" on page 27.

Pool state transition rules

Previous state	Current state	Uptime update	Downtime update
Available	Available	+ diff time	No update
Available	Unavailable	+ diff time	No update
Unavailable	Available	+ diff time	No update
Unavailable	Unavailable	No update	+ diff time

Enable or disable data availability tracking

Tracking is disabled by default. To enable data availability, run the following command:

```
ceph config set mon enable_availability_tracking true
```

- To start or resume data availability tracking, set **`enable_availability_tracking`** to `true`.
- To disable data availability tracking, set **`enable_availability_tracking`** to `false`. While disabled, the last calculated score is preserved.

View pool availability status

Run the following command to display data availability information for each pool:

```
ceph osd pool availability-status
```

Use JSON output to view exact time values:

```
ceph osd pool availability-status --format json-pretty
```

For example,

```
POOL         UPTIME  DOWNTIME  NUMFAILURES  MTBF  MTTR  SCORE     AVAILABLE
rbd             2m     21s           1        2m   21s  0.888889     1
.mgr             86s     0s           0        0s   0s        1       1
cephfs.a.meta    77s     0s           0        0s   0s        1       1
cephfs.a.data    76s     0s           0        0s   0s        1       1
```

Update interval control
Ceph updates data availability metrics every second by default. You can modify the update interval with the following command:

```
ceph config set mon pool_availability_update_interval 2
```

> **Note:** You cannot set the update interval lower than the value of **`paxos_propose_interval`**.

Transient state changes that occur and resolve between update cycles are not recorded.

Clear a pool's availability status
To remove historical availability data for a pool, run the following command:

```
ceph osd pool clear-availability-status pool-name
```

> **Note:** You cannot clear a pool’s data availability status when tracking is disabled.

Low-level monitoring

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.

Monitoring placement group sets

Learn and understand about monitoring placement group sets.

When CRUSH assigns placement groups to Ceph OSDs, it looks at the number of replicas for the pool and assigns the placement group to Ceph OSDs such that each replica of the placement group gets assigned to a different Ceph 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 Ceph OSDs in a large cluster. We refer to the set of Ceph 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, do not panic. Common examples include:

- You added or removed an OSD. Then, CRUSH reassigned the placement group to other Ceph OSDs, thereby changing the composition of the acting set and spawning the migration of data with a "backfill" process.
- A Ceph OSD was `down`, was restarted and is now `recovering`.
- A Ceph OSD in the acting set is `down` or unable to service requests, and another Ceph OSD has temporarily assumed its duties.

Ceph processes a client request using the **Up Set**, which is the set of Ceph OSDs that actually handle the requests. In most cases, the up set and the Acting Set are virtually identical. When they are not, it can indicate that Ceph is migrating data, a Ceph 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 Ceph 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
   ```

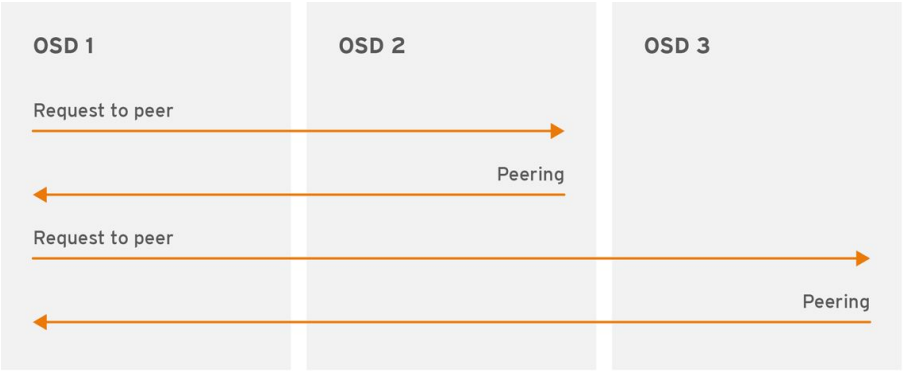
   > **Note:** If the Up Set and Acting Set do not match, this may be an indicator that the storage cluster rebalancing itself or of a potential problem with the storage cluster.

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.

"Figure: Peering" on page 29 is an example that assumes a pool with three replicas of the PG.

Figure: Peering



Placement group states

Learn about and understand the different placement group (PG) states.

When running commands 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 does **NOT** echo `HEALTH OK` in a number of placement group peering-related circumstances:

- You have just created a pool and placement groups have not 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 does not have enough storage capacity to complete backfilling operations.

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

To see the status of all placement groups, run:

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 `cc` 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": "2022-10-17T01:32:03.763988+0000",
              "last_deep_scrub": "0'0",
              "last_deep_scrub_stamp": "2022-10-17T01:32:03.763988+0000",
              "last_clean_scrub_stamp": "2022-10-17T01:32:03.763988+0000",
              "prior_readable_until_ub": 0
          },
          "stats": {
              "version": "0'0",
              "reported_seq": "2989",
              "reported_epoch": "2449",
              "state": "active+clean",
  ```

```
            "last_fresh": "2022-10-18T05:16:59.401080+0000",
            "last_change": "2022-10-17T01:32:03.764162+0000",
            "last_active": "2022-10-18T05:16:59.401080+0000",
....
```

Reference

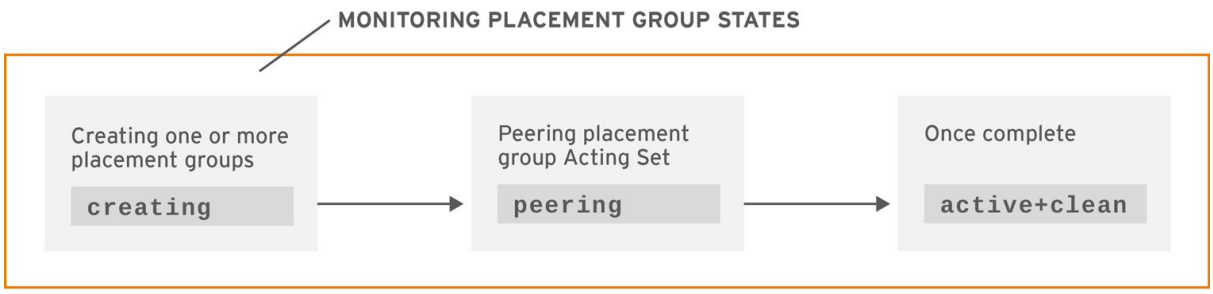
- For more details about the snapshot trimming settings, see Object Storage Daemon (OSD) configuration options.

Placement group creating state

When you create a pool, it creates the number of placement groups you specified.

.Ceph echos `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 peers. Once peering is complete, the placement group status should be `active+clean`, which means a Ceph client can begin writing to the placement group.

Figure: Creating PGs



CEPH_459704_1017

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.

Completion of the peering process does **NOT** mean that each replica has the latest contents.

Authoritative History

Ceph does **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.

Placement group active state

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.

Once Ceph completes the peering process, a placement group may become `active`.

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.

The `clean` state indicates that Ceph replicated all objects in the placement group the correct number of times.

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 acknowledgment 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 Ceph OSDs must peer again when the Ceph 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.

For example, if there are nine OSDs in a three way replica pool. If OSD number 9 goes down, the PGs assigned to OSD 9 goes into a degraded state. If OSD 9 does not recover, it goes out of the storage cluster and the storage cluster rebalances. In that scenario, the PGs are degraded and then recover to an active state.

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 is not always trivial, because a hardware failure might cause a cascading failure of multiple Ceph 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 storage 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 Ceph OSDs can fail, restart and re-peer at staggered rates. The `osd recovery max active` setting limits the number of recovery requests a Ceph OSD works on simultaneously to prevent the Ceph OSD from failing to serve. The `osd recovery max chunk` setting limits the size of the recovered data chunks to prevent network congestion.

Back fill state

When a new Ceph OSD joins the storage cluster, CRUSH will reassign placement groups from OSDs in the cluster to the newly added Ceph OSD. Forcing the new OSD to accept the reassigned placement groups immediately can put excessive load on the new Ceph 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 might 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 can be considered `incomplete`.

Ceph provides a number of settings to manage the load spike associated with reassigning placement groups to a Ceph OSD, especially a new Ceph OSD. By default, `osd_max_backfills` sets the maximum number of concurrent backfills to or from a Ceph OSD to 10. The `osd backfill full ratio` enables a Ceph 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. You can force a backfill rather than a recovery by setting the `osd_min_pg_log_entries` option to `1`, and setting the `osd_max_pg_log_entries` option to `2`. Contact your Red Hat Support account team for details on when this situation is appropriate for your workload.

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.

Placement group stale state

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`.

While Ceph uses heartbeats to ensure that hosts and daemons are running, the `ceph-osd` daemons may also get into a `stuck` state where they are not reporting statistics in a timely manner. For example, a temporary network fault. By default, OSD daemons report their placement group, up through boot and failure statistics every half a second, that is, `0.5`, which is more frequent than the heartbeat thresholds.

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 a while, seeing placement groups in the `stale` state indicates that the primary OSD for those placement groups is `down` or not reporting placement group statistics to the monitor.

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 three copies.

Syntax

```
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.

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.

For example, OSD 1, 2, and 3 are the acting OSD set and it switches to OSD 1, 4, and 3, then `osd.1` requests 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` is the only one which might not have fully backfilled all the data. At this time, the PG goes `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 monitors have not heard anything on that PG since the acting set changed. The reason being there are no more OSDs to notify the new OSDs.

Identifying stuck placement groups

A placement group is not necessarily problematic just because it is not in a `active+clean` state. Generally, Ceph's ability to self repair might not be working when placement groups get stuck.

PREREQUISITE

Before you begin, make sure that you have root-level access to the node.
The stuck states include `unclean`, `inactive`, and `stale`.

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.

- Identify the stuck placement group by running the **`pg dump_stuck`** command.

```
ceph pg dump_stuck {inactive|unclean|stale|undersized|degraded [inactive|unclean|stale|
undersized|degraded...]} {INT}
```

For example,

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

Finding 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

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
```

Monitor a Ceph cluster with the Dashboard

As a storage administrator, you can use Red Hat Ceph Storage Dashboard to monitor specific aspects of the cluster based on types of hosts, services, data access methods, and more.

Monitor a Ceph cluster with the Dashboard

As a storage administrator, you can use Red Hat Ceph Storage Dashboard to monitor specific aspects of the cluster based on types of hosts, services, data access methods, and more.

Monitoring hosts of the Ceph cluster

Monitor the hosts of the cluster on the Red Hat Ceph Storage Dashboard.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Dashboard is installed.
- Hosts are added to the storage cluster.
- All the services, monitor, manager and OSD daemons are deployed on the storage cluster.

The following are the different tabs on the hosts page. Each tab contains a table with the relavent information. The tables are searchable and customizable by column and row.

To change the order of the columns, select the column name and drag to place within the table.

To select which columns are displaying, click the **toggle columns** button and select or clear column names. Enter the number of rows to be displayed in the row selector field.

Devices

This tab has a table that details the device ID, state of the device health, life expectancy, device name, prediction creation date, and the daemons on the hosts.

Physical Disks

This tab has a table that details all disks attached to a selected host, as well as their type, size and others. It has details such as device path, type of device, available, vendor, model, size, and the OSDs deployed.

To identify which disk is where on the physical device, select the device and click **Identify**. Select the duration of how long the LED should blink for to find the selected disk.

Daemons

This tab has a table that details all services that have been deployed on the selected host, which container they are running in, and their current status.
The table has details such as daemon name, daemon version, status, when the daemon was last refreshed, CPU usage, memory usage (in MiB), and daemon events.

Daemon actions can be run from this tab. For more details, see “Running Ceph services” on page 0.

Performance details

This tab has details such as OSDs deployed, CPU utilization, RAM usage, network load, network drop rate, and OSD disk performance statistics. View performance information through the embedded Grafana Dashboard.

Device health

For SMART-enabled devices, you can get the individual health status and SMART data only on the OSD deployed hosts.

1. From the dashboard navigation, go to **Cluster › Hosts**.
2. On the **Hosts List** tab, expand the host row and select the host with the daemon to perform the action on.
3. On the **Daemons** tab of the host, select the row with the daemon.

   > **Note:** The **Daemons** table can be searched and filtered.

4. Select the action that needs to be run on the daemon. The options are **Start**, **Stop**, **Restart**, and **Redeploy**.

Viewing and editing the configuration of the Ceph cluster

View various configuration options of the Ceph cluster on the dashboard. You can edit only some configuration options.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Dashboard is installed.
- All the services are deployed on the storage cluster.

1. From the dashboard navigation, go to **Administration › Configuration**.
2. View the details of the configuration by expanding the row contents.
3. Optional: Use the search field to find a configuration.
4. Optional: You can filter for a specific configuration.
   To filter, use the following filters:
   - *Level* - Basic, advanced or dev
   - *Service* - Any, mon, mgr, osd, mds, common, mds_client, rgw, and similar filters.
   - *Source* - Any, mon, and similar filters
   - *Modified* - yes or no
5. Edit a configuration by selecting the configuration row and click **Edit**.
   a. Use the **Edit** form to edit the required parameters, and click **Update**.
   A notification displays that the configuration was updated successfully.

Viewing and editing the manager modules of the Ceph cluster

Manager modules are used to manage module-specific configuration settings. For example, you can enable alerts for the health of the cluster. You can view, enable or disable, and edit the manager modules of a cluster on the Red Hat Ceph Storage dashboard.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Dashboard is installed.

1. From the dashboard navigation, go to **Administration › Manager Modules**.
2. View the details of a specific manager module by expanding the row contents.

Enabling a manger module

Select the row and click **Enable** from the action drop-down.

Disabling a manger module

Select the row and click **Disable** from the action drop-down.

Editing a manger module

1. Select the row and click **Edit**.

   > **Note:** Not all modules have configurable parameters. If a module is not configurable, the **Edit** button is disabled.

2. Edit the required parameters and click **Update**.

A notification displays that the module was updated successfully.

Monitoring monitors of the Ceph cluster

Monitor the performance of the Ceph monitors on the landing page of the Red Hat Ceph Storage dashboard. You can also view the details such as status, quorum, number of open session, and performance counters of the monitors in the **Monitors** panel.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Dashboard is installed.
- Monitors are deployed in the storage cluster.

For more information about monitoring Ceph clusters, see:

- Ceph monitors.
- Ceph performance counters.

1. From the dashboard navigation, go to **Cluster › Monitors**.
   The **Monitors** panel displays information about the overall monitor status and monitor hosts that are in and out of quorum.
2. To see the number of open sessions, hover the cursor over the **Open Sessions**.
3. To see performance counters for any monitor, click **Name** in the **In Quorum** and **Not In Quorum** tables.

Monitoring services of the Ceph cluster

Monitor the services of the cluster on the Red Hat Ceph Storage Dashboard. You can view the details such as hostname, daemon type, daemon ID, container ID, container image name, container image ID, version status, and last refreshed time.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Dashboard is installed.
- Hosts are added to the storage cluster.
- All the services are deployed on the storage cluster.

1. From the dashboard navigation, go to **Administration › Services**.
2. To view the details of a specific service, click the **Expand/Collapse** icon on it's row.

Monitoring Ceph OSDs

Monitor the status of the Ceph OSDs on the landing page of the Red Hat Ceph Storage Dashboard. You can also view the details such as host, status, device class, number of placement groups (PGs), size flags, usage, and read or write operations time in the *OSDs* tab.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Dashboard is installed.
- Hosts are added to the storage cluster.
- All the services including OSDs are deployed on the storage cluster.

The following are the different tabs on the OSDs page:

- **Devices** - This tab has details such as Device ID, state of health, life expectancy, device name, and the daemons on the hosts.
- **Attributes (OSD map)** - This tab shows the cluster address, details of heartbeat, OSD state, and the other OSD attributes.
- **Metadata** - This tab shows the details of the OSD object store, the devices, the operating system, and the kernel details.
- **Device health** - For SMART-enabled devices, you can get the individual health status and SMART data.
- **Performance counter** - This tab gives details of the bytes written on the devices.
- **Performance Details** - This tab has details such as OSDs deployed, CPU utilization, RAM usage, network load, network drop rate, and OSD disk performance statistics.

1. From the dashboard navigation, go to **Cluster › OSDs**.
2. To view more information about a specific service, expand the OSD row.

Monitoring HAProxy

The Ceph Object Gateway allows you to assign many instances of the object gateway to a single zone, so that you can scale out as load increases. Since each object gateway instance has its own IP address, you can use HAProxy to balance the load across Ceph Object Gateway servers.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Admin-level access on the storage dashboard.
- An existing Ceph Object Gateway service, without SSL. If you want SSL service, the certificate should be configured on the ingress service, not the Ceph Object Gateway service.
- Ingress service deployed using the Ceph Orchestrator.
- Monitoring stack components are created on the dashboard.

You can monitor the following HAProxy metrics:

- Total responses by HTTP code.
- Total requests/responses.
- Total number of connections.
- Current total number of incoming / outgoing bytes.

You can also get the Grafana details by running the **`ceph dashboard get-grafana-api-url`** command.

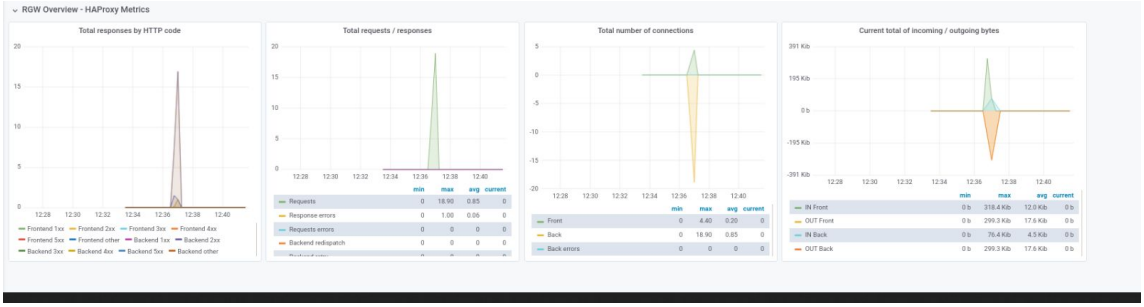
1. Log in to the Grafana URL and select **RGW Overview** panel.

   ```
   https://DASHBOARD_URL:3000
   ```

   For example,

   ```
   https://dashboard_url:3000
   ```

2. Verify the HAProxy metrics on the Grafana URL.
3. From the Ceph dashboard navigation, go to **Object › Gateways**.
4. From the **Overall Performance** tab, verify the Ceph Object Gateway HAProxy metrics.
   Figure: Viewing the Ceph Object Gateway HAProxy metrics



Related information

Configuring high availability for the Ceph Object Gateway

Viewing the CRUSH map of the Ceph cluster

View the The CRUSH map that contains a list of OSDs and related information on the Red Hat Ceph Storage dashboard. Together, the CRUSH map and CRUSH algorithm determine how and where data is stored. The dashboard allows you to view different aspects of the CRUSH map, including OSD hosts, OSD daemons, ID numbers, device class, and more.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Ceph Dashboard is installed.
- OSD daemons deployed on the storage cluster.

The CRUSH map allows you to determine which host a specific OSD ID is running on. This is helpful if there is an issue with an OSD.

1. From the dashboard navigation, go to **Cluster › CRUSH map**.
2. To view CRUSH map information, select a CRUSH map from the **CRUSH map viewer**.

Filtering logs of the Ceph cluster

View and filter logs of the Red Hat Ceph Storage cluster based on several criteria. The criteria includes *Priority*, *Keyword*, *Date*, and *Time range*. For further analysis, the logs are available for download to the system or copying to the clipboard.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- The Ceph Dashboard is installed.
- Log entries have been generated since the Ceph Monitor was last started.

> **Note:** The Dashboard logging feature only displays the thirty latest high level events. The events are stored in memory by the Ceph Monitor. The entries disappear after restarting the Monitor. If you need to review detailed or older logs, refer to the file based logs.

For more information about logs, see

- Configuring Logging.
- Understanding Ceph Logs.

1. From the dashboard navigation, go to **Observability › Logs**.

2. View and filter logs.
   The logs are divided up by type in different tabs: **Cluster Logs**, **Audit Logs**, and **Daemon Logs**.
   - To filter by priority, click the **Priority** drop-down menu and select either *Debug*, *Info*, *Warning*, *Error*, or *All*.
   - To filter by keyword, enter text into the **Keyword** field.
   - To filter by date, click the **Date** field and either use the date picker to select a date from the menu, or enter a date in the form of `YYYY-MM-DD`.
   - To filter by time, enter a range in the **Time range** fields using the `HH:MM - HH:MM` format. Hours must be entered using numbers `0` to `23`.
   - To combine filters, set two or more filters.
3. Download or copy the logs.
   - Download, by clicking the **Download** icon.
   - Copy, by clicking the **Copy to Clipboard** icon.

Viewing centralized logs of the Ceph cluster

Use the Ceph Dashboard to view logs from all the clients in a centralized space in the Red Hat Ceph Storage cluster. Centralized logs provide efficient monitoring by using Loki and alloy.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Dashboard is installed.
- Grafana is configured and the cluster is logged in to.

Loki is a log aggregation system that is designed to store and query logs. Alloy is an agent that provides the contents of local logs to a private Grafana Loki instance.

1. From the dashboard navigation, go to **Administration › Services**.
2. From **Services**, click **Create**.
3. In the **Create Service** form, from the **Type** list, select **loki**.
4. Complete the rest of the form.
5. To save the service, click **Create Service**.
6. Repeat steps “2” on page 41 through “5” on page 41 to create the `alloy` service.
   When creating the `alloy` service, select **alloy** from the **Type** list, instead of **loki**.

   > **Note:** By default, the alloy service is deployed on all running hosts.

   To see whether the services were created successfully, verify that **loki** and **alloy** services are running in the services list.
7. Enable logging to the file.
   a. Go to **Administration › Configuration**.
   b. From the search bar, search for `log_to_file`.

      > **Note:** If the file does not show in the search results, clear any filter settings that might be selected.

   c. Select `log_to_file` and click **Edit**.
      The **Edit log_to_file** form opens.
   d. In the **Values** section of the form, from the **global** list, select **true**.

e. Click **Update**.

8. The **Updated config option log_to_file** notification displays and you are returned to the **Configuration** table.
9. Repeat step "7" on page 41 to configure the **global** value to **true** for the `mon_cluster_log_to_file` file. Search for `mon_cluster_log_to_file` instead of `log_to_file`.

> **Important:** Both the `log_to_file` and `mon_cluster_log_to_file` files need to be configured.

10. Optional: To view the Ceph Object Gateway *ops_log*, `rgw_enable_ops_log` must be set to `true` by using the following command:

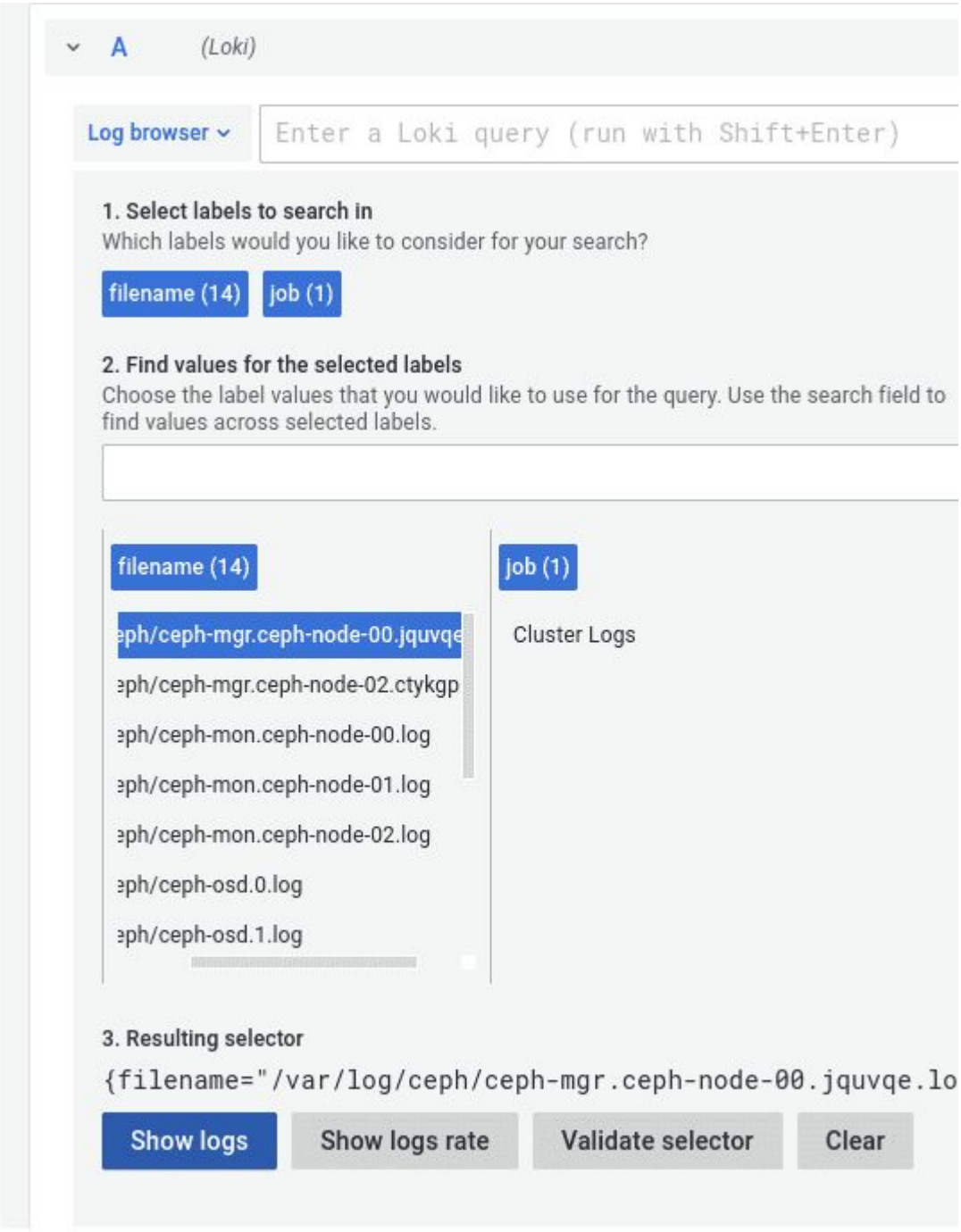
```
$ ceph config set client.rgw rgw_enable_ops_log true
```

a. Go to **Administration** -> **Configuration**.
b. Change level from *basic* to *Dev*.
c. Search for `rgw_enable_ops_log` and edit the value to `true`.
d. From the **Daemon Logs** tab, locate the logs file in the **filename** field and run the query to view the ops log.

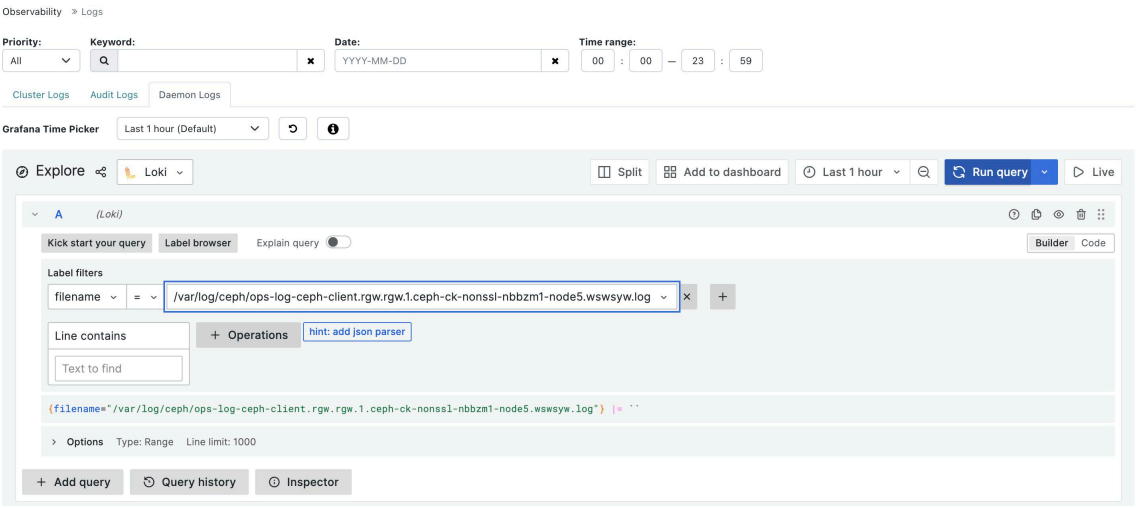
11. View the centralized logs by going to **Cluster › Logs** and select the **Daemon Logs** tab.
The **Logs** view is displayed.

> **Note:** If the **Daemon Logs** tab has a general **Welcome to Grafana** page and does not display the **Logs** view, sign in to Grafana and reload the page.

12. Use the **Log browser** field to select the relevant file and label values.
"Figure: Using the Log browser" on page 42 shows an example of selecting the relevant labels and values for the search query.
*Figure: Using the **Log browser***



13. Click **Show logs** to view the logs from the selected file.
"Figure: Viewing centralized logs" on page 43 shows the **Daemon Logs** view after a log query has been set.
Figure: Viewing centralized logs



Viewing and managing dashboard data

Learn how to use the Red Hat Ceph Storage Dashboard to view cluster health, explore system data, and manage alerts.

Before you begin, make sure that you have the following prerequisites in place:

- Ensure that you can access the Red Hat Ceph Storage Dashboard. The Dashboard module is enabled by default in Red Hat-supported installations. No additional configuration is required.
- Sign in with username and password generated during bootstrap.

You can monitor cluster data, review system health, and manage alerts directly from the Dashboard interface.

The dashboard displays temporary loading indicators or empty states while data loads. Large tables are paginated to improve loading time and navigation. The **Status card** card summarizes cluster health and displays active alerts. You can view and verify alert states directly from the **Status card** card.

- To view cluster status and alerts, open the **Status** card on the main dashboard.
- To navigate large tables, open a page that lists multiple records, such as **OSDs** or **Hosts**. Use the pagination controls to move between pages or change the number of rows displayed. Enter search or filter criteria, and wait for the table to refresh.
- To view cluster status and alerts, open the **Status** card on the main dashboard. The **Status** card shows summarized cluster health and active alerts.
- Click **View alerts** to open the **Active Alerts** page.
  Expand an alert row to view details, including the description, severity, state, and Total occurrences. To view the alert source, click **Source** in the alert row. Alerts update automatically as cluster conditions change.

Monitoring pools of the Ceph cluster

A pool plays a critical role in how the Ceph storage cluster distributes and stores data. If you have deployed a cluster without creating a pool, Ceph uses the default pools for storing data.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Dashboard is installed.
- Pools are created

1. From the dashboard navigation, go to **Cluster › Pools**.
2. View the **Pools List** tab, which gives the details of Data protection and the application for which the pool is enabled.

Hover the mouse over **Usage**, **Read bytes**, and **Write bytes** for the required details.

3. Expand the pool row for detailed information about a specific pool.

Monitoring Ceph File Systems

Use the Red Hat Ceph Storage Dashboard to monitor Ceph File Systems (CephFS) and related components.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Dashboard is installed.
- MDS service is deployed on at least one of the hosts.
- Ceph File System is installed.

For each File System listed, the following tabs are available:

Details

View the metadata servers (MDS) and their rank plus any standby daemons, pools and their usage,and performance counters.

Directories

View list of directories, their quotas and snapshots. Select a directory to set and unset maximum file and size quotas and to create and delete snapshots for the specific directory.

Subvolumes

Create, edit, and view subvolume information. These can be filtered by subvolume groups.

Subvolume groups

Create, edit, and view subvolume group information.

Snapshots

Create, clone, and view snapshot information. These can be filtered by subvolume groups and subvolumes.

Snapshot schedules

Enable, create, edit, and delete snapshot schedules.

Clients

View and evict Ceph File System client information.

Performance Details

View the performance of the file systems through the embedded Grafana Dashboard.

1. From the dashboard navigation, go to **File › File Systems**.
2. To view more information about an individual file system, expand the file system row.

Monitoring Ceph Object Gateway daemons

Use the Red Hat Ceph Storage Dashboard to monitor Ceph Object Gateway daemons. You can view the details, performance counters and performance details of the Ceph Object Gateway daemons.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Dashboard is installed.
- At least one Ceph Object Gateway daemon configured in the storage cluster.

1. From the dashboard navigation, go to **Object › Gateways**.

2. To view more information about a Ceph Object Gateway daemon, expand the Ceph Object Gateway row.
3. Optional: If you have configured multiple Ceph Object Gateway daemons, click on **Sync Performance** tab and view the multi-site performance counters.

Monitoring block device images

Use the Red Hat Ceph Storage Dashboard to monitor and manage block device images. You can view the details, snapshots, configuration details, and performance details of the images.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Dashboard is installed.
- A pool with the `rbd` application enabled is created.
- An image is created.

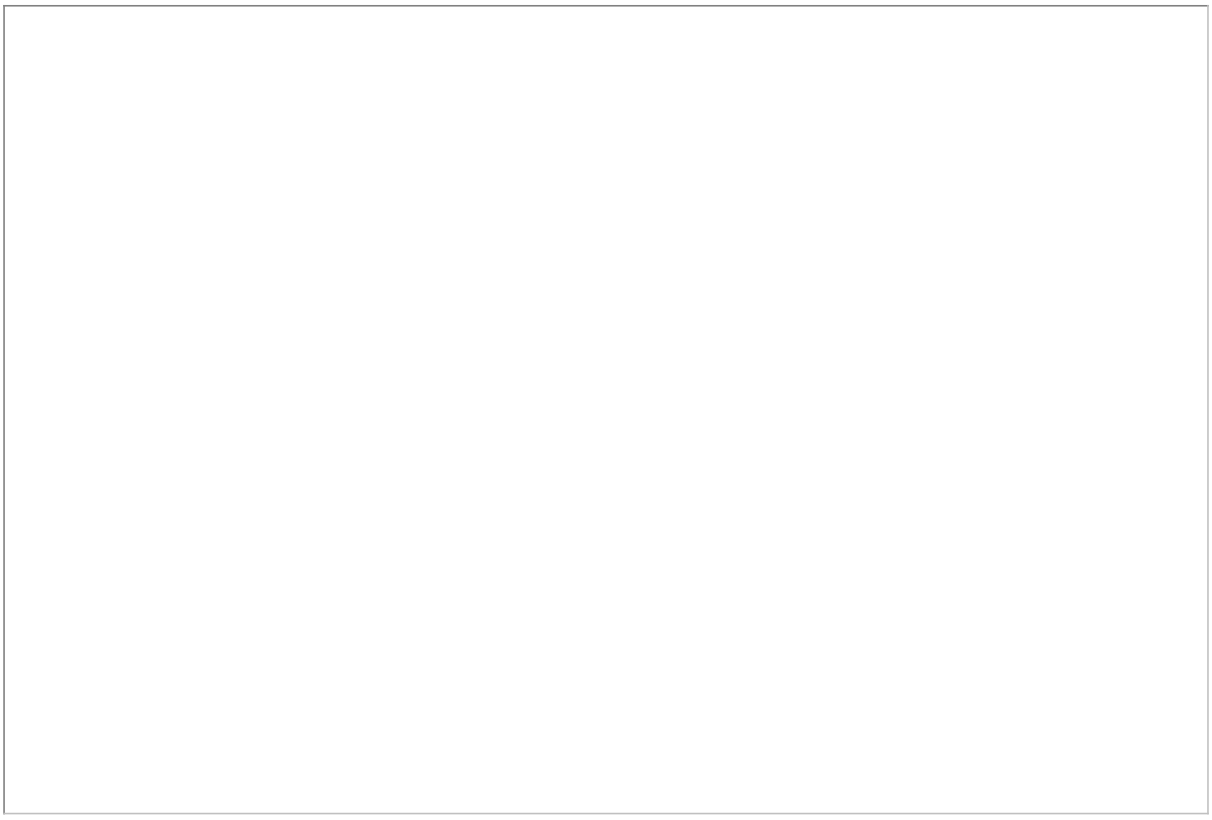
1. From the dashboard navigation, go to **Block › Images**.
2. View image information from the **Images** tab. Expand image rows for more information.
3. View namespace information from the **Namespaces** tab. Expand namespace rows for more information.

Stretch clusters for Ceph storage

As a storage administrator, you can configure a two-site stretched cluster by enabling stretch mode in Ceph.

Red Hat Ceph Storage, when configured with stretch mode on the cluster, offer the ability to expand the failure domain beyond the OSD or host level to span entire data centers or cloud availability zones, enhancing fault tolerance and availability in geographically distributed deployments.

“Figure: Stretch clusters for Ceph storage” on page 46 depicts a simplified representation of a Ceph cluster operating in stretch mode, where the tiebreaker host is provisioned in data center (DC) 3.
Figure: Stretch clusters for Ceph storage



A stretch cluster operates over a Wide Area Network (WAN), unlike a typical Ceph cluster, which operates over a Local Area Network (LAN).

For illustration purposes, a data center is chosen as the failure domain, though this could also represent a cloud availability zone. Data Center 1 (DC1) and Data Center 2 (DC2) contain OSDs and Monitors within their respective domains, while Data Center 3 (DC3) contains only a single monitor. The latency between DC1 and DC2 should not exceed 10 ms RTT, as higher latency can significantly impact Ceph performance in terms of replication, recovery, and related operations. However, DC3, a non-data site typically hosted on a virtual machine, can tolerate higher latency compared to the two data sites.

A stretch cluster, like the one depicted in "Figure: Stretch clusters for Ceph storage" on page 46, can withstand a complete data center failure or a network partition between data centers as long as at least two sites remain connected.

There are no additional steps for powering down a stretch cluster. For more information, see "Powering down and rebooting the cluster" on page 4.

Stretch mode

To improve availability in Stretched Clusters (geographically distributed deployments), you must enter the stretch mode. When stretch mode is enabled, the Ceph OSDs only take placement groups (PGs) as active when they peer across data centers, or whichever other CRUSH bucket type you specified, assuming both are active. Pools increase in size from the default three to four, with two copies on each site.

In stretch mode, Ceph OSDs are only allowed to connect to monitors within the same data center. New monitors are not allowed to join the cluster without specified location.

If all the OSDs and monitors from a data center become inaccessible at once, the surviving data center will enter a `degraded` stretch mode. This issues a warning, reduces the `min_size` to `1`, and allows the cluster to reach an `active` state with the data from the remaining site.

Stretch mode is designed to handle netsplit scenarios between two data centers and the loss of one data center. Stretch mode handles the netsplit scenario by choosing the surviving data center with a better connection to the *tiebreaker monitor*. Stretch mode handles the loss of one data center by reducing the `min_size` of all pools to `1`, allowing the cluster to continue operating with the remaining data center. When the lost data center comes back, the cluster will recover the lost data and return to normal operation.

> **Note:** In a stretch cluster, when a site goes down and the cluster enters a `degraded` state, the `min_size` of the pool may be temporarily reduced (e.g., to 1) to allow the placement groups (PGs) to become active and continue serving I/O. However, the `size` of the pool remains unchanged. The **`peering_crush_bucket_count`** stretch mode flag ensures that PGs does not become active unless they are backed by OSDs in a minimum number of distinct CRUSH buckets (e.g., different data centers). This mechanism prevents the system from creating redundant copies solely within the surviving site, ensuring that data is only fully replicated once the downed site recovers."

When the missing data center becomes accessible again, the cluster enters `recovery` stretch mode. This changes the warning and allows peering, but still requires only the OSDs from the data center, which was up the whole time.

When all PGs are in a known state and are not degraded or incomplete, the cluster goes back to the regular stretch mode, ends the warning, and restores `min_size` to its starting value `2`. The cluster again requires both sites to peer, not only the site that stayed up the whole time, therefore you can fail over to the other site, if necessary.

For troubleshooting information, see Troubleshooting clusters in stretch mode.

Stretch mode limitations

- It is not possible to exit from stretch mode once it is entered.
- You cannot use erasure-coded pools with clusters in stretch mode. You can neither enter the stretch mode with erasure-coded pools, nor create an erasure-coded pool when the stretch mode is active.
- Device class is not supported in stretch mode. In the following example, the `class hdd` is not supported.

```
rule stretch_replicated_rule
{id 2
type replicated class hdd
step take default
step choose firstn 0 type datacenter
step chooseleaf firstn 2 type host
step emit
}
```

To achieve same weights on both sites, the Ceph OSDs deployed in the two sites should be of equal size, that is, storage capacity in the first site is equivalent to storage capacity in the second site.

- While it is not enforced, you should run two Ceph monitors on each site and a tiebreaker, for a total of five. This is because OSDs can only connect to monitors in their own site when in stretch mode.
- You have to create your own CRUSH rule, which provides two copies on each site, which totals to four on both sites.
- You cannot enable stretch mode if you have existing pools with non-default size or `min_size`.
- Because the cluster runs with `min_size 1` when degraded, you should only use stretch mode with all-flash OSDs. This minimizes the time needed to recover once connectivity is restored, and minimizes the potential for data loss.

Stretch peering rule

In Ceph stretch cluster mode, a critical safeguard is enforced through the *stretch peering rule*, which ensures that a Placement Group (PG) cannot become active if all acting replicas reside within a single failure domain, such as a single data center or cloud availability zone.

This behavior is essential for protecting data integrity during site failures. If a PG were allowed to go active with all replicas confined to one site, write operations could be falsely acknowledged without true redundancy. In the event of a site outage, this would result in complete data loss for those PGs. By enforcing zone diversity in the acting set, Ceph stretch clusters maintain high availability while minimizing the risk of data inconsistency or loss.

.

Deployment requirements

This information details important hardware, software, and network requirements that are needed for deploying a generalized stretch cluster configuration for three availability zones.

Software requirements

Red Hat Ceph Storage 9

Hardware requirements

Use the following minimum hardware requirements before deploying a stretch cluster configuration.

- ceph-osd hardware requirements
- ceph-mon hardware requirements
- ceph-mds hardware requirements

ceph-osd hardware requirements	
Hardware criteria	**Minimum and recommended**
Processor	– 1 core minimum, 2 recommended. – 1 core per 200-500 MB/s throughput. – 1 core per 1000-3000 IOPS. – Results are before replication. – Results can vary across CPU and drive models and Ceph configuration (erasure coding, and compression). – ARM processors specifically can require more cores for performance. – SSD OSDs, especially NVMe, benefit from extra cores per OSD. – Actual performance depends on various factors, including drives, network, and client throughput and latency. Bench marking is recommended to assess performance accurately.
RAM	– 4 GB or more per daemon is required (higher is recommended). – 2-4 GB can work, but performance might be slower. – Less than 2 GB is not recommended for optimal performance.
Network	A single 1 Gb/s (bonded 10+ Gb/s recommended). The network latency between the two data sites should be less than 10ms.

ceph-mon hardware requirements	
Hardware criteria	**Minimum and recommended**
Processor	2 cores minimum
Storage drives	100 GB per daemon. SSD is recommended.
Network	A single 1 Gb/s (10+ Gb/s recommended)

ceph-mds hardware requirements	
Hardware criteria	**Minimum and recommended**
Processor	2 cores minimum
RAM	2 GB per daemon (more for production)
Disk space	1 GB per daemon
Network	A single 1 Gb/s (10+ Gb/s recommended)

Daemon placement

"Daemon placement" on page 49 lists the daemon placement details across various hosts and data centers.

Daemon placement		
Hostname	**Data center**	**Services**
host01	DC1	OSD+MON+MGR

Hostname	Data center	Services
host02	DC1	OSD+MON+MGR
host03	DC1	OSD+MDS+RGW
host04	DC2	OSD+MON+MGR
host05	DC2	OSD+MON+MGR
host06	DC2	OSD+MDS+RGW
host07	DC3 (Tiebreaker)	MON

Network configuration requirements

Use the following network configuration requirements before deploying stretch cluster configuration.

> **Note:** You can use different subnets for each of the data centers.

- Have two separate networks, one public network and one cluster network.
- The latencies between data centers that run the Red Hat Ceph Storage Object Storage Devices (OSDs) cannot exceed 10 ms RTT.

The following is an example of a basic network configuration:

- DC1
  Ceph public/private network: 10.0.40.0/24
- DC2
  Ceph public/private network: 10.0.40.0/24
- Tiebreaker
  Ceph public/private network: 10.0.40.0/24

Cluster setup requirements

Ensure that the hostname is configured by using the bare or short hostname in all hosts.

```
hostnamectl set-hostname SHORT_NAME
```

> **Important:** The **hostname** command should only return the short hostname, when run on all nodes. If the FQDN is returned, the cluster configuration will not be successful.

Configuring a cluster for stretch mode

As a storage administrator, you can configure stretch mode in Ceph to distribute your cluster across multiple data centers, ensuring high availability and fault tolerance.

The following is an example user case for using the stretch mode feature.

A global e-commerce company with multiple regional data centers can implement stretch mode to ensure their Ceph storage cluster remains highly available and resilient. In the event of a failure in one data center, stretch mode ensures that the cluster continues to operate by redistributing workloads to other active sites, minimizing downtime and preventing data loss, thereby maintaining continuous service for customers worldwide.

Setting CRUSH location for daemons

Before you enter the stretch mode, you need to prepare the cluster by setting the CRUSH location to the daemons in the cluster. Add details of host locations in crush map for mon daemons & OSD hosts. This can either be done with the cluster deployment with spec file, or can be added on existing clusters as well.

Prepare the stretch mode in one of the following ways:

- Bootstrap the cluster through a service configuration file, where the locations are added to the hosts as part of deployment.

– Set the locations manually through **ceph osd crush add-bucket** and **ceph osd crush move** commands after the cluster is deployed.

Setting the CRUSH location during bootstrap

For hosts with multiple public networks, update the public networks in CIDR format, within the configuration file. Enhance the specification file by setting the monitor CRUSH location. Set this information during the bootstrap procedure.

PREREQUISITE

Before you begin, be sure that you have root-level access to the nodes.

For more information about Ceph bootstrapping and different **cephadm bootstrap** command options, see Bootstrapping a new storage cluster.

1. Create a `cluster-spec.yaml` file.
   The specification file adds the nodes to the Red Hat Ceph Storage cluster and also sets specific labels for where the services run.
   For example,

```
service_type: host
addr: <host01 address>
hostname: host01
location:
 root: default
 datacenter: DC1
labels:
 - osd
 - mon
 - mgr
---
service_type: host
addr: <host02 address>
hostname: host02
location:
 datacenter: DC1
labels:
 - osd
 - mon
 - mgr
---
service_type: host
addr: <host03 address>
hostname: host03
location:
 datacenter: DC1
labels:
 - osd
 - mds
 - rgw
---
service_type: host
addr: <host04 address>
hostname: host04
location:
 root: default
 datacenter: DC2
labels:
 - osd
 - mon
 - mgr
---
service_type: host
addr: host05
hostname: host05
location:
 datacenter: DC2
labels:
 - osd
 - mon
 - mgr
---
service_type: host
addr: <host06 address>
hostname: host06
location:
```

```
 datacenter: DC2
labels:
 - osd
 - mds
 - rgw
---
service_type: host
addr: <host07 address>
hostname: host07
labels:
 - mon
------
service_type: mon
spec:
 crush_locations:
   host01:
     - datacenter=DC1
   host02:
     - datacenter=DC1
   host04:
     - datacenter=DC2
   host05:
     - datacenter=DC2
placement:
  label: mon
---
service_type: mgr
service_name: mgr
placement:
 label: "mgr"
---
service_type: osd
service_id: all-available-devices
service_name: osd.all-available-devices
placement:
 label: "osd"
spec:
 data_devices:
   all: true
---
service_type: rgw
service_id: objectgw
service_name: rgw.objectgw
placement:
 count: 2
 label: "rgw"
spec:
 rgw_frontend_port: 8080
```

2. Run the **cephadm bootstrap** command as the root user on the node that will serve as the initial Monitor node in the cluster.
   The `MONITOR_IP_ADDRESS` value is the node's IP address that you are using to run the **cephadm bootstrap** command. Run one of the following commands, based on your configuration needs.

   – For hosts that are present on the same network.

   ```
   cephadm bootstrap --apply-spec CONFIGURATION_FILE_NAME --mon-ip
   MONITOR_IP_ADDRESS --ssh-private-key PRIVATE_KEY --ssh-public-key PUBLIC_KEY --
   registry-url REGISTRY_URL --registry-username USER_NAME --registry-password
   PASSWORD
   ```

   – For hosts that are present on different networks.

   ```
   cephadm bootstrap --apply-spec CONFIGURATION_FILE_NAME --mon-ip
   MONITOR_IP_ADDRESS --ssh-private-key PRIVATE_KEY --ssh-public-key PUBLIC_KEY --
   registry-url REGISTRY_URL --registry-username USER_NAME --registry-password
   PASSWORD --config ceph.conf
   ```

   The following configuration in the `ceph.conf` file defines the public network settings for the cluster:

   ```
   [global]
   public_network = 10.1.172.0/23,10.0.64.0/22
   ```

   After the bootstrap process completes, the following output is emitted.

```
Or, if you are only running a single cluster on this host:
    sudo /usr/sbin/cephadm shell
Please consider enabling telemetry to help improve Ceph:
    ceph telemetry on
For more information see:
    https://docs.ceph.com/en/latest/mgr/telemetry/
Bootstrap complete.
```

AFTER COMPLETING THE TASK

1. Verify the network configuration.

   ```
   ceph config dump | grep network
   ```

   In the following example, the output confirms that there are multiple networks and that `10.1.172.0/23` and `10.0.64.0/22` are properly configured.

   ```
   [ceph: root@host01 /]#  ceph config dump | grep network
   global   advanced   public_network   10.1.172.0/23,10.0.64.0/22   *
   ```

2. Verify that the status of storage cluster deployment is in the HEALTH_OK state.
   For example,

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

    cluster:
      id:     ff19789c-f5c7-11ef-8e1c-fa163e4e1f7e
      health: HEALTH_OK

    services:
      mon: 5 daemons, quorum host01,host05,host02,host07-installer,host04 (age 10m)
      mgr: host05.aswlzq(active, since 43m), standbys: host02.ctajlt, host01.napqyw,
   host04.wdglem
      mds: 1/1 daemons up, 1 standby
      osd: 24 osds: 24 up (since 31m), 24 in (since 32m)
      rgw: 4 daemons active (2 hosts, 1 zones)

    data:
      volumes: 1/1 healthy
      pools:   7 pools, 1019 pgs
      objects: 216 objects, 456 KiB
      usage:   2.7 GiB used, 357 GiB / 360 GiB avail
   ```

3. Verify that all the nodes that were added in the `cluster-spec.yaml` file are added to the cluster.

   ```
   ceph orch host ls
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph orch host ls
   HOST                                    ADDR         LABELS       STATUS
   host01            10.0.56.37   mgr,mon,osd
   host02            10.0.59.35   mgr,mon,osd
   host03            10.0.58.106  osd,mds,rgw
   host04            10.0.56.13   osd,mon,mgr
   host05            10.0.59.188  mgr,mon,osd
   host06            10.0.56.223  rgw,mds,osd
   host07            10.0.56.189  _admin,mon
   7 hosts in cluster
   ```

4. Use the **ceph osd tree** command to verify the following:
   - CRUSH locations for the OSD hosts
   - That each host has one OSD configured
   - That each host OSD is in the up state
   - That each node is in the correct data center bucket

   For example,

```
[ceph: root@host01 /]# ceph osd tree
ID   CLASS  WEIGHT   TYPE NAME               STATUS REWEIGHT PRI-AFF
-1         0.35010  root default
-3         0.17505      datacenter DC1
-2         0.05835        host host01
5    hdd    0.01459              osd.5       up     1.00000  1.00000
11   hdd    0.01459             osd.11       up     1.00000  1.00000
17   hdd    0.01459             osd.17       up     1.00000  1.00000
23   hdd    0.01459             osd.23       up     1.00000  1.00000
-4         0.05835      host host02
1    hdd    0.01459              osd.1       up     1.00000  1.00000
6    hdd    0.01459              osd.6       up     1.00000  1.00000
12   hdd    0.01459             osd.12       up     1.00000  1.00000
18   hdd    0.01459             osd.18       up     1.00000  1.00000
-5         0.05835      host host03
3    hdd    0.01459              osd.3       up     1.00000  1.00000
10   hdd    0.01459             osd.10       up     1.00000  1.00000
16   hdd    0.01459             osd.16       up     1.00000  1.00000
22   hdd    0.01459             osd.22       up     1.00000  1.00000
-7         0.17505      datacenter DC2
-6         0.05835      host host04
2    hdd    0.01459              osd.2       up     1.00000  1.00000
8    hdd    0.01459              osd.8       up     1.00000  1.00000
14   hdd    0.01459             osd.14       up     1.00000  1.00000
20   hdd    0.01459             osd.20       up     1.00000  1.00000
-8         0.05835      host host05
0    hdd    0.01459              osd.0       up     1.00000  1.00000
7    hdd    0.01459              osd.7       up     1.00000  1.00000
13   hdd    0.01459             osd.13       up     1.00000  1.00000
19   hdd    0.01459             osd.19       up     1.00000  1.00000
-9         0.05835      host host06
4    hdd    0.01459              osd.4       up     1.00000  1.00000
9    hdd    0.01459              osd.9       up     1.00000  1.00000
15   hdd    0.01459             osd.15       up     1.00000  1.00000
21   hdd    0.01459             osd.21       up     1.00000  1.00000
```

5. Verify the CRUSH locations for the MON hosts. Check the `mon map` to ensure that each MON host has a `crush_location` specified.

```
ceph mon dump
```

The output displays details about the MON map, including the `crush_location` for each host.

For example,

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

epoch 5

fsid 4158287e-169e-11f0-b1ad-fa163e98b991

last_changed 2025-04-11T06:32:20.332479+0000

created 2025-04-11T06:29:24.974553+0000

min_mon_release 19 (squid)

election_strategy: 1

0: [v2:10.0.57.33:3300/0,v1:10.0.57.33:6789/0] mon.host07

1: [v2:10.0.58.200:3300/0,v1:10.0.58.200:6789/0] mon.host05; crush_location
{datacenter=DC2}

2: [v2:10.0.58.47:3300/0,v1:10.0.58.47:6789/0] mon.host02; crush_location
{datacenter=DC1}

3: [v2:10.0.58.104:3300/0,v1:10.0.58.104:6789/0] mon.host04; crush_location
{datacenter=DC2}

4: [v2:10.0.58.38:3300/0,v1:10.0.58.38:6789/0] mon.host01; crush_location
{datacenter=DC1}
```

Manually setting the CRUSH location for daemons

Set the locations manually through **ceph osd crush add-bucket** and **ceph osd crush move** commands after the cluster is deployed.

PREREQUISITE

Before you begin, be sure that you have root-level access to the nodes.

1. Add two buckets to which you plan to set the location of your non-tiebreaker monitors to the CRUSH map. Specify the bucket type as `datacenter`.

   ```
   ceph osd crush add-bucket BUCKET_NAME datacenter
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph osd crush add-bucket DC1 datacenter
   [ceph: root@host01 /]# ceph osd crush add-bucket DC2 datacenter
   ```

2. . `root=default`.

   ```
   ceph osd crush move BUCKET_NAME root=default
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph osd crush move DC1 root=default
   [ceph: root@host01 /]# ceph osd crush move DC2 root=default
   ```

3. Move the OSD hosts, according to the required CRUSH placement.

   ```
   ceph osd crush move HOST datacenter=DATACENTER
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph osd crush move host01 datacenter=DC1
   ```

4. Verify the CRUSH locations for OSD hosts.

   ```
   ceph osd tree
   ```

   For example,

```
[ceph: root@host01 /]# ceph osd tree
ID   CLASS  WEIGHT   TYPE NAME               STATUS REWEIGHT PRI-AFF
-1         0.35010  root default
-3         0.17505      datacenter DC1
-2         0.05835        host host01
5    hdd    0.01459              osd.5       up     1.00000  1.00000
11   hdd    0.01459             osd.11       up     1.00000  1.00000
17   hdd    0.01459             osd.17       up     1.00000  1.00000
23   hdd    0.01459             osd.23       up     1.00000  1.00000
-4         0.05835        host host02
1    hdd    0.01459              osd.1       up     1.00000  1.00000
6    hdd    0.01459              osd.6       up     1.00000  1.00000
12   hdd    0.01459             osd.12       up     1.00000  1.00000
18   hdd    0.01459             osd.18       up     1.00000  1.00000
-5         0.05835        host host03
3    hdd    0.01459              osd.3       up     1.00000  1.00000
10   hdd    0.01459             osd.10       up     1.00000  1.00000
16   hdd    0.01459             osd.16       up     1.00000  1.00000
22   hdd    0.01459             osd.22       up     1.00000  1.00000
-7         0.17505      datacenter DC2
-6         0.05835        host host04
2    hdd    0.01459              osd.2       up     1.00000  1.00000
8    hdd    0.01459              osd.8       up     1.00000  1.00000
14   hdd    0.01459             osd.14       up     1.00000  1.00000
20   hdd    0.01459             osd.20       up     1.00000  1.00000
-8         0.05835        host host05
0    hdd    0.01459              osd.0       up     1.00000  1.00000
7    hdd    0.01459              osd.7       up     1.00000  1.00000
13   hdd    0.01459             osd.13       up     1.00000  1.00000
19   hdd    0.01459             osd.19       up     1.00000  1.00000
-9         0.05835        host host06
4    hdd    0.01459              osd.4       up     1.00000  1.00000
9    hdd    0.01459              osd.9       up     1.00000  1.00000
15   hdd    0.01459             osd.15       up     1.00000  1.00000
21   hdd    0.01459             osd.21       up     1.00000  1.00000
```

5. Set the location of each monitor, matching your CRUSH map.

```
ceph mon set_location HOST datacenter=DATACENTER
```

For example,

```
[ceph: root@host01 /]# ceph mon set_location host01 datacenter=DC1
[ceph: root@host01 /]# ceph mon set_location host02 datacenter=DC1
[ceph: root@host01 /]# ceph mon set_location host04 datacenter=DC2
[ceph: root@host01 /]# ceph mon set_location host05 datacenter=DC2
```

6. Verify the CRUSH locations for the MON hosts.
Check the `mon map` to ensure that each MON host has a `crush_location` specified.

```
ceph mon dump
```

The output displays details about the MON map, including the `crush_location` for each host.

For example,

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

epoch 5

fsid 4158287e-169e-11f0-b1ad-fa163e98b991

last_changed 2025-04-11T06:32:20.332479+0000

created 2025-04-11T06:29:24.974553+0000

min_mon_release 19 (squid)

election_strategy: 1

0: [v2:10.0.57.33:3300/0,v1:10.0.57.33:6789/0] mon.host07

1: [v2:10.0.58.200:3300/0,v1:10.0.58.200:6789/0] mon.host05; crush_location
{datacenter=DC2}

2: [v2:10.0.58.47:3300/0,v1:10.0.58.47:6789/0] mon.host02; crush_location
{datacenter=DC1}

3: [v2:10.0.58.104:3300/0,v1:10.0.58.104:6789/0] mon.host04; crush_location
{datacenter=DC2}

4: [v2:10.0.58.38:3300/0,v1:10.0.58.38:6789/0] mon.host01; crush_location
{datacenter=DC1}
```

Configuring a CRUSH map for stretch mode

Use this information to configure a CRUSH map for stretch mode. In this step, we modify few configs on the cluster, add new crush rule and enable stretch mode on the cluster.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- Root-level access to the nodes.
- The CRUSH location is set to the hosts.

1. Create a CRUSH rule that makes use of this OSD crush topology by installing the ceph-base RPM package in order to use the **`crushtool`** command.

   ```
   dnf -y install ceph-base
   ```

2. Get the compiled CRUSH map from the cluster.

   ```
   ceph osd getcrushmap > /etc/ceph/crushmap.bin
   ```

3. Decompile the CRUSH map and convert it to a text file to edit it.

   ```
   crushtool -d /etc/ceph/crushmap.bin -o /etc/ceph/crushmap.txt
   ```

4. Add the following rule to the CRUSH map by editing the `/etc/ceph/crushmap.txt` at the end of the file.
   This rule distributes reads and writes evenly across the data center.

   ```
   rule stretch_rule {
           id 1
           type replicated
           step take default
           step choose firstn 0 type datacenter
           step chooseleaf firstn 2 type host
           step emit
    }
   ```

   a. Optionally have the cluster with a read/write affinity towards data center 1.

      ```
      rule stretch_rule {
              id 1
              type replicated
      ```

```
        step take DC1
        step chooseleaf firstn 2 type host
        step emit
        step take DC2
        step chooseleaf firstn 2 type host
        step emit
}
```

```
The CRUSH rule declared contains the following information:
     Rule name
          Description: A unique name for identifying the rule.
          Value: stretch_rule
     id
          Description: A unique whole number for identifying the rule.
          Value: 1
     type
          Description: Describes a rule for either a storage drive replicated or
erasure-coded.
          Value: replicated
     step take default
          Description: Takes the root bucket called default, and begins iterating
down the tree.
     step take DC1
          Description: Takes the bucket called DC1, and begins iterating down the
tree.
     step choose firstn 0 type datacenter
          Description: Selects the datacenter bucket, and goes into its subtrees.
     step chooseleaf firstn 2 type host
          Description: Selects the number of buckets of the given type. In this
case, it is two different hosts located in the datacenter it entered at the
previous level.
     step emit
          Description: Outputs the current value and empties the stack. Typically
used at the end of a rule, but may also be used to pick from different trees in
the same rule.
```

5. Compile the new CRUSH map from `/etc/ceph/crushmap.txt` and convert it to a binary file `/etc/ceph/crushmap2.bin`.

   ```
   crushtool -c /path/to/crushmap.txt -o /path/to/crushmap2.bin
   ```

   For example,

   ```
   [ceph: root@host01 /]# crushtool -c /etc/ceph/crushmap.txt -o /etc/ceph/crushmap2.bin
   ```

6. Inject the newly created CRUSH map back into the cluster.

   ```
   ceph osd setcrushmap -i /path/to/compiled_crushmap
   ```

   For example,

   ```
   ceph osd setcrushmap -i /etc/ceph/crushmap2.bin
   ```

   > **Note:** The number 17 is a counter and increases (18,19, and so on) depending on the changes that are made to the CRUSH map.

AFTER COMPLETING THE TASK

Verify that the newly created `stretch_rule` available for use.

```
ceph osd crush rule ls
```

For example,

```
[ceph: root@host01 /]# ceph osd crush rule ls

replicated_rule
stretch_rule
```

Changing stretch mode

Change the stretch mode state by entering or exiting stretch mode as needed to support your cluster's availability and data-placement requirements.

Entering stretch mode

Stretch mode is designed to handle two sites. There is a lesser risk of component availability outages with 2-site clusters.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- Root-level access to the nodes.
- The CRUSH location is set to the hosts.
- The CRUSH map configured to include stretch rule.
- No erasure coded pools in the cluster.
- Weights of the two sites are the same.

1. Check the current election strategy being used by the monitors.

```
ceph mon dump | grep election_strategy
```

> **Note:** The Ceph cluster `election_strategy` is set to `1`, by default.

For example,

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

dumped monmap epoch 9
election_strategy: 1
```

2. Change the election strategy to `connectivity`.

```
ceph mon set election_strategy connectivity
```

For more information about configuring the election strategy, see Configuring monitor election strategy.

3. Use the **ceph mon dump** command to verify that the election strategy was updated to `3`.
For example,

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

dumped monmap epoch 22
election_strategy: 3
```

4. Set the location of the tiebreaker monitor so that it is split across the data centers.

```
ceph mon set_location TIEBREAKER_HOST datacenter=DC3
```

For example,

```
[ceph: root@host01 /]# ceph mon set_location host07 datacenter=DC3
```

5. Verify that the tiebreaker monitor is set as expected.

```
ceph mon dump
```

For example,

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

epoch 8

fsid 4158287e-169e-11f0-b1ad-fa163e98b991

last_changed 2025-04-11T07:14:48.652801+0000

created 2025-04-11T06:29:24.974553+0000

min_mon_release 19 (squid)

election_strategy: 3

0: [v2:10.0.57.33:3300/0,v1:10.0.57.33:6789/0] mon.host07; crush_location
{datacenter=DC3}

1: [v2:10.0.58.200:3300/0,v1:10.0.58.200:6789/0] mon.host05; crush_location
{datacenter=DC2}

2: [v2:10.0.58.47:3300/0,v1:10.0.58.47:6789/0] mon.host02; crush_location
{datacenter=DC1}

3: [v2:10.0.58.104:3300/0,v1:10.0.58.104:6789/0] mon.host04; crush_location
{datacenter=DC2}

4: [v2:10.0.58.38:3300/0,v1:10.0.58.38:6789/0] mon.host01; crush_location
{datacenter=DC1}

dumped monmap epoch 8
0
```

6. Enter stretch mode.

```
ceph mon enable_stretch_mode TIEBREAKER_HOST STRETCH_RULE STRETCH_BUCKET
```

In the following example:

- The tiebreaker node is set as `host07`.
- The stretch rule is `stretch_rule`, as created in "Configuring a CRUSH map for stretch mode" on page 57.
- The stretch bucket is set as `datacenter`.

```
[ceph: root@host01 /]# ceph mon enable_stretch_mode host07 stretch_rule datacenter
```

AFTER COMPLETING THE TASK

Verify that stretch mode was implemented correctly by continuing to "Verifying stretch mode" on page 60.

Exiting stretch mode

Disable stretch mode by moving pools to a specified CRUSH rule or to the default replicated rule.

- Disable stretch mode.
  You can specify a CRUSH rule to move all pools to. If you do not specify a rule, Ceph moves the pools to the default replicated CRUSH rule.

```
ceph mon disable_stretch_mode CRUSH_RULE --yes-i-really-mean-it
```

Verifying stretch mode

Use this information to verify that stretch mode was created correctly with the implemented CRUSH rules.

1. Verify that all pools are using the CRUSH rule that was created in the Ceph cluster.
   In these examples, the CRUSH rule is set as `stretch_rule`, per the settings that were created in "Configuring a CRUSH map for stretch mode" on page 57.

```
for pool in $(rados lspools);do echo -n "Pool: ${pool}; ";ceph osd pool get ${pool}
crush_rule;done
```

For example,

```
[ceph: root@host01 /]# for pool in $(rados lspools);do echo -n "Pool: ${pool}; ";ceph
osd pool get ${pool} crush_rule;done
Pool: device_health_metrics; crush_rule: stretch_rule
Pool: cephfs.cephfs.meta; crush_rule: stretch_rule
Pool: cephfs.cephfs.data; crush_rule: stretch_rule
Pool: .rgw.root; crush_rule: stretch_rule
Pool: default.rgw.log; crush_rule: stretch_rule
Pool: default.rgw.control; crush_rule: stretch_rule
Pool: default.rgw.meta; crush_rule: stretch_rule
Pool: rbdpool; crush_rule: stretch_rule
```

2. Verify that stretch mode is enabled.
Ensure that `stretch_mode_enabled` is set to `true`.

```
ceph osd dump
```

The output includes the following information:

`stretch_mode_enabled`

Set to `true` if stretch mode is enabled.

`stretch_bucket_count`

The number of data centers with OSDs.

`degraded_stretch_mode`

Output of `0` if not degraded. If the stretch mode is degraded, this outputs the number of up sites.

`recovering_stretch_mode`

Output of `0` if not recovering. If the stretch mode is recovering, the output is `1`.

`stretch_mode_bucket`

A unique value set for each CRUSH bucket type. This value is usually set to 8, for data center.

For example,

```
stretch_mode_enabled true
stretch_bucket_count 2
degraded_stretch_mode 0
recovering_stretch_mode 0
stretch_mode_bucket 8
```

3. Verify that stretch mode is using the mon map, by using the **`ceph mon dump`** command.
Ensure the following:
   - `stretch_mode_enabled` is set to `1`
   - The correct mon host is set as `tiebreaker_mon`
   - The correct mon host is set as `disallowed_leaders`

```
ceph mon dump
```

For example,

```
[ceph: root@host01 /]# ceph mon dump
epoch 16
fsid ff19789c-f5c7-11ef-8e1c-fa163e4e1f7e
last_changed 2025-02-28T12:12:51.089706+0000
created 2025-02-28T11:34:59.325503+0000
min_mon_release 19 (squid)
election_strategy: 3
stretch_mode_enabled 1
tiebreaker_mon host07
disallowed_leaders host07
0: [v2:10.0.56.37:3300/0,v1:10.0.56.37:6789/0] mon.host01; crush_location
{datacenter=DC1}
1: [v2:10.0.59.188:3300/0,v1:10.0.59.188:6789/0] mon.host05; crush_location
{datacenter=DC2}
2: [v2:10.0.59.35:3300/0,v1:10.0.59.35:6789/0] mon.host02; crush_location
{datacenter=DC1}
3: [v2:10.0.56.189:3300/0,v1:10.0.56.189:6789/0] mon.host07; crush_location
{datacenter=DC3}
4: [v2:10.0.56.13:3300/0,v1:10.0.56.13:6789/0] mon.host04; crush_location
{datacenter=DC2}
dumped monmap epoch 16
```

AFTER COMPLETING THE TASK

1. Deploy, configure, and administer a Ceph Object Gateway. For more information, see Ceph Object Gateway.
2. Manage, create, configure, and use Ceph Block Devices. For more information, see Using Ceph Block Devices.
3. Create, mount, and work the Ceph File System (CephFS). For more information, see Ceph File Systems.

Using and maintaining stretch mode

Use and maintain stretch mode by adding OSD hosts, managing data center monitor service hosts, and replacing tiebreakers with a monitor both with and without a quorum.

Adding OSD hosts in stretch mode

You can add Ceph OSDs in the stretch mode. The procedure is similar to the addition of the OSD hosts on a cluster where stretch mode is not enabled.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Stretch mode is enabled on a cluster.
- Root-level access to the nodes.

1. List the available devices to deploy OSDs.

   ```
   ceph orch device ls [--hostname=HOST_1 HOST_2] [--wide] [--refresh]
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph orch device ls
   ```

2. Deploy the OSDs on specific hosts or on all the available devices.
   - Create an OSD from a specific device on a specific host.

     ```
     ceph orch daemon add osd HOST:DEVICE_PATH
     ```

     For example,

     ```
     [ceph: root@host01 /]# ceph orch daemon add osd host03:/dev/sdb
     ```

- Deploy OSDs on any available and unused devices.

  Important: This command creates collocated WAL and DB devices. Do not use this command to create non-collocated devices.

  ```
  ceph orch apply osd --all-available-devices
  ```

3. Move the OSD hosts under the CRUSH bucket.

   ```
   ceph osd crush move HOST datacenter=DATACENTER
   ```

   Note: Ensure you add the same topology nodes on both sites. Issues can occur if hosts are added only on one site.

   For example,

   ```
   [ceph: root@host01 /]# ceph osd crush move host03 datacenter=DC1
    [ceph: root@host01 /]# ceph osd crush move host06 datacenter=DC2
   ```

AFTER COMPLETING THE TASK

Verify that the OSD hosts and OSDs are added in the CRUSH bucket, by using the **`ceph osd tree`** command.

Related information

Adding OSDs

Managing data center monitor service hosts in stretch mode

Use this information to add and remove data center monitor service (`mon`) hosts in stretch mode. Managing data centers can be done by using the specification file or directly on the Ceph cluster.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Stretch mode is enabled on a cluster.
- Root-level access to the nodes.

Managing a `mon` service with a service specification file

These steps detail how to add a `mon` service. To remove the service, use the same steps of updating the service specification file, with removing the needed information.

1. Export the specification file for `mon` and save the output to `mon-spec.yaml`.

   ```
   ceph orch ls mon --export > mon-spec.yaml
   ```

   After the file is exported, the YAML file can be edited.

2. Add the new host details.
   In the following example, `host08` is being added to the cluster into the DC2 data center bucket.

```
service_type: host

addr: 10.1.172.225

hostname: host08

labels:

- mon

---

service_type: mon

service_name: mon

placement:

label: mon

spec:

crush_locations:

host01:

- datacenter=DC1

host02:

- datacenter=DC1

host04:

- datacenter=DC2

host05:

- datacenter=DC2

host08:

- datacenter=DC2
```

3. Apply the specification file.

```
ceph orch apply -i mon-spec.yaml
```

For example,

```
[ceph: root@host01 /]# eph orch apply -i mon-spec.yaml
Added host 'host08' with addr '10.1.172.225'
Scheduled mon update...
```

AFTER COMPLETING THE TASK

1. Use the **ceph mon dump** command to verify that the `mon` service was deployed and that the appropriate CRUSH location was added to the monitor.
For example,

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

epoch 10

fsid 4158287e-169e-11f0-b1ad-fa163e98b991

last_changed 2025-04-11T08:45:13.049347+0000

created 2025-04-11T06:29:24.974553+0000

min_mon_release 19 (squid)

election_strategy: 3

stretch_mode_enabled 1

tiebreaker_mon host07

disallowed_leaders host07

0: [v2:10.0.57.33:3300/0,v1:10.0.57.33:6789/0] mon.host07; crush_location
{datacenter=DC3}

1: [v2:10.0.58.200:3300/0,v1:10.0.58.200:6789/0] mon.host05; crush_location
{datacenter=DC2}

2: [v2:10.0.58.47:3300/0,v1:10.0.58.47:6789/0] mon.host02; crush_location
{datacenter=DC1}

3: [v2:10.0.58.104:3300/0,v1:10.0.58.104:6789/0] mon.host04; crush_location
{datacenter=DC2}

4: [v2:10.0.58.38:3300/0,v1:10.0.58.38:6789/0] mon.host01; crush_location
{datacenter=DC1}

5: [v2:10.0.66.96:3300/0,v1:10.0.66.96:6789/0] mon.host08; crush_location
{datacenter=DC2}

dumped monmap epoch 10
```

2. Use the **ceph orch host ls** command to verify that the host was added to the cluster.
   For example,

```
[ceph: root@host01 /]# ceph orch host ls

HOST ADDR LABELS STATUS

host01 10.0.56.37 mgr,mon,osd

host02 10.0.59.35 mgr,mon,osd

host03 10.0.58.106 osd,mds,rgw

host04 10.0.56.13 osd,mon,mgr

host05 10.0.59.188 mgr,mon,osd

host06 10.0.56.223 rgw,mds,osd

host07 10.0.56.189 _admin,mon

host08 10.1.172.225 mon

8 hosts in cluster
```

Managing a `mon` with the command-line interface

These steps detail how to add a `mon` service. To remove the service, use the same steps of updating with the CLI, with removing the needed information.

1. Set the monitor service tounmanaged.

```
ceph orch set-unmanaged mon
```

2. Optional: Use the **ceph orch ls** command to verify that the service was set, as expected.
   For example,

```
[ceph: root@host01 /]# ceph orch host ls
NAME                                PORTS             RUNNING  REFRESHED  AGE
 PLACEMENT
mon                                                       8/8  10m ago    19s
 <unmanaged>
```

3. Add a new host with the `mon` label.

```
ceph orch host add HOST_NAME IP_ADDRESS_OF_HOST [--label=LABEL_NAME_1,LABEL_NAME_2]
```

For example,

```
[ceph: root@host01 /]# ceph orch host add host08 10.1.172.205 --labels=mon
```

4. Add a monitor service with CRUSH locations.

> **Note:** At this point, the `mon` is not running and is not managed by `cephadm`.

```
ceph mon add NODE:IP_ADDRESS datacenter=DC2
```

For example,

```
[ceph: root@host01 /]# ceph mon add host08 10.1.172.205 datacenter=DC2
```

5. Deploy the monitor daemon using `cephadm`.

```
ceph orch daemon add mon host08
```

For example,

```
[ceph: root@host01 /]# ceph orch daemon add mon host08
Deployed mon.host08 on host 'host08'
```

6. Enable `cephadm` management for the monitor service.

```
ceph orch set-managed mon
```

7. Start the newly added `mon` daemon.

```
ceph orch set-managed mgr
```

AFTER COMPLETING THE TASK

Verify that the service, monitor, and host are added and running.

1. Use the **`ceph orch ls`** command to verify that the service is running.
For example,

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

NAME PORTS RUNNING  REFRESHED  AGE PLACEMENT

mon   8/8  7m ago 4d label:mon
```

2. Use the **`ceph mon dump`** command to verify that the `mon` service was deployed and that the appropriate CRUSH location was added to the monitor.
For example,

```
[ceph: root@host01 /]# ceph mon dump
epoch 16
fsid ff19789c-f5c7-11ef-8e1c-fa163e4e1f7e
last_changed 2025-02-28T12:12:51.089706+0000
created 2025-02-28T11:34:59.325503+0000
min_mon_release 19 (squid)
election_strategy: 3
stretch_mode_enabled 1
tiebreaker_mon host07
disallowed_leaders host07
0: [v2:10.0.56.37:3300/0,v1:10.0.56.37:6789/0] mon.host01; crush_location
{datacenter=DC1}
1: [v2:10.0.59.188:3300/0,v1:10.0.59.188:6789/0] mon.host05; crush_location
{datacenter=DC2}
2: [v2:10.0.59.35:3300/0,v1:10.0.59.35:6789/0] mon.host02; crush_location
{datacenter=DC1}
3: [v2:10.0.56.189:3300/0,v1:10.0.56.189:6789/0] mon.host07; crush_location
{datacenter=DC3}
4: [v2:10.0.56.13:3300/0,v1:10.0.56.13:6789/0] mon.host04; crush_location
{datacenter=DC2}
dumped monmap epoch 16
```

3. Use the **ceph orch host ls** command to verify that the host was added to the cluster.
   For example,

```
[ceph: root@host01 /]# ceph orch host ls

HOST ADDR LABELS STATUS

host01 10.0.56.37 mgr,mon,osd

host02 10.0.59.35 mgr,mon,osd

host03 10.0.58.106 osd,mds,rgw

host04 10.0.56.13 osd,mon,mgr

host05 10.0.59.188 mgr,mon,osd

host06 10.0.56.223 rgw,mds,osd

host07 10.0.56.189 _admin,mon

host08 10.1.172.205 mon

8 hosts in cluster
```

Replacing a tiebreaker with a monitor in quorum

If your tiebreaker monitor fails, you can replace it with an existing monitor in quorum and remove it from the cluster.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Stretch mode is enabled on a cluster.

1. Disable automated monitor deployment.

```
ceph orch apply mon --unmanaged
```

   For example,

```
[ceph: root@host01 /]# ceph orch apply mon --unmanaged

 Scheduled mon update…
```

2. View the monitors in quorum, by using the **ceph -s** command.
   For example,

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

 mon: 5 daemons, quorum host01, host02, host04, host05 (age 30s), out of quorum:
host07
```

3. Set the monitor in quorum as a new tiebreaker.

```
ceph mon set_new_tiebreaker NEW_HOST
```

For example,

```
[ceph: root@host01 /]# ceph mon set_new_tiebreaker host02
```

> **Important:** If the monitor is in the same location as existing nontiebreaker monitors, an error message is displayed.
> For example,
>
> ```
> [ceph: root@host01 /]# ceph mon set_new_tiebreaker host02
>
>   Error EINVAL: mon.host02 has location DC1, which matches mons host02 on the
> datacenter dividing bucket for stretch mode.
> ```
>
> If there is an error message, change the location of the monitor.
>
> ```
> ceph mon set_location HOST datacenter=DATACENTER
> ```
>
> For example,
>
> ```
> [ceph: root@host01 /]# ceph mon set_location host02 datacenter=DC3
> ```

4. Remove the failed tiebreaker monitor.

```
ceph orch daemon rm FAILED_TIEBREAKER_MONITOR --force
```

For example,

```
[ceph: root@host01 /]# ceph orch daemon rm mon.host07 --force

 Removed mon.host07 from host 'host07'
```

5. Redeploy the monitor after the monitor is removed from the host.

```
ceph mon add HOST IP_ADDRESS datacenter=DATACENTER
 ceph orch daemon add mon HOST
```

For example,

```
[ceph: root@host01 /]# ceph mon add host07 213.222.226.50 datacenter=DC1
 [ceph: root@host01 /]# ceph orch daemon add mon host07
```

6. Ensure that there are 5 monitors in quorum, by using the **ceph -s** command.

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

 mon: 5 daemons, quorum host01, host02, host04, host05, host07 (age 15s)
```

7. Verify that everything is configured properly.

```
ceph mon dump
```

For example,

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

 epoch 19
 fsid 1234ab78-1234-11ed-b1b1-de456ef0a89d
 last_changed 2023-01-17T04:12:05.709475+0000
 created 2023-01-16T05:47:25.631684+0000
 min_mon_release 16 (pacific)
 election_strategy: 3
 stretch_mode_enabled 1
 tiebreaker_mon host02
 disallowed_leaders host02
 0: [v2:132.224.169.63:3300/0,v1:132.224.169.63:6789/0] mon.host02; crush_location
{datacenter=DC3}
 1: [v2:220.141.179.34:3300/0,v1:220.141.179.34:6789/0] mon.host04; crush_location
{datacenter=DC2}
 2: [v2:40.90.220.224:3300/0,v1:40.90.220.224:6789/0] mon.host01; crush_location
{datacenter=DC1}
 3: [v2:60.140.141.144:3300/0,v1:60.140.141.144:6789/0] mon.host07; crush_location
{datacenter=DC1}
 4: [v2:186.184.61.92:3300/0,v1:186.184.61.92:6789/0] mon.host03; crush_location
{datacenter=DC2}
 dumped monmap epoch 19
```

8. Redeploy the monitors.

```
ceph orch apply mon --placement="HOST_1, HOST_2, HOST_3, HOST_4, HOST_5”
```

For example,

```
[ceph: root@host01 /]# ceph orch apply mon --placement="host01, host02, host04,
host05, host07"

 Scheduled mon update...
```

Replacing a tiebreaker with a new monitor

When a tiebreaker monitor fails you can replace it with a new monitor and remove the failed one from the cluster.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Stretch mode in enabled on a cluster.

1. Add a new monitor to the cluster.
   a. Manually add the `crush_location` to the new monitor.

> **Note:** The new monitor must be in a different location than existing non-tiebreaker monitors.

```
ceph mon add NEW_HOST IP_ADDRESS datacenter=DATACENTER
```

For example,

```
[ceph: root@host01 /]# ceph mon add host06 213.222.226.50 datacenter=DC3

 adding mon.host06 at [v2:213.222.226.50:3300/0,v1:213.222.226.50:6789/0]
```

   b. Disable the automated monitor deployment.
   For example,

```
[ceph: root@host01 /]# ceph orch apply mon --unmanaged

 Scheduled mon update…
```

   c. Deploy the new monitor.

```
ceph orch daemon add mon NEW_HOST
```

For example,

```
[ceph: root@host01 /]# ceph orch daemon add mon host06
```

2. Ensure that there are six monitors.
Five of the monitors must be in quorum.
For example,

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

 mon: 6 daemons, quorum host01, host02, host04, host05, host06 (age 30s), out of
quorum: host07
```

3. Set the new monitor as a new tiebreaker.

```
ceph mon set_new_tiebreaker NEW_HOST
```

For example,

```
[ceph: root@host01 /]# ceph mon set_new_tiebreaker host06
```

4. Remove the failed tiebreaker monitor.

```
ceph orch daemon rm FAILED_TIEBREAKER_MONITOR --force
```

For example,

```
[ceph: root@host01 /]# ceph orch daemon rm mon.host07 --force

  Removed mon.host07 from host 'host07'
```

5. Verify that everything is properly configured.
For example,

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

 epoch 19
 fsid 1234ab78-1234-11ed-b1b1-de456ef0a89d
 last_changed 2023-01-17T04:12:05.709475+0000
 created 2023-01-16T05:47:25.631684+0000
 min_mon_release 16 (pacific)
 election_strategy: 3
 stretch_mode_enabled 1
 tiebreaker_mon host06
 disallowed_leaders host06
 0: [v2:213.222.226.50:3300/0,v1:213.222.226.50:6789/0] mon.host06; crush_location
{datacenter=DC3}
 1: [v2:220.141.179.34:3300/0,v1:220.141.179.34:6789/0] mon.host04; crush_location
{datacenter=DC2}
 2: [v2:40.90.220.224:3300/0,v1:40.90.220.224:6789/0] mon.host01; crush_location
{datacenter=DC1}
 3: [v2:60.140.141.144:3300/0,v1:60.140.141.144:6789/0] mon.host02; crush_location
{datacenter=DC1}
 4: [v2:186.184.61.92:3300/0,v1:186.184.61.92:6789/0] mon.host05; crush_location
{datacenter=DC2}
 dumped monmap epoch 19
```

6. Redeploy the monitors.

```
ceph orch apply mon --placement="HOST_1, HOST_2, HOST_3, HOST_4, HOST_5”
```

For example,

```
[ceph: root@host01 /]# ceph orch apply mon --placement="host01, host02, host04,
host05, host06"

 Scheduled mon update…
```

Read Affinity in stretch clusters

Read Affinity reduces cross-zone traffic by keeping the data access within the respective data centers.

For stretched clusters deployed in multi-zone environments, the read affinity topology implementation provides a mechanism to help keep traffic within the data center it originated from. Ceph Object Gateway volumes have the ability to read data from an OSD in proximity to the client, according to OSD locations defined in the CRUSH map and topology labels on nodes.

For example, a stretch cluster contains a Ceph Object Gateway primary OSD and replicated OSDs spread across two data centers A and B. If a *GET* action is performed on an *Object* in data center A, the *READ* operation is performed on the data of the OSDs closest to the client in data center A.

Performing localized reads

Run a localized read on a replicated pool in a stretch cluster.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A stretch cluster with two data centers and a Ceph Object Gateway that is configured on both.
- A user who is created with a bucket having primary and replicated OSDs.

When a localized read request is made on a replicated pool, Ceph selects the local OSDs closest to the client based on the client location specified in **`crush_location`**.

- To run a localized read, set **`rados_replica_read_policy`** to `localize` in the OSD daemon configuration by using the **`ceph config set`** command.

```
ceph config set client.rgw.rgw.1 rados_replica_read_policy localize
```

AFTER COMPLETING THE TASK

Verify the localized read from an OSD set.

1. Run the **`ceph osd tree`** command to view the OSDs and the data centers.
   For example,

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

ID  CLASS  WEIGHT   TYPE NAME                                STATUS  REWEIGHT  PRI-
AFF
-1         0.58557  root default
-3         0.29279      datacenter DC1
-2         0.09760          host ceph-ci-fbv67y-ammmck-node2
 2    hdd  0.02440              osd.2                            up   1.00000
 1.00000
11    hdd  0.02440              osd.11                           up   1.00000
 1.00000
17    hdd  0.02440              osd.17                           up   1.00000
 1.00000
22    hdd  0.02440              osd.22                           up   1.00000
 1.00000
-4         0.09760          host ceph-ci-fbv67y-ammmck-node3
 0    hdd  0.02440              osd.0                            up   1.00000
 1.00000
 6    hdd  0.02440              osd.6                            up   1.00000
 1.00000
12    hdd  0.02440              osd.12                           up   1.00000
 1.00000
18    hdd  0.02440              osd.18                           up   1.00000
 1.00000
-5         0.09760          host ceph-ci-fbv67y-ammmck-node4
 5    hdd  0.02440              osd.5                            up   1.00000
 1.00000
10    hdd  0.02440              osd.10                           up   1.00000
 1.00000
16    hdd  0.02440              osd.16                           up   1.00000
 1.00000
23    hdd  0.02440              osd.23                           up   1.00000
 1.00000
-7         0.29279      datacenter DC2
-6         0.09760          host ceph-ci-fbv67y-ammmck-node5
 3    hdd  0.02440              osd.3                            up   1.00000
 1.00000
 8    hdd  0.02440              osd.8                            up   1.00000
 1.00000
14    hdd  0.02440              osd.14                           up   1.00000
 1.00000
20    hdd  0.02440              osd.20                           up   1.00000
 1.00000
-8         0.09760          host ceph-ci-fbv67y-ammmck-node6
 4    hdd  0.02440              osd.4                            up   1.00000
 1.00000
 9    hdd  0.02440              osd.9                            up   1.00000
 1.00000
15    hdd  0.02440              osd.15                           up   1.00000
 1.00000
21    hdd  0.02440              osd.21                           up   1.00000
 1.00000
-9         0.09760          host ceph-ci-fbv67y-ammmck-node7
 1    hdd  0.02440              osd.1                            up   1.00000
 1.00000
 7    hdd  0.02440              osd.7                            up   1.00000
 1.00000
13    hdd  0.02440              osd.13                           up   1.00000
 1.00000
19    hdd  0.02440              osd.19                           up   1.00000
 1.00000
```

2. Run the **ceph orch** command to identify the Ceph Object Gateway daemons in the data centers. For example,

```
[ceph: root@host01 /]# ceph orch ps | grep rg

rgw.rgw.1.ceph-ci-fbv67y-ammmck-node4.dmsmex         ceph-ci-fbv67y-ammmck-node4
     *:80              running (4h)     10m ago  22h    93.3M        -
 19.1.0-55.el9cp  0ee0a0ad94c7  34f27723ccd2
rgw.rgw.1.ceph-ci-fbv67y-ammmck-node7.pocecp         ceph-ci-fbv67y-ammmck-node7
     *:80              running (4h)     10m ago  22h    96.4M        -
 19.1.0-55.el9cp  0ee0a0ad94c7  40e4f2a6d4c4
```

3. Verify if the localized read took place by running the **vim** command on the Ceph Object Gateway logs.

```
vim /var/log/ceph/<fsid>/<ceph-client-rgw>.log
```

For example,

```
[ceph: root@host01 /]# vim /var/log/ceph/<fsid>/<ceph-client-rgw>.log

2024-08-26T08:07:45.471+0000 7fc623e63640  1 ====== starting new request
req=0x7fc5b93694a0 =====
2024-08-26T08:07:45.471+0000 7fc623e63640  1 -- 10.0.67.142:0/279982082 -->
[v2:10.0.66.23:6816/73244434,v1:10.0.66.23:6817/73244434] -- osd_op(unknown.0.0:9081
11.55 11:ab26b168:::3acf4091-c54c-43b5-a495-c505fe545d25.27842.1_f1:head
[getxattrs,stat] snapc 0=[]
ondisk+read+localize_reads+known_if_redirected+supports_pool_eio e3533) --
0x55f781bd2000 con 0x55f77f0e8c00
```

You can see in the logs that a localized read has taken place.

Important: To be able to view the debug logs, you must first enable `debug_ms 1` in the configuration by running the **`ceph config set`** command.
For example,

```
[ceph: root@host01 /]#ceph config set client.rgw.rgw.1.ceph-ci-gune2w-mysx73-
node4.dgvrmx    advanced  debug_ms    1/1

[ceph: root@host01 /]#ceph config set client.rgw.rgw.1.ceph-ci-gune2w-mysx73-
node7.rfkqqq    advanced  debug_ms    1/1
```

Performing balanced reads

Perform a balanced read on a pool to retrieve evenly distributed OSDs across data centers.

PREREQUISITE

- A stretch cluster with two data centers and Ceph Object Gateway configured on both.
- A user created with a bucket and OSDs - primary and replicated OSDs.

When a balanced *READ* is issued on a pool, the read operations are distributed evenly across all OSDs that are spread across the data centers.

1. To perform a balanced read, set **`rados_replica_read_policy`** to `balance` for the Ceph Object Gateway daemon configuration using the `ceph config set` command.

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

   client.rgw.rgw.1 advanced rados_replica_read_policy balance
   ```

   The IO load is divided between the two data centers when a GET operation is performed.

2. **Verification**: Perform the below steps to verify the localized read from an OSD set.
   a. Run the `ceph osd tree` command to view the OSDs and the data centers.
   Example

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

ID  CLASS  WEIGHT   TYPE NAME                                STATUS  REWEIGHT
 PRI-AFF
-1         0.58557  root default
-3         0.29279      datacenter DC1
-2         0.09760          host ceph-ci-fbv67y-ammmck-node2
 2    hdd  0.02440              osd.2                            up   1.00000
 1.00000
11    hdd  0.02440              osd.11                           up   1.00000
 1.00000
17    hdd  0.02440              osd.17                           up   1.00000
 1.00000
22    hdd  0.02440              osd.22                           up   1.00000
 1.00000
-4         0.09760          host ceph-ci-fbv67y-ammmck-node3
 0    hdd  0.02440              osd.0                            up   1.00000
 1.00000
 6    hdd  0.02440              osd.6                            up   1.00000
 1.00000
12    hdd  0.02440              osd.12                           up   1.00000
 1.00000
18    hdd  0.02440              osd.18                           up   1.00000
 1.00000
-5         0.09760          host ceph-ci-fbv67y-ammmck-node4
 5    hdd  0.02440              osd.5                            up   1.00000
 1.00000
10    hdd  0.02440              osd.10                           up   1.00000
 1.00000
16    hdd  0.02440              osd.16                           up   1.00000
 1.00000
23    hdd  0.02440              osd.23                           up   1.00000
 1.00000
-7         0.29279      datacenter DC2
-6         0.09760          host ceph-ci-fbv67y-ammmck-node5
 3    hdd  0.02440              osd.3                            up   1.00000
 1.00000
 8    hdd  0.02440              osd.8                            up   1.00000
 1.00000
14    hdd  0.02440              osd.14                           up   1.00000
 1.00000
20    hdd  0.02440              osd.20                           up   1.00000
 1.00000
-8         0.09760          host ceph-ci-fbv67y-ammmck-node6
 4    hdd  0.02440              osd.4                            up   1.00000
 1.00000
 9    hdd  0.02440              osd.9                            up   1.00000
 1.00000
15    hdd  0.02440              osd.15                           up   1.00000
 1.00000
21    hdd  0.02440              osd.21                           up   1.00000
 1.00000
-9         0.09760          host ceph-ci-fbv67y-ammmck-node7
 1    hdd  0.02440              osd.1                            up   1.00000
 1.00000
 7    hdd  0.02440              osd.7                            up   1.00000
 1.00000
13    hdd  0.02440              osd.13                           up   1.00000
 1.00000
19    hdd  0.02440              osd.19                           up   1.00000
 1.00000
```

b. Run the `ceph orch` command to identify the Ceph Object Gateway daemons in the data centers.
Example

```
ceph orch ps | grep rg

rgw.rgw.1.ceph-ci-fbv67y-ammmck-node4.dmsmex         ceph-ci-fbv67y-ammmck-node4
           *:80              running (4h)     10m ago  22h    93.3M        -
 19.1.0-55.el9cp  0ee0a0ad94c7  34f27723ccd2
rgw.rgw.1.ceph-ci-fbv67y-ammmck-node7.pocecp         ceph-ci-fbv67y-ammmck-node7
           *:80              running (4h)     10m ago  22h    96.4M        -
 19.1.0-55.el9cp  0ee0a0ad94c7  40e4f2a6d4c4
```

c. Verify if a balanced read has happened by running the `vim` command on the Ceph Object Gateway logs.
For example,

```
[ceph: root@host01 /]# vim /var/log/ceph/<fsid>/<ceph-client-rgw>.log

2024-08-27T09:32:25.510+0000 7f2a7a284640  1 ====== starting new request
req=0x7f2a31fcf4a0 =====
2024-08-27T09:32:25.510+0000 7f2a7a284640  1 -- 10.0.67.142:0/3116867178 -->
[v2:10.0.64.146:6816/2838383288,v1:10.0.64.146:6817/2838383288] --
osd_op(unknown.0.0:268731 11.55 11:ab26b168:::3acf4091-c54c-43b5-a495-
c505fe545d25.27842.1_f1:head [getxattrs,stat] snapc 0=[]
ondisk+read+balance_reads+known_if_redirected+supports_pool_eio e3554) --
0x55cd1b88dc00 con 0x55cd18dd6000
```

You can see in the logs that a balanced read has taken place.

Important: To be able to view the debug logs, you must first enable `debug_ms 1` in the configuration by running the `ceph config set` command.

```
[ceph: root@host01 /]#ceph config set client.rgw.rgw.1.ceph-ci-gune2w-
mysx73-node4.dgvrmx    advanced  debug_ms    1/1

[ceph: root@host01 /]#ceph config set client.rgw.rgw.1.ceph-ci-gune2w-
mysx73-node7.rfkqqq    advanced  debug_ms    1/1
```

Performing default reads

Perform a default read on a pool to retrieve data from primary data centers.

PREREQUISITE

- A stretch cluster with two data centers and Ceph Object Gateway configured on both.
- A user created with a bucket and OSDs - primary and replicated OSDs.

When a default *READ* is issued on a pool, the IO operations are retrieved directly from each OSD in the data center.

1. To perform a default read, set **`rados_replica_read_policy`** to `default` in the OSD daemon configuration by using the `ceph config set` command.

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

   client.rgw.rgw.1 advanced rados_replica_read_policy default
   ```

   The IO operations from the closest OSD in a data center are retrieved when a GET operation is performed.

2. **Verification**: Perform the below steps to verify the localized read from an OSD set.
   a. Run the `ceph osd tree` command to view the OSDs and the data centers.
   For example,

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

ID  CLASS  WEIGHT   TYPE NAME                                STATUS  REWEIGHT
 PRI-AFF
-1         0.58557  root default
-3         0.29279      datacenter DC1
-2         0.09760          host ceph-ci-fbv67y-ammmck-node2
 2    hdd  0.02440              osd.2                            up   1.00000
 1.00000
11    hdd  0.02440              osd.11                           up   1.00000
 1.00000
17    hdd  0.02440              osd.17                           up   1.00000
 1.00000
22    hdd  0.02440              osd.22                           up   1.00000
 1.00000
-4         0.09760          host ceph-ci-fbv67y-ammmck-node3
 0    hdd  0.02440              osd.0                            up   1.00000
 1.00000
 6    hdd  0.02440              osd.6                            up   1.00000
 1.00000
12    hdd  0.02440              osd.12                           up   1.00000
 1.00000
18    hdd  0.02440              osd.18                           up   1.00000
 1.00000
-5         0.09760          host ceph-ci-fbv67y-ammmck-node4
 5    hdd  0.02440              osd.5                            up   1.00000
 1.00000
10    hdd  0.02440              osd.10                           up   1.00000
 1.00000
16    hdd  0.02440              osd.16                           up   1.00000
 1.00000
23    hdd  0.02440              osd.23                           up   1.00000
 1.00000
-7         0.29279      datacenter DC2
-6         0.09760          host ceph-ci-fbv67y-ammmck-node5
 3    hdd  0.02440              osd.3                            up   1.00000
 1.00000
 8    hdd  0.02440              osd.8                            up   1.00000
 1.00000
14    hdd  0.02440              osd.14                           up   1.00000
 1.00000
20    hdd  0.02440              osd.20                           up   1.00000
 1.00000
-8         0.09760          host ceph-ci-fbv67y-ammmck-node6
 4    hdd  0.02440              osd.4                            up   1.00000
 1.00000
 9    hdd  0.02440              osd.9                            up   1.00000
 1.00000
15    hdd  0.02440              osd.15                           up   1.00000
 1.00000
21    hdd  0.02440              osd.21                           up   1.00000
 1.00000
-9         0.09760          host ceph-ci-fbv67y-ammmck-node7
 1    hdd  0.02440              osd.1                            up   1.00000
 1.00000
 7    hdd  0.02440              osd.7                            up   1.00000
 1.00000
13    hdd  0.02440              osd.13                           up   1.00000
 1.00000
19    hdd  0.02440              osd.19                           up   1.00000
 1.00000
```

b. Run the `ceph orch` command to identify the Ceph Object Gateway daemons in the data centers. For example,

```
ceph orch ps | grep rg

rgw.rgw.1.ceph-ci-fbv67y-ammmck-node4.dmsmex         ceph-ci-fbv67y-ammmck-node4
           *:80              running (4h)     10m ago  22h    93.3M        -
 19.1.0-55.el9cp  0ee0a0ad94c7  34f27723ccd2
rgw.rgw.1.ceph-ci-fbv67y-ammmck-node7.pocecp         ceph-ci-fbv67y-ammmck-node7
           *:80              running (4h)     10m ago  22h    96.4M        -
 19.1.0-55.el9cp  0ee0a0ad94c7  40e4f2a6d4c4
```

c. Verify if a default read has happened by running the `vim` command on the Ceph Object Gateway logs.
For example,

```
[ceph: root@host01 /]# vim /var/log/ceph/<fsid>/<ceph-client-rgw>.log

2024-08-28T10:26:05.155+0000 7fe6b03dd640  1 ====== starting new request
req=0x7fe6879674a0 =====
2024-08-28T10:26:05.156+0000 7fe6b03dd640  1 -- 10.0.64.251:0/2235882725 -->
[v2:10.0.65.171:6800/4255735352,v1:10.0.65.171:6801/4255735352] --
osd_op(unknown.0.0:1123 11.6d 11:b69767fc:::699c2d80-5683-43c5-bdcd-
e8912107c176.24827.3_f1:head [getxattrs,stat] snapc 0=[]
ondisk+read+known_if_redirected+supports_pool_eio e4513) -- 0x5639da653800 con
0x5639d804d800
```

You can see in the logs that a default read has taken place.

Important: To be able to view the debug logs, you must first enable `debug_ms 1` in the configuration by running the `ceph config set` command.

```
[ceph: root@host01 /]#ceph config set client.rgw.rgw.1.ceph-ci-gune2w-
mysx73-node4.dgvrmx    advanced  debug_ms    1/1

[ceph: root@host01 /]#ceph config set client.rgw.rgw.1.ceph-ci-gune2w-
mysx73-node7.rfkqqq    advanced  debug_ms    1/1
```

Generalized stretch cluster configuration for three availability zones

As a storage administrator, you can configure a generalized stretch cluster configuration for three availability zones with Ceph OSDs.

Red Hat Ceph Storage can withstand the loss of Ceph OSDs because of its network and cluster, which are equally reliable with failures that are randomly distributed across the CRUSH map. If multiple OSDs are shut down, the remaining OSDs and monitors still manage to operate.

Using a single cluster limits data availability to a single location with a single point of failure. However, in some situations, higher availability might be required. Using three availability zones allows the cluster to withstand power loss and even a full data center loss in the event of a natural disaster.

With a generalized stretch cluster configuration for three availability zones, three data centers are supported, with each site holding two copies of the data. This helps ensure that even during a data center outage, the data remains accessible and writeable from another site. With this configuration, the pool replication size is `6` and the pool `min_size` is `3`.

Note: The standard Ceph configuration survives many failures of the network or data centers and it never compromises data consistency. If you restore enough Ceph servers following a failure, it recovers. Red Hat Ceph Storage maintains availability if you lose a data center, but can still form a quorum of monitors and have all the data available with enough copies to satisfy pools' `min_size`, or CRUSH rules that replicate again to meet the size.

Limitations

When using generalized stretch clusters, the following limitations should be considered.

- Generalized stretch cluster configuration for three availability zones does not support I/O operations during a netsplit scenario between two or more zones. While the cluster remains accessible for basic Ceph commands, I/O usage remains unavailable until the netsplit is resolved. This is different from stretch mode, where the tiebreaker monitor can isolate one zone of the cluster and continue I/O operations in degraded mode during a netsplit. For more information about stretch mode, see "Stretch mode" on page 47.
- In a three availability zone configuration, Red Hat Ceph Storage is designed to tolerate multiple host failures. However, if more than 25% of the OSDs in the cluster go down, Ceph may stop marking OSDs as `out`. This behavior is controlled by the **`mon_osd_min_in_ratio`** parameter. By default, **`mon_osd_min_in_ratio`** is set to `0.75`, meaning that at least 75% of the OSDs in the cluster must remain `in` (active) before any additional OSDs can be marked `out`. This setting prevents too many OSDs from being marked `out` as this might lead to significant data movement. The data movement can cause high client I/O impact and long recovery times when the OSDs are returned to service.

If Red Hat Ceph Storage stops marking OSDs as `out`, some placement groups (PGs) may fail to rebalance to surviving OSDs, potentially leading to inactive placement groups (PGs).

> **Important:** While adjusting the **`mon_osd_min_in_ratio`** value can allow more OSDs to be marked `out` and trigger rebalancing, this should be done with caution. For more information on the **`mon_osd_min_in_ratio`** parameter, see Ceph Monitor and OSD configuration options.

Generalized stretch cluster deployment requirements

This information details important hardware, software, and network requirements that are needed for deploying a generalized stretch cluster configuration for three availability zones.

Hardware requirements

Use the following minimum hardware requirements before deploying generalized stretch cluster configuration for three availability zones. "Hardware requirements" on page 78 lists the physical server locations and Ceph component layout for an example three availability zone deployment.

Hardware requirements

Host name	Datacenter	Ceph services
host01	DC1	OSD+MON+MGR
host02	DC1	OSD+MON+MGR+RGW
host03	DC1	OSD+MON+MDS
host04	DC2	OSD+MON+MGR
host05	DC2	OSD+MON+MGR+RGW
host06	DC2	OSD+MON+MDS
host07	DC3	OSD+MON+MGR
host08	DC3	OSD+MON+MGR+RGW
host09	DC3	OSD+MON+MDS

Network configuration requirements

Use the following network configuration requirements before deploying generalized stretch cluster configuration for three availability zones.

- Have two separate networks, one public network and one cluster network.
- Have three different data centers that support VLANS and subnets for Ceph cluster and public networks for all data centers.

  > **Note:** You can use different subnets for each of the data centers.

- The latencies between data centers running the Red Hat Ceph Storage Object Storage Devices (OSDs) cannot exceed 10 ms RTT.

For more information about network considerations, see "Network considerations for Red Hat Ceph Storage" on page 0.

Cluster setup requirements

Ensure that the hostname is configured by using the bare or short hostname in all hosts.

```
hostnamectl set-hostname SHORT_NAME
```

> **Important:** The **`hostname`** command should only return the short hostname, when run on all nodes. If the FQDN is returned, the cluster configuration will not be successful.

Bootstrapping the Ceph cluster with a specification file

Deploy the generalized stretch cluster by setting the CRUSH location to the daemons in the cluster with the spec configuration file.

PREREQUISITE

Before you begin, be sure that you have root-level access to the nodes.
Set the CRUSH location to the daemons in the cluster with a service configuration file. Use the configuration file to add the hosts to the proper locations during deployment.

For more information about Ceph bootstrapping and different **cephadm bootstrap** command options, see Bootstrapping a new storage cluster.

> **Important:** Run **cephadm bootstrap** on the node that you want to be the initial Monitor node in the cluster. The *IP_ADDRESS* option should be the IP address of the node you are using to run **cephadm bootstrap**.

> **Note:**
> - If the storage cluster includes multiple networks and interfaces, be sure to choose a network that is accessible by any node that uses the storage cluster.
> - To deploy a storage cluster by using IPV6 addresses, use the IPV6 address format for the `--mon-ip` *IP_ADDRESS* option. For example: **cephadm bootstrap --mon-ip 2620:52:0:880:225:90ff:fefc:2536 --registry-json /etc/mylogin.json**.
> - To route the internal cluster traffic over the public network, omit the `--cluster-network` *SUBNET* option.

Within this procedure the network Classless Inter-Domain Routing (CIDR) is referred to as *subnet*.

1. Create the service configuration YAML file.
   The YAML file adds the nodes to the Red Hat Ceph Storage cluster and also sets specific labels for where the services run.
   The following example depends on the specific OSD and Ceph Object Gateway (RGW) configuration that is needed.

```
---
service_type: host
hostname: HOST01
addr: IP_ADDRESS01
labels: ['alertmanager', 'osd', 'installer', '_admin', 'mon', 'prometheus', 'mgr',
'grafana']
location:
  root: default
  datacenter: DC1
---
service_type: host
hostname: HOST02
addr: IP_ADDRESS02
labels: ['osd', 'mon', 'mgr', 'rgw']
location:
  root: default
  datacenter: DC1
---
service_type: host
hostname: HOST03
addr: IP_ADDRESS03
labels: ['osd', 'mon', 'mds']
location:
  root: default
  datacenter: DC1
---
service_type: host
hostname: HOST04
addr: IP_ADDRESS04
labels: ['osd', '_admin', 'mon', 'mgr']
```

```
location:
  root: default
  datacenter: DC2
---
service_type: host
hostname: HOST05
addr: IP_ADDRESS05
labels: ['osd', 'mon', 'mgr', 'rgw']
location:
  root: default
  datacenter: DC2
---
service_type: host
hostname: HOST06
addr: IP_ADDRESS06
labels: ['osd', 'mon', 'mds']
location:
  root: default
  datacenter: DC2
---
service_type: host
hostname: HOST07
addr: IP_ADDRESS07
labels: ['osd', '_admin', 'mon', 'mgr']
location:
  root: default
  datacenter: DC3
---
service_type: host
hostname: HOST08
addr: IP_ADDRESS08
labels: ['osd', 'mon', 'mgr', 'rgw']
location:
  root: default
  datacenter: DC3
---
service_type: host
hostname: HOST09
addr: IP_ADDRESS09
labels: ['osd', 'mon', 'mds']
location:
  root: default
  datacenter: DC3
---
service_type: mon
service_name: mon
placement:
  label: mon
spec:
  crush_locations:
    HOST01:
    - datacenter=DC1
    HOST02:
    - datacenter=DC1
    HOST03:
    - datacenter=DC1
    HOST04:
    - datacenter=DC2
    HOST05:
    - datacenter=DC2
    HOST06:
    - datacenter=DC2
    HOST07:
    - datacenter=DC3
    HOST08:
    - datacenter=DC3
    HOST09:
    - datacenter=DC3

---
service_type: mgr
service_name: mgr
placement:
  label: mgr
------
service_type: osd
service_id: osds
placement:
  label: osd
spec:
  data_devices:
```

```
     all: true
---------
service_type: rgw
service_id: rgw.rgw.1
placement:
label: rgw
------------
```

For more information about changing the custom spec for OSD and Object Gateway, see the following deployment instructions::

- “Deploying Ceph OSDs using advanced service specifications” on page 249
- “Deploying the Ceph Object Gateway using the service specification” on page 277

2. Bootstrap the storage cluster with the **--apply-spec** option.

```
cephadm bootstrap --apply-spec CONFIGURATION_FILE_NAME --mon-ip MONITOR_IP_ADDRESS --
ssh-private-key PRIVATE_KEY --ssh-public-key PUBLIC_KEY --registry-url REGISTRY_URL --
registry-username USER_NAME --registry-password PASSWORD
```

For example,

```
[root@host01 ~]# cephadm bootstrap --apply-spec initial-config.yaml --mon-ip
10.10.128.68 --ssh-private-key /home/ceph/.ssh/id_rsa --ssh-public-key /home/
ceph/.ssh/id_rsa.pub --registry-url registry.redhat.io --registry-username myuser1 --
registry-password mypassword1
```

> **Important:** You can use different command options with the **cephadm bootstrap** command but always include the **--apply-spec** option to use the service configuration file and configure the host locations.

3. Log in to the `cephadm` shell.

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

4. Configure the public network with the subnet.
For more information about configuring multiple public networks to the cluster, see “Configuring multiple public networks to the cluster” on page 0.

```
ceph config set global public_network "SUBNET_1,SUBNET_2, ..."
```

For example,

```
[ceph: root@host01 /]# ceph config set global public_network
"10.0.208.0/22,10.0.212.0/22,10.0.64.0/22,10.0.56.0/22"
```

5. Optional: Configure a cluster network.
For more information about configuring multiple cluster networks to the cluster, see “Configuring a private network” on page 0.

```
ceph config set global cluster_network "SUBNET_1,SUBNET_2, ..."
```

For example,

```
[ceph: root@host01 /]# ceph config set global cluster_network
"10.0.208.0/22,10.0.212.0/22,10.0.64.0/22,10.0.56.0/22"
```

6. Optional: Verify the network configurations.

```
[ceph: root@host01 /]# ceph config dump | grep network
```

7. Restart the daemons.
Ceph daemons bind dynamically, so you do not have to restart the entire cluster at once if you change the network configuration for a specific daemon.

```
ceph orch restart mon
```

8. Optional: Restart the cluster on the admin node as a root user, by running the **systemctl restart** command.

> **Note:** To get the FSID of the cluster, use the **ceph fsid** command.

```
systemctl restart ceph-FSID_OF_CLUSTER.target
```

For example,

```
[root@host01 ~]# systemctl restart ceph-1ca9f6a8-d036-11ec-8263-fa163ee967ad.target
```

AFTER COMPLETING THE TASK

Verify the specification file details and that the bootstrap was installed successfully.

1. Verify that all hosts were placed in the expected data centers, as specified in step “1” on page 79.

```
ceph osd tree
```

Check that there are three data centers under `root` and that the hosts are placed in each of the expected data centers.

> **Note:** The hosts with OSDs will only be present after bootstrap if OSDs are deployed during bootstrap with the specification file.

For example,

```
[root@host01 ~]# ceph osd tree
ID   CLASS  WEIGHT   TYPE NAME                                STATUS
 REWEIGHT  PRI-AFF
 -1         0.87836  root default
 -3         0.29279      datacenter DC1
 -2         0.09760          host host01-installer
  0    hdd  0.02440              osd.0                            up
1.00000  1.00000
 12    hdd  0.02440              osd.12                           up
1.00000  1.00000
 21    hdd  0.02440              osd.21                           up
1.00000  1.00000
 29    hdd  0.02440              osd.29                           up
1.00000  1.00000
 -4         0.09760          host host02
  1    hdd  0.02440              osd.1                            up
1.00000  1.00000
  9    hdd  0.02440              osd.9                            up
1.00000  1.00000
 18    hdd  0.02440              osd.18                           up
1.00000  1.00000
 28    hdd  0.02440              osd.28                           up
1.00000  1.00000
 -5         0.09760          host host03
  8    hdd  0.02440              osd.8                            up
1.00000  1.00000
 16    hdd  0.02440              osd.16                           up
1.00000  1.00000
 24    hdd  0.02440              osd.24                           up
1.00000  1.00000
 34    hdd  0.02440              osd.34                           up
1.00000  1.00000
 -7         0.29279      datacenter DC2
 -6         0.09760          host host04
  4    hdd  0.02440              osd.4                            up
1.00000  1.00000
 13    hdd  0.02440              osd.13                           up
1.00000  1.00000
 20    hdd  0.02440              osd.20                           up
```

```
1.00000  1.00000
 27    hdd  0.02440              osd.27                                  up
1.00000  1.00000
 -8         0.09760          host host05
  3    hdd  0.02440              osd.3                                   up
1.00000  1.00000
 10    hdd  0.02440              osd.10                                  up
1.00000  1.00000
 19    hdd  0.02440              osd.19                                  up
1.00000  1.00000
 30    hdd  0.02440              osd.30                                  up
1.00000  1.00000
 -9         0.09760          host host06
  7    hdd  0.02440              osd.7                                   up
1.00000  1.00000
 17    hdd  0.02440              osd.17                                  up
1.00000  1.00000
 26    hdd  0.02440              osd.26                                  up
1.00000  1.00000
 35    hdd  0.02440              osd.35                                  up
1.00000  1.00000
-11         0.29279      datacenter DC3
-10         0.09760          host host07
  5    hdd  0.02440              osd.5                                   up
1.00000  1.00000
 14    hdd  0.02440              osd.14                                  up
1.00000  1.00000
 23    hdd  0.02440              osd.23                                  up
1.00000  1.00000
 32    hdd  0.02440              osd.32                                  up
1.00000  1.00000
-12         0.09760          host host08
  2    hdd  0.02440              osd.2                                   up
1.00000  1.00000
 11    hdd  0.02440              osd.11                                  up
1.00000  1.00000
 22    hdd  0.02440              osd.22                                  up
1.00000  1.00000
 31    hdd  0.02440              osd.31                                  up
1.00000  1.00000
-13         0.09760          host host09
  6    hdd  0.02440              osd.6                                   up
1.00000  1.00000
 15    hdd  0.02440              osd.15                                  up
1.00000  1.00000
 25    hdd  0.02440              osd.25                                  up
1.00000  1.00000
 33    hdd  0.02440              osd.33                                  up
1.00000  1.00000
```

2. From the `cephadm` shell, verify that the mon daemons are deployed with CRUSH locations, as specified in step “1” on page 79.

```
ceph mon dump
```

Check that all mon daemons are in the output and that the correct CRUSH locations are added.

```
[root@host01 ~]# ceph mon dump
epoch 19
fsid b556497a-693a-11ef-b9d1-fa163e841fd7
last_changed 2024-09-03T12:47:08.419495+0000
created 2024-09-02T14:50:51.490781+0000
min_mon_release 19 (squid)
election_strategy: 3
0: [v2:10.0.67.43:3300/0,v1:10.0.67.43:6789/0] mon.host01-installer; crush_location
{datacenter=DC1}
1: [v2:10.0.67.20:3300/0,v1:10.0.67.20:6789/0] mon.host02; crush_location
{datacenter=DC1}
2: [v2:10.0.64.242:3300/0,v1:10.0.64.242:6789/0] mon.host03; crush_location
{datacenter=DC1}
3: [v2:10.0.66.17:3300/0,v1:10.0.66.17:6789/0] mon.host06; crush_location
{datacenter=DC2}
4: [v2:10.0.66.228:3300/0,v1:10.0.66.228:6789/0] mon.host09; crush_location
{datacenter=DC3}
5: [v2:10.0.65.125:3300/0,v1:10.0.65.125:6789/0] mon.host05; crush_location
{datacenter=DC2}
6: [v2:10.0.66.252:3300/0,v1:10.0.66.252:6789/0] mon.host07; crush_location
{datacenter=DC3}
7: [v2:10.0.64.145:3300/0,v1:10.0.64.145:6789/0] mon.host08; crush_location
{datacenter=DC3}
8: [v2:10.0.64.125:3300/0,v1:10.0.64.125:6789/0] mon.host04; crush_location
{datacenter=DC2}
dumped monmap epoch 19
```

3. Verify that the service spec and all location attributes are added correctly.
   a. Check the service name for mon daemons on the cluster, by using the **ceph orch ls** command.
   For example,

```
[root@host01 ~]# ceph orch ls
NAME                       PORTS        RUNNING  REFRESHED  AGE  PLACEMENT
alertmanager               ?:9093,9094      1/1  8m ago     6d   count:1
ceph-exporter                               9/9  8m ago     6d   *
crash                                       9/9  8m ago     6d   *
grafana                    ?:3000           1/1  8m ago     6d   count:1
mds.cephfs                                  3/3  8m ago     6d   label:mds
mgr                                         6/6  8m ago     6d   label:mgr
 mon                                         9/9  8m ago     5d   label:mon
node-exporter              ?:9100           9/9  8m ago     6d   *
osd.all-available-devices                    36  8m ago     6d   label:osd
prometheus                 ?:9095           1/1  8m ago     6d   count:1
rgw.rgw.1                  ?:80             3/3  8m ago     6d   label:rgw
```

   b. Confirm the mon daemon services, by using the **ceph orch ls mon --export** command.
   For example,

```
[root@host01 ~]# ceph orch ls mon --export
service_type: mon
service_name: mon
placement:
  label: mon
spec:
  crush_locations:
    host01-installer:
    - datacenter=DC1
    host02:
    - datacenter=DC1
    host03:
    - datacenter=DC1
    host04:
    - datacenter=DC2
    host05:
    - datacenter=DC2
    host06:
    - datacenter=DC2
    host07:
    - datacenter=DC3
    host08:
    - datacenter=DC3
    host09:
    - datacenter=DC3
```

4. Verify that the bootstrap was installed successfully, by running the **cephadm shell ceph -s** command.
   For more information, see “Verifying the cluster installation” on page 0.

Enabling three availability zones on the pool

Use this information to enable and integrate three availability zones within a generalized stretch cluster configuration.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- Root-level access to the nodes.
- The CRUSH location is set to the hosts.

1. Get the most recent CRUSH map and decompile the map into a text file.

   ```
   ceph osd getcrushmap > COMPILED_CRUSHMAP_FILENAME
   crushtool -d COMPILED_CRUSHMAP_FILENAME -o DECOMPILED_CRUSHMAP_FILENAME
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph osd getcrushmap > crush.map.bin
   [ceph: root@host01 /]# crushtool -d crush.map.bin -o crush.map.txt
   ```

2. Add the new CRUSH rule into the decompiled CRUSH map file from step “1” on page 85.
   In this example, the rule name is `3az_rule`.

   ```
   rule 3az_rule {
            id 1
            type replicated
            step take default
            step choose firstn 3 type datacenter
            step chooseleaf firstn 2 type host
            step emit
    }
   ```

   With this rule, the placement groups will be replicated with two copies in each of the three data centers.

3. Inject the CRUSH map to make the rule available to the cluster.

   ```
   crushtool -c DECOMPILED_CRUSHMAP_FILENAME -o COMPILED_CRUSHMAP_FILENAME
   ceph osd setcrushmap -i COMPILED_CRUSHMAP_FILENAME
   ```

   For example,

   ```
   [ceph: root@host01 /]# crushtool -c crush.map.txt -o crush2.map.bin
   [ceph: root@host01 /]# ceph osd setcrushmap -i crush2.map.bin
   ```

   You can verify that the rule was injected successfully, by using the following steps.

   a. List the rules on the cluster.

      ```
      ceph osd crush rule ls
      ```

      For example,

      ```
      [ceph: root@host01 /]# ceph osd crush rule ls
      replicated_rule
      ec86_pool
      3az_rule
      ```

   b. Dump the CRUSH rule.

      ```
      ceph osd crush rule dump CRUSH_RULE
      ```

      For example,

```
[ceph: root@host01 /]# ceph osd crush rule dump 3az_rule
{
    "rule_id": 1,
    "rule_name": "3az_rule",
    "type": 1,
    "steps": [
        {
            "op": "take",
            "item": -1,
            "item_name": "default"
        },
        {
            "op": "choose_firstn",
            "num": 3,
            "type": "datacenter"
        },
        {
            "op": "chooseleaf_firstn",
            "num": 2,
            "type": "host"
        },
        {
            "op": "emit"
        }
    ]
}
```

4. Set the MON election strategy to `connectivity`.

```
ceph mon set election_strategy connectivity
```

When updated successfully, the `election_strategy` is updated to `3`. The default `election_strategy` is `1`.

5. Optional: Verify the election strategy that was set in step “4” on page 86.

```
ceph mon dump
```

Check that all mon daemons are in the output and that the correct CRUSH locations are added.

```
[ceph: root@host01 /]# ceph mon dump
epoch 19
fsid b556497a-693a-11ef-b9d1-fa163e841fd7
last_changed 2024-09-03T12:47:08.419495+0000
created 2024-09-02T14:50:51.490781+0000
min_mon_release 19 (squid)
election_strategy: 3
0: [v2:10.0.67.43:3300/0,v1:10.0.67.43:6789/0] mon.host01-installer; crush_location
{datacenter=DC1}
1: [v2:10.0.67.20:3300/0,v1:10.0.67.20:6789/0] mon.host02; crush_location
{datacenter=DC1}
2: [v2:10.0.64.242:3300/0,v1:10.0.64.242:6789/0] mon.host03; crush_location
{datacenter=DC1}
3: [v2:10.0.66.17:3300/0,v1:10.0.66.17:6789/0] mon.host06; crush_location
{datacenter=DC2}
4: [v2:10.0.66.228:3300/0,v1:10.0.66.228:6789/0] mon.host09; crush_location
{datacenter=DC3}
5: [v2:10.0.65.125:3300/0,v1:10.0.65.125:6789/0] mon.host05; crush_location
{datacenter=DC2}
6: [v2:10.0.66.252:3300/0,v1:10.0.66.252:6789/0] mon.host07; crush_location
{datacenter=DC3}
7: [v2:10.0.64.145:3300/0,v1:10.0.64.145:6789/0] mon.host08; crush_location
{datacenter=DC3}
8: [v2:10.0.64.125:3300/0,v1:10.0.64.125:6789/0] mon.host04; crush_location
{datacenter=DC2}
dumped monmap epoch 19
```

6. Set the pool to associate with three availability zone stretch clusters.
For more information about available pool values, see Pool values.

```
ceph osd pool stretch
set POOL_NAME PEERING_CRUSH_BUCKET_COUNT PEERING_CRUSH_BUCKET_TARGET PEERING_CRUSH_BUC
KET_BARRIER CRUSH_RULE SIZE MIN_SIZE [--yes-i-really-mean-it]
```

Replace the variables as follows:

POOL_NAME

The name of the pool. It must be an existing pool, this command doesn't create a new pool.

PEERING_CRUSH_BUCKET_COUNT

The value is used along with peering_crush_bucket_barrier to determined whether the set of OSDs in the chosen acting set can peer with each other, based on the number of distinct buckets there are in the acting set.

PEERING_CRUSH_BUCKET_TARGET

This value is used along with peering_crush_bucket_barrier and size to calculate the value `bucket_max` which limits the number of OSDs in the same bucket from getting chose to be in the acting set of a PG.

PEERING_CRUSH_BUCKET_BARRIER

The type of bucket a pool is stretched across. For example, rack, row, or datacenter.

CRUSH_RULE

The crush rule to use for the stretch pool. The type of pool must match the type of **`crush_rule`** (replicated or erasure).

SIZE

The number of replicas for objects in the stretch pool.

MIN_SIZE

The minimum number of replicas required for I/O in the stretch pool.

Important: The **`--yes-i-really-mean-it`** flag is required when setting the *PEERING_CRUSH_BUCKET_COUNT* and *PEERING_CRUSH_BUCKET_TARGET* to be more than the number of buckets in the CRUSH map. Use the optional flag to confirm that you want to bypass the safety checks and set the values for a stretch pool.

For example,

```
[ceph: root@host01 /]# ceph osd pool stretch set pool01 2 3 datacenter 3az_rule 6 3
```

Note: To revert a pool to a nonstretched cluster, use the **`ceph osd pool stretch unset POOL_NAME`** command. Using this command does not unset the `crush_rule`, `size`, and `min_size` values. If needed, these need to be reset manually.

A success message is emitted that the pool stretch values were set correctly.

7. Optional: Verify the pools associated with the stretch clusters, by using the **`ceph osd pool stretch show`** commands.
For example,

```
[ceph: root@host01 /]# ceph osd pool stretch show pool01
pool: pool01
pool_id: 1
is_stretch_pool: 1
peering_crush_bucket_count: 2
peering_crush_bucket_target: 3
peering_crush_bucket_barrier: 8
crush_rule: 3az_rule
size: 6
min_size: 3
```

Adding OSD hosts with three availability zones

You can add Ceph OSDs with three availability zones on a generalized stretch cluster. The procedure is similar to the addition of the OSD hosts on a cluster where a generalized stretch cluster is not enabled.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Three availability zones enabled on a cluster. For more information, see "Enabling three availability zones on the pool" on page 85.
- Root-level access to the nodes.

For more information, see Adding OSDs.

1. From the node that contains the admin keyring, install the storage cluster's public SSH key in the root user's `authorized_keys` file on the new host.

   ```
   ssh-copy-id -f -i /etc/ceph/ceph.pub user@NEWHOST
   ```

   For example,

   ```
   [root@host10 ~]# ssh-copy-id -f -i /etc/ceph/ceph.pub root@host11
   [root@host10 ~]# ssh-copy-id -f -i /etc/ceph/ceph.pub root@host12
   ```

2. Add nodes by IP address.
   If you do not have DNS configured in your storage cluster environment, you can add the hosts by IP address, along with the host names.

   ```
   ceph orch host add HOSTNAME IP_ADDRESS
   ```

   For example,

   ```
   [ceph: root@host10 /]# ceph orch host add host11 10.10.128.69
   Added host 'host11' with addr '10.10.128.69'
   ```

3. Optional: Verify the status of the storage cluster and that each new host has been added by using the **`ceph orch host ls`** command.
   See that the new host has been added and that the **Status** of each host is blank in the output.

4. List the available devices to deploy OSDs.

   ```
   ceph orch device ls [--hostname=HOST11 HOST12] [--wide] [--refresh]
   ```

5. Deploy the OSDs on specific hosts or on all the available devices.
   Deploy in one of the following ways:
   - Create an OSD from a specific device on a specific host.

     ```
     ceph orch daemon add osd HOST:DEVICE_PATH
     ```

     For example,

     ```
     [ceph: root@host10 /]# ceph orch daemon add osd host11:/dev/sdb
     ```

   - Deploy OSDs on any available and unused devices.

     > **Important:** This command creates collocated WAL and DB devices. If you want to create non-collocated devices, do not use this command.

     ```
     ceph orch apply osd --all-available-devices
     ```

6. Move the OSD hosts under the CRUSH bucket.

   ```
   ceph osd crush move HOST datacenter=DATACENTER
   ```

```
[ceph: root@host10 /]# ceph osd crush move host10 datacenter=DC1
[ceph: root@host10 /]# ceph osd crush move host11 datacenter=DC2
[ceph: root@host10 /]# ceph osd crush move host12 datacenter=DC3
```

> **Note:** Ensure you add the same topology nodes on all sites. Issues might arise if hosts are added only on one site.

AFTER COMPLETING THE TASK

Verify that all hosts are moved to the assigned data centers, by using the **`ceph osd tree`** command.

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.

Setting and unsetting 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
`noin`	Prevents OSDs from being treated as `in` the cluster.
`noout`	Prevents OSDs from being treated as `out` of the cluster.
`noup`	Prevents OSDs from being treated as `up` and running.

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

Override use cases

Understand various overrride 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.

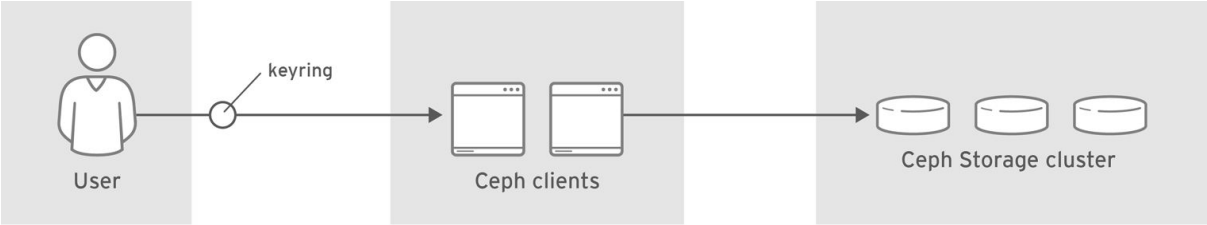
  > **Note:** Setting the cluster to full will prevent write operations.

- 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.

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.

Figure: OSD states



Note: `cephadm` manages the client keyrings for the Red Hat Ceph Storage cluster as long as the clients are within the scope of `cephadm`. Do not modify the keyrings that are managed by the `cephadm`, unless otherwise specified. If troubleshooting is needed, see “cephadm troubleshooting” on page 0.

Ceph user management background

When Ceph runs with authentication and authorization that is enabled, you must specify a username. If a username is not specified, Ceph uses the `client.admin` administrative user as the default username.

The `CEPH_ARGS` environment variable may also be used to avoid re-entry of the username 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. Also, administrative Ceph users must have permissions to run Ceph’s administrative commands.

For more information on configuring the use of authentication, see “Configuring Red Hat Ceph Storage” on page 0.

The following concepts can help you understand Ceph user management:

- “Storage cluster users” on page 91
- “Authorization capabilities” on page 92
- “Pool” on page 93
- “Namespace” on page 93

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 is always `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, instead, as `user1`. If you specify `--name` or `-n`, you must specify the type and name, such as `client.user1`. Use the type and name, 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. The gateway also 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:

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

Monitor Caps

Monitor capabilities include `r`, `w`, `x`, `allow profile CAP`, and `profile rbd`.
For 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`. OSD capabilities also allow for pool and namespace settings.
For example:

```
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 is not represented as a Ceph storage cluster daemon type.

"OSD Caps capabilities" on page 92 describes each capability.

OSD Caps capabilities

Capability	Description
`allow`	Precedes access settings for a daemon.
`r`	Gives the user read access. Required with monitors to retrieve the CRUSH map.
`w`	Gives the user write access to objects.
`x`	Gives the user the capability to call class methods (that is, both read and write) and to conduct `auth` operations on monitors.
`class-read`	Gives the user the capability to call class read methods. Subset of `x`.
`class-write`	Gives the user the capability to call class write methods. Subset of `x`.
`*`	Gives the user read, write and run permissions for a particular daemon or pool, and the ability to run admin commands.

Capability	Description
`profile osd`	Gives a user permissions to connect as an OSD to other OSDs or monitors. Conferred on OSDs to enable OSDs to handle replication heartbeat traffic and status reporting.
`profile bootstrap-osd`	Gives a user permissions to bootstrap an OSD so that they have permissions to add keys when bootstrapping an OSD.
`profile rbd`	Gives a user read/write access to the Ceph Block Devices.
`profile rbd-read-only`	Gives a user read-only access to the Ceph Block Devices.

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 that are 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 can 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.

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.

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

- To list the users in the storage cluster, run 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/JIhAAPsLBNG+SfC2eMVEFkl3vfA==
        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/QANBAATS9DuP3DbxEl86MTyKEmdw==
        caps: [mon] allow profile bootstrap-osd
client.bootstrap-rbd
        key: AQAHYEtgjxEBNBAANho25V9tWNNvIKnHknW59A==
        caps: [mon] allow profile bootstrap-rbd
client.bootstrap-rbd-mirror
        key: AQAHYEtgdE8BNBAAr6rLYxZci0b2hoIgH9GXYw==
        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: AQDuYUtgqgfdOhAAsyX+Mo35M+HFpURGad7nJA==
        caps: [mgr] profile crash
        caps: [mon] profile crash
client.crash.host03
        key: AQB98E5g5jHZAxAAklWSvmDsh2JaL5G7FvMrrA==
        caps: [mgr] profile crash
        caps: [mon] profile crash
client.rgw.test_realm.test_zone.host01.hgbvnq
        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: AQD0RE9gkxA4ExAAFXp3pLJWdIhsyTe2ZR6Ilw==
        caps: [mgr] allow rw
        caps: [mon] allow *
        caps: [osd] allow rwx tag rgw *=*
mgr.host01.hdhzwn
        key: AQAEYEtg3lhIBxAAmHodoIpdvnxK0llWF80ltQ==
        caps: [mds] allow *
        caps: [mon] profile mgr
```

```
        caps: [osd] allow *
mgr.host02.eobuuv
        key: AQAn6U5gzUuiABAA2Fed+jPM1xwb4XDYtrQxaQ==
        caps: [mds] allow *
        caps: [mon] profile mgr
        caps: [osd] allow *
mgr.host03.wquwpj
        key: AQAd6U5gIzWsLBAAbOKUKZlUcAVe9kBLfajMKw==
        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.

Displaying 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, run 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 `auid`, which isn't relevant to end users.

Adding a 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 Red Hat Ceph Storage cluster. The user's capabilities authorize the user to read, write, or run 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.

Modifying 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 ' '
   ```

Reference

For more information about capabilities, see “Ceph user management background” on page 91.

Deleting 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
   ```

Printing 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

- Print a user’s authentication key to standard output:

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

  Example

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

Using `ceph-volume` utility

Use the `ceph-volume` utility to prepare, list, create, activate, deactivate, batch, trigger, zap, and migrate Ceph OSDs.

The `ceph-volume` utility is a single-purpose command-line tool to deploy logical volumes as OSDs. It uses a plugin-type framework to deploy OSDs with different device technologies. The `ceph-volume` utility follows a similar workflow of the `ceph-disk` utility for deploying OSDs, with a predictable, and robust way of preparing, activating, and starting OSDs. Currently, the `ceph-volume` utility only supports the `lvm` plug-in, with the plan to support others technologies in the future.

> **Important:** The **`ceph-disk`** command is deprecated.

Running ceph-volume in a `cephadm` shell

> **Important:** Do not run `ceph-volume` from a container session that is started with `cephadm shell` when relying on the shell's default bind mounts.
> By design, `cephadm shell` omits several host bind mounts that `ceph-volume` expects, such as `/run/udev` and `/run/lvm`. As a result, running `ceph-volume` in this environment can fail or behave incorrectly.

Running `ceph-volume` manually, outside operations driven by `ceph orch` or `cephadm`, is not the normal operational path. This approach is intended mainly for debugging, testing, or development.

> **Important:** Using `ceph-volume` directly outside the `ceph orch` (`cephadm`) workflow is not encouraged. The utility assumes a host environment with access to device metadata and system services that are normally available on the host when operating under the `cephadm`-driven workflow.
> Although it is possible to extend the `cephadm shell` container environment by adding host bind mounts, this configuration is not typical. For guidance specific to your environment, contact IBM Support.

Ceph volume `lvm` plugin

By making use of LVM tags, the `lvm` sub-command is able to store and re-discover by querying devices associated with OSDs so they can be activated. This includes support for lvm-based technologies like `dm-cache` as well.

When using `ceph-volume`, the use of `dm-cache` is transparent, and treats `dm-cache` like a logical volume. The performance gains and losses when using `dm-cache` will depend on the specific workload. Generally, random and sequential reads will see an increase in performance at smaller block sizes. While random and sequential writes will see a decrease in performance at larger block sizes.

To use the LVM plugin, add `lvm` as a subcommand to the `ceph-volume` command within the `cephadm` shell:

```
[ceph: root@host01 /]# ceph-volume lvm
```

Following are the `lvm` subcommands:

- `prepare` - Format an LVM device and associate it with an OSD.
- `activate` - Discover and mount the LVM device associated with an OSD ID and start the Ceph OSD.
- `list` - List logical volumes and devices associated with Ceph.
- `batch` - Automatically size devices for multi-OSD provisioning with minimal interaction.
- `deactivate` - Deactivate OSDs.
- `create` - Create a new OSD from an LVM device.
- `trigger` - A systemd helper to activate an OSD.
- `zap` - Removes all data and filesystems from a logical volume or partition.
- `migrate` - Migrate BlueFS data from to another LVM device.
- `new-wal` - Allocate new WAL volume for the OSD at specified logical volume.
- `new-db` - Allocate new DB volume for the OSD at specified logical volume.

> **Note:** Using the **`create`** subcommand combines the **`prepare`** and **`activate`** subcommands into one subcommand.

Reference

For more information, see "Creating Ceph OSDs" on page 103.

Why does `ceph-volume` replace `ceph-disk`?

Previous versions of Ceph used the `ceph-disk` utility to prepare, activate, and create OSDs. The `ceph-disk` is replaced by the `ceph-volume` utility that aims to be a single purpose command-line tool to deploy logical volumes as OSDs, while maintaining a similar API to `ceph-disk` when preparing, activating, and creating OSDs.

How does `ceph-volume` work?

The `ceph-volume` is a modular tool that currently supports two ways of provisioning hardware devices, legacy `ceph-disk` devices and LVM (Logical Volume Manager) devices. The `ceph-volume lvm` command uses the LVM tags to store information about devices specific to Ceph and its relationship with OSDs. It uses these tags to later re-discover and query devices associated with OSDS so that it can activate them. It supports technologies based on LVM and `dm-cache` as well.

The `ceph-volume` utility uses `dm-cache` transparently and treats it as a logical volume. You might consider the performance gains and losses when using `dm-cache`, depending on the specific workload you are handling. Generally, the performance of random and sequential read operations increases at smaller block sizes; while the performance of random and sequential write operations decreases at larger block sizes. Using `ceph-volume` does not introduce any significant performance penalties.

Important: The `ceph-disk` utility is deprecated.

Note: The **`ceph-volume simple`** command can handle legacy `ceph-disk` devices, if these devices are still in use.

How does `ceph-disk` work?

The `ceph-disk` utility was required to support many different types of init systems, such as `upstart` or `sysvinit`, while being able to discover devices. For this reason, `ceph-disk` concentrates only on GUID Partition Table (GPT) partitions. Specifically on GPT GUIDs that label devices in a unique way to answer questions like:

- Is this device a `journal`?
- Is this device an encrypted data partition?
- Was the device partially prepared?

To solve these questions, `ceph-disk` uses UDEV rules to match the GUIDs.

What are disadvantages of using `ceph-disk`?

Using the UDEV rules to call `ceph-disk` can lead to a back-and-forth between the `ceph-disk systemd` unit and the `ceph-disk` executable. The process is very unreliable and time consuming and can cause OSDs to not come up at all during the boot process of a node. Moreover, it is hard to debug, or even replicate these problems given the asynchronous behavior of UDEV.

Because `ceph-disk` works with GPT partitions exclusively, it cannot support other technologies, such as Logical Volume Manager (LVM) volumes, or similar device mapper devices.

To ensure the GPT partitions work correctly with the device discovery workflow, `ceph-disk` requires a large number of special flags to be used. In addition, these partitions require devices to be exclusively owned by Ceph.

Preparing OSDs

The `prepare` subcommand prepares an OSD back-end object store and consumes logical volumes (LV) for both the OSD data and journal. It does not modify the logical volumes, except for adding some extra metadata tags using LVM. These tags make volumes easier to discover, and they also identify the volumes as part of the Ceph Storage Cluster and the roles of those volumes in the storage cluster.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- Root-level access to the OSD nodes.
- Optionally, create logical volumes. If you provide a path to a physical device, the subcommand turns the device into a logical volume. This approach is simpler, but you cannot configure or change the way the logical volume is created.

The BlueStore OSD backend supports the following configurations:

- A block device, a `block.wal` device, and a `block.db` device
- A block device and a `block.wal` device
- A block device and a `block.db` device
- A single block device

The `prepare` subcommand accepts a whole device or partition, or a logical volume for `block`.

For more information, see:

- "Activating OSDs" on page 101
- "Creating Ceph OSDs" on page 103

1. Extract the Ceph keyring.

   ```
   ceph auth get client.ID -o ceph.client.ID.keyring
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph auth get client.bootstrap-osd -o /var/lib/ceph/bootstrap-
   osd/ceph.keyring
   ```

2. Prepare the LVM volumes.

   ```
   ceph-volume lvm prepare --bluestore --data VOLUME_GROUP/LOGICAL_VOLUME
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph-volume lvm prepare --bluestore --data example_vg/data_lv
   ```

   a. To use a separate device for RocksDB, specify the **`--block.db`** and **`--block.wal`** options.

      ```
      ceph-volume lvm prepare --bluestore --block.db --block.wal --data VOLUME_GROUP/
      LOGICAL_VOLUME
      ```

      For example,

      ```
      ceph: root@host01 /]# ceph-volume lvm prepare --bluestore --block.db --block.wal
      --data example_vg/data_lv
      ```

   b. To encrypt data, use the **`--dmcrypt`** flag.

      ```
      ceph-volume lvm prepare --bluestore --dmcrypt --data VOLUME_GROUP/LOGICAL_VOLUME
      ```

      For example,

      ```
      [ceph: root@host01 /]# ceph-volume lvm prepare --bluestore --dmcrypt --data
      example_vg/data_lv
      ```

Listing devices

You can use the **`ceph-volume lvm list`** subcommand to list logical volumes and devices associated with a Ceph cluster, as long as they contain enough metadata to allow for that discovery. The output is grouped by the OSD ID associated with the devices. For logical volumes, the `devices key` is populated with the physical devices associated with the logical volume.

Sometimes, the output of the **`ceph -s`** command shows the following error message:

```
1 devices have fault light turned on
```

In such cases, you can list the devices with **ceph device ls-lights** command that gives the details about the lights on the devices. Based on the information, you can turn off the lights on the devices.

Prerequisites

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

Procedure

1. List the devices in the Ceph cluster:

   Example

   ```
   [ceph: root@host01 /]# ceph-volume lvm list


   ====== osd.6 =======

     [block]       /dev/ceph-83909f70-95e9-4273-880e-5851612cbe53/osd-
   block-7ce687d9-07e7-4f8f-a34e-d1b0efb89920

         block device              /dev/ceph-83909f70-95e9-4273-880e-5851612cbe53/osd-
   block-7ce687d9-07e7-4f8f-a34e-d1b0efb89920
         block uuid                4d7gzX-Nzxp-UUG0-bNxQ-Jacr-l0mP-IPD8cX
         cephx lockbox secret
         cluster fsid              1ca9f6a8-d036-11ec-8263-fa163ee967ad
         cluster name              ceph
         crush device class        None
         encrypted                 0
         osd fsid                  7ce687d9-07e7-4f8f-a34e-d1b0efb89920
         osd id                    6
         osdspec affinity          all-available-devices
         type                      block
         vdo                       0
         devices                   /dev/vdc
   ```

2. List the devices in the storage cluster with the lights.

   ```
   [ceph: root@host01 /]# ceph device ls-lights

   {
       "fault": [
           "SEAGATE_ST12000NM002G_ZL2KTGCK0000C149"
       ],
       "ident": []
   }
   ```

3. Turn off the lights on the device.

   ```
   ceph device light off DEVICE_NAME FAULT/INDENT --force
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph device light off SEAGATE_ST12000NM002G_ZL2KTGCK0000C149
   fault --force
   ```

Activating OSDs

The activation process for a Ceph OSD enables a `systemd` unit at boot time, which allows the correct OSD identifier and its UUID to be enabled and mounted.

Before you begin

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

- Ceph OSDs prepared by the `ceph-volume` utility.

Procedure

1. Get the OSD ID and OSD FSID from an OSD node.

   Example

   ```
   [ceph: root@host01 /]# ceph-volume lvm list
   ```

2. Activate the OSD.

   ```
   ceph-volume lvm activate --bluestore OSD_ID OSD_FSID
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph-volume lvm activate --bluestore 10 7ce687d9-07e7-4f8f-
   a34e-d1b0efb89920
   ```

   To activate all OSDs that are prepared for activation, use the `--all` option.

   Example

   ```
   [ceph: root@host01 /]# ceph-volume lvm activate --all
   ```

3. Optionally, you can use the `trigger` subcommand. This command cannot be used directly, and it is used by `systemd` so that it proxies input to `ceph-volume lvm activate`. This parses the metadata coming from systemd and startup, detecting the UUID and ID associated with an OSD.

   ```
   ceph-volume lvm trigger SYSTEMD_DATA
   ```

   Here the *SYSTEMD_DATA* is in *OSD_ID-OSD_FSID* format.

   Example

   ```
   [ceph: root@host01 /]# ceph-volume lvm trigger 10 7ce687d9-07e7-4f8f-a34e-d1b0efb89920
   ```

Reference

For more information, see:

- “Preparing OSDs” on page 99
- “Creating Ceph OSDs” on page 103

Deactivating OSDs

You can deactivate the Ceph OSDs using the **`ceph-volume lvm`** subcommand. This subcommand removes the volume groups and the logical volume.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the Ceph OSD node.
- The Ceph OSDs are activated using the `ceph-volume` utility.

Procedure

1. Get the OSD ID from the OSD node.

   ```
   [ceph: root@host01 /]# ceph-volume lvm list
   ```

2. Deactivate the OSD:

```
ceph-volume lvm deactivate OSD_ID
```

Example

```
[ceph: root@host01 /]# ceph-volume lvm deactivate 16
```

Reference

For more information, see:

- “Activating OSDs” on page 101
- “Preparing OSDs” on page 99
- “Creating Ceph OSDs” on page 103

Creating Ceph OSDs

The `create` subcommand calls the `prepare` subcommand, and then calls the `activate` subcommand.

Prerequisites

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

> **Note:** If you prefer to have more control over the creation process, you can use the **`prepare`** and **`activate`** subcommands separately to create the OSD instead of using **`create`**. You can use the two subcommands to gradually introduce new OSDs into a storage cluster, and avoid having to rebalance large amounts of data. Both approaches work the same way, except that using the **`create`** subcommand causes the OSD to become `up` and `in` immediately after completion.

Procedure

- Create an OSD.

  Syntax

  ```
  ceph-volume lvm create --bluestore --data VOLUME_GROUP/LOGICAL_VOLUME
  ```

  Example

  ```
  [root@osd ~]# ceph-volume lvm create --bluestore --data example_vg/data_lv
  ```

Reference

- “Preparing OSDs” on page 99
- “Activating OSDs” on page 101

Migrating BlueFS data

You can migrate the BlueStore file system (BlueFS) data, that is the RocksDB data, from the source volume to the target volume using the `migrate` LVM subcommand.

When using the `migrate` LVM subcommand, the source volume, except the main one, is removed on success.

LVM volumes are primarily for the target only.

The new volumes are attached to the OSD, replacing one of the source drives.

Following are the placement rules for the LVM volumes:

- If source list has DB or WAL volume, then the target device replaces it.

- If source list has slow volume only, then explicit allocation using the `new-db` or `new-wal` command is needed.

The `new-db` and `new-wal` commands attaches the given logical volume to the given OSD as a DB or a WAL volume respectively.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the Ceph OSD node.
- Ceph OSDs prepared by the `ceph-volume` utility.
- Volume groups and Logical volumes are created.

Procedure

1. Log in the `cephadm` shell:

   Example

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

2. Stop the OSD to which you have to add the DB or the WAL device:

   Example

   ```
   [ceph: root@host01 /]# ceph orch daemon stop osd.1
   ```

3. Mount the new devices to the container:

   Example

   ```
   [root@host01 ~]# cephadm shell --mount /var/lib/ceph/72436d46-ca06-11ec-9809-
   ac1f6b5635ee/osd.1:/var/lib/ceph/osd/ceph-1
   ```

4. Attach the given logical volume to OSD as a DB/WAL device:

   > **Note:** This command fails if the OSD has an attached DB.

   Syntax

   ```
   ceph-volume lvm new-db --osd-id OSD_ID --osd-fsid OSD_FSID --target VOLUME_GROUP_NAME/
   LOGICAL_VOLUME_NAME
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph-volume lvm new-db --osd-id 1 --osd-fsid
   7ce687d9-07e7-4f8f-a34e-d1b0efb89921 --target vgname/new_db
   [ceph: root@host01 /]# ceph-volume lvm new-wal --osd-id 1 --osd-fsid
   7ce687d9-07e7-4f8f-a34e-d1b0efb89921 --target vgname/new_wal
   ```

5. You can migrate BlueFS data in the following ways:

   - Move BlueFS data from main device to LV that is already attached as DB:

     Syntax

     ```
     ceph-volume lvm migrate --osd-id OSD_ID --osd-fsid OSD_UUID --from data --target
     VOLUME_GROUP_NAME/LOGICAL_VOLUME_NAME
     ```

     Example

     ```
     [ceph: root@host01 /]# ceph-volume lvm migrate --osd-id 1 --osd-fsid
     0263644D-0BF1-4D6D-BC34-28BD98AE3BC8 --from data --target vgname/db
     ```

- Move BlueFS data from shared main device to LV which shall be attached as a new DB:

  Syntax

  ```
  ceph-volume lvm migrate --osd-id OSD_ID --osd-fsid OSD_UUID --from data --target
  VOLUME_GROUP_NAME/LOGICAL_VOLUME_NAME
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph-volume lvm migrate --osd-id 1 --osd-fsid
  0263644D-0BF1-4D6D-BC34-28BD98AE3BC8 --from data --target vgname/new_db
  ```

- Move BlueFS data from DB device to new LV, and replace the DB device:

  Syntax

  ```
  ceph-volume lvm migrate --osd-id OSD_ID --osd-fsid OSD_UUID --from db --target
  VOLUME_GROUP_NAME/LOGICAL_VOLUME_NAME
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph-volume lvm migrate --osd-id 1 --osd-fsid
  0263644D-0BF1-4D6D-BC34-28BD98AE3BC8 --from db --target vgname/new_db
  ```

- Move BlueFS data from main and DB devices to new LV, and replace the DB device:

  Syntax

  ```
  ceph-volume lvm migrate --osd-id OSD_ID --osd-fsid OSD_UUID --from data db --
  target VOLUME_GROUP_NAME/LOGICAL_VOLUME_NAME
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph-volume lvm migrate --osd-id 1 --osd-fsid
  0263644D-0BF1-4D6D-BC34-28BD98AE3BC8 --from data db --target vgname/new_db
  ```

- Move BlueFS data from main, DB, and WAL devices to new LV, remove the WAL device, and replace the DB device:

  Syntax

  ```
  ceph-volume lvm migrate --osd-id OSD_ID --osd-fsid OSD_UUID --from data db wal --
  target VOLUME_GROUP_NAME/LOGICAL_VOLUME_NAME
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph-volume lvm migrate --osd-id 1 --osd-fsid
  0263644D-0BF1-4D6D-BC34-28BD98AE3BC8 --from data db --target vgname/new_db
  ```

- Move BlueFS data from main, DB, and WAL devices to the main device, remove the WAL and DB devices:

  Syntax

  ```
  ceph-volume lvm migrate --osd-id OSD_ID --osd-fsid OSD_UUID --from db wal --
  target VOLUME_GROUP_NAME/LOGICAL_VOLUME_NAME
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph-volume lvm migrate --osd-id 1 --osd-fsid
  0263644D-0BF1-4D6D-BC34-28BD98AE3BC8 --from db wal --target vgname/data
  ```

Expanding BlueFS DB device

Expand the storage of BlueStore File System.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Ceph OSDs are prepared by the `ceph-volume` utility.
- Volume groups and Logical volumes are created.

> **Note:** Run these steps on the host where the OSD is deployed.

You can expand the storage of the BlueStore file system (BlueFS) data that is the RocksDB data of `ceph-volume` created OSDs with the `ceph-bluestore` tool.

1. **Optional:** Inside the `cephadm` shell, list the devices in the Red Hat Ceph Storage cluster.

```
ceph-volume lvm list
```

   For example,

```
[ceph: root@host01 /]# ceph-volume lvm list

====== osd.3 =======

  [db]          /dev/db-test/db1

      block device              /dev/test/lv1
      block uuid                N5zoix-FePe-uExe-UngY-D9YG-BMs0-1tTDyB
      cephx lockbox secret
      cluster fsid              1a6112da-ed05-11ee-bacd-525400565cda
      cluster name              ceph
      crush device class
      db device                 /dev/db-test/db1
      db uuid                   1TUaDY-3mEt-fReP-cyB2-JyZ1-oUPa-hKPfo6
      encrypted                 0
      osd fsid                  94ff742c-7bfd-4fb5-8dc4-843d10ac6731
      osd id                    3
      osdspec affinity          None
      type                      db
      vdo                       0
      devices                   /dev/vdh

  [block]       /dev/test/lv1

      block device              /dev/test/lv1
      block uuid                N5zoix-FePe-uExe-UngY-D9YG-BMs0-1tTDyB
      cephx lockbox secret
      cluster fsid              1a6112da-ed05-11ee-bacd-525400565cda
      cluster name              ceph
      crush device class
      db device                 /dev/db-test/db1
      db uuid                   1TUaDY-3mEt-fReP-cyB2-JyZ1-oUPa-hKPfo6
      encrypted                 0
      osd fsid                  94ff742c-7bfd-4fb5-8dc4-843d10ac6731
      osd id                    3
      osdspec affinity          None
      type                      block
      vdo                       0
      devices                   /dev/vdg
```

2. Get the volume group information.

```
vgs
```

   For example,

```
[root@host01 ~]# vgs
VG                                        #PV #LV #SN Attr   VSize    VFree
db-test                                     1   1   0 wz--n- <200.00g <160.00g
test                                        1   1   0 wz--n- <200.00g <170.00g
```

3. Stop the Ceph OSD service.

```
systemctl stop SERVICE_ID
```

For example,

```
[root@host01 ~]# systemctl stop host01a6112da-ed05-11ee-
bacd-525400565cda@osd.3.service
```

4. Resize, shrink, and expand the logical volumes.

```
lvresize -l 100%FREE PATH_OF_DB_DEVICE
```

For example,

```
[root@host01 ~]# lvresize -l 100%FREE /dev/db-test/db1
Size of logical volume db-test/db1 changed from 40.00 GiB (10240 extents) to <160.00
GiB (40959 extents).
Logical volume db-test/db1 successfully resized.
```

5. Launch the `cephadm` shell.

```
cephadm shell -m /var/lib/ceph/CLUSTER_FSID/osd.OSD_ID:/var/lib/ceph/osd/ceph-OSD_ID:z
```

For example,

```
[root@host01 ~]# cephadm shell -m /var/lib/ceph/1a6112da-ed05-11ee-bacd-525400565cda/
osd.3:/var/lib/ceph/osd/ceph-3:z
```

The `ceph-bluestore-tool` needs to access the BlueStore data from within the `cephadm` shell container, so it must be bind-mounted. Use the `-m` option to make the BlueStore data available.

6. Check the size of the Rocks DB before expansion.

```
ceph-bluestore-tool show-label --path OSD_DIRECTORY_PATH
```

For example,

```
[ceph: root@host01 /]# ceph-bluestore-tool show-label --path /var/lib/ceph/osd/ceph-3/
inferring bluefs devices from bluestore path
{
    "/var/lib/ceph/osd/ceph-3/block": {
        "osd_uuid": "94ff742c-7bfd-4fb5-8dc4-843d10ac6731",
        "size": 32212254720,
        "btime": "2024-04-03T08:34:12.742848+0000",
        "description": "main",
        "bfm_blocks": "7864320",
        "bfm_blocks_per_key": "128",
        "bfm_bytes_per_block": "4096",
        "bfm_size": "32212254720",
        "bluefs": "1",
        "ceph_fsid": "1a6112da-ed05-11ee-bacd-525400565cda",
        "ceph_version_when_created": "ceph version 19.0.0-2493-gd82c9aa1
(d82c9aa17f09785fe698d262f9601d87bb79f962) squid (dev)",
        "created_at": "2024-04-03T08:34:15.637253Z",
        "elastic_shared_blobs": "1",
        "kv_backend": "rocksdb",
        "magic": "ceph osd volume v026",
        "mkfs_done": "yes",
        "osd_key": "AQCEFA1m9xuwABAAwKEHkASVbgB1GVt5jYC2Sg==",
        "osdspec_affinity": "None",
        "ready": "ready",
        "require_osd_release": "19",
        "whoami": "3"
    },
    "/var/lib/ceph/osd/ceph-3/block.db": {
        "osd_uuid": "94ff742c-7bfd-4fb5-8dc4-843d10ac6731",
        "size": 40794497536,
        "btime": "2024-04-03T08:34:12.748816+0000",
        "description": "bluefs db"
    }
}
```

7. Expand the BlueStore device.

```
ceph-bluestore-tool bluefs-bdev-expand --path OSD_DIRECTORY_PATH
```

For example,

```
[ceph: root@host01 /]# ceph-bluestore-tool bluefs-bdev-expand --path /var/lib/ceph/
osd/ceph-3/
inferring bluefs devices from bluestore path
1 : device size 0x27ffbfe000 : using 0x2300000(35 MiB)
2 : device size 0x780000000 : using 0x52000(328 KiB)
Expanding DB/WAL...
1 : expanding  to 0x171794497536
1 : size label updated to 171794497536
```

8. Verify the `block.db` is expanded.

```
ceph-bluestore-tool show-label --path OSD_DIRECTORY_PATH
```

For example,

```
[ceph: root@host01 /]# ceph-bluestore-tool show-label --path /var/lib/ceph/osd/ceph-3/
inferring bluefs devices from bluestore path
{
    "/var/lib/ceph/osd/ceph-3/block": {
        "osd_uuid": "94ff742c-7bfd-4fb5-8dc4-843d10ac6731",
        "size": 32212254720,
        "btime": "2024-04-03T08:34:12.742848+0000",
        "description": "main",
        "bfm_blocks": "7864320",
        "bfm_blocks_per_key": "128",
        "bfm_bytes_per_block": "4096",
        "bfm_size": "32212254720",
        "bluefs": "1",
        "ceph_fsid": "1a6112da-ed05-11ee-bacd-525400565cda",
        "ceph_version_when_created": "ceph version 19.0.0-2493-gd82c9aa1
(d82c9aa17f09785fe698d262f9601d87bb79f962) squid (dev)",
        "created_at": "2024-04-03T08:34:15.637253Z",
        "elastic_shared_blobs": "1",
        "kv_backend": "rocksdb",
        "magic": "ceph osd volume v026",
        "mkfs_done": "yes",
        "osd_key": "AQCEFA1m9xuwABAAwKEHkASVbgB1GVt5jYC2Sg==",
        "osdspec_affinity": "None",
        "ready": "ready",
        "require_osd_release": "19",
        "whoami": "3"
    },
    "/var/lib/ceph/osd/ceph-3/block.db": {
        "osd_uuid": "94ff742c-7bfd-4fb5-8dc4-843d10ac6731",
        "size": 171794497536,
        "btime": "2024-04-03T08:34:12.748816+0000",
        "description": "bluefs db"
    }
```

9. Exit the shell and restart the OSD.

```
systemctl start SERVICE_ID
```

For example,

```
[root@host01 ~]# systemctl start host01a6112da-ed05-11ee-
bacd-525400565cda@osd.3.service
osd.3              host01               running (15s)     0s ago  13m    46.9M
 4096M  19.0.0-2493-gd82c9aa1  3714003597ec  02150b3b6877
```

Using batch mode

The `batch` subcommand automates the creation of multiple OSDs when single devices are provided.

The `ceph-volume` command decides the best method to use to create the OSDs, based on drive type. Ceph OSD optimization depends on the available devices:

- If all devices are traditional hard drives, `batch` creates one OSD per device.
- If all devices are solid state drives, `batch` creates two OSDs per device.

– If there is a mix of traditional hard drives and solid state drives, `batch` uses the traditional hard drives for data, and creates the largest possible journal (`block.db`) on the solid state drive.

> **Note:** The **`batch`** subcommand does not support the creation of a separate logical volume for the write-ahead-log (`block.wal`) device.

Prerequisites

– A running Red Hat Ceph Storage cluster.

– Root-level access to the Ceph OSD nodes.

Procedure

1. To create OSDs on several drives:

```
ceph-volume lvm batch --bluestore PATH_TO_DEVICE [PATH_TO_DEVICE]
```

Example

```
[ceph: root@host01 /]# ceph-volume lvm batch --bluestore /dev/sda /dev/sdb /dev/
nvme0n1
```

Reference

– For more information, see “Creating Ceph OSDs” on page 103.

Zapping data

The `zap` subcommand removes all data and file systems from a logical volume or partition. Use the `zap` subcommand to zap logical volumes, partitions, or raw devices that are used by Ceph OSDs for reuse. Any file system present on the given logical volume or partition are removed and all data is purged.

Optionally, you can use the `--destroy` flag for complete removal of a logical volume, partition, or the physical device.

Prerequisites

– A running Red Hat Ceph Storage cluster.

– Root-level access to the Ceph OSD node.

Procedure

– Zap the logical volume:

Syntax

```
ceph-volume lvm zap VOLUME_GROUP_NAME/LOGICAL_VOLUME_NAME [--destroy]
```

Example

```
[ceph: root@host01 /]# ceph-volume lvm zap osd-vg/data-lv
```

– Zap the partition:

Syntax

```
ceph-volume lvm zap DEVICE_PATH_PARTITION [--destroy]
```

Example

```
[ceph: root@host01 /]# ceph-volume lvm zap /dev/sdc1
```

- Zap the raw device:

  Syntax

  ```
  ceph-volume lvm zap DEVICE_PATH --destroy
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph-volume lvm zap /dev/sdc --destroy
  ```

- Purge multiple devices with the OSD ID:

  Syntax

  ```
  ceph-volume lvm zap --destroy --osd-id OSD_ID
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph-volume lvm zap --destroy --osd-id 16
  ```

  Note: All the relative devices are zapped.

- Purge OSDs with the FSID:

  Syntax

  ```
  ceph-volume lvm zap --destroy --osd-fsid OSD_FSID
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph-volume lvm zap --destroy --osd-fsid 65d7b6b1-e41a-4a3c-
  b363-83ade63cb32b
  ```

  Note: All the relative devices are zapped.

Ceph performance benchmark

As a storage administrator, you can benchmark performance of the Red Hat Ceph Storage cluster.

Use this information to gain 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.

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.

Reference

- For more details about some of the available tools, see this Knowledge base article.

Benchmarking Ceph performance

Ceph includes the **rados bench** command to do performance benchmarking on a RADOS storage cluster. The command runs a write test and two types of read tests. The `--no-cleanup` option is important to use when testing both read and write performance.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

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

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

1. Create a new storage pool.
   For example,

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

2. Run a write test for 10 seconds to the storage pool created in the previous step.
   For 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         -         0
        1      16        16         0         0         0         -         0
        2      16        16         0         0         0         -         0
        3      16        16         0         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
   ```

3. Run a sequential read test for 10 seconds to the storage pool.
   For 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         -         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
```

4. Run a random read test for 10 seconds to the storage pool.
   For 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
```

5. Use additional parameters, as needed.
   - Increase the number of concurrent reads and writes, by using the **-t** parameter.
     The default value is 16 threads.
   - Adjust the size of the object being written, by using the **-b** parameter.
     The default object size is 4 MB.

     The maximum object size is 16 MB.
   - Control the names of the objects that get written during the benchmark test, by using the **--run-name** option, with the *LABEL* value.

> **Tip:** Run multiple copies of these benchmark tests to different pools to view the changes in performance from multiple clients.

Run multiple **rados bench** commands simultaneously, by changing the **--run-name** label for each running command instance. Changing the label prevents potential I/O errors that can occur when multiple clients are trying to access the same object. The multiple labels also allows for different clients to access different objects.

Use the **--run-name** option when trying to simulate a real world workload.

```
[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
```

6. Remove the data created by the **rados bench** command.
   For example,

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

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.
- Run 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
    ..
```

Reference

- For more information about the **rbd** command, see “Using Ceph Block Devices” on page 0.

Benchmarking CephFS performance

Benchmark Ceph File System (CephFS) performance with the FIO tool.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Root-level access to the node.
- FIO tool is installed on the nodes.
- Block Device or the Ceph File System mounted on the node.

You can use the FIO tool to benchmark Ceph File System (CephFS) performance. This tool can also be used to benchmark Ceph Block Device.

1. Navigate to the node or the application where the Block Device or the CephFS is mounted.

   ```
   [root@host01 ~]# cd /mnt/ceph-block-device
    [root@host01 ~]# cd /mnt/ceph-file-system
   ```

2. Run FIO command. Start the `bs` value from 4k and repeat in power of 2 increments (4k, 8k, 16k, 32k ... 128k... 512k, 1 m, 2 m, 4 m) and with different `iodepth` settings. Also run tests at your expected workload operation size.
   For example, 4 K tests with different `iodepth` values:

   ```
   fio --name=randwrite --rw=randwrite --direct=1 --ioengine=libaio --bs=4k --iodepth=32
   --size=5G --runtime=60 --group_reporting=1
   ```

   For example, 8 K tests with different `iodepth` values:

   ```
   fio --name=randwrite --rw=randwrite --direct=1 --ioengine=libaio --bs=8k --iodepth=32
    --size=5G --runtime=60 --group_reporting=1
   ```

   > **Note:** For more information on the usage of `fio` command, see the `fio` man page.

Benchmarking Ceph Object Gateway performance

Benchmark Ceph Object Gateway performance with the `s3cmd` tool.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Root-level access to the node.
- `s3cmd` installed on the nodes.

You can use the `s3cmd` tool to benchmark Ceph Object Gateway performance.

Use `get` and `put` requests to determine the performance.

1. Upload a file and measure the speed. The `time` command measures the duration of upload.

   ```
   time s3cmd put PATH_OF_SOURCE_FILE PATH_OF_DESTINATION_FILE
   ```

   For example,

   ```
   time s3cmd put /path-to-local-file s3://bucket-name/remote/file
   ```

   Replace `/path-to-local-file` with the file you want to upload and `s3://bucket-name/remote/file` with the destination in your S3 bucket.

2. Download a file and measure the speed. The `time` command measures the duration of download.

```
time s3cmd get PATH_OF_DESTINATION_FILE DESTINATION_PATH
```

For example,

```
time s3cmd get s3://bucket-name/remote/file /path-to-local-destination
```

Replace `s3://bucket-name/remote/file` with the S3 object that you want to download and `/path-to-local-destination` with the local directory where you want to save the file.

3. List all the objects in the specified bucket and measure response time.

```
time s3cmd ls s3://BUCKET_NAME
```

For example,

```
time s3cmd ls s3://bucket-name
```

4. Analyze the output to calculate upload and download speed. Measure response time reported by the `time` command.

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.

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 collections names represent a subsystem or an instance of a subsystem.

The following is a 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 placement groups.

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 I/O.

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 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

Displays the statistics on how the OSD is throttling.

Display 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:** 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
   ```

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.

Reference

- “Average count and sum” on page 119

Viewing the performance counters for users and buckets

The Ceph Object Gateway uses the performance counters to track metrics. You can visualize a cluster-wide view of the usage data over time in the Ceph Exporter port, which is usually, `9926`, which includes PUT operations for objects in a bucket.

PREREQUISITE

1. A running Red Hat Ceph Storage cluster with Ceph Object Gateway installed.
2. Monitoring stack enabled which includes Prometheus and `ceph-exporter`.

To track the operation metrics by users, set the `rgw_user_counters_cache` to `true` and to track the operation metrics by buckets, set the `rgw_bucket_counters_cache` to `true`.

You can use both `rgw_user_counters_cache_size` and `rgw_bucket_counters_cache_size` to set the number of entries in each cache.

Counters are evicted from a cache once the number of counters in the cache are greater than the cache size configuration variable. The counters that are evicted are the least recently used (LRU).

For example, if the number of buckets exceeded `rgw_bucket_counters_cache_size` by `1` and the counters with label `bucket1` were the last to be updated, the counters for `bucket1` get evicted from the cache. If S3 operations tracked by the operation metrics were done on `bucket1` after eviction, all the metrics in the cache for `bucket1` start at `0`.

Cache sizing can depend on several factors, which include the following:

- Number of users in the cluster.
- Number of buckets in the cluster.
- Memory usage of the Ceph Object Gateway.
- Disk and memory usage of Prometheus.
- To help calculate the Ceph Object Gateway’s memory usage of a cache, it should be noted that each cache entry, encompassing all the operation metrics, is 1360 bytes. This value is an estimate and subject to change if metrics are added or removed from the operation metrics list.

> **Important:**
> Since the operation metrics are labeled as performance counters, they live in memory. If the Ceph Object Gateway is restarted or crashes, all counters in the Ceph Object Gateway, whether in a cache or not, are lost.

1. Set the performance counters for users and buckets.
   a. Set the performance counter for users.
   For example,

```
[ceph: root@host01 /]# ceph config set client.rgw.rgw.1.host04
rgw_user_counters_cache true
```

b. Set the performance counter for buckets.
For example,

```
[ceph: root@host01 /]# ceph config set client.rgw.rgw.1.host04
rgw_bucket_counters_cache true
```

c. Restart the Ceph Object Gateway service.

```
[ceph: root@host01 /]# ceph orch restart rgw.rgw.1
```

2. Create users. For more information, see “User management” on page 0.
3. Create buckets and upload objects into the bucket.
   a. Configure `s3cmd`.

```
[root@host01 ~]# s3cmd --configure
```

   b. Create the S3 bucket.

```
s3cmd mb s3://NAME_OF_THE_BUCKET_FOR_S3
```

   For example,

```
[root@host01 ~]# s3cmd mb s3://bucket
Bucket 's3://bucket/' created
```

   c. Create your file, input all the data, upload buckets on S3.

```
s3cmd put FILE_NAME s3://NAME_OF_THE_BUCKET_FOR_S3
```

   For example,

```
[root@host01 ~]# s3cmd put test.txt s3://bucket

upload: 'test.txt' -> 's3://bucket/test.txt'  [1 of 1]
 21 of 21   100% in    1s    16.75 B/s  done
```

   d. Verify that the objects are uploaded.

```
config dump ceph daemon DAEMON_ID counter dump
```

4. Verify that the metrics are running on the local host.

```
http://RGW_IP_ADDRESS:CEPH-EXPORTER_PORT
```

   For example,

```
http://10.0.210.100:9926/
ceph_rgw_op_get_obj_ops{Bucket="bucket",instance_id="host04"} 0
ceph_rgw_op_get_obj_ops{User="admin",instance_id="host04"} 0
```

5. Verify the metrics on Prometheus.

```
https://RGW_IP_ADDRESS:PROMETHEUS_PORT
```

   For example,

```
https://10.0.210.100:9283/
```

Dump 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: 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
   ```

Reference

- To view a short description of each Monitor metric available, see "Ceph Monitor metrics" on page 119.

Average count and sum

The `avgcount` is the number of operations within this range and the `sum` is the total latency in seconds. To know approximately how much latency there is per operation, divide the `sum` by the `avgcount`.

All latency numbers have a bit field value of 5. This field contains floating point values for the average count and sum.

For a short description of each OSD metric available, see "Ceph OSD metrics" on page 123.

Ceph Monitor metrics

Use this information to understand different Ceph Monitor metrics.

Cluster metrics

"Cluster metrics table" on page 119 describes the cluster metrics for the `cluster` collection.

Cluster metrics table

Metric Name	Bit Field Value	Short Description
`num_mon`	2	Number of monitors
`num_mon_quorum`	2	Number of monitors in quorum

Metric Name	Bit Field Value	Short Description
num_osd	2	Total number of OSD
num_osd_up	2	Number of OSDs that are up
num_osd_in	2	Number of OSDs that are in cluster
osd_epoch	2	Current epoch of OSD map
osd_bytes	2	Total capacity of cluster in bytes
osd_bytes_used	2	Number of used bytes on cluster
osd_bytes_avail	2	Number of available bytes on cluster
num_pool	2	Number of pools
num_pg	2	Total number of placement groups
num_pg_active_clean	2	Number of placement groups in active+clean state
num_pg_active	2	Number of placement groups in active state
num_pg_peering	2	Number of placement groups in peering state
num_object	2	Total number of objects on cluster
num_object_degraded	2	Number of degraded (missing replicas) objects
num_object_misplaced	2	Number of misplaced (wrong location in the cluster) objects
num_object_unfound	2	Number of unfound objects
num_bytes	2	Total number of bytes of all objects
num_mds_up	2	Number of MDSs that are up
num_mds_in	2	Number of MDS that are in cluster
num_mds_failed	2	Number of failed MDS
mds_epoch	2	Current epoch of MDS map

Level database metrics

"Level database metrics table" on page 120 describes the level database metrics for the `leveldb` collection.

Level database metrics table

Metric Name	Bit Field Value	Short Description
leveldb_get	10	Gets
leveldb_transaction	10	Transactions

Metric Name	Bit Field Value	Short Description
leveldb_compact	10	Compactions
leveldb_compact_range	10	Compactions by range
leveldb_compact_queue_merge	10	Merging of ranges in compaction queue
leveldb_compact_queue_len	2	Length of compaction queue

General monitor metrics

"General monitor metrics" on page 121 describes the general monitor metrics for the mon collection.

General monitor metrics table

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

Paxos metrics

"Paxos metrics" on page 121 describes the Paxos metrics for the paxos collection.

Paxos metrics table

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

Metric Name	Bit Field Value	Short Description
begin_bytes	6	Data in transaction on begin
begin_latency	5	Latency of begin operation
commit	10	Commits
commit_keys	6	Keys in transaction on commit
commit_bytes	6	Data in transaction on commit
commit_latency	5	Commit latency
collect	10	Peon collects
collect_keys	6	Keys in transaction on peon collect
collect_bytes	6	Data in transaction on peon collect
collect_latency	5	Peon collect latency
collect_uncommitted	10	Uncommitted values in started and handled collects
collect_timeout	10	Collect timeouts
accept_timeout	10	Accept timeouts
lease_ack_timeout	10	Lease acknowledgment timeouts
lease_timeout	10	Lease timeouts
store_state	10	Store a shared state on disk
store_state_keys	6	Keys in transaction in stored state
store_state_bytes	6	Data in transaction in stored state
store_state_latency	5	Storing state latency
share_state	10	Sharing of state
share_state_keys	6	Keys in shared state
share_state_bytes	6	Data in shared state
new_pn	10	New proposal number queries
new_pn_latency	5	New proposal number getting latency

Throttle metrics

"Throttle metrics" on page 122 describes the throttle metrics for the `throttle-*` collection.

Throttle Metrics Table		
Metric Name	**Bit Field Value**	**Short Description**
val	10	Currently available throttle

Metric Name	Bit Field Value	Short Description
max	10	Max value for throttle
get	10	Gets
get_sum	10	Got data
get_or_fail_fail	10	Get blocked during get_or_fail
get_or_fail_success	10	Successful get during get_or_fail
take	10	Takes
take_sum	10	Taken data
put	10	Puts
put_sum	10	Put data
wait	5	Waiting latency

Ceph OSD metrics

Use this information to understand different Ceph OSD metrics.

Write back throttle metrics

"Write back throttle metrics table" on page 123 describes the write back throttle metrics for the `WBThrottle` collection.

Write back throttle metrics table

Metric Name	Bit Field Value	Short Description
bytes_dirtied	2	Dirty data
bytes_wb	2	Written data
ios_dirtied	2	Dirty operations
ios_wb	2	Written operations
inodes_dirtied	2	Entries waiting for write
inodes_wb	2	Written entries

Level database metrics

"Level database metrics" on page 123 describes the level database metrics for the `leveldb` collection.

Level database metrics table			
Collection Name	**Metric Name**	**Bit Field Value**	**Short Description**
`leveldb`	`leveldb_get`	10	Gets
	`leveldb_transaction`	10	Transactions
	`leveldb_compact`	10	Compactions
	`leveldb_compact_range`	10	Compactions by range
	`leveldb_compact_queue_merge`	10	Merging of ranges in compaction queue
	`leveldb_compact_queue_len`	2	Length of compaction queue

Objecter metrics

"Objecter metrics" on page 124 describes the objecter metrics for the `objecter` collection.

Objecter metrics table		
Metric Name	**Bit Field Value**	**Short Description**
`op_active`	2	Active operations
`op_laggy`	2	Laggy operations
`op_send`	10	Sent operations
`op_send_bytes`	10	Sent data
`op_resend`	10	Resent operations
`op_ack`	10	Commit callbacks
`op_commit`	10	Operation commits
`op`	10	Operation
`op_r`	10	Read operations
`op_w`	10	Write operations
`op_rmw`	10	Read-modify-write operations
`op_pg`	10	PG operation
`osdop_stat`	10	Stat operations
`osdop_create`	10	Create object operations
`osdop_read`	10	Read operations
`osdop_write`	10	Write operations
`osdop_writefull`	10	Write full object operations
`osdop_append`	10	Append operation
`osdop_zero`	10	Set object to zero operations

Metric Name	Bit Field Value	Short Description
osdop_truncate	10	Truncate object operations
osdop_delete	10	Delete object operations
osdop_mapext	10	Map extent operations
osdop_sparse_read	10	Sparse read operations
osdop_clonerange	10	Clone range operations
osdop_getxattr	10	Get xattr operations
osdop_setxattr	10	Set xattr operations
osdop_cmpxattr	10	Xattr comparison operations
osdop_rmxattr	10	Remove xattr operations
osdop_resetxattrs	10	Reset xattr operations
osdop_tmap_up	10	TMAP update operations
osdop_tmap_put	10	TMAP put operations
osdop_tmap_get	10	TMAP get operations
osdop_call	10	Call (run) operations
osdop_watch	10	Watch by object operations
osdop_notify	10	Notify about object operations
osdop_src_cmpxattr	10	Extended attribute comparison in multi operations
osdop_other	10	Other operations
linger_active	2	Active lingering operations
linger_send	10	Sent lingering operations
linger_resend	10	Resent lingering operations
linger_ping	10	Sent pings to lingering operations
poolop_active	2	Active pool operations
poolop_send	10	Sent pool operations
poolop_resend	10	Resent pool operations
poolstat_active	2	Active get pool stat operations
poolstat_send	10	Pool stat operations sent
poolstat_resend	10	Resent pool stats
statfs_active	2	Statfs operations
statfs_send	10	Sent FS stats

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

Read and write operations metrics

"Read and write operations metrics" on page 126 describes the read and write operations metrics for the osd collection.

Read and write operations metrics table

Metric Name	Bit Field Value	Short Description
op_wip	2	Replication operations currently being processed (primary)
op_in_bytes	10	Client operations total write size
op_out_bytes	10	Client operations total read size
op_latency	5	Latency of client operations (including queue time)
op_process_latency	5	Latency of client operations (excluding queue time)
op_r	10	Client read operations
op_r_out_bytes	10	Client data read
op_r_latency	5	Latency of read operation (including queue time)
op_r_process_latency	5	Latency of read operation (excluding queue time)
op_w	10	Client write operations
op_w_in_bytes	10	Client data written
op_w_rlat	5	Client write operation readable/ applied latency

Metric Name	Bit Field Value	Short Description
op_w_latency	5	Latency of write operation (including queue time)
op_w_process_latency	5	Latency of write operation (excluding queue time)
op_rw	10	Client read-modify-write operations
op_rw_in_bytes	10	Client read-modify-write operations write in
op_rw_out_bytes	10	Client read-modify-write operations read out
op_rw_rlat	5	Client read-modify-write operation readable/applied latency
op_rw_latency	5	Latency of read-modify-write operation (including queue time)
op_rw_process_latency	5	Latency of read-modify-write operation (excluding queue time)
subop	10	Sub operations
subop_in_bytes	10	Sub operations total size
subop_latency	5	Sub operations latency
subop_w	10	Replicated writes
subop_w_in_bytes	10	Replicated written data size
subop_w_latency	5	Replicated writes latency
subop_pull	10	Sub operations pull requests
subop_pull_latency	5	Sub operations pull latency
subop_push	10	Sub operations push messages
subop_push_in_bytes	10	Sub operations pushed size
subop_push_latency	5	Sub operations push latency
pull	10	Pull requests sent
push	10	Push messages sent
push_out_bytes	10	Pushed size
push_in	10	Inbound push messages
push_in_bytes	10	Inbound pushed size
recovery_ops	10	Started recovery operations
loadavg	2	CPU load

Metric Name	Bit Field Value	Short Description
buffer_bytes	2	Total allocated buffer size
numpg	2	Placement groups
numpg_primary	2	Placement groups for which this osd is primary
numpg_replica	2	Placement groups for which this osd is replica
numpg_stray	2	Placement groups ready to be deleted from this osd
heartbeat_to_peers	2	Heartbeat (ping) peers we send to
heartbeat_from_peers	2	Heartbeat (ping) peers we recv from
map_messages	10	OSD map messages
map_message_epochs	10	OSD map epochs
map_message_epoch_dups	10	OSD map duplicates
stat_bytes	2	OSD size
stat_bytes_used	2	Used space
stat_bytes_avail	2	Available space
copyfrom	10	RADOS *copy-from* operations
tier_promote	10	Tier promotions
tier_flush	10	Tier flushes
tier_flush_fail	10	Failed tier flushes
tier_try_flush	10	Tier flush attempts
tier_try_flush_fail	10	Failed tier flush attempts
tier_evict	10	Tier evictions
tier_whiteout	10	Tier whiteouts
tier_dirty	10	Dirty tier flag set
tier_clean	10	Dirty tier flag cleaned
tier_delay	10	Tier delays (agent waiting)
tier_proxy_read	10	Tier proxy reads
agent_wake	10	Tiering agent wake up
agent_skip	10	Objects skipped by agent
agent_flush	10	Tiering agent flushes
agent_evict	10	Tiering agent evictions

Metric Name	Bit Field Value	Short Description
object_ctx_cache_hit	10	Object context cache hits
object_ctx_cache_total	10	Object context cache lookups
ceph_cluster_osd_blocklist_count	2	Number of clients blocklisted

Recovery state metrics

"Recovery state metrics" on page 129 describes the recovery state metrics for the recoverystate_perf collection.

Recovery state metrics table

Metric Name	Bit Field Value	Short Description
initial_latency	5	Initial recovery state latency
started_latency	5	Started recovery state latency
reset_latency	5	Reset recovery state latency
start_latency	5	Start recovery state latency
primary_latency	5	Primary recovery state latency
peering_latency	5	Peering recovery state latency
backfilling_latency	5	Backfilling recovery state latency
waitremotebackfillreserved_latency	5	Wait remote backfill reserved recovery state latency
waitlocalbackfillreserved_latency	5	Wait local backfill reserved recovery state latency
notbackfilling_latency	5	Notbackfilling recovery state latency
repnotrecovering_latency	5	Repnotrecovering recovery state latency
repwaitrecoveryreserved_latency	5	Rep wait recovery reserved recovery state latency
repwaitbackfillreserved_latency	5	Rep wait backfill reserved recovery state latency
RepRecovering_latency	5	RepRecovering recovery state latency
activating_latency	5	Activating recovery state latency
waitlocalrecoveryreserved_latency	5	Wait local recovery reserved recovery state latency
waitremoterecoveryreserved_latency	5	Wait remote recovery reserved recovery state latency
recovering_latency	5	Recovering recovery state latency

Metric Name	Bit Field Value	Short Description
recovered_latency	5	Recovered recovery state latency
clean_latency	5	Clean recovery state latency
active_latency	5	Active recovery state latency
replicaactive_latency	5	Replicaactive recovery state latency
stray_latency	5	Stray recovery state latency
getinfo_latency	5	Getinfo recovery state latency
getlog_latency	5	Getlog recovery state latency
waitactingchange_latency	5	Waitactingchange recovery state latency
incomplete_latency	5	Incomplete recovery state latency
getmissing_latency	5	Get missing recovery state latency
waitupthru_latency	5	Waitupthru recovery state latency

OSD throttle metrics

"OSD throttle metrics" on page 130 describes the OSD throttle metrics for the `throttle-*` collection.

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

Ceph Object Gateway metrics

Use this information to understand different Ceph Object Gateway metrics.

- "Ceph Object Gateway client metrics" on page 131
- "Objecter metrics" on page 131

– “Ceph Object Gateway throttle metrics” on page 134

Ceph Object Gateway client metrics

“Ceph Object Gateway client metrics” on page 131 describes the Ceph Object Gateway client metrics for the `client.rgw.<rgw_node_name>` collection.

Ceph Object Gateway client metrics table

Metric Name	Bit Field Value	Short Description
`req`	10	Requests
`failed_req`	10	Aborted requests
`copy_obj_ops`	10	Copy objects
`copy_obj_bytes`	10	Size of copy objects
`copy_obj_lat`	10	Copy object latency
`del_obj_ops`	10	Delete objects
`del_obj_bytes`	10	Size of delete objects
`del_obj_lat`	10	Delete object latency
`del_bucket_ops`	10	Delete Buckets
`del_bucket_lat`	10	Delete bucket latency
`get`	10	Gets
`get_b`	10	Size of gets
`get_initial_lat`	5	Get latency
`list_obj_ops`	10	List objects
`list_obj_lat`	10	List object latency
`list_buckets_ops`	10	List buckets
`list_buckets_lat`	10	List buckets latency
`put`	10	Puts
`put_b`	10	Size of puts
`put_initial_lat`	5	Put latency
`qlen`	2	Queue length
`qactive`	2	Active requests queue
`cache_hit`	10	Cache hits
`cache_miss`	10	Cache miss
`keystone_token_cache_hit`	10	Keystone token cache hits
`keystone_token_cache_miss`	10	Keystone token cache miss

Objecter metrics

“Objecter metrics table” on page 132 describes the objecter metrics for the `objecter` collection.

Objecter metrics table		
Metric Name	**Bit Field Value**	**Short Description**
`op_active`	2	Active operations
`op_laggy`	2	Laggy operations
`op_send`	10	Sent operations
`op_send_bytes`	10	Sent data
`op_resend`	10	Resent operations
`op_ack`	10	Commit callbacks
`op_commit`	10	Operation commits
`op`	10	Operation
`op_r`	10	Read operations
`op_w`	10	Write operations
`op_rmw`	10	Read-modify-write operations
`op_pg`	10	PG operation
`osdop_stat`	10	Stat operations
`osdop_create`	10	Create object operations
`osdop_read`	10	Read operations
`osdop_write`	10	Write operations
`osdop_writefull`	10	Write full object operations
`osdop_append`	10	Append operation
`osdop_zero`	10	Set object to zero operations
`osdop_truncate`	10	Truncate object operations
`osdop_delete`	10	Delete object operations
`osdop_mapext`	10	Map extent operations
`osdop_sparse_read`	10	Sparse read operations
`osdop_clonerange`	10	Clone range operations
`osdop_getxattr`	10	Get xattr operations
`osdop_setxattr`	10	Set xattr operations
`osdop_cmpxattr`	10	Xattr comparison operations
`osdop_rmxattr`	10	Remove xattr operations
`osdop_resetxattrs`	10	Reset xattr operations
`osdop_tmap_up`	10	TMAP update operations

Metric Name	Bit Field Value	Short Description
osdop_tmap_put	10	TMAP put operations
osdop_tmap_get	10	TMAP get operations
osdop_call	10	Call (run) operations
osdop_watch	10	Watch by object operations
osdop_notify	10	Notify about object operations
osdop_src_cmpxattr	10	Extended attribute comparison in multi operations
osdop_other	10	Other operations
linger_active	2	Active lingering operations
linger_send	10	Sent lingering operations
linger_resend	10	Resent lingering operations
linger_ping	10	Sent pings to lingering operations
poolop_active	2	Active pool operations
poolop_send	10	Sent pool operations
poolop_resend	10	Resent pool operations
poolstat_active	2	Active get pool stat operations
poolstat_send	10	Pool stat operations sent
poolstat_resend	10	Resent pool stats
statfs_active	2	Statfs operations
statfs_send	10	Sent FS stats
statfs_resend	10	Resent FS stats
command_active	2	Active commands
command_send	10	Sent commands
command_resend	10	Resent commands
map_epoch	2	OSD map epoch
map_full	10	Full OSD maps received
map_inc	10	Incremental OSD maps received
osd_sessions	2	Open sessions
osd_session_open	10	Sessions opened
osd_session_close	10	Sessions closed
osd_laggy	2	Laggy OSD sessions

Ceph Object Gateway throttle metrics

"Ceph Object Gateway throttle metrics table" on page 134 describes the Ceph Object Gateway throttle metrics for the `throttle-*` collection.

Ceph Object Gateway throttle metrics table

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

mClock OSD scheduler

As a storage administrator, you can implement the Red Hat Ceph Storage's quality of service (QoS) using mClock queueing scheduler. This is based on an adaptation of the mClock algorithm called dmClock.

The mClock OSD scheduler provides the desired QoS using configuration profiles to allocate proper reservation, weight, and limit tags to the service types.

The mClock OSD scheduler performs the QoS calculations for the different device types, that is SSD or HDD, by using the OSD's IOPS capability (determined automatically) and maximum sequential bandwidth capability (See `osd_mclock_max_sequential_bandwidth_hdd` and `osd_mclock_max_sequential_bandwidth_ssd` in the mclock configuration options.

Comparison of mClock OSD scheduler with WPQ OSD scheduler

The mClock OSD scheduler replaces the Weighted Priority Queue (WPQ) OSD scheduler as a default scheduler in Red Hat Ceph Storage 7.

> **Important:** The mClock scheduler is supported for BlueStore OSDs. For Filestore OSDs the `osd_op_queue` is set to `wpq` and it is enforced even if the user attempts to change it.

The WPQ OSD scheduler features a strict sub-queue, which is de-queued before the normal queue. The WPQ removes operations from a queue in relation to their priorities to prevent depletion of any queue. This helps in cases where some Ceph OSDs are more overloaded than others.

The mClock OSD scheduler currently features an immediate queue, into which operations that require immediate response are queued. The immediate queue is not handled by mClock and functions as a simple first in, first out queue and is given the first priority.

Operations, such as OSD replication operations, OSD operation replies, peering, recoveries marked with the highest priority, and so forth, are queued into the immediate queue. All other operations are enqueued into the mClock queue that works according to the mClock algorithm.

The mClock queue, `mclock_scheduler`, prioritizes operations based on which bucket they belong to, that is `pg recovery`, `pg scrub`, `snap trim`, `client op`, and `pg deletion`.

With background operations in progress, the average client throughputs, that is the input and output operations per second (IOPS), are significantly higher and latencies are lesser with the mClock profiles when compared to the WPQ scheduler. That is because of mClock's effective allocation of the QoS parameters.

Reference

- For automated OSD capacity determination, see mClock profiles.

Allocation of input and output resources

This section describes how the QoS controls work internally with reservation, limit, and weight allocation. The user is not expected to set these controls as the mClock profiles automatically set them. Tuning these controls can only be performed using the available mClock profiles.

The dmClock algorithm allocates the input and output (I/O) resources of the Ceph cluster in proportion to weights. It implements the constraints of minimum reservation and maximum limitation to ensure the services can compete for the resources fairly.

Currently, the `mclock_scheduler` operation queue divides Ceph services involving I/O resources into following buckets:

- `client op`: the input and output operations per second (IOPS) issued by a client.
- `pg deletion`: the IOPS issued by primary Ceph OSD.
- `snap trim`: the snapshot trimming-related requests.
- `pg recovery`: the recovery-related requests.
- `pg scrub`: the scrub-related requests.

The resources are partitioned using the following three sets of tags, meaning that the share of each type of service is controlled by these three tags:

- Reservation
- Limit
- Weight

Reservation

The minimum IOPS allocated for the service. The more reservation a service has, the more resources it is guaranteed to possess, as long as it requires so.

For example, a service with the reservation set to 0.1 (or 10%) always has 10% of the OSD's IOPS capacity allocated for itself. Therefore, even if the clients start to issue large amounts of I/O requests, they do not exhaust all the I/O resources and the service's operations are not depleted even in a cluster with high load.

Limit

The maximum IOPS allocated for the service. The service does not get more than the set number of requests per second serviced, even if it requires so and no other services are competing with it. If a service crosses the enforced limit, the operation remains in the operation queue until the limit is restored.

> **Note:** If the value is set to `0` (disabled), the service is not restricted by the limit setting and it can use all the resources if there is no other competing operation. This is represented as "MAX" in the mClock profiles.

> **Note:** The reservation and limit parameter allocations are per-shard, based on the type of backing device, that is HDD or SSD, under the Ceph OSD.

Weight

The proportional share of capacity if extra capacity or system is not enough. The service can use a larger portion of the I/O resource, if its weight is higher than its competitor's.

Note: The reservation and limit values for a service are specified in terms of a proportion of the total IOPS capacity of the OSD. The proportion is represented as a percentage in the mClock profiles. The weight does not have a unit. The weights are relative to one another, so if one class of requests has a weight of 9 and another a weight of 1, then the requests are performed at a 9 to 1 ratio. However, that only happens once the reservations are met and those values include the operations performed under the reservation phase.

Important: If the weight is set to `W`, then for a given class of requests the next one that enters has a weight tag of `1/W` and the previous weight tag, or the current time, whichever is larger. That means, if `W` is too large and thus `1/W` is too small, the calculated tag might never be assigned as it gets a value of the current time. Therefore, values for weight should be always under the number of requests expected to be serviced each second.

Factors impacting mClock operation queues

Understand the different factors that can reduce the impact of the mClock operation queues within Red Hat Ceph Storage.

There are three factors that can reduce the impact of the mClock operation queues within Red Hat Ceph Storage

- The number of shards for client operations.
- The number of operations in the operation sequencer.
- The usage of distributed system for Ceph OSDs

The number of shards for client operations

Requests to a Ceph OSD are sharded by their placement group identifier. Each shard has its own mClock queue and these queues neither interact, nor share information amongst them.

The number of shards can be controlled with these configuration options:

- `osd_op_num_shards`
- `osd_op_num_shards_hdd`
- `osd_op_num_shards_ssd`

A lesser number of shards increase the impact of the mClock queues, but might have other damaging effects.

Note: Use the default number of shards as defined by the configuration options `osd_op_num_shards`, `osd_op_num_shards_hdd`, and `osd_op_num_shards_ssd`.

The number of operations in the operation sequencer

Requests are transferred from the operation queue to the operation sequencer, in which they are processed. The mClock scheduler is located in the operation queue. It determines which operation to transfer to the operation sequencer.

The number of operations allowed in the operation sequencer is a complex issue. The aim is to keep enough operations in the operation sequencer so it always works on some, while it waits for disk and network access to complete other operations.

However, mClock no longer has control over an operation that is transferred to the operation sequencer. Therefore, to maximize the impact of mClock, the goal is also to keep as few operations in the operation sequencer as possible.

The configuration options that influence the number of operations in the operation sequencer are:

- `bluestore_throttle_bytes`

- `bluestore_throttle_deferred_bytes`
- `bluestore_throttle_cost_per_io`
- `bluestore_throttle_cost_per_io_hdd`
- `bluestore_throttle_cost_per_io_ssd`

> **Note:** Use the default values as defined by the `bluestore_throttle_bytes` and `bluestore_throttle_deferred_bytes` options. However, these options can be determined during the benchmarking phase.

The usage of distributed system for Ceph OSDs

The third factor that affects the impact of the mClock algorithm is the usage of a distributed system, where requests are made to multiple Ceph OSDs, and each Ceph OSD can have multiple shards.

> **Note:** dmClock is the distributed version of mClock.

Reference

- For more details about `osd_op_num_shards_hdd` and `osd_op_num_shards_ssd` parameters, see Object Storage Daemon (OSD) configuration options.
- For more details about BlueStore throttle parameters, see BlueStore configuration options.
- Manually benchmarking OSDs

mClock configuration

The mClock profiles hide the low-level details from users, making it easier to configure and use mClock.

The following input parameters are required for an mClock profile to configure the quality of service (QoS) related parameters:

- The total capacity of input and output operations per second (IOPS) for each Ceph OSD. This is determined automatically.
- The maximum sequential bandwidth capacity (MiB/s) of each OS. See `osd_mclock_max_sequential_bandwidth_[hdd/ssd]` option.
- An mClock profile type to be enabled. The default is `balanced`.

Using the settings in the specified profile, a Ceph OSD determines and applies the lower-level mClock and Ceph parameters. The parameters applied by the mClock profile make it possible to tune the QoS between the client I/O and background operations in the OSD.

Reference

- For automated OSD capacity determination, see “Ceph OSD capacity determination” on page 147 .

mClock clients

The mClock scheduler handles requests from different types of Ceph services. Each service is considered by mClock as a type of client. Depending on the type of requests handled, mClock clients are classified into different bucket types.

The different bucket classification types are as follows:

- Client - Handles input and output (I/O) requests issued by external clients of Ceph.
- Background recovery - Handles internal recovery requests.
- Background best-effort - Handles internal backfill, scrub, snap trim, and placement group (PG) deletion requests.

The mClock scheduler derives the cost of an operation used in the QoS calculations from **osd_mclock_max_capacity_iops_hdd** | **osd_mclock_max_capacity_iops_ssd**, **osd_mclock_max_sequential_bandwidth_hdd** | **osd_mclock_max_sequential_bandwidth_ssd**, and **osd_op_num_shards_hdd** | **osd_op_num_shards_ssd** parameters.

mClock profiles

An mClock profile is a configuration setting. When applied to a running Red Hat Ceph Storage cluster, it enables the throttling of the IOPS operations belonging to different client classes, such as `background recovery`, `scrub`, `snap trim`, `client op`, and `pg deletion`.

The mClock profile uses the capacity limits and the mClock profile type selected by the user to determine the low-level mClock resource control configuration parameters and applies them transparently. Other Red Hat Ceph Storage configuration parameters are also applied. The low-level mClock resource control parameters are the reservation, limit, and weight that provide control of the resource shares. The mClock profiles allocate these parameters differently for each client type.

mClock profile types

mClock profiles can be classified into *built-in* and *custom* profiles.

If any mClock profile is active, the following Red Hat Ceph Storage configuration sleep options get disabled, which means they are set to `0`:

- `osd_recovery_sleep`
- `osd_recovery_sleep_hdd`
- `osd_recovery_sleep_ssd`
- `osd_recovery_sleep_hybrid`
- `osd_scrub_sleep`
- `osd_delete_sleep`
- `osd_delete_sleep_hdd`
- `osd_delete_sleep_ssd`
- `osd_delete_sleep_hybrid`
- `osd_snap_trim_sleep`
- `osd_snap_trim_sleep_hdd`
- `osd_snap_trim_sleep_ssd`
- `osd_snap_trim_sleep_hybrid`

It is to ensure that mClock scheduler is able to determine when to pick the next operation from its operation queue and transfer it to the operation sequencer. This results in the desired QoS being provided across all its clients.

Custom profile

This profile gives users complete control over all the mClock configuration parameters.

Built-in profiles

When a `built-in` profile is enabled, the mClock scheduler calculates the low-level mClock parameters, that is, reservation, weight, and limit, based on the profile enabled for each client type.

The mClock parameters are calculated based on the maximum Ceph OSD capacity provided beforehand. Therefore, the following mClock configuration options cannot be modified when using any of the built-in profiles:

- `osd_mclock_scheduler_client_res`
- `osd_mclock_scheduler_client_wgt`
- `osd_mclock_scheduler_client_lim`
- `osd_mclock_scheduler_background_recovery_res`

- `osd_mclock_scheduler_background_recovery_wgt`
- `osd_mclock_scheduler_background_recovery_lim`
- `osd_mclock_scheduler_background_best_effort_res`
- `osd_mclock_scheduler_background_best_effort_wgt`
- `osd_mclock_scheduler_background_best_effort_lim`

> **Note:** These defaults cannot be changed using any of the config subsystem commands like `config set`, `config daemon` or `config tell` commands. Although the command(s) report success, the mclock QoS parameters are reverted to their respective built-in profile defaults.

The following recovery and backfill related Ceph options are overridden to mClock defaults:

> **Warning:** Do not change these options as the built-in profiles are optimized based on them. Changing these defaults can result in unexpected performance outcomes.

- `osd_max_backfills`
- `osd_recovery_max_active`
- `osd_recovery_max_active_hdd`
- `osd_recovery_max_active_ssd`

The following options show the mClock defaults which is same as the current defaults to maximize the performance of the foreground client operations:

`osd_max_backfills`

Original default

`1`

mClock default

`1`

`osd_recovery_max_active`

Original default

`0`

mClock default

`0`

`osd_recovery_max_active_hdd`

Original default

`3`

mClock default

`3`

`osd_recovery_max_active_sdd`

Original default

`10`

mClock default

`10`

Note: The mClock defaults can be modified, only if necessary, by enabling `osd_mclock_override_recovery_settings` that is by default set as `false`. See Modifying backfill and recovery options to modify these parameters.

Built-in profiles

Users can choose from the following *built-in* profile types:

- `balanced` (default)
- `high_client_ops`
- `high_recovery_ops`

Note: The values mentioned in the list represent the proportion of the total IOPS capacity of the Ceph OSD allocated for the service type.

- `balanced`:

`balanced`:

The default mClock profile is set to `balanced` because it represents a compromise between prioritizing client IO or recovery IO. It allocates equal reservation or priority to client operations and background recovery operations. Background best-effort operations are given lower reservation and therefore take longer to complete when there are competing operations. This profile meets the normal or steady state requirements of the cluster which is the case when external client performance requirements is not critical and there are other background operations that still need attention within the OSD.

There might be instances that necessitate giving higher priority to either client operations or recovery operations. To meet such requirements you can choose either the `high_client_ops` profile to prioritize client IO or the `high_recovery_ops` profile to prioritize recovery IO. These profiles are discussed further.

Service type: client

Reservation

50%

Limit

MAX

Weight

1

Service type: background recovery

Reservation

50%

Limit

MAX

Weight

1

Service type: background best-effort

Reservation

MIN

Limit

90%

Weight

1

- high_client_ops

high_client_ops:

This profile optimizes client performance over background activities by allocating more reservation and limit to client operations as compared to background operations in the Ceph OSD. This profile, for example, can be enabled to provide the needed performance for I/O intensive applications for a sustained period of time at the cost of slower recoveries. The list shows the resource control parameters set by the profile:

Service type: client

Reservation

60%

Limit

MAX

Weight

2

Service type: background recovery

Reservation

40%

Limit

MAX

Weight

1

Service type: background best-effort

Reservation

MIN

Limit

70%

Weight

1

- high_recovery_ops

high_recovery_ops:

This profile optimizes background recovery performance as compared to external clients and other background operations within the Ceph OSD.

For example, it could be temporarily enabled by an administrator to accelerate background recoveries during non-peak hours. The list shows the resource control parameters set by the profile:

Service type: client

Reservation

30%

Limit

MAX

Weight

1

Service type: background recovery

Reservation

70%

Limit

MAX

Weight

2

Service type: background best-effort

Reservation

MIN

Limit

MAX

Weight

1

Reference

- For more information about mClock configuration options, see the mClock configuration options.

Changing mClock profiles

The default mClock profile is set to `balanced`. The other types of the `built-in` profile are `high_client_ops` and `high_recovery_ops`.

> **Note:** The `custom` profile is not recommended unless you are an advanced user.

Prerequisites

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

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. Set the `osd_mclock_profile` option:

   Syntax

   ```
   ceph config set osd._OSD_ID_ osd_mclock_profile VALUE
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph config set osd.0 osd_mclock_profile high_recovery_ops
   ```

   This example changes the profile to allow faster recoveries on `osd.0`.

Note: For optimal performance the profile must be set on all Ceph OSDs by using the following command:
Syntax

```
ceph config set osd osd_mclock_profile VALUE
```

Switching between `built-in` and `custom` profiles

Understand how to switch between the `built-in` and `custom` profiles.

In some cases, you might want to switch to the `custom` profile if you want complete control over all the mClock configuration options. However, it is recommended not to use the `custom` profile unless you are an advanced user.

Prerequisites

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

Procedure

Switch from *built-in* profile to *custom* profile

1. Log into the `cephadm` shell:

   Example

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

2. Switch to the *custom* profile:

   Syntax

   ```
   ceph config set osd.OSD_ID osd_mclock_profile custom
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph config set osd.0 osd_mclock_profile custom
   ```

   Note: For optimal performance the profile must be set on all Ceph OSDs by using the following command:

   ```
   [ceph: root@host01 /]# ceph config set osd osd_mclock_profile custom
   ```

3. Optional: After switching to the *custom* profile, modify the desired mClock configuration options:

   Syntax

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

   Example

   ```
   [ceph: root@host01 /]# ceph config set osd.0 osd_mclock_scheduler_client_res 0.5
   ```

   This example changes the client reservation IOPS ratio for a specific OSD `osd.0` to 0.5 (50%)

> **Important:** Change the reservations of other services, such as background recovery and background best-effort accordingly to ensure that the sum of the reservations does not exceed the maximum proportion (1.0) of the IOPS capacity of the OSD.

Switch from *custom* profile to *built-in* profile

1. Log into the cephadm shell:

   Example

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

2. Set the desired *built-in* profile:

   Syntax

   ```
   ceph config set osd osd_mclock_profile MCLOCK_PROFILE
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph config set osd osd_mclock_profile high_client_ops
   ```

   This example sets the *built-in* profile to `high_client_ops` on all Ceph OSDs.

3. Determine the existing custom mClock configuration settings in the database:

   Example

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

4. Remove the custom mClock configuration settings determined earlier:

   Syntax

   ```
   ceph config rm osd MCLOCK_CONFIGURATION_OPTION
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph config rm osd osd_mclock_scheduler_client_res
   ```

   This example removes the configuration option `osd_mclock_scheduler_client_res` that was set on all Ceph OSDs.

   After all existing custom mClock configuration settings are removed from the central configuration database, the configuration settings related to `high_client_ops` are applied.

5. Verify the settings on Ceph OSDs:

   Syntax

   ```
   ceph config show osd.OSD_ID
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph config show osd.0
   ```

Reference

- For the list of the mClock configuration options that cannot be modified with _*built-in*_ profiles, see mClock profile types.

Switching temporarily between mClock profiles

Understand how to temporarily switch between mClock profiles.

Warning: This section is for advanced users or for experimental testing. Do not use these commands on a running storage cluster as it could have unexpected outcomes.

Note: The configuration changes on a Ceph OSD using these commands are temporary and are lost when the Ceph OSD is restarted.

Important: The configuration options that are overridden using the commands described in this section cannot be modified further using the `ceph config set osd.OSD_ID` command. The changes do not take effect until a given Ceph OSD is restarted. This is intentional, as per the configuration subsystem design. However, any further modifications can still be made temporarily using these commands.

Prerequisites

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

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. Run the following command to override the mClock settings:

   Syntax

   ```
   ceph tell osd.OSD_ID injectargs '--MCLOCK_CONFIGURATION_OPTION=VALUE'
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph tell osd.0 injectargs '--
   osd_mclock_profile=high_recovery_ops'
   ```

   This example overrides the `osd_mclock_profile` option on `osd.0`.

3. Optional: You can use the alternative to the previous `ceph tell osd.OSD_ID injectargs` command:

   Syntax

   ```
   ceph daemon osd.OSD_ID config set MCLOCK_CONFIGURATION_OPTION VALUE
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph daemon osd.0 config set osd_mclock_profile
   high_recovery_ops
   ```

Note: The individual QoS related configuration options for the `custom` profile can also be modified temporarily using these commands.

Degraded and misplaced object recovery rate with mClock profiles

Degraded object recovery is categorized into the background recovery bucket. Across all mClock profiles, degraded object recovery is given higher priority when compared to misplaced object recovery because degraded objects present a data safety issue not present with objects that are merely misplaced. Backfill or the misplaced object recovery operation is categorized into the background best-effort bucket.

According to the `balanced` and `high_client_ops` mClock profiles, background best-effort client is not constrained by reservation (set to zero) but is limited to use a fraction of the participating OSD's capacity if there are no other competing services.

Therefore, with the `balanced` or `high_client_ops` profile and with other background competing services active, backfilling rates are expected to be slower when compared to the previous *WeightedPriorityQueue* (WPQ) scheduler.

If higher backfill rates are desired, follow the steps mentioned in "Modifying backfills and recovery options" on page 146.

Improving backfilling rates

For faster backfilling rate when using either `balanced` or `high_client_ops` profile, run the following steps:

- Switch to the 'high_recovery_ops' mClock profile for the duration of the backfills. See changing an clock profile to achieve this. Once the backfilling phase is complete, switch the mClock profile to the previously active profile. In case there is no significant improvement in the backfilling rate with the `high_recovery_ops` profile, continue to the next step.
- Switch the mClock profile back to the previously active profile.
- Modify `osd_max_backfills` to a higher value, for example, `3`. See Modifying backfills and recovery options.
- Once the backfilling is complete, *osd_max_backfills* can be reset to the default value of 1 by following the same procedure mentioned in step 3.

> **Important:** Modifying `osd_max_backfills` may result in other operations, for example, client operations may experience higher latency during the backfilling phase. Therefore, users are recommended to increase `osd_max_backfills` in small increments to minimize performance impact to other operations in the cluster.

Modifying `backfills` and `recovery` options

Modify the `backfills` and `recovery` options with the **`ceph config set`** command.

The backfill or recovery options that can be modified are listed in mClock profile types.

> **Warning:** This section is for advanced users or for experimental testing. Do not use these commands on a running storage cluster as it could have unexpected outcomes.
> Modify the values only for experimental testing, or if the cluster is unable to handle the values or it shows poor performance with the default settings.

> **Important:** The modification of the mClock default backfill or recovery options is restricted by the `osd_mclock_override_recovery_settings` option, which is set to `false` by default.
> If you attempt to modify any default backfill or recovery options without setting `osd_mclock_override_recovery_settings` to `true`, it resets the options back to the mClock defaults along with a warning message logged in the cluster log.

Prerequisites

- A running Red Hat Ceph Storage cluster.

– Root-level access to the Ceph Monitor host.

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. Set the `osd_mclock_override_recovery_settings` configuration option to `true` on all Ceph OSDs:

   Example

   ```
   [ceph: root@host01 /]# ceph config set osd osd_mclock_override_recovery_settings true
   ```

3. Set the desired `backfills` or `recovery` option:

   Syntax

   ```
   ceph config set osd OPTION VALUE
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph config set osd osd_max_backfills 5
   ```

4. Wait a few seconds and verify the configuration for the specific OSD:

   Syntax

   ```
   ceph config show osd.OSD_ID | grep OPTION
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph config show osd.0 | grep osd_max_backfills
   ```

5. Reset the `osd_mclock_override_recovery_settings` configuration option to `false` on all OSDs:

   Example

   ```
   [ceph: root@host01 /]# ceph config set osd osd_mclock_override_recovery_settings false
   ```

Ceph OSD capacity determination

The Ceph OSD capacity in terms of total IOPS is determined automatically during the Ceph OSD initialization. This is achieved by running the Ceph OSD bench tool and overriding the default value of `osd_mclock_max_capacity_iops_[hdd, ssd]` option depending on the device type. No other action or input is expected from the user to set the Ceph OSD capacity.

Mitigation of unrealistic Ceph OSD capacity from the automated procedure

In certain conditions, the Ceph OSD bench tool might show unrealistic or inflated results depending on the drive configuration and other environment related conditions.

To mitigate the performance impact due to this unrealistic capacity, a couple of threshold configuration options depending on the OSD device type are defined and used:

– `osd_mclock_iops_capacity_threshold_hdd` = 500

– `osd_mclock_iops_capacity_threshold_ssd` = 80000

You can verify these parameters by running the following commands:

```
[ceph: root@host01 /]# ceph config show osd.0 osd_mclock_iops_capacity_threshold_hdd
500.000000
[ceph: root@host01 /]# ceph config show osd.0 osd_mclock_iops_capacity_threshold_ssd
80000.000000
```

Note: If you want to manually benchmark OSD(s) or manually tune the BlueStore throttle parameters, see “Manually benchmarking OSDs” on page 149.

You can verify the capacity of an OSD after the cluster is up by running the following command:

Syntax

```
ceph config show osd.N osd_mclock_max_capacity_iops_[hdd, ssd]
```

Example

```
[ceph: root@host01 /]# ceph config show osd.0 osd_mclock_max_capacity_iops_ssd
```

In the example, you can view the maximum capacity for `osd.0` on a Red Hat Ceph Storage node whose underlying device is an SSD.

The following automated step is performed:

Fallback to using default OSD capacity

If the Ceph OSD bench tool reports a measurement that exceeds the threshold values, the fallback mechanism reverts to the default value of `osd_mclock_max_capacity_iops_hdd` or `osd_mclock_max_capacity_iops_ssd`. The threshold configuration options can be reconfigured based on the type of drive used.

A cluster warning is logged in case the measurement exceeds the threshold:

Example

```
3403 Sep 11 11:52:50 dell-r640-039.dsal.lab.eng.rdu2.redhat.com ceph-osd[70342]:
log_channel(cluster) log [WRN] : OSD bench result of 49691.213005 IOPS exceeded the
threshold limit of 500.000000 IOPS for osd.27. IOPS capacity is unchanged at 315.000000
IOPS. The recommendation is to establish the osd's IOPS capacity using other benchmark
tools (e.g. Fio) and then override osd_mclock_max_capacity_iops_[hdd|ssd].
```

Important: If the default capacity does not accurately represent the Ceph OSD capacity, it is highly recommended to run a custom benchmark using the preferred tool, for example Fio, on the drive and then override the `osd_mclock_max_capacity_iops_[hdd, ssd]` option as described in Specifying maximum OSD capacity

Reference

- To manually benchmark Ceph OSDs or manually tune the BlueStore throttle parameters, see Manually benchmarking OSDs.
- For more information about the `osd_mclock_max_capacity_iops_[hdd, ssd]` and `osd_mclock_iops_capacity_threshold_[hdd, ssd]` options, see the mClock configuration options.

Verifying the capacity of an OSD

You can verify the capacity of a Ceph OSD after setting up the storage cluster.

Prerequisites

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

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. Verify the capacity of a Ceph OSD:

   Syntax

   ```
   ceph config show osd.OSD_ID osd_mclock_max_capacity_iops_[hdd, ssd]
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph config show osd.0 osd_mclock_max_capacity_iops_ssd
   21500.000000
   ```

Manually benchmarking OSDs

To manually benchmark a Ceph OSD, any existing benchmarking tool, for example Fio, can be used. Regardless of the tool or command used, steps documented here remain the same.

Important: The number of shards and BlueStore throttle parameters have an impact on the mClock operation queues. Therefore, it is critical to set these values carefully in order to maximize the impact of the mclock scheduler. For more information on these values, see factors that impact mClock operation queues.

Note: The steps in this section are only necessary if you want to override the Ceph OSD capacity determined automatically during the OSD initialization.
If you have already determined the benchmark data and wish to manually override the maximum OSD capacity for a Ceph OSD, see Specifying maximum OSD capacity.

Prerequisites

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

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. Benchmark a Ceph OSD:

   Syntax

   ```
   ceph tell osd.OSD_ID bench [TOTAL_BYTES] [BYTES_PER_WRITE] [OBJ_SIZE] [NUM_OBJS]
   ```

   where:

   - *TOTAL_BYTES*: Total number of bytes to write.
   - *BYTES_PER_WRITE*: Block size per write.
   - *OBJ_SIZE*: Bytes per object.
   - *NUM_OBJS*: Number of objects to write.

   Example

```
[ceph: root@host01 /]# ceph tell osd.0 bench 12288000 4096 4194304 100
{
    "bytes_written": 12288000,
    "blocksize": 4096,
    "elapsed_sec": 1.3718913019999999,
    "bytes_per_sec": 8956977.8466311768,
    "iops": 2186.7621695876896
}
```

Determining BlueStore throttle values

Determining BlueStore throttle values is optional. It is used to determine the correct BlueStore throttle values. The steps use the default shards.

Important: Before running the test, clear the caches to get an accurate measurement. Clear the OSD caches between each benchmark run using the following command:
Syntax

```
ceph tell osd.OSD_ID cache drop
```

Example

```
[ceph: root@host01 /]# ceph tell osd.0 cache drop
```

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the Ceph Monitor node hosting the OSDs that you wish to benchmark.

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. Run a simple 4KiB random write workload on an OSD:

   Syntax

   ```
   ceph tell osd.OSD_ID bench 12288000 4096 4194304 100
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph tell osd.0 bench 12288000 4096 4194304 100
   {
       "bytes_written": 12288000,
       "blocksize": 4096,
       "elapsed_sec": 1.3718913019999999,
       "bytes_per_sec": 8956977.8466311768,
       "iops": 2186.7621695876896 1
   }
   ```

   a. The overall throughput obtained from the output of the `osd bench` command. This value is the baseline throughput, when the default BlueStore throttle options are in effect.

3. Note the overall throughput, that is IOPS, obtained from the output of the previous command.

4. If the intent is to determine the BlueStore throttle values for your environment, set `bluestore_throttle_bytes` and `bluestore_throttle_deferred_bytes` options to 32 KiB, that is, 32768 Bytes:

Syntax

```
ceph config set osd.OSD_ID bluestore_throttle_bytes 32768
ceph config set osd.OSD_ID bluestore_throttle_deferred_bytes 32768
```

Example

```
[ceph: root@host01 /]# ceph config set osd.0 bluestore_throttle_bytes 32768
[ceph: root@host01 /]# ceph config set osd.0 bluestore_throttle_deferred_bytes 32768
```

Otherwise, you can skip to the next section, specifying maximum OSD capacity.

5. Run the 4KiB random write test as before using an OSD bench command:

   Example

   ```
   [ceph: root@host01 /]# ceph tell osd.0 bench 12288000 4096 4194304 100
   ```

6. Notice the overall throughput from the output and compare the value against the baseline throughput recorded earlier.
7. If the throughput does not match with the baseline, increase the BlueStore throttle options by multiplying by 2.
8. Repeat the steps by running the 4KiB random write test, comparing the value against the baseline throughput, and increasing the BlueStore throttle options by multiplying by 2, until the obtained throughput is very close to the baseline value.

> **Note:** For example, during benchmarking on a machine with NVMe SSDs, a value of 256 KiB for both BlueStore throttle and deferred bytes was determined to maximize the impact of mClock. For HDDs, the corresponding value was 40 MiB, where the overall throughput was roughly equal to the baseline throughput.
> In general for HDDs, the BlueStore throttle values are expected to be higher when compared to SSDs.

Specifying maximum OSD capacity

You can override the maximum Ceph OSD capacity automatically set during OSD initialization.

These steps are optional. Perform the following steps if the default capacity does not accurately represent the Ceph OSD capacity.

> **Note:** Ensure that you determine the benchmark data first, as described in manually benchmarking OSDs.

Prerequisites

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

Procedure

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. Set `osd_mclock_max_capacity_iops_[hdd, ssd]` option for an OSD:

   Syntax

```
ceph config set osd.OSD_ID osd_mclock_max_capacity_iops_[hdd,ssd] VALUE
```

Example

```
[ceph: root@host01 /]# ceph config set osd.0 osd_mclock_max_capacity_iops_hdd 350
```

This example sets the maximum capacity for `osd.0`, where an underlying device type is HDD, to 350 IOPS.

mClock configuration options

Learn the different mClock configuration options.

Following are the list of mClock configuration options:

`osd_mclock_profile`

Description

It sets the type of mClock profile to provide the quality of service (QoS) based on operations that belong to different classes, such as background recovery, `backfill`, `pg scrub`, `snap trim`, `client op`, and `pg deletion`.

On enabling a `built-in` profile, the lower-level mClock resource control parameters, that is reservation, weight, and limit, and some Ceph configuration parameters are set transparently. This does not apply for the `custom` profile.

Type

String

Default

`balanced`

Valid choices

`balanced`, `high_recovery_ops`, `high_client_ops`, `custom`

`osd_mclock_max_capacity_iops_hdd`

Description

It sets a maximum random write IOPS capacity, at 4 KiB block size, to consider per OSD for rotational media. Contributes in QoS calculations when enabling a dmclock profile. It is only considered for `osd_op_queue = mclock_scheduler`

Type

Float

Default

`315.0`

`osd_mclock_max_capacity_iops_ssd`

Description

It sets a maximum random write IOPS capacity, at 4 KiB block size, to consider per OSD for solid-state media.

Type

Float

Default

`21500.0`

`osd_mclock_cost_per_byte_usec_ssd`

Description

Indicates cost per byte in microseconds to consider per OSD for SDDs. Contributes in QoS calculations when enabling a dmclock profile. It is only considered for `osd_op_queue = mclock_scheduler`

Type

Float

Default

`0.011`

`osd_mclock_max_sequential_bandwidth_hdd`

Description

Indicates the maximum sequential bandwidth in bytes to consider for an OSD whose underlying device type is rotational media. This option is considered by the mclock scheduler to derive the cost factor to be used in QoS calculations. Only considered for `osd_op_queue = mclock_scheduler`

Type

Size

Default

`150_M`

`osd_mclock_max_sequential_bandwidth_ssd`

Description

Indicates the maximum sequential bandwidth in bytes to consider for an OSD whose underlying device type is solid-state media. This is considered by the mclock scheduler to derive the cost factor to be used in QoS calculations. Only considered for `osd_op_queue = mclock_scheduler`

Type

Size

Default

`1200_M`

`osd_mclock_force_run_benchmark_on_init`

Description

This force-runs the OSD benchmark on OSD initialization or boot-up.

Type

Boolean

Default

False

See also

`osd_mclock_max_capacity_iops_hdd`, `osd_mclock_max_capacity_iops_ssd`

`osd_mclock_skip_benchmark`

Description

Setting this option skips the OSD benchmark on OSD initialization or boot-up.

Type

Boolean

Default

False

See also

`osd_mclock_max_capacity_iops_hdd`, `osd_mclock_max_capacity_iops_ssd`

osd_mclock_override_recovery_settings

Description

Setting this option enables the override of the recovery or backfill limits for the mClock scheduler as defined by the `osd_recovery_max_active_hdd`, `osd_recovery_max_active_ssd`, and `osd_max_backfills` options.

Type

Boolean

Default

False

See also

`osd_recovery_max_active_hdd`, `osd_recovery_max_active_ssd`, `osd_max_backfills`

osd_mclock_iops_capacity_threshold_hdd

Description

It indicates the threshold IOPS capacity, at 4 KiB block size, beyond which to ignore the Ceph OSD bench results for an OSD for HDDs.

Type

Float

Default

`500.0`

osd_mclock_iops_capacity_threshold_ssd

Description

It indicates the threshold IOPS capacity, at 4 KiB block size, beyond which to ignore the Ceph OSD bench results for an OSD for SSDs.

Type

Float

Default

`80000.0`

osd_mclock_scheduler_client_res

Description

It is the default I/O proportion that is reserved for each client. The default value of `0` specifies the lowest possible reservation. Any value greater than 0 and up to 1.0 specifies the minimum IO proportion to reserve for each client in terms of a fraction of the OSD's maximum IOPS capacity.

Type

Float

Default

`0`

min

0

max

1.0

osd_mclock_scheduler_client_wgt

Description

It is the default I/O share for each client over reservation.

Type

Unsigned integer

Default

1

osd_mclock_scheduler_client_lim

Description

It is the default I/O limit for each client over reservation. The default value of 0 specifies no limit enforcement, which means each client can use the maximum possible IOPS capacity of the OSD. Any value greater than 0 and up to 1.0 specifies the upper IO limit over reservation that each client receives in terms of a fraction of the OSD's maximum IOPS capacity.

Type

Float

Default

0

min

0

max

1.0

osd_mclock_scheduler_background_recovery_res

Description

It is the default I/O proportion reserved for background recovery. The default value of 0 specifies the lowest possible reservation. Any value greater than 0 and up to 1.0 specifies the minimum IO proportion to reserve for background recovery operations which is in terms of a fraction of the OSD's maximum IOPS capacity.

Type

Float

Default

0

min

0

max

1.0

osd_mclock_scheduler_background_recovery_wgt

Description

It indicates the I/O share for each background recovery over reservation.

Type

Unsigned integer

Default

1

osd_mclock_scheduler_background_recovery_lim

Description

It indicates the I/O limit for background recovery over reservation. The default value of 0 specifies no limit enforcement, which means background recovery operation can use the maximum possible IOPS capacity of the OSD. Any value greater than 0 and up to 1.0 specifies the upper IO limit over reservation that background recovery operation receives in terms of a fraction of the OSD's maximum IOPS capacity.

Type

Float

Default

`0`

min

0

max

1.0

`osd_mclock_scheduler_background_best_effort_res`

Description

It indicates the default I/O proportion that is reserved for background `best_effort`. The default value of 0 specifies the lowest possible reservation. Any value greater than 0 and up to 1.0 specifies the minimum IO proportion to reserve for background best_effort operations in terms of a fraction of the OSD's maximum IOPS capacity.

Type

Float

Default

`0`

min

0

max

1.0

`osd_mclock_scheduler_background_best_effort_wgt`

Description

It indicates the I/O share for each background `best_effort` over reservation.

Type

Unsigned integer

Default

`1`

`osd_mclock_scheduler_background_best_effort_lim`

Description

It indicates the I/O limit for background `best_effort` over reservation. The default value of 0 specifies no limit enforcement, which means background best_effort operation can use the maximum IOPS capacity of the OSD. Any value greater than 0 and up to 1.0 specifies the upper IO limit over reservation that background best_effort operation receives in terms of a fraction of the OSD's maximum IOPS capacity.

Type

float

Default

`0`

min

0

max

1.0

Reference

- For more details about this `osd_op_queue` option, see Object Storage Daemon (OSD) configuration options .

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, contact Red Hat Support and share the details of your configuration to help Red Hat improve the auto-tuning capability. Red Hat looks forward to your feedback and appreciates your recommendations.

BlueStore features

Learn about the main BlueStore features.

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 under 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). For more information, see “BlueStore devices” on page 158.

Efficient block device usage

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

BlueStore devices

BlueStore manages either one, two, or three storage devices in the backend. The backend devices include primary, write-ahead-log (WAL), and database (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 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 device 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 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 provides while also allowing for storing additional metadata.

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. Afterward, a write acknowledgment 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:** Before enabling the `bluestore_default_buffered_write` option, contact Red Hat Support.

Sizing considerations

When mixing traditional and solid state drives using BlueStore OSDs, it is important to size the RocksDB logical volume (`block.db`) appropriately.

The RocksDB logical volume be no less than 4% of the block size with object, file and mixed workloads. Red Hat Ceph Storage 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.

Tuning BlueStore

Use this procedure for freshly deployed OSDs.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- Ceph Monitors and Managers are deployed in the cluster.
- Servers that can be freshly provisioned as OSD nodes.
- The admin keyring.
- Check the value used to deploy the OSD by running the following command:

  ```
  ceph osd metadata OSD_ID | grep bluestore_min_alloc_size
  ```

  For example,

  ```
  [ceph: root@host01 /]# ceph osd metadata 11 | grep bluestore_min_alloc_size
  "bluestore_min_alloc_size": "4096",
  ```

In BlueStore, the raw underlying storage is allocated in chunks of **`bluestore_min_alloc_size`**. By default, **`bluestore_min_alloc_size`** is `4096`, equivalent to 4 KiB for HDDs and SSDs. Starting with Red Hat Ceph Storage 9, erasure-coded pools no longer add zero padding to unwritten areas in chunks. To enable erasure coding optimization, also known as FastEC, see "Enabling coding optimizations" on page 0.

> **Important:** Changing the value of **`bluestore_min_alloc_size`** is not recommended. For any assistance, contact Red Hat Support.

> **Note:**
>
> - You can change specifically the SSD and HDD options with the **`bluestore_min_alloc_size`** parameter within the **`config set`** command. For more information, contact Red Hat Support.
> - For already deployed OSDs, you cannot modify the **`bluestore_min_alloc_size`** parameter. To modify, remove and then redeploy each affected OSD.

For OSD removal and additional information, see Managing OSDs.

1. From the `cephadm` shell, change the **`bluestore_min_alloc_size`** parameter value.
   Set the allocation size, in one of the following ways.

   > **Note:** The selected **`bluestore_min_alloc_size`** values should be power of 2 aligned.

   - Set the allocation size per OSD.

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

In the following example, **bluestore_min_alloc_size** is set to 8192 bytes, which is equivalent to 8 KiB.

```
[ceph: root@host01 /]# ceph config set osd.4 bluestore_min_alloc_size 8192
```

- Set the allocation size per CRUSH device class.

```
ceph config set class:CLASS bluestore_min_alloc_size VALUE
```

Replace *CLASS* with the CRUSH device class, usuallyhdd or ssd.
In the following example, the CRUSH device class is hdd and the **bluestore_min_alloc_size** is set to 8192 bytes, which is equivalent to 8 KiB.

```
[ceph: root@host01 /]# ceph config set class:hdd bluestore_min_alloc_size
8192
```

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

For example,

```
[ceph: root@host01 /]# ceph config set osd.4 bluestore_min_alloc_size 8192
```

2. Redeploy the affected OSDs.
For more information, see Managing OSDs.

AFTER COMPLETING THE TASK

Verify the BlueStore tuning settings, by using the **ceph osd metadata** command.

For example,

```
[ceph: root@host01 /]# ceph osd metadata OSD.1 | grep bluestore_min_alloc_size
"bluestore_min_alloc_size": "8192",
```

RocksDB cache behavior and metrics

Understand how RocksDB caching works in BlueStore OSDs, how to interpret cache performance counters, and how to identify and tune cache inefficiencies.

RocksDB is used by Ceph components, such as monitors and object storage daemons (OSDs). This topic describes RocksDB caching behavior for BlueStore OSDs.

RocksDB uses a block cache to store portions of SST files in memory. In BlueStore, Red Hat Ceph Storage uses an integrated cache implementation that combines the following elements:

- RocksDB block cache
- BlueStore metadata cache
- BlueStore data cache

These caches share available memory and dynamically compete for resources. This design allows memory to be allocated where it is most beneficial for performance.

Cache sharding

To improve concurrency and reduce lock contention, the RocksDB cache is divided into multiple shards. Each cached object is assigned to a shard based on a hash of its key, enabling concurrent access across threads.

The number of shards is controlled by the **rocksdb_cache_shard_bits** configuration option. Set **rocksdb_cache_shard_bits** to select how many of bits from the key hash used to determine the cache shard. The shards input is an unsigned integer and the default is 4.

For example, a value of 4 results in 16 cache shards.

RocksDB cache performance counters

BlueStore exposes RocksDB cache activity through performance counters.

By default, two cache instances are created:

- `rocksdb-cache-0`: metadata cache for onode structures
- `rocksdb-cache-default`: cache for all other data

These counters can be used to evaluate cache efficiency and effectiveness.

Use the following command to view performance counters:

```
ceph tell osd.ID perf dump rocksdb-cache-0
```

The following is an example output:

```
"rocksdb-cache-0": {
  "capacity": 134217728,
  "usage": 134182832,
  "pinned": 0,
  "elems": 24502,
  "inserts": 25806978,
  "lookups": 150436987,
  "hits": 124629911,
  "misses": 25807076
  "read_bytes": 35265735
}
```

The following metrics are provided:

`capacity`
: Total cache size

`usage`
: Currently used memory

`pinned`
: Number of temporarily pinned entries

`elems`
: Number of cached elements

`inserts`
: Number of items added to the cache

`lookups`
: Number of cache lookup requests

`hits`
: Number of successful lookups

`misses`
: Number of failed lookups

`read_bytes`
: Size of read operations being used, in bytes.

Viewing detailed cache statistics

Performance counters provide aggregated values. For deeper analysis, you can inspect cache statistics at the shard level.

Use per-shard statistics to help you identify imbalance or inefficient cache usage.

- Use the following command to display shard statistics:

  ```
  ceph tell osd.ID rocksdb show cache 0
  ```

- Use the following command to reset cache counters:

  ```
  ceph tell osd.ID rocksdb reset cache 0
  ```

Identifying cache inefficiencies

Imbalanced shard utilization can negatively affect performance.

Indicators of inefficiency include:

- Very low element count in a shard compared to others.
- High number of cache misses relative to hits.
- Rapid increase in `bluefs.read_bytes`.

> **Note:** To monitor `read_bytes` counters, use the `perf dump` commands. For example,
>
> ```
> ceph tell osd.0 perf dump bluefs | jq -r '.bluefs.read_bytes'
> ```

For example,

```
# ceph tell osd.ID rocksdb show cache 0
shard  capacity     usage   pinned   elems  inserts  lookups     hits   misses
  ...
    2  13631488  11060608        0    2232   135076   908468   773313   135155
    3  13631488   8166896        0       1   134006   427070   293147   133923
    4  13631488  11117984        0    2297   133367   700242   567318   132924
  ...
```

This pattern can indicate frequent eviction or ineffective cache usage within a shard.

Cache behavior during compaction

Large OSD deployments can store millions of objects, which can result in large RocksDB index blocks.

During compaction:

- Multiple index blocks can be required at the same time.
- Index blocks can be large compared to shard size.
- If multiple large blocks are assigned to the same shard, repeated eviction can occur.

This behavior can reduce cache efficiency and negatively impact performance.

Tuning cache shard configuration

The default configuration of 16 shards (`rocksdb_cache_shard_bits=4`) is suitable for most environments.

In some cases, reducing the number of shards can improve cache behavior by increasing shard size. When larger size shards are required, decrease the value of **`rocksdb_cache_shard_bits`**.

> **Important:** Reducing the number of shards can increase lock contention and might slightly affect baseline performance. Evaluate performance in a test environment before applying changes in production.

For more information about restructuring the RocksDB database to improve performance, see “Resharding the RocksDB database” on page 162.

Resharding the RocksDB database

Use the BlueStore administration tool to reshard the RocksDB database. Resharding splits the BlueStore block.db into several column families without redeploying the OSDs, which can be beneficial for write performance.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.

- The object store configured as BlueStore.
- OSD nodes deployed on the hosts.
- Root level access to the all the hosts.
- The `ceph-common` and `cephadm` packages installed on all the hosts.

Reshard the RocksDB database to split the BlueStore block.db into several columns. 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.

Reshard the OSD in one of the following ways:

- "Use the rocksdb-resharding.yml playbook" on page 163.
- "Manually resharding the OSDs" on page 165.

For more information, see Initial installation.

Use the `rocksdb-resharding.yml` playbook

Reshard the database by using the `rocksdb-resharding.yml` playbook.

1. As a root user, on the administration node, navigate to the `cephadm` folder where the playbook is installed.

   ```
   cd /usr/share/cephadm-ansible
   ```

2. Run the playbook.

   ```
   ansible-playbook -i hosts rocksdb-resharding.yml -e osd_id=OSD_ID -e
   admin_node=HOST_NAME
   ```

   For example,

```
[root@host01 ~]# ansible-playbook -i hosts rocksdb-resharding.yml -e osd_id=7 -e
admin_node=host03

.............
TASK [stop the osd]
*********************************************************************************
*********************************************************************************
******************
Wednesday 29 November 2023  11:25:18 +0000 (0:00:00.037)       0:00:03.864 ****
changed: [localhost -> host03]
TASK [set_fact ceph_cmd]
*********************************************************************************
*********************************************************************************
**************
Wednesday 29 November 2023  11:25:32 +0000 (0:00:14.128)       0:00:17.992 ****
ok: [localhost -> host03]

TASK [check fs consistency with fsck before resharding]
*********************************************************************************
*************************************************************************
Wednesday 29 November 2023  11:25:32 +0000 (0:00:00.041)       0:00:18.034 ****
ok: [localhost -> host03]

TASK [show current sharding]
*********************************************************************************
*********************************************************************************
**********
Wednesday 29 November 2023  11:25:43 +0000 (0:00:11.053)       0:00:29.088 ****
ok: [localhost -> host03]

TASK [reshard]
*********************************************************************************
*********************************************************************************
************************
Wednesday 29 November 2023  11:25:45 +0000 (0:00:01.446)       0:00:30.534 ****
ok: [localhost -> host03]

TASK [check fs consistency with fsck after resharding]
*********************************************************************************
*************************************************************************
Wednesday 29 November 2023  11:25:46 +0000 (0:00:01.479)       0:00:32.014 ****
ok: [localhost -> host03]

TASK [restart the osd]
*********************************************************************************
*********************************************************************************
****************
Wednesday 29 November 2023  11:25:57 +0000 (0:00:10.699)       0:00:42.714 ****
changed: [localhost -> host03]
```

AFTER COMPLETING THE TASK

Verify that the sharding is complete.

1. Stop the OSD that is resharded.

```
[ceph: root@host01 /]# ceph orch daemon stop osd.7
```

2. Enter the OSD container.

```
[root@host03 ~]# cephadm shell --name osd.7
```

3. Check for resharding.

```
[ceph: root@host03 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-9/ show-
sharding
    m(3) p(3,0-12) O(3,0-13) L P
```

4. Start the OSD.

```
[ceph: root@host01 /]# ceph orch daemon start osd.7
```

Manually resharding the OSDs

Manually reshard the OSDs with the `cephadm` shell.

1. Log into the `cephadm` shell.
   For example,

   ```
   [root@host01 ~]# cephadm shell
   [ceph: root@host01 /]#
   ```

2. Fetch the *OSD_ID* and the host details from the administration node.

   ```
   ceph orch ps
   ```

3. Log into the respective host as a root user and stop the OSD.

   ```
   cephadm unit --name OSD_ID stop
   ```

   For example,

   ```
   [root@host02 ~]# cephadm unit --name osd.0 stop
   ```

4. Enter into the stopped OSD daemon container.

   ```
   cephadm shell --name OSD_ID
   ```

   For example,

   ```
   root@host02 ~]# cephadm shell --name osd.0
   ```

5. Log into the `cephadm shell` and check the file system consistency.

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

   For example,

   ```
   [ceph: root@host02 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-0/ fsck

   fsck success
   ```

6. Check the sharding status of the node.

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

   For example,

   ```
   [ceph: root@host02 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-9/ show-
   sharding

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

7. Reshard by running the **`ceph-bluestore-tool`** command.
   Use the command, along with the following given parameters.

   ```
   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)=block_cache={type=binned_lru} L P" reshard
   ```

   For example,

   ```
   [ceph: root@host02 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-9/ --
   sharding="m(3) p(3,0-12) O(3,0-13)=block_cache={type=binned_lru} L P" reshard

   reshard success
   ```

8. Use the **`show-sharding`** command to check the sharding status of the OSD node.

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

For example,

```
[ceph: root@host02 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-9/ show-sharding

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

9. Exit from the `cephadm` shell.
   For example,

```
[ceph: root@host02 /]# exit
[root@host02 ~]#
```

10. Log into the respective host as a root user and start the OSD.

```
cephadm unit --name OSD_ID start
```

For example,

```
[root@host02 ~]# cephadm unit --name osd.0 start
```

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.

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.

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: For help resolving severe fragmentation, contact Red Hat Support.

Checking for fragmentation

Check the fragmentation level of BlueStore OSDs while the BlueStore OSD process is either online or offline.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

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

Getting an online BlueStore fragmentation score

Inspect a running BlueStore OSD process by getting either a simple or more detailed report.

- Get a simple report.

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

For example,

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

- Get a detailed report.

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

For example,

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

Getting an offline BlueStore fragmentation score

1. Reshard to check the offline BlueStore OSD.
   For more information about resharding, see “Resharding the RocksDB database” on page 162.

   ```
   [root@host01 ~]# cephadm shell --name osd.ID
   ```

   For example,

   ```
   [root@host01 ~]# cephadm shell --name osd.2
   Inferring fsid 110bad0a-bc57-11ee-8138-fa163eb9ffc2
   Inferring config /var/lib/ceph/110bad0a-bc57-11ee-8138-fa163eb9ffc2/osd.2/config
   Using ceph image with id `17334f841482` and tag `ceph-9-rhel-9-containers-
   candidate-59483-20240301201929` created on 2026-03-01 20:22:41 +0000 UTC
   registry-proxy.engineering.redhat.com/rh-osbs/
   rhceph@sha256:09fc3e5baf198614d70669a106eb87dbebee16d4e91484375778d4adbccadacd
   ```

2. Inspect a non-running BlueStore OSD process.
   Inspect a running BlueStore OSD process by getting either a simple or more detailed report.
   - Get a simple report.

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

     For example,

     ```
     [root@host01 /]# ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-123 --
     allocator block free-score
     ```

   - Get a detailed report.

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

     For example,

```
[root@host01 /]# 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"
        },
```

Ceph BlueStore BlueFS

BlueStore block database stores metadata as key-value pairs in a RocksDB database. The block database resides on a small BlueFS partition on the storage device. BlueFS is a minimal file system that is designed to hold the RocksDB files.

BlueFS files

There are three types of files that RocksDB produces.

- Control files, for example `CURRENT`, `IDENTITY`, and `MANIFEST-00011`.
- Database (DB) table files, for example `004112.sst`.
- Write ahead logs (WAL), for example `00038.log`.

There is also an internal, hidden file that serves as BlueFS replay log, `ino 1`, that works as directory structure, file mapping, and operations log.

Fallback hierarchy

With BlueFS it is possible to put any file on any device. Parts of file can even reside on different devices, that is WAL, DB, and SLOW. There is an order to where BlueFS puts files. File is put to secondary storage only when primary storage is exhausted, and tertiary only when secondary is exhausted.

The order for the specific files is as follows, for each device type.

Write ahead logs

WAL, DB, SLOW

Replay log `ino 1`

DB, SLOW

Control and DB files

DB, SLOW
Control and DB file order when running out of space: SLOW

> **Important:** There is an exception to control and DB file order. When RocksDB detects that you are running out of space on DB file, it directly notifies you to put file to SLOW device.

Viewing the bluefs_buffered_io setting

As a storage administrator, you can view the current setting for the **bluefs_buffered_io** parameter.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Root-level access to the Ceph Monitor node.
- Log into the cephadm shell, using the **cephadm shell** command.

The option **bluefs_buffered_io** is set to True by default for Red Hat Ceph Storage. This option enable BlueFS to perform buffered reads in some cases, and enables the kernel page cache to act as a secondary cache for reads like RocksDB block reads.

> **Important:** Changing the value of **bluefs_buffered_io** is not recommended. Before changing the **bluefs_buffered_io** parameter, contact your Red Hat Support account team.

View the current value of the **bluefs_buffered_io** parameter, using one of the following procedures.

- View the value stored in the configuration database.

```
ceph config get osd bluefs_buffered_io
```

For example,

```
[ceph: root@host01 /]# ceph config get osd bluefs_buffered_io
```

- View the value stored in the configuration database for a specific OSD.

```
ceph config get OSD_ID bluefs_buffered_io
```

For example,

```
[ceph: root@host01 /]# ceph config get osd.2 bluefs_buffered_io
```

- View the running value for an OSD where the running value is different from the value stored in the configuration database.

```
ceph config show OSD_ID bluefs_buffered_io
```

For example,

```
[ceph: root@host01 /]# ceph config show osd.3 bluefs_buffered_io
```

Viewing Ceph BlueFS statistics for Ceph OSDs

View the BluesFS related information about collocated and non-collocated Ceph OSDs with the **bluefs stats** command.

About this task

For more information about BlueStore devices, see “BlueStore devices” on page 158.

Before you begin

- A running Red Hat Ceph Storage cluster.
- Root-level access to the OSD node.
- The object store configured as BlueStore.
- Log into the cephadm shell, using the **cephadm shell** command.

Procedure

View the BlueStore OSD statistics.

```
ceph daemon osd.OSD_ID bluefs stats
```

- Example for collocated OSDs

```
[ceph: root@host01 /]# ceph daemon osd.1 bluefs stats

1 : device size 0x3bfc00000 : using 0x1a428000(420 MiB)
wal_total:0, db_total:15296836403, slow_total:0
```

- Example for non-collocated OSDs

```
[ceph: root@host01 /]# ceph daemon osd.1 bluefs stats

0 :
1 : device size 0x1dfbfe000 : using 0x1100000(17 MiB)
2 : device size 0x27fc00000 : using 0x248000(2.3 MiB)
RocksDBBlueFSVolumeSelector: wal_total:0, db_total:7646425907, slow_total:10196562739,
db_avail:935539507
Usage matrix:
DEV/LEV     WAL         DB          SLOW        *           *           REAL
 FILES
LOG         0 B         4 MiB       0 B         0 B         0 B         756 KiB     1
WAL         0 B         4 MiB       0 B         0 B         0 B         3.3 MiB     1
DB          0 B         9 MiB       0 B         0 B         0 B         76 KiB      10
SLOW        0 B         0 B         0 B         0 B         0 B         0 B         0
TOTALS      0 B         17 MiB      0 B         0 B         0 B         0 B         12
MAXIMUMS:
LOG         0 B         4 MiB       0 B         0 B         0 B         756 KiB
WAL         0 B         4 MiB       0 B         0 B         0 B         3.3 MiB
DB          0 B         11 MiB      0 B         0 B         0 B         112 KiB
SLOW        0 B         0 B         0 B         0 B         0 B         0 B
TOTALS      0 B         17 MiB      0 B         0 B         0 B         0 B
```

In this example,

0

Refers to the dedicated WAL device, which is `block.wal`.

1

Refers to the dedicated DB device, which is `block.db`.

2

Refers to the main block device, which is `block` or `slow`.

device size

Represents an actual size of the device.

using

Represents the total usage. It is not restricted to BlueFS.

> **Note:** DB and WAL devices are used only by BlueFS. For a main device, usage from stored BlueStore data is also included. In this example, 2.3 MiB is the data from BlueStore.

wal_total, db_total, and slow_total

Values that reiterate the device values previously stated.

db_avail

Represents how many bytes can be taken from the SLOW device, if necessary.
Usage matrix:

Rows WAL, DB, and SLOW

Describes where the specific file was intended to be put.

Row LOG

Describes the BlueFS replay log `ino 1`.

Columns WAL, DB, and SLOW

Describe where data is actually put. The values are in allocation units. WAL and DB have bigger allocation units for performance reasons.

Columns *

Relate to virtual devices `new-db` and `new-wal` that are used for `ceph-bluestore-tool`. It should always show `0 B`.

Column REAL

Shows actual usage in bytes.

Column FILES

Shows count of files.

MAXIMUMS

This table captures the maximum value of each entry from the usage matrix.

Using the `ceph-bluestore-tool`

`ceph-bluestore-tool` is a utility to perform low-level administrative operations on a BlueStore instance.

The following commands are available with the `ceph-bluestore-tool`.

```
ceph-bluestore-tool COMMAND [ --dev DEVICE … ] [ -i OSD_ID ] [ --path OSD_PATH ] [ --out-
dir DIR ] [ --log-file | -l filename ] [ --deep ]

ceph-bluestore-tool fsck|repair --path OSD_PATH [ --deep ]

ceph-bluestore-tool qfsck --path OSD_PATH

ceph-bluestore-tool allocmap --path OSD_PATH

ceph-bluestore-tool restore_cfb --path OSD_PATH

ceph-bluestore-tool show-label --dev DEVICE …

ceph-bluestore-tool prime-osd-dir --dev DEVICE --path OSD_PATH

ceph-bluestore-tool bluefs-export --path OSD_PATH --out-dir DIR

ceph-bluestore-tool bluefs-bdev-new-wal --path OSD_PATH --dev-target NEW_DEVICE

ceph-bluestore-tool bluefs-bdev-new-db --path OSD_PATH --dev-target NEW_DEVICE

ceph-bluestore-tool bluefs-bdev-migrate --path OSD_PATH --dev-target NEW_DEVICE --devs-
source DEVICE1 [--devs-source DEVICE2]

ceph-bluestore-tool free-dump|free-score --path OSD_PATH [ --allocator block/bluefs-wal/
bluefs-db/bluefs-slow ]

ceph-bluestore-tool reshard --path OSD_PATH --sharding NEW_SHARDING [ --sharding-ctrl
CONTROL_STRING ]

ceph-bluestore-tool show-sharding --path OSD_PATH
```

Every BlueStore block device has a single block label at the beginning of the device. You can dump the contents of the label with:

```
ceph-bluestore-tool show-label --dev DEVICE
```

The main device contains a lot of metadata, including information that used to be stored in small files in the OSD data directory. The auxiliary devices (db and wal) only have the minimum required fields: OSD UUID, size, device type, and birth time).

Generate the content for an OSD data directory that can start up a BlueStore OSD with the **`prime-osd-dir`** command.

```
ceph-bluestore-tool prime-osd-dir --dev MAIN_DEVICE --path /var/lib/ceph/osd/ceph-ID
```

ceph-bluestore-tool commands	
Command	**Description**
`help`	Show help
`fsck [--deep]`	Options:on/`off`,`yes`/`no`,`1`/`0`, or`true`/`false`. Run consistency check on BlueStore metadata. If --deep is specified, also read all object data and verify checksums.
`repair`	Run a consistency check and repair any errors.
`qfsck`	Run consistency check on BlueStore metadata comparing allocator data with ONodes state. The allocator data comes from the RocksDB CFB, when exists, and if not uses allocation-file.
`allocmap`	Performs the same check done by **`qfsck`** and then stores a new allocation-file. This command is disabled by default and requires a special build.
`restore_cfb`	Reverses changes done by the new NCB code (either through **`ceph restart`** or when running the **`allocmap`** command) and restores RocksDB B Column-Family (allocator-map).
`bluefs-export`	Export the contents of BlueFS to an output directory.
`bluefs-bdev-sizes --path` *`OSD_PATH`*	Print the device sizes, as understood by BlueFS, to stdout.
`bluefs-bdev-expand --path` *`OSD_PATH`*	Instruct BlueFS to check the size of its block devices and, if they have expanded, make use of the additional space. Note that only the new files created by BlueFS will be allocated on the preferred block device if it has enough free space, and the existing files that have spilled over to the slow device will be gradually removed when RocksDB performs compaction. In other words, if there is any data spilled over to the slow device, it will be moved to the fast device over time.
`bluefs-bdev-new-wal --path` *`OSD_PATH`* `--dev-target` *`NEW_DEVICE`*	Adds WAL device to BlueFS, fails if WAL device already exists.
`bluefs-bdev-new-db --path` *`OSD_PATH`* `--dev-target` *`NEW_DEVICE`*	Adds DB device to BlueFS, fails if DB device already exists.
`bluefs-bdev-migrate --dev-target` *`NEW_DEVICE`* `--devs-source` *`DEVICE1`* `[--devs-source` *`DEVICE2`*`]`	Moves BlueFS data from source device(s) to the target one, source devices (except the main one) are removed on success. Target device can be both already attached or new device. In the latter case it's added to OSD replacing one of the source devices. Following replacement rules apply (in the order of precedence, stop on the first match): (1) if source list has DB volume - target device replaces it. (2) if source list has WAL volume - target device replace it. (3) if source list has slow volume only - operation isn't permitted, requires explicit allocation via new **`-db/new-wal`** command.
`show-label --dev` *`DEVICE`* `[…]`	Show any device labels.
`free-dump --path` *`OSD_PATH`* `[ --allocator block/bluefs-wal/bluefs-db/bluefs-slow ]`	Dump all free regions in allocator.

Command	Description
`free-score --path OSD_PATH [ --allocator block/bluefs-wal/bluefs-db/bluefs-slow ]`	Give a [0-1] number that represents quality of fragmentation in allocator. 0 represents case when all free space is in one chunk. 1 represents worst possible fragmentation.
`reshard --path OSD_PATH --sharding NEW_SHARDING [ --resharding-ctrl CONTROL_STRING ]`	Changes sharding of BlueStore's RocksDB. Sharding is build on top of RocksDB column families. This option allows to test performance of new sharding without need to redeploy OSD. Resharding is usually a long process, which involves walking through entire RocksDB key space and moving some of them to different column families. The **`--resharding-ctrl`** option provides performance control over resharding process. Interrupted resharding will prevent OSD from running. Interrupted resharding does not corrupt data. It is always possible to continue previous resharding, or select any other sharding scheme, including reverting to original one. For more information, see "Manually resharding the OSDs" on page 165.
`show-sharding --path OSD_PATH`	Show sharding that is currently applied to BlueStore's RocksDB.

ceph-bluestore-tool command options	
Command	**Description**
`--dev DEVICE`	Add the *device* to the list of devices to consider.
`-i OSD_ID`	Operate as OSD *OSD_ID*. Connect to monitor for OSD specific options. If monitor is unavailable, add **`--no-mon-config`** to read from ceph.conf instead.
`--devs-source DEVICE`	Add the _device_ to the list of devices to consider as sources for migration.
`--dev-target DEVICE`	Specify _target_ device migrate operation or device to add for adding new DB/WAL.
`--path OSD_PATH`	Specify an OSD path. In most cases, the device list is inferred from the symlinks present in osd path. This is usually simpler than explicitly specifying the device(s) with --dev. This option is not necessary if **`-i osd_id`** is provided.
`--out-dir DIR`	Output directory for `bluefs-export`
`-l`, `--log-file LOG_FILE`	The file to log to.
`--log-level NUM`	Debug log level. Default is 30 (extremely verbose), 20 is very verbose, 10 is verbose, and 1 is not very verbose.
`--deep`	Deep scrub/repair (read and validate object data, not just metadata).
`--allocator NAME`	Useful for free-dump and free-score actions. Selects allocator(s).

Command	Description
`--resharding-ctrl CONTROL_STRING`	Provides control over resharding process. Specifies how often refresh RocksDB iterator, and how large should commit batch be before committing to RocksDB. Option format is: `<iterator_refresh_bytes>/<iterator_refresh_keys>/<batch_commit_bytes>/<batch_commit_keys>` Default: 10000000/10000/1000000/1000

1. Stop the OSD before using the `ceph-bluestore-tool`.

   ```
   ceph orch daemon stop osd.ID
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph orch daemon stop osd.2
   ```

2. From the OSD node, log into the target OSD container.

   ```
   cephadm shell --name osd.ID
   ```

   For example,

   ```
   [ceph: root@host01 /]# cephadm shell --name osd.2
   ```

3. Run the needed command.
   The following example is adding a new wal device.

   ```
   [ceph: root@host01 /]# ceph-bluestore-tool bluefs-bdev-new-wal --dev-target /dev/test/
   newdb --path /var/lib/ceph/osd/ceph-0
   ```

4. From the `cephadm` shell, restart the OSD.

   ```
   ceph orch daemon start osd.ID
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph orch daemon start osd.2
   ```

Related information

BlueStore configuration options

`cephadm` operations

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

Prerequisites

- A running Red Hat Ceph Storage cluster.

Monitor `cephadm` log messages

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

- To monitor progress in real time, run the following command:

  Example

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

Example

```
2022-11-02T17:51:36.335728+0000 mgr.Ceph5-1.nqikfh [INF] refreshing Ceph5-adm facts
2022-11-02T17:51:37.170982+0000 mgr.Ceph5-1.nqikfh [INF] deploying 1 monitor(s) instead
of 2 so monitors may achieve consensus
2022-11-02T17:51:37.173487+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]
2022-11-02T17:51:37.179658+0000 mgr.Ceph5-1.nqikfh [INF] Found osd claims -> {}
2022-11-02T17:51:37.180116+0000 mgr.Ceph5-1.nqikfh [INF] Found osd claims for
drivegroup all-available-devices -> {}
2022-11-02T17:51:37.182138+0000 mgr.Ceph5-1.nqikfh [INF] Applying all-available-devices
on host Ceph5-adm...
2022-11-02T17:51:37.182987+0000 mgr.Ceph5-1.nqikfh [INF] Applying all-available-devices
on host Ceph5-1...
2022-11-02T17:51:37.183395+0000 mgr.Ceph5-1.nqikfh [INF] Applying all-available-devices
on host Ceph5-2...
2022-11-02T17:51:43.373570+0000 mgr.Ceph5-1.nqikfh [INF] Reconfiguring node-
exporter.Ceph5-1 (unknown last config time)...
2022-11-02T17:51:43.373840+0000 mgr.Ceph5-1.nqikfh [INF] Reconfiguring daemon node-
exporter.Ceph5-1 on Ceph5-1
```

– By default, the log displays info-level events. 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
  [ceph: root@host01 /]# ceph -W cephadm --verbose
  ```

– Return debugging level to default `info`:

  Example

  ```
  [ceph: root@host01 /]# ceph config set mgr mgr/cephadm/log_to_cluster_level info
  ```

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

  Example

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

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

Ceph daemon logs

You can view the Ceph daemon logs through `stderr` or files.

Logging to stderr directory

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**:

  ```
  [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:

  ```
  [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:

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

Important: Log rotation to a non-default path is not currently supported.

Note: Disable logging to `stderr` to avoid double logs.

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`.

Data location

Know where `cephadm` daemon data and logs are located.

- `/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 a 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 move this directory to its own disk, partition, or logical volume so that the root file system is not filled up.

`cephadm` health checks

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

`cephadm` operations health checks

Health checks are run 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 daemon, 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 run 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.

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.

Enabling cephadm configuration health checks

Enable cephadm configuration health checks by using the following command:

```
[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.

Viewing the current state

Use the **ceph -W cephadm** command to show log entries of the current state and display the output of the configuration checks.

Example output for disabled state

```
ALL cephadm checks are disabled, use ceph config set mgr mgr/cephadm/
config_checks_enabled true command to enable
```

Example output for enabled state

```
CEPHADM 8/8 checks enabled and executed (0 bypassed, 0 disabled). No issues detected
```

Managing configuration checks

The configuration checks themselves are managed through several cephadm subcommands.

Determine whether the configuration checks are enabled by running the following command:

```
ceph cephadm config-check status
```

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

List all the configuration checks and their current state by running the following command:

```
ceph cephadm config-check ls
```

For 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 health check.

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 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.

cephadm agent (Technology Preview)

The cephadm agent is a service that runs on each host in a Ceph cluster and provides host and daemon metadata to the Ceph Manager.

By default, the Ceph Manager retrieves host information by using SSH connections. When the cephadm agent is enabled, each host collects metadata locally and sends updates to the manager, reducing the need for repeated SSH operations and improving responsiveness.

Important: Technology Preview features are not supported with production service level agreements (SLAs), might not be functionally complete, and does not recommend using them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

Benefits

- Reduces SSH-based communication overhead
- Improves scalability for large clusters
- Provides near real-time updates of host and daemon state
- Improves responsiveness of orchestration operations

How the cephadm agent works

When enabled, the cephadm agent integrates with the cephadm orchestrator and the Ceph Manager module to provide agent-based host monitoring and metadata collection.

1. The `mgr/cephadm/use_agent` configuration option is enabled
2. The cephadm module deploys the agent service on all hosts
3. Each host runs the agent as a systemd service
4. The agent collects host and daemon metadata
5. The agent sends metadata updates to the Ceph Manager over HTTPS
6. The manager processes the data and updates cluster state

Agent-based and SSH-based host management

In SSH-based mode, the Ceph Manager connects to each host on demand to retrieve host and daemon information. This approach relies on repeated SSH operations.

In agent-based mode, each host runs the cephadm agent and periodically sends metadata updates to the manager. This reduces the need for SSH connections and enables more efficient and timely host monitoring.

Metadata collection and synchronization

The cephadm agent replaces SSH-based metadata collection when enabled. Each host periodically collects and sends metadata to the manager at a defined refresh interval.

The Ceph Manager waits until host metadata is up to date before applying service specifications, ensuring consistent and accurate orchestration behavior.

Configuration behavior

The cephadm agent is controlled by configuration options stored in the monitor (MON) database. These options belong to the `mgr` subsystem.

When configuration values are updated, monitors notify active manager daemons, which then apply the changes across the cluster.

Managing the cephadm agent

Enable, disable, and verify the cephadm agent that provides host metadata to the Ceph Manager.

PREREQUISITE

Ensure that you have administrator access to run Ceph CLI commands.

Use the following commands to manage the cephadm agent. When enabled, the agent collects host metadata and sends updates to the Ceph Manager. For more information about how the agent works, see “cephadm agent (Technology Preview)” on page 178.

- Enable the cephadm agent.

```
ceph config set mgr mgr/cephadm/use_agent true
```

- Disable the cephadm agent.

```
ceph config set mgr mgr/cephadm/use_agent false
```

- Verify whether the cephadm agent is enabled or disabled.

```
ceph config get mgr mgr/cephadm/use_agent
```

- Verify that the cephadm agent is running on hosts.

```
ceph orch ps --daemon-type agent
```

- View the cephadm agent logs on a host.

```
journalctl -u ceph-<fsid>@agent.<host>
```

Result

The cephadm agent is enabled or disabled based on the configuration. When enabled, the agent runs on each host and reports metadata to the Ceph Manager.

Using `cephadm-ansible` modules

Use `cephadm-ansible` modules in Ansible playbooks to administer your Red Hat Ceph Storage cluster.

The `cephadm-ansible` package provides several modules that wrap `cephadm` calls to let you write your own unique Ansible playbooks to administer your cluster.

> **Note:** `cephadm-ansible` modules only support the most important tasks. Any operation not covered by `cephadm-ansible` modules must be completed using either the `command` or `shell` Ansible modules in your playbooks.

`cephadm-ansible` modules

The `cephadm-ansible` modules are a collection of modules that simplify writing Ansible playbooks by providing a wrapper around `cephadm` and `ceph orch` commands. You can use the modules to write your own unique Ansible playbooks to administer your cluster using one or more of the modules.

The `cephadm-ansible` package includes the following modules:

- `cephadm_bootstrap`
- `ceph_orch_host`
- `ceph_config`
- `ceph_orch_apply`
- `ceph_orch_daemon`
- `cephadm_registry_login`

`cephadm-ansible` module options

This information lists available options for the `cephadm-ansible` modules.

Options listed as required need to be set when using the modules in your Ansible playbooks. Options listed with a default value of `true` indicate that the option is automatically set when using the modules and you do not need to specify it in your playbook. For example, for the `cephadm_bootstrap` module, the Ceph Dashboard is installed unless you set `dashboard: false`.

Available options for the `cephadm_bootstrap` module			
cephadm_bootstrap	**Description**	**Required**	**Default**
`mon_ip`	Ceph Monitor IP address.	true	
`image`	Ceph container image.	false	
`docker`	Use `docker` instead of `podman`.	false	
`fsid`	Define the Ceph FSID.	false	
`pull`	Pull the Ceph container image.	false	true
`dashboard`	Deploy the Ceph Dashboard.	false	true
`dashboard_user`	Specify a specific Ceph Dashboard user.	false	
`dashboard_password`	Ceph Dashboard password.	false	
`monitoring`	Deploy the monitoring stack.	false	true
`firewalld`	Manage firewall rules with firewalld.	false	true
`allow_overwrite`	Allow overwrite of existing --output-config, --output-keyring, or --output-pub-ssh-key files.	false	false
`registry_url`	URL for custom registry.	false	
`registry_username`	Username for custom registry.	false	
`registry_password`	Password for custom registry.	false	
`registry_json`	JSON file with custom registry login information.	false	
`ssh_user`	SSH user to use for `cephadm` ssh to hosts.	false	
`ssh_config`	SSH config file path for `cephadm` SSH client.	false	
`allow_fqdn_hostname`	Allow hostname that is a fully-qualified domain name (FQDN).	false	false
`cluster_network`	Subnet to use for cluster replication, recovery and heartbeats.	false	

Available options for the `ceph_orch_host` module			
`ceph_orch_host`	**Description**	**Required**	**Default**
`fsid`	The FSID of the Ceph cluster to interact with.	false	
`image`	The Ceph container image to use.	false	
`name`	Name of the host to add, remove, or update.	true	
`address`	IP address of the host.	true when `state` is `present`.	
`set_admin_label`	Set the `_admin` label on the specified host.	false	false
`labels`	The list of labels to apply to the host.	false	[]
`state`	If set to `present`, it ensures the name specified in `name` is present. If set to `absent`, it removes the host specified in `name`. If set to `drain`, it schedules to remove all daemons from the host specified in `name`.	false	present

Available options for the `ceph_config` module			
`ceph_config`	**Description**	**Required**	**Default**
`fsid`	The FSID of the Ceph cluster to interact with.	false	
`image`	The Ceph container image to use.	false	
`action`	Whether to `set` or `get` the parameter specified in `option`.	false	set
`who`	Which daemon to set the configuration to.	true	
`option`	Name of the parameter to `set` or `get`.	true	
`value`	Value of the parameter to set.	true if action is `set`	

Available options for the `ceph_orch_apply` module		
`ceph_orch_apply`	**Description**	**Required**
`fsid`	The FSID of the Ceph cluster to interact with.	false
`image`	The Ceph container image to use.	false

ceph_orch_apply	Description	Required
spec	The service specification to apply.	true

Available options for the ceph_orch_daemon module		
ceph_orch_daemon	**Description**	**Required**
fsid	The FSID of the Ceph cluster to interact with.	false
image	The Ceph container image to use.	false
state	The desired state of the service specified in name.	true If started, it ensures the service is started. If stopped, it ensures the service is stopped. If restarted, it will restart the service.
daemon_id	The ID of the service.	true
daemon_type	The type of service.	true

Available options for the cephadm_registry_login module			
cephadm_registry_login	**Description**	**Required**	**Default**
state	Login or logout of a registry.	false	login
docker	Use docker instead of podman.	false	
registry_url	The URL for custom registry.	false	
registry_username	Username for custom registry.	true when state is login.	
registry_password	Password for custom registry.	true when state is login.	
registry_json	The path to a JSON file. This file must be present on remote hosts prior to running this task. This option is currently not supported.		

Bootstrapping a storage cluster with Ansible

As a storage administrator, you can bootstrap a storage cluster using Ansible by using the cephadm_bootstrap and cephadm_registry_login modules in your Ansible playbook.

Before you begin

- An IP address for the first Ceph Monitor container, which is also the IP address for the first node in the storage cluster.
- Login access to registry.redhat.io.

- A minimum of 10 GB of free space for `/var/lib/containers/`.
- Installation of the `cephadm-ansible` package on the Ansible administration node.
- Passwordless SSH is set up on all hosts in the storage cluster.
- Hosts are registered with CDN.

For the latest supported Red Hat Enterprise Linux versions, see Compatibility matrix.

Procedure

1. Log in to the Ansible administration node.
2. Navigate to the `/usr/share/cephadm-ansible` directory on the Ansible administration node.
   For example,

   ```
   [ansible@admin ~]$ cd /usr/share/cephadm-ansible
   ```

3. Create the `hosts` file and add hosts, labels, and monitor IP address of the first host in the storage cluster.

   ```
   sudo vi INVENTORY_FILE

   HOST1 labels="['LABEL1', 'LABEL2']"
   HOST2 labels="['LABEL1', 'LABEL2']"
   HOST3 labels="['LABEL1']"

   [admin]
   ADMIN_HOST monitor_address=MONITOR_IP_ADDRESS labels="['ADMIN_LABEL', 'LABEL1',
   'LABEL2']"
   ```

   For example,

   ```
   [ansible@admin cephadm-ansible]$ sudo vi hosts

   host02 labels="['mon', 'mgr']"
   host03 labels="['mon', 'mgr']"
   host04 labels="['osd']"
   host05 labels="['osd']"
   host06 labels="['osd']"

   [admin]
   host01 monitor_address=10.10.128.68 labels="['_admin', 'mon', 'mgr']"
   ```

4. Run the preflight playbook:

   ```
   ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --extra-vars
   "ceph_origin=rhcs"
   ```

   For example,

   ```
   [ansible@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --
   extra-vars "ceph_origin=rhcs""
   ```

5. Create a playbook to bootstrap your cluster.

   ```
   sudo vi PLAYBOOK_FILENAME.yml

   ---
   - name: NAME_OF_PLAY
     hosts: BOOTSTRAP_HOST
     become: USE_ELEVATED_PRIVILEGES
     gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
     tasks:
       -name: NAME_OF_TASK
        cephadm_registry_login:
          state: STATE
          registry_url: REGISTRY_URL
          registry_username: REGISTRY_USER_NAME
          registry_password: REGISTRY_PASSWORD

       - name: NAME_OF_TASK
         cephadm_bootstrap:
   ```

```
        mon_ip: "{{ monitor_address }}"
        dashboard_user: DASHBOARD_USER
        dashboard_password: DASHBOARD_PASSWORD
        allow_fqdn_hostname: ALLOW_FQDN_HOSTNAME
        cluster_network: NETWORK_CIDR
```

For example,

```
sudo vi bootstrap.yml

---
- name: bootstrap the cluster
  hosts: host01
  become: true
  gather_facts: false
  tasks:
    - name: login to registry
      cephadm_registry_login:
        state: login
        registry_url: registry.redhat.io
        registry_username: user1
        registry_password: mypassword1

    - name: bootstrap initial cluster
      cephadm_bootstrap:
        mon_ip: "{{ monitor_address }}"
        dashboard_user: mydashboarduser
        dashboard_password: mydashboardpassword
        allow_fqdn_hostname: true
        cluster_network: 10.10.128.0/28
```

6. Run the playbook.

```
ansible-playbook -i INVENTORY_FILE PLAYBOOK_FILENAME.yml -vvv
```

For example,

```
ansible@admin cephadm-ansible]$ ansible-playbook -i hosts bootstrap.yml -vvv
```

Verification

Review the Ansible output after running the playbook.

Adding and removing hosts

Add and remove hosts in your storage cluster by using the `ceph_orch_host` module in your Ansible playbook.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Register the nodes to the CDN and attach subscriptions.
- Ansible user with sudo and passwordless SSH access to all nodes in the storage cluster.
- Installation of the `cephadm-ansible` package on the Ansible administration node.
- New hosts have the storage cluster's public SSH key.
  For more information about copying the storage cluster's public SSH keys to new hosts, see "Adding hosts" on page 201.

Adding hosts

Add new hosts to the storage cluster.

1. Log in to the Ansible administration node.
2. Navigate to the `/usr/share/cephadm-ansible` directory on the Ansible administration node.
   For example,

```
[ansible@admin ~]$ cd /usr/share/cephadm-ansible
```

3. Add the new hosts and labels to the Ansible inventory file.

> **Note:** If you have previously added the new hosts to the Ansible inventory file and ran the preflight playbook on the hosts, skip to step “5” on page 186.

```
sudo vi INVENTORY_FILE

NEW_HOST1 labels="['LABEL1', 'LABEL2']"
NEW_HOST2 labels="['LABEL1', 'LABEL2']"
NEW_HOST3 labels="['LABEL1']"

[admin]
ADMIN_HOST monitor_address=MONITOR_IP_ADDRESS labels="['ADMIN_LABEL', 'LABEL1',
'LABEL2']"
```

For example,

```
[ansible@admin cephadm-ansible]$ sudo vi hosts

host02 labels="['mon', 'mgr']"
host03 labels="['mon', 'mgr']"
host04 labels="['osd']"
host05 labels="['osd']"
host06 labels="['osd']"

[admin]
host01 monitor_address= 10.10.128.68 labels="['_admin', 'mon', 'mgr']"
```

4. Run the preflight playbook with the `--limit` option.

```
ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --extra-vars
"ceph_origin=rhcs" --limit NEWHOST
```

For example,

```
[ansible@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --
extra-vars "ceph_origin=rhcs" --limit host02
```

The preflight playbook installs `podman`, `lvm2`, `chrony`, and `cephadm` on the new host. After installation is complete, `cephadm` resides in the `/usr/sbin/` directory.

5. Create a playbook to add the new hosts to the cluster.

```
sudo vi PLAYBOOK_FILENAME.yml

---
- name: PLAY_NAME
  hosts: HOSTS_OR_HOST_GROUPS
  become: USE_ELEVATED_PRIVILEGES
  gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
  tasks:
    - name: NAME_OF_TASK
      ceph_orch_host:
        name: "{{ ansible_facts[hostname] }}"
        address: "{{ ansible_facts[default_ipv4][address] }}"
        labels: "{{ labels }}"
      delegate_to: HOST_TO_DELEGATE_TASK_TO

    - name: NAME_OF_TASK
      when: inventory_hostname in groups[admin]
      ansible.builtin.shell:
        cmd: CEPH_COMMAND_TO_RUN
      register: REGISTER_NAME

    - name: NAME_OF_TASK
      when: inventory_hostname in groups[admin]
      debug:
        msg: "{{ REGISTER_NAME.stdout }}"
```

> **Note:** By default, Ansible runs all tasks on the host that matches the `hosts` line of your playbook. The **ceph orch** commands must run on the host that contains the admin keyring and the Ceph configuration file. Use the `delegate_to` keyword to specify the admin host in your cluster.

In the following example, the playbook adds the new hosts to the cluster and displays a current list of hosts.

```
[ansible@admin cephadm-ansible]$ sudo vi add-hosts.yml

---
- name: add additional hosts to the cluster
  hosts: all
  become: true
  gather_facts: true
  tasks:
    - name: add hosts to the cluster
      ceph_orch_host:
        name: "{{ ansible_facts['hostname'] }}"
        address: "{{ ansible_facts['default_ipv4']['address'] }}"
        labels: "{{ labels }}"
      delegate_to: host01

    - name: list hosts in the cluster
      when: inventory_hostname in groups['admin']
      ansible.builtin.shell:
        cmd: ceph orch host ls
      register: host_list

    - name: print current list of hosts
      when: inventory_hostname in groups['admin']
      debug:
        msg: "{{ host_list.stdout }}"
```

6. Run the playbook to add additional hosts to the cluster.

```
ansible-playbook -i INVENTORY_FILE PLAYBOOK_FILENAME.yml
```

For example,

```
[ansible@admin cephadm-ansible]$ ansible-playbook -i hosts add-hosts.yml
```

AFTER COMPLETING THE TASK

Review the Ansible task output displaying the current list of hosts in the cluster.
For example,

```
TASK [print current hosts]
*****************************************************************************************
***********
Friday 24 April 2026  14:52:40 -0400 (0:00:03.365)       0:02:31.702 ***********
ok: [host01] =>
  msg: |-
    HOST    ADDR           LABELS          STATUS
    host01  10.10.128.68   _admin mon mgr
    host02  10.10.128.69   mon mgr
    host03  10.10.128.70   mon mgr
    host04  10.10.128.71   osd
    host05  10.10.128.72   osd
    host06  10.10.128.73   osd
```

Removing hosts

Remove hosts from the cluster.

1. Log in to the Ansible administration node.
2. Navigate to the `/usr/share/cephadm-ansible` directory on the Ansible administration node.
   For example,

```
[ansible@admin ~]$ cd /usr/share/cephadm-ansible
```

3. Create a playbook to remove a host or multiple hosts from the cluster.

```
sudo vi PLAYBOOK_FILENAME.yml

---
- name: NAME_OF_PLAY
  hosts: ADMIN_HOST
  become: USE_ELEVATED_PRIVILEGES
  gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
  tasks:
    - name: NAME_OF_TASK
      ceph_orch_host:
        name: HOST_TO_REMOVE
        state: STATE

    - name: NAME_OF_TASK
      ceph_orch_host:
        name: HOST_TO_REMOVE
        state: STATE
      retries: NUMBER_OF_RETRIES
      delay: DELAY
      until: CONTINUE_UNTIL
      register: REGISTER_NAME

    - name: NAME_OF_TASK
      ansible.builtin.shell:
        cmd: ceph orch host ls
      register: REGISTER_NAME

    - name: NAME_OF_TASK
        debug:
          msg: "{{ REGISTER_NAME.stdout }}"
```

In the following example, the playbook tasks drain all daemons on `host07`, removes the host from the cluster, and displays a current list of hosts.

```
[ansible@admin cephadm-ansible]$ sudo vi remove-hosts.yml

---
- name: remove host
  hosts: host01
  become: true
  gather_facts: true
  tasks:
    - name: drain host07
      ceph_orch_host:
        name: host07
        state: drain

    - name: remove host from the cluster
      ceph_orch_host:
        name: host07
        state: absent
      retries: 20
      delay: 1
      until: result is succeeded
      register: result

     - name: list hosts in the cluster
       ansible.builtin.shell:
         cmd: ceph orch host ls
       register: host_list

     - name: print current list of hosts
       debug:
         msg: "{{ host_list.stdout }}"
```

4. Run the playbook to remove the hosts from the cluster.

```
ansible-playbook -i INVENTORY_FILE PLAYBOOK_FILENAME.yml
```

For example,

```
[ansible@admin cephadm-ansible]$ ansible-playbook -i hosts remove-hosts.yml
```

AFTER COMPLETING THE TASK

Review the Ansible task output displaying the current list of hosts in the cluster.
For example,

```
TASK [print current hosts]
*****************************************************************************************
***********
Friday 24 April 2026  14:52:40 -0400 (0:00:03.365)       0:02:31.702 ***********
ok: [host01] =>
  msg: |-
    HOST    ADDR           LABELS          STATUS
    host01  10.10.128.68   _admin mon mgr
    host02  10.10.128.69   mon mgr
    host03  10.10.128.70   mon mgr
    host04  10.10.128.71   osd
    host05  10.10.128.72   osd
    host06  10.10.128.73   osd
```

Setting configuration options

As a storage administrator, you can set or get Red Hat Ceph Storage configuration options using the `ceph_config` module.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Ansible user with sudo and passwordless SSH access to all nodes in the storage cluster.
- Installation of the `cephadm-ansible` package on the Ansible administration node.
- The Ansible inventory file contains the cluster and admin hosts.

Procedure

1. Log in to the Ansible administration node.
2. Navigate to the `/usr/share/cephadm-ansible` directory on the Ansible administration node:

   Example

   ```
   [ansible@admin ~]$ cd /usr/share/cephadm-ansible
   ```

3. Create a playbook with configuration changes:

   Syntax

   ```
   sudo vi PLAYBOOK_FILENAME.yml

   ---
   - name: PLAY_NAME
     hosts: ADMIN_HOST
     become: USE_ELEVATED_PRIVILEGES
     gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
     tasks:
       - name: NAME_OF_TASK
         ceph_config:
           action: GET_OR_SET
           who: DAEMON_TO_SET_CONFIGURATION_TO
           option: CEPH_CONFIGURATION_OPTION
           value: VALUE_OF_PARAMETER_TO_SET

       - name: NAME_OF_TASK
         ceph_config:
           action: GET_OR_SET
           who: DAEMON_TO_SET_CONFIGURATION_TO
           option: CEPH_CONFIGURATION_OPTION
         register: REGISTER_NAME

       - name: NAME_OF_TASK
         debug:
           msg: "MESSAGE_TO_DISPLAY {{ REGISTER_NAME.stdout }}"
   ```

   Example

```
[ansible@admin cephadm-ansible]$ sudo vi change_configuration.yml

---
- name: set pool delete
  hosts: host01
  become: true
  gather_facts: false
  tasks:
    - name: set the allow pool delete option
      ceph_config:
        action: set
        who: mon
        option: mon_allow_pool_delete
        value: true

    - name: get the allow pool delete setting
      ceph_config:
        action: get
        who: mon
        option: mon_allow_pool_delete
      register: verify_mon_allow_pool_delete

    - name: print current mon_allow_pool_delete setting
      debug:
        msg: "the value of 'mon_allow_pool_delete' is
{{ verify_mon_allow_pool_delete.stdout }}"
```

In this example, the playbook first sets the `mon_allow_pool_delete` option to `false`. The playbook then gets the current `mon_allow_pool_delete` setting and displays the value in the Ansible output.

4. Run the playbook:

   Syntax

   ```
   ansible-playbook -i INVENTORY_FILE PLAYBOOK_FILENAME.yml
   ```

   Example

   ```
   [ansible@admin cephadm-ansible]$ ansible-playbook -i hosts change_configuration.yml
   ```

Verification

- Review the output from the playbook tasks.

  Example

  ```
  TASK [print current mon_allow_pool_delete setting]
  ************************************************************
  Wednesday 29 June 2022  13:51:41 -0400 (0:00:05.523)       0:00:17.953 ********
  ok: [host01] =>
    msg: the value of 'mon_allow_pool_delete' is true
  ```

Reference

- For more information about configuration options, see Configuring.

Applying a service specification

As a storage administrator, you can apply service specifications to your storage cluster using the `ceph_orch_apply` module in your Ansible playbooks.

A service specification is a data structure to specify the service attributes and configuration settings that is used to deploy the Ceph service. You can use a service specification to deploy Ceph service types like `mon`, `crash`, `mds`, `mgr`, `osd`, `rdb`, or `rbd-mirror`.

For more information about service specification options, see .

Prerequisites

- A running Red Hat Ceph Storage cluster.

- Ansible user with sudo and passwordless SSH access to all nodes in the storage cluster.
- Installation of the `cephadm-ansible` package on the Ansible administration node.
- The Ansible inventory file contains the cluster and admin hosts.

Procedure

1. Log in to the Ansible administration node.
2. Navigate to the `/usr/share/cephadm-ansible` directory on the Ansible administration node:

   Example

   ```
   [ansible@admin ~]$ cd /usr/share/cephadm-ansible
   ```

3. Create a playbook with the service specifications.

   ```
   sudo vi PLAYBOOK_FILENAME.yml

   ---
   - name: PLAY_NAME
     hosts: HOSTS_OR_HOST_GROUPS
     become: USE_ELEVATED_PRIVILEGES
     gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
     tasks:
       - name: NAME_OF_TASK
         ceph_orch_apply:
           spec: |
             service_type: SERVICE_TYPE
             service_id: UNIQUE_NAME_OF_SERVICE
             placement:
               host_pattern: HOST_PATTERN_TO_SELECT_HOSTS
               label: LABEL
             spec:
               SPECIFICATION_OPTIONS:
   ```

   Example

   ```
   [ansible@admin cephadm-ansible]$ sudo vi deploy_osd_service.yml

   ---
   - name: deploy osd service
     hosts: host01
     become: true
     gather_facts: true
     tasks:
       - name: apply osd spec
         ceph_orch_apply:
           spec: |
             service_type: osd
             service_id: osd
             placement:
               host_pattern: '*'
               label: osd
             spec:
               data_devices:
                 all: true
   ```

   In this example, the playbook deploys the Ceph OSD service on all hosts with the label `osd`.

4. Run the playbook.

   ```
   ansible-playbook -i INVENTORY_FILE PLAYBOOK_FILENAME.yml
   ```

   Example

   ```
   [ansible@admin cephadm-ansible]$ ansible-playbook -i hosts deploy_osd_service.yml
   ```

Verification

- Review the output from the playbook tasks.

Managing Ceph daemon states

As a storage administrator, you can start, stop, and restart Ceph daemons on hosts using the `ceph_orch_daemon` module in your Ansible playbooks.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Ansible user with sudo and passwordless SSH access to all nodes in the storage cluster.
- Installation of the `cephadm-ansible` package on the Ansible administration node.
- The Ansible inventory file contains the cluster and admin hosts.

Procedure

1. Log in to the Ansible administration node.
2. Navigate to the `/usr/share/cephadm-ansible` directory on the Ansible administration node:

   Example

   ```
   [ansible@admin ~]$ cd /usr/share/cephadm-ansible
   ```

3. Create a playbook with daemon state changes:

   Syntax

   ```
   sudo vi PLAYBOOK_FILENAME.yml

   ---
   - name: PLAY_NAME
     hosts: ADMIN_HOST
     become: USE_ELEVATED_PRIVILEGES
     gather_facts: GATHER_FACTS_ABOUT_REMOTE_HOSTS
     tasks:
       - name: NAME_OF_TASK
         ceph_orch_daemon:
           state: STATE_OF_SERVICE
           daemon_id: DAEMON_ID
           daemon_type: TYPE_OF_SERVICE
   ```

   Example

   ```
   [ansible@admin cephadm-ansible]$ sudo vi restart_services.yml

   ---
   - name: start and stop services
     hosts: host01
     become: true
     gather_facts: false
     tasks:
       - name: start osd.0
         ceph_orch_daemon:
           state: started
           daemon_id: 0
           daemon_type: osd

       - name: stop mon.host02
         ceph_orch_daemon:
           state: stopped
           daemon_id: host02
           daemon_type: mon
   ```

   In this example, the playbook starts the OSD with an ID of `0` and stops a Ceph Monitor with an id of `host02`.

4. Run the playbook:

   Syntax

```
ansible-playbook -i INVENTORY_FILE PLAYBOOK_FILENAME.yml
```

Example

```
[ansible@admin cephadm-ansible]$ ansible-playbook -i hosts restart_services.yml
```

Verification

- Review the output from the playbook tasks.

Operations

Learn to do operational tasks for Red Hat Ceph Storage.

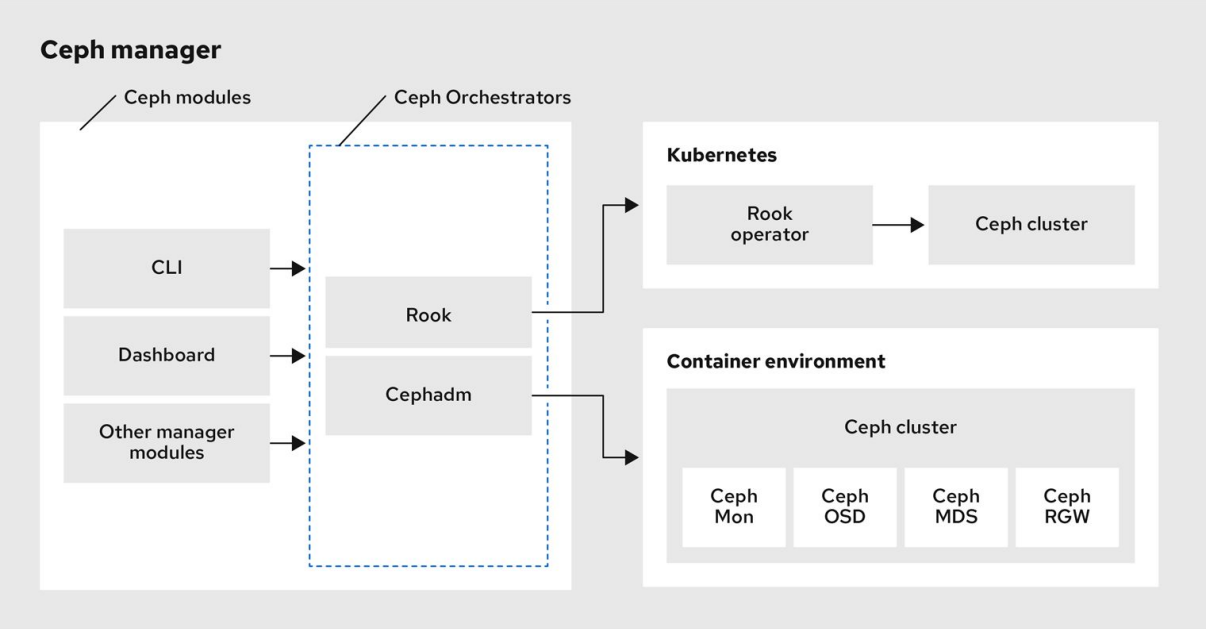
Introduction to the Ceph Orchestrator

Red Hat Ceph Storage Orchestrators are manager modules that primarily act as a bridge between a Red Hat Ceph Storage cluster and deployment tools like Rook and `cephadm` for a unified experience. They also integrate with the Ceph command line interface and Ceph Dashboard.

As a storage administrator, you can use the Ceph Orchestrator with `cephadm` utility that provides the ability to discover devices and create services in a Red Hat Ceph Storage cluster.

"Figure: Ceph Orchestrator" on page 193 is a workflow diagram of Ceph Orchestrator.

Figure: Ceph Orchestrator



Types of Red Hat Ceph Storage

There are three main types of Red Hat Ceph Storage Orchestrators:

Orchestrator CLI

These are common APIs used in Orchestrators and include a set of commands that can be implemented. These APIs also provide a common command line interface (CLI) to orchestrate `ceph-mgr` modules with external orchestration services. The following are the nomenclature that is used with the Ceph Orchestrator:

- Host: This is the host name of the physical host and not the pod name, DNS name, container name, or host name inside the container.

- Service type: This is the type of the service, such as `mds`, `osd`, `mon`, `rgw`, and `mgr`.
- Service: A functional service that is provided by a Ceph storage cluster such as monitors service, managers service, OSD services, and Ceph Object Gateway service.
- Daemon: A specific instance of a service that is deployed by one or more hosts such as Ceph Object Gateway services can have different Ceph Object Gateway daemons running in three different hosts.

`cephadm` Orchestrator

This is a Ceph Orchestrator module that does not rely on an external tool such as Rook or Ansible, but rather manages nodes in a cluster by establishing an SSH connection and issuing explicit management commands. This module is intended for day-one and day-two operations.

Using the `cephadm` Orchestrator is the recommended way of installing a Ceph storage cluster without using any deployment frameworks like Ansible. The idea is to provide the manager daemon with access to an SSH configuration and key that is able to connect to all nodes in a cluster to perform any management operations, like creating an inventory of storage devices, deploying and replacing OSDs, or starting and stopping Ceph daemons. In addition, the `cephadm` Orchestrator will deploy container images that are managed by `systemd` in order to allow independent upgrades of colocated services.

This orchestrator also highlights a tool that encapsulates all necessary operations to manage the deployment of container image-based services on the current host, including a command that bootstraps a minimal cluster that runs a Ceph Monitor and a Ceph Manager.

Rook Orchestrator

Rook is an orchestration tool that uses the Kubernetes Rook operator to manage a Ceph storage cluster running inside a Kubernetes cluster. The rook module provides integration between Ceph's Orchestrator framework and Rook. Rook is an open source cloud-native storage operator for Kubernetes.

Rook follows the "operator" model, in which a custom resource definition (CRD) object is defined in Kubernetes to describe a Ceph storage cluster and its desired state, and a rook operator daemon is running in a control loop that compares the current cluster state to desired state and takes steps to make them converge. The main object describing Ceph's desired state is the Ceph storage cluster CRD, which includes information about which devices should be consumed by OSDs, how many monitors should be running, and what version of Ceph must be used. Rook defines several other CRDs to describe RBD pools, CephFS file systems, and so on.

The Rook Orchestrator module is the glue that runs in the `ceph-mgr` daemon and implements the Ceph orchestration API by making changes to the Ceph storage cluster in Kubernetes that describe desired cluster state. A Rook cluster's `ceph-mgr` daemon is running as a Kubernetes pod, and hence, the rook module can connect to the Kubernetes API without any explicit configuration.

Managing services

As a storage administrator, after installing the Red Hat Ceph Storage cluster, you can monitor and manage the services in a storage cluster. A service is a group of daemons that are configured together.

Checking service status

Check the service status of the storage cluster, by using the **`ceph orch ls`** command.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Log in to the `cephadm` shell.

You can check the service status of the storage cluster with the **`ceph orch ls`** command. This command checks the status of the following services of the storage cluster:

- Print a list of services.
- Locate the service whose status you want to check.
- Print the status of the service.

Note: If the services are applied with the **ceph orch apply** command while bootstrapping, changing the service specification file is complicated. Instead, you can use the `--export` option with the **ceph orch ls** command to export the running specification, update the `yaml` file, and reapply the service.

- Print a list of services.

```
ceph orch ls [--service_type SERVICE_TYPE] [--service_name SERVICE_NAME] [--export] [--
format FORMAT] [--refresh]
```

  The format can be plain, `json`, `json-pretty`, `yaml`, `xml-pretty`, or `xml`.

  For example,

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

- Check the status of a particular service or a daemon.

```
ceph orch ls [--service_type SERVICE_TYPE] [--service_name SERVICE_NAME] [--refresh]
```

  For example,

```
ceph: root@host01 /]# ceph orch ls --service-type mds
[ceph: root@host01 /]# ceph orch ls --service-name rgw.realm.myzone
```

- Export the service specification.
  For example,

```
[ceph: root@host01 /]# ceph orch ls --service-type mgr --export > mgr.yaml
[ceph: root@host01 /]# ceph orch ls --export > cluster.yaml
```

  This exports the file in the `.yaml` file format. This file can be used with the **ceph orch apply -i** command to retrieve the service specification of a single service.

Checking daemon status

A daemon is a `systemd` unit that is running and is part of the service. The daemon status can be checked by using the **ceph orch ps** command.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Log into the `cephadm` shell.

You can check the following status of the daemons of the storage cluster using the `ceph orch ps` command. This command checks provides the following information:

- Print a list of all the daemons.
- Query the status of the target daemon.

1. Print a list of daemons.

```
ceph orch ps [--daemon-type DAEMON_TYPE] [--service_name SERVICE_NAME] [--
daemon_id DAEMON_ID] [--format FORMAT] [--refresh]
```

   For example,

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

2. Verify the status of the service instance.

```
ceph orch ls [--daemon-type DAEMON_TYPE] [--daemon_id DAEMON_ID] [--refresh]
```

For example,

```
[ceph: root@host01 /]# ceph orch ps --daemon_type osd --daemon_id 0
```

Placement specification of the Ceph Orchestrator

You can use the Ceph Orchestrator to deploy `osds`, `mons`, `mgrs`, `mds`, and `rgw` services. Red Hat recommends deploying services using placement specifications. You need to know where and how many daemons have to be deployed to deploy a service. Placement specifications can either be passed as command line arguments or as a service specification in a `yaml` file.

There are two ways of deploying the services using the placement specification:

- Using the placement specification directly in the command line interface. For example, if you want to deploy three monitors on the hosts, running the following command deploys three monitors on `host01`, `host02`, and `host03`.

  Example

  ```
  [ceph: root@host01 /]# ceph orch apply mon --placement="3 host01 host02 host03"
  ```

- Using the placement specification in the YAML file. For example, if you want to deploy `node-exporter` on all the hosts, then you can specify the following in the `yaml` file.

  Example

  ```
  service_type: node-exporter
  placement:
  host_pattern: '*'
  extra_entrypoint_args:
      - "--collector.textfile.directory=/var/lib/node_exporter/textfile_collector2"
  ```

Deploying the Ceph daemons using the command line interface

Using the Ceph Orchestrator, you can deploy the daemons such as Ceph Manager, Ceph Monitors, Ceph OSDs, monitoring stack, and others using the `ceph orch` command. Placement specification is passed as `--placement` argument with the Orchestrator commands.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Hosts are added to the storage cluster.

Procedures

1. Log into the `cephadm` shell:

   Example

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

2. Use one of the following methods to deploy the daemons on the hosts:

   - **Method 1**: Specify the number of daemons and the host names:

     Syntax

     ```
     ceph orch apply SERVICE_NAME --placement="NUMBER_OF_DAEMONS HOST_NAME_1
     HOST_NAME_2 HOST_NAME_3"
     ```

     Example

     ```
     [ceph: root@host01 /]# ceph orch apply mon --placement="3 host01 host02 host03"
     ```

   - **Method 2**: Add the labels to the hosts and then deploy the daemons using the labels:

- Add the labels to the hosts:

  Syntax

  ```
  ceph orch host label add HOSTNAME_1 LABEL
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph orch host label add host01 mon
  ```

- Deploy the daemons with labels:

  Syntax

  ```
  ceph orch apply DAEMON_NAME label:LABEL
  ```

  Example

  ```
  ceph orch apply mon label:mon
  ```

- **Method 3**: Add the labels to the hosts and deploy using the `--placement` argument:
  - Add the labels to the hosts:

    Syntax

    ```
    ceph orch host label add HOSTNAME_1 LABEL
    ```

    Example

    ```
    [ceph: root@host01 /]# ceph orch host label add host01 mon
    ```

  - Deploy the daemons using the label placement specification:

    Syntax

    ```
    ceph orch apply DAEMON_NAME --placement="label:LABEL"
    ```

    Example

    ```
    ceph orch apply mon --placement="label:mon"
    ```

Verification

- List the service:

  Example

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

- List the hosts, daemons, and processes:

  Syntax

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ceph orch ps --service_name=SERVICE_NAME
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=mon
  [ceph: root@host01 /]# ceph orch ps --service_name=mon
  ```

Reference

- Adding hosts

Deploying the Ceph daemons on a subset of hosts using the command line interface

You can use the `--placement` option to deploy daemons on a subset of hosts. You can specify the number of daemons in the placement specification with the name of the hosts to deploy the daemons.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Hosts are added to the cluster.

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. List the hosts on which you want to deploy the Ceph daemons:

   Example

   ```
   [ceph: root@host01 /]# ceph orch host ls
   ```

3. Deploy the daemons:

   Syntax

   ```
   ceph orch apply SERVICE_NAME --placement="NUMBER_OF_DAEMONS HOST_NAME_1 _HOST_NAME_2
   HOST_NAME_3"
   ```

   Example

   ```
   ceph orch apply mgr --placement="2 host01 host02 host03"
   ```

   In this example, the `mgr` daemons are deployed only on two hosts.

Verification

- List the hosts:

  Example

  ```
  [ceph: root@host01 /]# ceph orch host ls
  ```

Reference

- *Listing hosts*

Service specification of the Ceph Orchestrator

A service specification is a data structure to specify the service attributes and configuration settings that is used to deploy the Ceph service.

The following is an example of the multi-document YAML file, `cluster.yaml`, for specifying service specifications:

Example

```
service_type: mon
placement:
  host_pattern: "mon*"
---
service_type: mgr
```

```
placement:
  host_pattern: "mgr*"
---
service_type: osd
service_id: default_drive_group
placement:
  host_pattern: "osd*"
data_devices:
  all: true
```

The following list are the parameters where the properties of a service specification are defined as follows:

- `service_type`: The type of service:
  - Ceph services like mon, crash, mds, mgr, osd, rbd, or rbd-mirror.
  - Ceph gateway like rgw.
  - Monitoring stack like Alertmanager, Prometheus, Grafana or Node-exporter.
  - Container for custom containers.
- `service_id`: A unique name of the service.
- `placement`: This is used to define where and how to deploy the daemons.
- `unmanaged`: If set to `true`, the Orchestrator will neither deploy nor remove any daemon associated with this service.

Stateless service of Orchestrators

A stateless service is a service that does not need information of the state to be available. For example, to start an `rgw` service, additional information is not needed to start or run the service. The `rgw` service does not create information about this state in order to provide the functionality. Regardless of when the `rgw` service starts, the state is the same.

Disabling automatic management of daemons

You can mark the `cephadm` services as `managed` or `unmanaged` without having to edit and re-apply the service specification.

PREREQUISITE

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

1. Set `unmanaged` to `True` for service mon by using this command:

   ```
   [root@host01 ~]# ceph orch set-unmanaged mon
   ```

2. Set `unmanaged` to `False` for service mon by using this command:

   ```
   [root@host01 ~]# ceph orch set-managed mon
   ```

3. Set service `osd.all-available-devices` as unmanaged:

   ```
   [root@host01 ~]# ceph orch set-managed osd.all-available-devices
   ```

Deploying the Ceph daemons using the service specification

Using the Ceph Orchestrator, you can deploy daemons such as Ceph Manager, Ceph Monitors, Ceph OSDs, monitoring stack, and others using the service specification in a YAML file.

Prerequisites

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

Procedure

1. Create the `yaml` file:

   Example

   ```
   [root@host01 ~]# touch mon.yaml
   ```

2. This file can be configured in two different ways:

   - Edit the file to include the host details in placement specification:

     Syntax

     ```
     service_type: SERVICE_NAME
     placement:
       hosts:
         - HOST_NAME_1
         - HOST_NAME_2
     ```

     Example

     ```
     service_type: mon
     placement:
       hosts:
         - host01
         - host02
         - host03
     ```

   - Edit the file to include the label details in placement specification:

     Syntax

     ```
     service_type: SERVICE_NAME
     placement:
       label: "LABEL_1"
     ```

     Example

     ```
     service_type: mon
     placement:
       label: "mon"
     ```

3. Optional: Use extra container arguments in the service specification files such as CPUs, CA certificates, and other files while deploying services. Use extra arguments to enable additional metrics in node-exporter deployed by `cephadm`.

   Example

   ```
   extra_container_args:
        - "-v"
        - "/etc/pki/ca-trust/extracted:/etc/pki/ca-trust/extracted:ro"
        - "--security-opt"
        - "label=disable"
        - "cpus=2"
        - "--collector.textfile.directory=/var/lib/node_exporter/textfile_collector2"
   ```

4. Mount the YAML file under a directory in the container:

   Example

   ```
   [root@host01 ~]# cephadm shell --mount mon.yaml:/var/lib/ceph/mon/mon.yaml
   ```

5. Navigate to the directory:

   Example

   ```
   [ceph: root@host01 /]# cd /var/lib/ceph/mon/
   ```

6. Deploy the Ceph daemons using service specification:

Syntax

```
ceph orch apply -i FILE_NAME.yaml
```

Example

```
[ceph: root@host01 mon]# ceph orch apply -i mon.yaml
```

Verification

- List the service:

  Example

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

- List the hosts, daemons, and processes:

  Syntax

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=mon
  ```

Reference

- *Listing hosts*

Managing hosts

As a storage administrator, you can use the Ceph Orchestrator with `cephadm` in the backend to add, list, and remove hosts in an existing Red Hat Ceph Storage cluster.

You can also add labels to hosts. Labels are free-form and have no specific meanings. Each host can have multiple labels. For example, apply the `mon` label to all hosts that have monitor daemons deployed, `mgr` for all hosts with manager daemons deployed, `rgw` for Ceph Object Gateways, and so on.

Labeling all the hosts in the storage cluster helps to simplify system management tasks by allowing you to quickly identify the daemons running on each host. In addition, you can use the Ceph Orchestrator or a YAML file to deploy or remove daemons on hosts that have specific host labels.

Adding hosts

You can use the Ceph Orchestrator with `cephadm` in the backend to add hosts to an existing Red Hat Ceph Storage.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to all nodes in the storage cluster.
- Register the nodes to the CDN and attach subscriptions.
- Ansible user with sudo and passwordless `ssh` access to all nodes in the storage cluster.
- The IP addresses of the new hosts should be updated in the `/etc/hosts` file.

Procedure

1. From the Ceph administration node, log into the `cephadm` shell.
   For example,

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

2. Extract the cluster's public SSH keys to a folder.

```
ceph cephadm get-pub-key > ~/PATH
```

For example,

```
[ceph: root@host01 /]# ceph cephadm get-pub-key > ~/ceph.pub
```

3. Copy Ceph cluster's public SSH keys to the root user's `authorized_keys` file on the new host.

```
ssh-copy-id -f -i ~/PATH root@HOST_NAME_2
```

For example,

```
[ceph: root@host01 /]# ssh-copy-id -f -i ~/ceph.pub root@host02
```

4. From the Ansible administration node, add the new host to the Ansible inventory file. The default location for the file is `/usr/share/cephadm-ansible/hosts`. The following example shows the structure of a typical inventory file:

```
host01
host02
host03

[admin]
host00
```

> **Note:** If you have previously added the new host to the Ansible inventory file and run the preflight playbook on the host, skip to step 6.

5. Run the preflight playbook with the `--limit` option:

```
ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --extra-vars
"ceph_origin=rhcs" --limit NEWHOST
```

For example,

```
[ansible@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --
extra-vars "ceph_origin=rhcs"
 --limit host02
```

The preflight playbook installs `podman`, `lvm2`, `chrony`, and `cephadm` on the new host. After installation is complete, `cephadm` resides in the `/usr/sbin/` directory.

6. From the Ceph administration node, log into the `cephadm` shell.
For example,

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

7. Use the `cephadm` orchestrator to add hosts to the storage cluster.

```
ceph orch host add HOST_NAME IP_ADDRESS_OF_HOST [--label=LABEL_NAME_1,LABEL_NAME_2]
```

For example,

```
[ceph: root@host01 /]# ceph orch host add host02 10.10.128.70 --labels=mon,mgr
```

Verification

Verify that the hosts were added successfully, by running the **`ceph orch host ls`** command.

For example,

```
[ceph: root@host01 /]# ceph orch host ls
```

Reference

For more information, see the following:

- Listing hosts
- “Running the preflight playbook” on page 0
- “Registering the Red Hat Ceph Storage nodes” on page 0
- “Creating an Ansible user with sudo access” on page 0

Adding multiple hosts

You can use the Ceph Orchestrator to add multiple hosts to a Red Hat Ceph Storage cluster at the same time using the service specification in YAML file format.

Prerequisites

- A running Red Hat Ceph Storage.

Procedure

1. Create the `hosts.yaml` file:

   Example

   ```
   [root@host01 ~]# touch hosts.yaml
   ```

2. Edit the `hosts.yaml` file to include the following details:

   Example

   ```
   service_type: host
   addr: host01
   hostname: host01
   labels:
   - mon
   - osd
   - mgr
   ---
   service_type: host
   addr: host02
   hostname: host02
   labels:
   - mon
   - osd
   - mgr
   ---
   service_type: host
   addr: host03
   hostname: host03
   labels:
   - mon
   - osd
   ```

3. Mount the YAML file under a directory in the container:

   Example

   ```
   [root@host01 ~]# cephadm shell --mount hosts.yaml:/var/lib/ceph/hosts.yaml
   ```

4. Navigate to the directory:

   Example

```
[ceph: root@host01 /]# cd /var/lib/ceph/
```

5. Deploy the hosts using service specification:

   Syntax

   ```
   ceph orch apply -i FILE_NAME.yaml
   ```

   Example

   ```
   [ceph: root@host01 hosts]# ceph orch apply -i hosts.yaml
   ```

Verification

- List the hosts:

  Example

  ```
  [ceph: root@host01 /]# ceph orch host ls
  ```

Reference

- *Listing hosts*.

Listing hosts

> **Note:** The *STATUS* of the hosts is blank, in the output of the `ceph orch host ls` command.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Hosts are added to the cluster.

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. List the hosts of the cluster:

   Example

   ```
   [ceph: root@host01 /]# ceph orch host ls
   ```

   You will see that the *STATUS* of the hosts is blank which is expected.

Adding labels to a host

Use the Ceph Orchestrator to add a label to a host. Labels can be used to specify placement of daemons.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A storage cluster is installed and bootstrapped.
- Root-level access to all nodes in the storage cluster.

- Hosts are added to the storage cluster.

A few examples of labels are `mgr`, `mon`, and `osd` based on the service deployed on the hosts. Each host can have multiple labels.

The following labels have special meaning to `cephadm`.

> **Note:** Special labels begin with _ (underscore).

`_no_schedule`

This label prevents `cephadm` from scheduling or deploying daemons on the host. If it is added to an existing host that already contains Ceph daemons, it causes `cephadm` to move those daemons elsewhere, except OSDs which are not removed automatically. When a host is added with the `_no_schedule` label, no daemons are deployed on it. When the daemons are drained before the host is removed, the `_no_schedule` label is set on that host.

`_no_autotune_memory`

This label does not autotune memory on the host. It prevents the daemon memory from being tuned even when the **`osd_memory_target_autotune`** option or other similar options are enabled for one or more daemons on that host.

`_admin`

By default, the `_admin` label is applied to the bootstrapped host in the storage cluster and the `client.admin` key is set to be distributed to that host with the `ceph orch client-keyring [ls|set|rm]` function. Adding this label to additional hosts normally causes `cephadm` to deploy configuration and keyring files in the `/etc/ceph` directory.

1. Launch the `cephadm` shell.
   For example,

   ```
   [root@host01 ~]# cephadm shell
   [ceph: root@host01 /]#
   ```

2. Add a label to a host.

   ```
   ceph orch host label add HOSTNAME LABEL
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph orch host label add host02 mon
   ```

   > **Note:** If your Red Hat Ceph Storage deployment comprises multiple CRUSH bucket types, such as `chassis` or `rack`, also known as failure domains, that are higher than `host`, assign the `mon` label to no more than 2 hosts per failure domain. Limiting the `mon` label spreads the daemons across failure domains, ensuring Monitor quorum when entire failure domain is lost.

AFTER COMPLETING THE TASK

Verify that the label is added to the host, by using the **`ceph orch host ls`** command.

Removing labels from a host

Use the Ceph Orchestrator to remove a label from a host.

PREREQUISITE

- A Red Hat Ceph Storage cluster is installed and bootstrapped.
- Root-level access to all nodes in the storage cluster.
- Hosts are added to the storage cluster.

1. Log into the `cephadm` shell.

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

2. Remove the label.

   ```
   ceph orch host label rm HOST_NAME LABEL_NAME
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph orch host label rm host02 mon
   ```

AFTER COMPLETING THE TASK

Verify that the label is moved from the host, by using the **`ceph orch host ls`** command.

Removing hosts

Remove Ceph cluster hosts with the Ceph Orchestrators.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Root-level access to all the nodes.
- Hosts are added to the storage cluster.
- All the services are deployed.
- `cephadm` is deployed on the nodes where the services have to be removed.

All daemons are removed with the **`drain`** option which adds the `_no_schedule` label to ensure that you cannot deploy any daemons or a cluster till the operation is complete.

> **Important:** If you are removing the bootstrap host, be sure to copy the admin keyring and the configuration file to another host in the storage cluster before you remove the host.

For more information, see the following operations information:

- Adding hosts using the Ceph Orchestrator
- Listing hosts using the Ceph Orchestrator

1. Log into the `cephadm` shell.

   ```
   cephadm shell
   ```

   For example,

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

2. Get the host details.

   ```
   ceph orch host ls
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph orch host ls
   ```

3. Drain all the daemons from the host.

   ```
   ceph orch host drain HOSTNAME
   ```

> **Note:** The ceph orch host drain command supports the **`--zap-osd-devices`** flag. When you set this flag, `cephadm` zaps the devices of the OSDs it removes during the drain process.

For example,

```
[ceph: root@host01 /]# ceph orch host drain host02
```

The `_no_schedule` label is automatically applied to the host which blocks deployment.

4. Check the status of OSD removal.

```
ceph orch osd rm status
```

For example,

```
[ceph: root@host01 /]# ceph orch osd rm status
```

When no placement groups (PGs) remain on the OSD, the OSD is decommissioned and removed from the storage cluster.

5. Check if all the daemons are removed from the storage cluster.

```
ceph orch ps HOSTNAME
```

For example,

```
[ceph: root@host01 /]# ceph orch ps host02
```

6. Remove the host from the orchestrator and the CRUSH map.
Use the **`--rm-crush-entry`** flag to remove the host from both the orchestrator and the CRUSH map in a single operation. This ensures the host is completely removed from the cluster and does not appear in the `ceph osd tree` output.

```
ceph orch host rm HOSTNAME --rm-crush-entry
```

For example,

```
[ceph: root@host01 /]# ceph orch host rm host02 --rm-crush-entry
Removed host 'host02'
```

Alternative method: If you have already removed the host without the **`--rm-crush-entry`** flag, you can manually remove the host from the CRUSH map.

```
ceph osd crush remove HOSTNAME
```

For example,

```
[ceph: root@host01 /]# ceph osd crush remove host02
removed item id -5 name 'host02' from crush map
```

7. Verify that the host has been removed from both the orchestrator and the CRUSH map.
   a. Check that the host is not listed in the orchestrator.

```
ceph orch host ls
```

For example,

```
[ceph: root@host01 /]# ceph orch host ls
```

   b. Verify that the host does not appear in the OSD tree.

```
ceph osd tree
```

For example,

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

The host should not appear in either output.

Placing hosts in the maintenance mode

You can use the Ceph Orchestrator to place the hosts in and out of the maintenance mode.

The `ceph orch host maintenance enter` command stops the `systemd target` which causes all the Ceph daemons to stop on the host. Similarly, the `ceph orch host maintenance exit` command restarts the `systemd target` and the Ceph daemons restart on their own.

The orchestrator adopts the following workflow when the host is placed in maintenance:

1. Confirms the removal of hosts does not impact data availability by running the `orch host ok-to-stop` command.
2. If the host has Ceph OSD daemons, it applies `noout` to the host subtree to prevent data migration from triggering during the planned maintenance slot.
3. Stops the Ceph target, thereby, stopping all the daemons.
4. Disables the `ceph target` on the host, to prevent a reboot from automatically starting Ceph services.

Exiting maintenance reverses this workflow sequence.

Prerequisites

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

Procedure

1. Log into the `cephadm` shell.

   Example

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

2. You can either place the host in maintenance mode or place it out of the maintenance mode:
   - Place the host in maintenance mode.

     Syntax

     ```
     ceph orch host maintenance enter HOST_NAME [--force]
     ```

     Example

     ```
     [ceph: root@host01 /]# ceph orch host maintenance enter host02 --force
     ```

     The `--force` flag allows the user to bypass warnings, but not alerts.

   - Place the host out of the maintenance mode.

     Syntax

     ```
     ceph orch host maintenance exit HOST_NAME
     ```

     Example

     ```
     [ceph: root@host01 /]# ceph orch host maintenance exit host02
     ```

Verification

- List the hosts.

  Example

  ```
  [ceph: root@host01 /]# ceph orch host ls
  ```

Configure SSH hardening for cephadm

Configure SSH hardening to control how cephadm connects to hosts and runs commands.

PREREQUISITE

Ensure that the cluster is deployed and hosts are added. Configure a non-root user with passwordless sudo access on each host.

SSH hardening improves security by restricting how cephadm runs commands on hosts. When enabled, cephadm does not run commands directly. Instead, it uses the `cephadm_invoker.py` script.

The invoker script verifies the cephadm binary before execution and allows only approved operations. This helps prevent unauthorized command execution.

SSH hardening restricts sudo access for non-root users. When a host is prepared for SSH hardening, the sudoers configuration is updated to limit the commands that can be run with sudo. This helps prevent unauthorized command execution while still allowing required cephadm operations.

The sudoers configuration allows passwordless access only to the `cephadm_invoker.py`, providing a controlled and secure execution environment.

- **Enable SSH hardening for the cluster.**

  ```
  ceph cephadm prepare-host-and-enable-ssh-hardening --ssh-user <user>
  ```

  SSH hardening is enabled for the cluster. cephadm now runs commands through the invoker script and uses a non-root user with restricted sudo access.

  Additional actions performed:

  - Installs or updates the `cephadm` package with the invoker script.
  - Configures restricted sudo access for the specified user.
  - Copies the cluster SSH key to hosts.
  - Enables SSH hardening for the cluster.

  How the workflow changes:

  - Add the cluster SSH key to the user's `authorized_keys` file before adding hosts.
  - Ensure that the user has passwordless sudo access. The configuration updates the `/etc/sudoers.d/<username>` file with the following entry:

    ```
    <username> ALL=(root) NOPASSWD: /usr/libexec/cephadm_invoker.py
    ```

  - cephadm runs commands through the invoker script instead of direct execution.
  - The invoker validates the cephadm binary before running commands.

  Note: SSH hardening restricts cephadm operations to the `cephadm_invoker.py` script. However, it does not remove or validate existing sudo permissions that are defined in `/etc/sudoers`, other sudoers files, or group memberships. It updates only the `/etc/sudoers.d/<username>` file. Review and remove any additional sudo permissions if stricter access control is required.

- **Manually prepare a host.**

  ```
  cephadm prepare-host-ssh-hardening --ssh-user <user> --ssh-pub-key <key>
  ```

- **Disable SSH hardening.**

  ```
  ceph config set mgr mgr/cephadm/sudo_hardening false
  ```

  Disabling SSH hardening does not remove the configuration changes on hosts. The `/etc/sudoers.d/<username>` file and its restrictions remain in place. You must review and update the sudoers configuration manually if required.

Managing monitors

As a storage administrator, you can deploy additional monitors using placement specification, add monitors using service specification, add monitors to a subnet configuration, and add monitors to specific hosts. Apart from this, you can remove the monitors.

By default, a typical Red Hat Ceph Storage has three or five monitor daemons deployed on different hosts.

Red Hat recommends deploying five monitors if there are five or more nodes in a cluster.

Ceph deploys monitor daemons automatically as the cluster grows, and scales back monitor daemons automatically as the cluster shrinks. The smooth execution of this automatic growing and shrinking depends upon proper subnet configuration.

If your monitor nodes or your entire cluster are located on a single subnet, then `cephadm` automatically adds up to five monitor daemons as you add new hosts to the cluster. `cephadm` automatically configures the monitor daemons on the new hosts. The new hosts reside on the same subnet as the bootstrapped host in the storage cluster.

`cephadm` can also deploy and scale monitors to correspond to changes in the size of the storage cluster.

Ceph Monitors

Ceph Monitors are lightweight processes that maintain a master copy of the storage cluster map. All Ceph clients contact a Ceph monitor and retrieve the current copy of the storage cluster map, enabling clients to bind to a pool and read and write data.

Ceph Monitors use a variation of the Paxos protocol to establish consensus about maps and other critical information across the storage cluster. Due to the nature of Paxos, Ceph requires a majority of monitors running to establish a quorum, thus establishing consensus.

> **Important:** Red Hat requires at least three monitors on separate hosts to receive support for a production cluster.

Red Hat recommends deploying an odd number of monitors. An odd number of Ceph Monitors has a higher resilience to failures than an even number of monitors. For example, to maintain a quorum on a two-monitor deployment, Ceph cannot tolerate any failures; with three monitors, one failure; with four monitors, one failure; with five monitors, two failures. This is why an odd number is advisable. Summarizing, Ceph needs a majority of monitors to be running and to be able to communicate with each other, two out of three, three out of four, and so on.

For an initial deployment of a multi-node Ceph storage cluster, Red Hat requires three monitors, increasing the number two at a time if a valid need for more than three monitors exists.

Since Ceph Monitors are lightweight, it is possible to run them on the same host as OpenStack nodes. However, Red Hat recommends running monitors on separate hosts.

> **Important:** Red Hat ONLY supports collocating Ceph services in containerized environments.

When you remove monitors from a storage cluster, consider that Ceph Monitors use the Paxos protocol to establish a consensus about the master storage cluster map. You must have a sufficient number of Ceph Monitors to establish a quorum.

Reference

- *Supported configurations* Knowledgebase article

Configuring monitor election strategy

The monitor election strategy identifies the net splits and handles failures. You can configure the election monitor strategy in three different modes: classic, disallow, and connectivity.

The monitor election strategy identifies the net splits and handles failures. You can configure the election monitor strategy in three different modes:

1. `classic` - This is the default mode in which the lowest ranked monitor is voted based on the elector module between the two sites.
2. `disallow` - This mode lets you mark monitors as disallowed, in which case they will participate in the quorum and serve clients, but cannot be an elected leader. This lets you add monitors to a list of disallowed leaders. If a monitor is in the disallowed list, it will always defer to another monitor.
3. `connectivity` - This mode is mainly used to resolve network discrepancies. It evaluates connection scores, based on pings that check liveness, provided by each monitor for its peers and elects the most connected and reliable monitor to be the leader. This mode is designed to handle net splits, which may happen if your cluster is stretched across multiple data centers or otherwise susceptible. This mode incorporates connection score ratings and elects the monitor with the best score. If a specific monitor is desired to be the leader, configure the election strategy so that the specific monitor is the first monitor in the list with a rank of `0`.

Red Hat recommends you to stay in the `classic` mode unless you require features in the other modes.

Before constructing the cluster, change the `election_strategy` to `classic`, `disallow`, or `connectivity` in the following command:

Syntax

```
ceph mon set election_strategy {classic|disallow|connectivity}
```

Deploying the Ceph monitor daemons using the command line interface

The Ceph Orchestrator deploys one monitor daemon by default. You can deploy additional monitor daemons by using the `placement` specification in the command line interface. To deploy a different number of monitor daemons, specify a different number. If you do not specify the hosts where the monitor daemons should be deployed, the Ceph Orchestrator randomly selects the hosts and deploys the monitor daemons to them.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Hosts are added to the cluster.

Procedure

1. Log into the `cephadm` shell.

   Example

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

2. There are four different ways of deploying Ceph monitor daemons:

Method 1

- Use placement specification to deploy monitors on hosts:

> **Note:** Red Hat recommends that you use the `--placement` option to deploy on specific hosts.

Syntax

```
ceph orch apply mon --placement="HOST_NAME_1 HOST_NAME_2 HOST_NAME_3"
```

Example

```
[ceph: root@host01 /]# ceph orch apply mon --placement="host01 host02 host03"
```

Note: Be sure to include the bootstrap node as the first node in the command.

Important:

- Do not add the monitors individually as `ceph orch apply mon` supersedes and will not add the monitors to all the hosts. For example, if you run the following commands, then the first command creates a monitor on `host01`. Then the second command supersedes the monitor on host1 and creates a monitor on `host02`. Then the third command supersedes the monitor on `host02` and creates a monitor on `host03`. Eventually, there is a monitor only on the third host.

  ```
  # ceph orch apply mon host01
  # ceph orch apply mon host02
  # ceph orch apply mon host03
  ```

Method 2

- Use placement specification to deploy specific number of monitors on specific hosts with labels:
  - Add the labels to the hosts:

    Syntax

    ```
    ceph orch host label add HOSTNAME_1 LABEL
    ```

    Example

    ```
    [ceph: root@host01 /]# ceph orch host label add host01 mon
    ```

  - Deploy the daemons.

    Syntax

    ```
    ceph orch apply mon --placement="HOST_NAME_1:mon HOST_NAME_2:mon HOST_NAME_3:mon"
    ```

    Example

    ```
    [ceph: root@host01 /]# ceph orch apply mon --placement="host01:mon host02:mon
    host03:mon"
    ```

Method 3

- Use placement specification to deploy specific number of monitors on specific hosts:

  Syntax

  ```
  ceph orch apply mon --placement="NUMBER_OF_DAEMONS HOST_NAME_1 HOST_NAME_2 HOST_NAME_3"
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph orch apply mon --placement="3 host01 host02 host03"
  ```

Method 4

- Deploy monitor daemons randomly on the hosts in the storage cluster:

  Syntax

  ```
  ceph orch apply mon NUMBER_OF_DAEMONS
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph orch apply mon 3
  ```

Verification

- List the service.

  Example

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

- List the hosts, daemons, and processes.

  Syntax

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=mon
  ```

Deploying the Ceph monitor daemons using the service specification

The Ceph Orchestrator deploys one monitor daemon by default. You can deploy additional monitor daemons by using the service specification, like a YAML format file.

Prerequisites

- A running Red Hat Ceph Storage .
- Hosts are added to the cluster.

Procedure

1. Create the `mon.yaml` file:

   Example

   ```
   [root@host01 ~]# touch mon.yaml
   ```

2. Edit the `mon.yaml` file to include the following details:

   Syntax

   ```
   service_type: mon
   placement:
     hosts:
       - HOST_NAME_1
       - HOST_NAME_2
   ```

   Example

   ```
   service_type: mon
   placement:
     hosts:
       - host01
       - host02
   ```

3. Mount the YAML file under a directory in the container:

Example

```
[root@host01 ~]# cephadm shell --mount mon.yaml:/var/lib/ceph/mon/mon.yaml
```

4. Navigate to the directory:

   Example

   ```
   [ceph: root@host01 /]# cd /var/lib/ceph/mon/
   ```

5. Deploy the monitor daemons:

   Syntax

   ```
   ceph orch apply -i FILE_NAME.yaml
   ```

   Example

   ```
   [ceph: root@host01 mon]# ceph orch apply -i mon.yaml
   ```

Verification

- List the service:

  Example

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

- List the hosts, daemons, and processes:

  Syntax

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=mon
  ```

Deploying the monitor daemons on specific network

The Ceph Orchestrator deploys one monitor daemon by default. You can explicitly specify the IP address or CIDR network for each monitor and control where each monitor is placed.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Hosts are added to the cluster.

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. Disable automated monitor deployment:

   Example

   ```
   [ceph: root@host01 /]# ceph orch apply mon --unmanaged
   ```

3. Deploy monitors on hosts on specific network:

Syntax

```
ceph orch daemon add mon _HOST_NAME_1_:_IP_OR_NETWORK_
```

Example

```
[ceph: root@host01 /]# ceph orch daemon add mon host03:10.1.2.123
```

Verification

- List the service:

  Example

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

- List the hosts, daemons, and processes:

  Syntax

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=mon
  ```

Managing the Ceph Monitor service with Red Hat OpenStack Platform

Understand how replacing Controller nodes affects the Ceph Monitor service in director-deployed Red Hat Ceph Storage environments.

In director-deployed Red Hat Ceph Storage environments, it is important to understand the impact of replacing Controller nodes. The Ceph Monitor service runs on the Controller nodes and typically uses IP addresses assigned from the Storage network.

The Ceph Monitor service IP addresses are associated with virtual machine instances that use Red Hat Ceph Storage. These IP addresses are not dynamically updated if a Ceph Monitor service IP address changes during replacement of a Controller node.

As a result, replacing Controller nodes can lead to a storage outage, particularly if multiple Controller nodes are replaced. In this situation, each affected virtual machine instance must be migrated, rebooted, or shelved and unshelved to resolve the IP address change and restore storage access.

To avoid storage outages during Controller node replacement, reuse the IP addresses of the removed Ceph Monitor service instances instead of assigning new IP addresses.

For example,

```
- name: Controller
  count: 3
  defaults:
    network_config:
      template: /home/stack/templates/nic-config/myController.j2
      default_route_network:
      - external
  instances:
  - hostname: overcloud-controller-0
    name: node00
    networks:
    - network: ctlplane
      vif: true
    - network: external
      subnet: external_subnet
    - network: internal_api
      subnet: internal_api_subnet01
      fixed_ip: 172.21.11.100
    - network: storage
      subnet: storage_subnet01
    - network: storage_mgmt
      subnet: storage_mgmt_subnet01
```

```
  - network: tenant
    subnet: tenant_subnet01
```

Note: To find the current Ceph Monitor service IP addresses on a Controller node, run the following command:

```
sudo cephadm shell -- ceph mon stat
```

For more information, see the Red Hat OpenStack Platform documentation.

Removing the monitor daemons

To remove the monitor daemons from the host, you can just redeploy the monitor daemons on other hosts.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Hosts are added to the cluster.
- At least one monitor daemon deployed on the hosts.

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. Run the `ceph orch apply` command to deploy the required monitor daemons:

   Syntax

   ```
   ceph orch apply mon “NUMBER_OF_DAEMONS HOST_NAME_1 HOST_NAME_3”
   ```

   If you want to remove monitor daemons from `host02`, then you can redeploy the monitors on other hosts.

   Example

   ```
   [ceph: root@host01 /]# ceph orch apply mon “2 host01 host03”
   ```

Verification

- List the hosts,daemons, and processes:

  Syntax

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=mon
  ```

Reference

- *Deploying the Ceph monitor daemons using the command line interface*
- *Deploying the Ceph monitor daemons using the service specification*

Removing a Ceph Monitor from an unhealthy storage cluster

You can remove a `ceph-mon` daemon from an unhealthy storage cluster. An unhealthy storage cluster is one that has placement groups persistently in not `active + clean` state.

Prerequisites

- Root-level access to the Ceph Monitor node.
- At least one running Ceph Monitor node.

Procedure

1. Identify a surviving monitor and log into the host:

   Syntax

   ```
   ssh root@MONITOR_ID
   ```

   Example

   ```
   [root@admin ~]# ssh root@host00
   ```

2. Log in to each Ceph Monitor host and stop all the Ceph Monitors:

   Syntax

   ```
   cephadm unit --name DAEMON_NAME.HOSTNAME stop
   ```

   Example

   ```
   [root@host00 ~]# cephadm unit --name mon.host00 stop
   ```

3. Set up the environment suitable for extended daemon maintenance and to run the daemon interactively:

   Syntax

   ```
   cephadm shell --name DAEMON_NAME.HOSTNAME
   ```

   Example

   ```
   [root@host00 ~]# cephadm shell --name mon.host00
   ```

4. Extract a copy of the `monmap` file:

   Syntax

   ```
   ceph-mon -i HOSTNAME --extract-monmap TEMP_PATH
   ```

   Example

   ```
   [ceph: root@host00 /]# ceph-mon -i host01 --extract-monmap /tmp/monmap
   2022-01-05T11:13:24.440+0000 7f7603bd1700 -1 wrote monmap to /tmp/monmap
   ```

5. Remove the non-surviving Ceph Monitor(s):

   Syntax

   ```
   monmaptool TEMPORARY_PATH --rm HOSTNAME
   ```

   Example

   ```
   [ceph: root@host00 /]# monmaptool /tmp/monmap --rm host01
   ```

6. Inject the surviving monitor map with the removed monitor(s) into the surviving Ceph Monitor:

   Syntax

   ```
   ceph-mon -i HOSTNAME --inject-monmap TEMP_PATH
   ```

   Example

   ```
   [ceph: root@host00 /]# ceph-mon -i host00 --inject-monmap /tmp/monmap
   ```

7. Start only the surviving monitors:

   Syntax

   ```
   cephadm unit --name DAEMON_NAME.HOSTNAME start
   ```

   Example

   ```
   [root@host00 ~]# cephadm unit --name mon.host00 start
   ```

8. Verify the monitors form a quorum:

   Example

   ```
   [ceph: root@host00 /]# ceph -s
   ```

9. Optional: Archive the removed Ceph Monitor's data directory in `/var/lib/ceph/CLUSTER_FSID/mon.HOSTNAME` directory.

Managing manager daemons

As a storage administrator, you can use the Ceph Orchestrator to deploy additional manager daemons. `cephadm` automatically installs a manager daemon on the bootstrap node during the bootstrapping process.

In general, you should set up a Ceph Manager on each of the hosts running the Ceph Monitor daemon to achieve same level of availability.

By default, whichever `ceph-mgr` instance comes up first is made active by the Ceph Monitors, and others are standby managers. There is no requirement that there should be a quorum among the `ceph-mgr` daemons.

If the active daemon fails to send a beacon to the monitors for more than the `mon mgr beacon grace`, then it is replaced by a standby.

If you want to pre-empt failover, you can explicitly mark a `ceph-mgr` daemon as failed with **`ceph mgr fail MANAGER_NAME`** command.

Prerequisites

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

Deploying the manager daemons

The Ceph Orchestrator deploys two Manager daemons by default. You can deploy additional manager daemons using the `placement` specification in the command line interface. To deploy a different number of Manager daemons, specify a different number. If you do not specify the hosts where the Manager daemons should be deployed, the Ceph Orchestrator randomly selects the hosts and deploys the Manager daemons to them.

Prerequisites

> **Note:** Ensure your deployment has at least three Ceph Managers in each deployment.

- A running Red Hat Ceph Storage cluster.
- Hosts are added to the cluster.

Procedure

1. Log into the `cephadm` shell.

   Example

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

2. You can deploy manager daemons in two different ways:

Method 1

- Deploy manager daemons using placement specification on specific set of hosts:

  > **Note:** Use the `--placement` option to deploy on specific hosts.

  Syntax

  ```
  ceph orch apply mgr --placement="HOST_NAME_1 HOST_NAME_2 HOST_NAME_3"
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph orch apply mgr --placement="host01 host02 host03"
  ```

Method 2

- Deploy manager daemons randomly on the hosts in the storage cluster.

  Syntax

  ```
  ceph orch apply mgr NUMBER_OF_DAEMONS
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph orch apply mgr 3
  ```

Verification

- List the service.

  Example

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

- List the hosts, daemons, and processes.

  Syntax

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=mgr
  ```

Removing the manager daemons

To remove the manager daemons from the host, you can just redeploy the daemons on other hosts.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to all the nodes.
- Hosts are added to the cluster.
- At least one manager daemon deployed on the hosts.

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. Run the `ceph orch apply` command to redeploy the required manager daemons:

   Syntax

   ```
   ceph orch apply mgr "NUMBER_OF_DAEMONS HOST_NAME_1 HOST_NAME_3"
   ```

   If you want to remove manager daemons from `host02`, then you can redeploy the manager daemons on other hosts.

   Example

   ```
   [ceph: root@host01 /]# ceph orch apply mgr "2 host01 host03"
   ```

Verification

- List the hosts,daemons, and processes:

  Syntax

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=mgr
  ```

Reference

- *Deploying the manager daemons*

Using Ceph Manager modules

Use the **`ceph mgr module ls`** command to see the available modules and the modules that are presently enabled.

Enable or disable modules with **`ceph mgr module enable MODULE`** command or **`ceph mgr module disable MODULE`** command respectively.

If a module is enabled, then the active `ceph-mgr` daemon loads and executes it. In the case of modules that provide a service, such as an HTTP server, the module might publish its address when it is loaded. To see the addresses of such modules, run the `ceph mgr services` command.

Some modules might also implement a special standby mode which runs on standby `ceph-mgr` daemon as well as the active daemon. This enables modules that provide services to redirect their clients to the active daemon, if the client tries to connect to a standby.

Following is an example to enable the dashboard module:

```
[ceph: root@host01 /]# ceph mgr module enable dashboard

[ceph: root@host01 /]# ceph mgr module ls

        MODULE
        balancer              on (always on)
        crash                 on (always on)
        devicehealth          on (always on)
        orchestrator          on (always on)
        pg_autoscaler         on (always on)
        progress              on (always on)
        rbd_support           on (always on)
        status                on (always on)
        telemetry             on (always on)
        volumes               on (always on)
        cephadm               on
        dashboard             on
        iostat                on
        prometheus            on
        restful               on
        alerts                -
        diskprediction_local  -
        influx                -
        insights              -
        k8sevents             -
        localpool             -
        mds_autoscaler        -
        mirroring             -
        osd_perf_query        -
        osd_support           -
        rgw                   -
        rook                  -
        selftest              -
        snap_schedule         -
        stats                 -
        telegraf              -
        test_orchestrator     -
        zabbix                -

[ceph: root@host01 /]# ceph mgr services
{
        "dashboard": "http://myserver.com:7789/",
        "restful": "https://myserver.com:8789/"
}
```

The first time the cluster starts, it uses the `mgr_initial_modules` setting to override which modules to enable. However, this setting is ignored through the rest of the lifetime of the cluster: only use it for bootstrapping. For example, before starting your monitor daemons for the first time, you might add a section like this to your `ceph.conf` file:

```
[mon]
    mgr initial modules = dashboard balancer
```

Where a module implements comment line hooks, the commands are accessible as ordinary Ceph commands and Ceph automatically incorporates module commands into the standard CLI interface and route them appropriately to the module:

```
[ceph: root@host01 /]# ceph <command | help>
```

Use the configuration parameters listed in “Configuration parameters” on page 221 with the **ceph** command.

Configuration parameters

Configuration	Description	Type	Default
`mgr module path`	Path to load modules from.	String	`"<library dir>/mgr"`
`mgr data`	Path to load daemon data (such as keyring)	String	`"/var/lib/ceph/mgr/$cluster-$id"`

Configuration	Description	Type	Default
`mgr tick period`	How many seconds between manager beacons to monitors, and other periodic checks.	Integer	`5`
`mon mgr beacon grace`	How long after last beacon should a manager be considered failed.	Integer	`30`

Using the Ceph Manager balancer module

The balancer is a module for Ceph Manager (`ceph-mgr`) that optimizes the placement of placement groups (PGs) across OSDs in order to achieve a balanced distribution, either automatically or in a supervised fashion.

> **Note:** The balancer module cannot be disabled but can be turned off to customize the configuration.

The following are the supported balancer modes:

crush-compat

The CRUSH compat mode uses the compat `weight-set` feature to manage an alternative set of weights for devices in the CRUSH hierarchy.
The normal weights should remain set to the size of the device to reflect the target amount of data that you want to store on the device. The balancer then optimizes the `weight-set` values, adjusting them up or down in small increments in order to achieve a distribution that matches the target distribution as closely as possible. Because PG placement is a pseudorandom process, there is a natural amount of variation in the placement; by optimizing the weights, the balancer counter-acts that natural variation.

This mode is fully backwards compatible with older clients. When an OSDMap and CRUSH map are shared with older clients, the balancer presents the optimized weights as the real weights.

The primary restriction of this mode is that the balancer cannot handle multiple CRUSH hierarchies with different placement rules if the subtrees of the hierarchy share any OSDs. Because this configuration makes managing space utilization on the shared OSDs difficult, it is generally not recommended. As such, this restriction is normally not an issue.

upmap

Use `upmap` to optimize PG placement and achieve uniform distribution across OSDs. In most cases, this distribution is nearly uniform, with each OSD hosting an equal number of PGs (+/-1 PG). In most cases, this distribution is "perfect", with an equal number of PGs on each OSD +/-1 PG, as they might not divide evenly.

1. Run the following command to check the PG distribution by device class:

   ```
   ceph osd df deviceclass
   ```

   If the `VAR` values for `hdd` or `ssd` are outside the range `[0.95 – 1.05]`, the balancer may be disabled or misconfigured.
   If any OSD shows a `REWEIGHT` value other than `1.000`, reset the OSD.

   ```
   ceph osd reweight osd.ID 1.000
   ```

2. Improve balancer effectiveness by ensuring that pools have enough PGs. If PGS per OSD is less than 100, add PGs to the pools served by that device class.
   When PG autoscaler is enabled, apply the following settings:

   ```
   ceph config set global max_pg_per_osd 500
   ceph config set global mon_target_pg_per_osd 300
   ceph config set mgr/balancer/upmap_max_deviation 1
   ```

> **Note:** Using **`ceph config set mgr/balancer/upmap_max_deviation 1`** is useful when OSD sizes vary significantly.

3. Overlapping CRUSH roots can prevent proper balancing.
   Verify the pool rules by using the following command:

   ```
   ceph osd pool ls detail
   ```

4. After verifying the pool rules, inspect the CRUSH rule.

   ```
   ceph osd crush rule dump | jq 'map(select(.rule_id == RULE_ID) | .steps)'
   ```

   Each pool should use a rule that specifies a device class, for example, `default~ssd`. If any pool uses a rule without a device class, contact Red Hat Support for guidance.

read

The OSDMap can store explicit mappings for individual primary OSDs as exceptions to the normal CRUSH placement calculation. These **`pg-upmap-primary`** entries provide fine-grained control over primary PG mappings. This mode optimizes the placement of individual primary PGs in order to achieve balanced reads, or primary PGs, in a cluster. In `read` mode, upmap behavior is not exercised, so this mode is best for uses cases in which only read balancing is desired.

> **Important:** To allow use of this feature, you must tell the cluster that it only needs to support rRef or later clients with the following command:
>
> ```
> ceph osd set-require-min-compat-client reef
> ```
>
> This command fails if any pre-Reef clients or daemons are connected to the monitors. To work around this issue, use the `--yes-i-really-mean-it` flag:
>
> ```
> ceph osd set-require-min-compat-client reef --yes-i-really-mean-it
> ```
>
> You can check what client versions are in use with: the **`ceph features`** command. For example,
>
> ```
> [ceph: root@host01 /]# ceph features
> ```

upmap-read

This balancer mode combines optimization benefits of both `upmap` and `read` mode. Like in `read` mode, `upmap-read` makes use of **`pg-upmap-primary`**.
Use `upmap-read` for achieving the `upmap` mode's offering of balanced PG distribution as well as the `read` mode's offering of balanced reads.

> **Important:** To allow use of this feature, you must tell the cluster that it only needs to support Reef or later clients with the following command:
>
> ```
> ceph osd set-require-min-compat-client reef
> ```
>
> This command fails if any pre-Reef clients or daemons are connected to the monitors. To work around this issue, use the `--yes-i-really-mean-it` flag:
>
> ```
> ceph osd set-require-min-compat-client reef --yes-i-really-mean-it
> ```
>
> You can check what client versions are in use with: the **`ceph features`** command. For example,
>
> ```
> [ceph: root@host01 /]# ceph features
> ```

Balancing Red Hat Ceph Storage cluster using the capacity balancer

Balance the cluster by using the capacity balancer module.

PREREQUISITE

Before you begin, make sure that you have a running Red Hat Ceph Storage cluster.

1. Check if the balancer module is enabled.

```
ceph mgr module enable balancer
```

2. Turn on the balancer module.

```
ceph balancer on
```

3. To change the mode use the following command. The default mode is `upmap`.

```
ceph balancer mode upmap
```

4. Check the current status of the balancer.

```
ceph balancer status
```

Automatic balancing

Use automatic balancing when turning on the balancer module.

Turn on the capacity balancer, by using the **`ceph balancer on`** command. By default, when turning on the balancer module, automatic balancing is used and the `upmap` mode is enabled.

```
[ceph: root@host01 /]# ceph balancer on
```

You can change the balancer mode from `upmap` to `crush-compact` mode. The `crush-compat` mode is backward compatible with older clients and makes small changes to the data distribution to ensure that OSDs are equally utilized.

After the balancer is on, you can verify the balancer status.

```
[ceph: root@host01 /]# ceph balancer on
[ceph: root@host01 /]# ceph balancer mode crush-compat
[ceph: root@host01 /]# ceph balancer status
{
    "active": true,
    "last_optimize_duration": "0:00:00.001174",
    "last_optimize_started": "Fri Nov 22 11:09:18 2024",
    "mode": "crush-compact",
    "no_optimization_needed": false,
    "optimize_result": "Unable to find further optimization, change balancer mode and retry
might help",
    "plans": []
}
```

To turn off the balancer module, use the **`ceph balancer off`** command. For example,

```
[ceph: root@host01 /]# ceph balancer off
[ceph: root@host01 /]#  ceph balancer status
{
    "active": false,
    "last_optimize_duration": "",
    "last_optimize_started": "",
    "mode": "crush-compat",
    "no_optimization_needed": false,
    "optimize_result": "",
    "plans": []
}
```

Throttling

Adjust the balancer with throttling.

When the cluster is healthy, the balancer throttles its changes such that the percentage of placement groups (PGs) that are misplaced, or need to be moved, is below a threshold of 5% by default. No adjustments are made to the PG distribution if the cluster is degraded. For example, if an OSD has failed and the system has not yet healed itself.

Adjust the balancer percentage by using the **target_max_misplaced_ratio** parameter.

```
ceph config-key set mgr target_max_misplaced_ratio THRESHOLD_PERCENTAGE
```

The following example increases the threshold to 7%.

```
[ceph: root@host01 /]# ceph config-key set mgr target_max_misplaced_ratio .07
```

Supervised optimization

Break down the balancer operations in a phased manner.

The balancer operation is broken into a few distinct phases.

1. Build a `plan`.
2. Evaluate the quality of the data distribution, either for the current PG distribution, or the PG distribution that might result after running a `plan`.
3. Run the `plan`.
    a. Evaluate and score the current distribution.

    ```
    ceph balancer eval
    ```

    b. Evaluate the distribution for a single pool.

    ```
    ceph balancer eval POOL_NAME
    ```

    For example,

    ```
    [ceph: root@host01 /]# ceph balancer eval rbd
    ```

    c. See a greater detail for the evaluation.

    ```
    ceph balancer eval-verbose ...
    ```

    d. Generate a plan using the currently configured mode.

    ```
    ceph balancer optimize PLAN_NAME
    ```

    Replace *PLAN_NAME* with a custom plan name.

    For example,

    ```
    [ceph: root@host01 /]# ceph balancer optimize rbd_123
    ```

    e. See the contents of a plan.

    ```
    ceph balancer show PLAN_NAME
    ```

    For example,

    ```
    [ceph: root@host01 /]# ceph balancer show rbd_123
    ```

    f. Discard old plans.

    ```
    ceph balancer rm PLAN_NAME
    ```

    For example,

```
[ceph: root@host01 /]# ceph balancer rm rbd_123
```

g. See currently recorded plans use the status command.

```
ceph balancer status
```

h. Calculate the quality of the distribution that would result after executing a plan.

```
ceph balancer eval PLAN_NAME
```

For example,

```
[ceph: root@host01 /]# ceph balancer eval rbd_123
```

i. Run the plan.

```
ceph balancer execute PLAN_NAME
```

For example,

```
[ceph: root@host01 /]# ceph balancer execute rbd_123
```

> **Note:** Only run the plan if it is expected to improve the distribution. After execution, the plan is discarded.

Balancing a Ceph cluster using read balancer

Balance primary placement groups (PGs) in a cluster. The balancer can optimize the allocation of placement groups across OSDs to achieve a balanced distribution. The balancer can operate either automatically (online), offline, or in a supervised fashion (offline).

Online optimization

Balance primary PGs in a cluster by using the balancer module. Using the balancer module, all offline optimization steps are completed automatically.

PREREQUISITE

Before you begin, make sure that you have a running Red Hat Ceph Storage cluster.

1. Enable the balancer module.

```
ceph mgr module enable balancer
```

For example,

```
[ceph: root@host01 /]# ceph mgr module enable balancer
```

2. Turn on the balancer module.

```
ceph balancer on
```

For example,

```
[ceph: root@host01 /]# ceph balancer on
```

3. Update the **`min_compat_client setting`** to ensure compatibility. Read balancing requires that no clients older than Red Hat Ceph Storage 7 connect to the cluster. Older clients are not supported and will fail to connect if read balancing is enabled.
To use online optimization, the support for Reef or later clients must be indicated on the cluster.

```
ceph osd set-require-min-compat-client reef
```

Note: This command fails if any pre-Reef clients or daemons are connected to the monitors. To work around this issue, use the `--yes-i-really-mean-it` flag:

```
ceph osd set-require-min-compat-client reef --yes-i-really-mean-it
```

You can check what client versions are in use with: the **`ceph features`** command. For example,

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

4. Change the mode to **`upmap-read`** or **`read`** for read balancing.
   The default mode is **upmap**, enable these modes by running one of the following commands:
   - `ceph balancer mode upmap-read`
   - `ceph balancer mode read`
5. Check the current status of the balancer.

   ```
   ceph balancer status
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph balancer status
   {
   "active": true,
   "last_optimize_duration": "0:00:00.013640",
   "last_optimize_started": "Mon Nov 22 14:47:57 2024",
   "mode": "upmap-read",
   "no_optimization_needed": true,
   "optimize_result": "Unable to find further optimization, or pool(s) pg_num is
   decreasing, or distribution is already perfect",
   "plans": []
   }
   ```

Offline optimization (Technology Preview)

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Before running the offline read balancer, run the capacity balancer to balance PG placement across OSDs. This will ensure optimal results. Execute the following steps:
  - Get the latest copy of your `osdmap`.

    ```
    [ceph: root@host01 /]# ceph osd  getmap -o map
    ```

  - Run the `upmap` balancer.

    ```
    ospmaptool map–upmap balance.sh
    ```

  - The file `balance.sh` contains the proposed solution.

    The commands in this procedure are normal Ceph CLI commands that are run to apply the changes to the cluster.

    Run the following command if there are any recommendations in the `balance.sh` file.

    ```
    [ceph: root@host01 /]# source balance.sh
    ```

For information, see "Balancing Red Hat Ceph Storage cluster using the capacity balancer" on page 224

Important: Technology Preview features are not supported with production service level agreements (SLAs), might not be functionally complete, and does not recommend using them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

If you have unbalanced primary OSDs, you can update them with an offline optimizer that is built into the `osdmaptool`. Run the capacity balancer before running the read balancer to ensure optimal results.

1. Check the `read_balance_score`, available for each pool.

   ```
   ceph osd pool ls detail
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph osd pool ls detail
   ```

   Note:
   You can use `json` formatting to get more details.

   If the `read_balance_score` is considerably above `1`, your pool has unbalanced primary OSDs.

   For a homogenous cluster the optimal score is [Ceil{(number of PGs/Number of OSDs)}/(number of PGs/Number of OSDs)]/[(number of PGs/Number of OSDs)/(number of PGs/Number of OSDs)].

   For example, if you have a pool with 32 PG and 10 OSDs then (number of PGs/Number of OSDs) = 32/10 = 3.2. So, the optimal score if all the devices are identical is the ceiling value of 3.2 divided by (number of PGs/Number of OSDs) that is 4/3.2 = 1.25. If you have another pool in the same system with 64 PGs the optimal score is 7/6.4 =1.09375).

   The following is an example output:

   ```
   $ ceph osd pool ls detail
   pool 1 '.mgr' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num 1
   pgp_num 1 autoscale_mode on last_change 17 flags hashpspool stripe_width 0 pg_num_max
   32 pg_num_min 1 application mgr read_balance_score 3.00
   pool 2 'cephfs.a.meta' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins
   pg_num 16 pgp_num 16 autoscale_mode on last_change 55 lfor 0/0/25 flags hashpspool
   stripe_width 0 pg_autoscale_bias 4 pg_num_min 16 recovery_priority 5 application
   cephfs read_balance_score 1.50
   pool 3 'cephfs.a.data' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins
   pg_num 128 pgp_num 128 autoscale_mode on last_change 27 lfor 0/0/25 flags
   hashpspool,bulk stripe_width 0 application cephfs read_balance_score 1.31
   ```

2. Get the latest copy of your `osdmap`.

   ```
   ceph osd getmap -o om
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph osd getmap -o om
   got osdmap epoch 56
   ```

3. Run the optimizer.
   The `balance.sh` file contains the proposed solution.

   ```
   osdmaptool om --read balance.sh --read-pool POOL_NAME [--vstart]
   ```

   For example,

```
[ceph: root@host01 /]# sdmaptool om --read balance.sh --read-pool cephfs.a.meta
./bin/osdmaptool: osdmap file 'om'
writing upmap command output to: balance.sh
---------- BEFORE ------------
 osd.0 | primary affinity: 1 | number of prims: 4
 osd.1 | primary affinity: 1 | number of prims: 8
 osd.2 | primary affinity: 1 | number of prims: 4

read_balance_score of 'cephfs.a.meta': 1.5

---------- AFTER ------------
 osd.0 | primary affinity: 1 | number of prims: 5
 osd.1 | primary affinity: 1 | number of prims: 6
 osd.2 | primary affinity: 1 | number of prims: 5

read_balance_score of 'cephfs.a.meta': 1.13


num changes: 2
```

4. Apply the changes to the cluster. The `balance.sh` file contains the proposed solution.
The commands in this step are normal Ceph CLI commands that are run to apply the changes to the cluster.

```
source balance.sh
```

For example,

```
[ceph: root@host01 /]# source balance.sh
```

The following shows an example of offline balancer output and execution.

```
$ cat balance.sh
ceph osd pg-upmap-primary 2.3 0
ceph osd pg-upmap-primary 2.4 2

$ source balance.sh
change primary for pg 2.3 to osd.0
change primary for pg 2.4 to osd.2
```

> **Note:** If you are running the command **`ceph osd pg-upmap-primary`** for the first time, you might get a warning as: `Error EPERM: min_compat_client luminous < reef, which is required for pg-upmap-primary.`. In this case, run the **`ceph osd set-require-min-compat-client reef`** command to adjust your cluster's min-compat-client.

AFTER COMPLETING THE TASK

Consider rechecking the scores and re-running the balancer if the number of PGs change or if any OSDs are added or removed from the cluster as these operations can considerably impact the read balancer effect on a pool.

Supervised optimization

The read balancer can also be used with supervised optimization. If using supervised optimization, use the information detailed in “Supervised optimization” on page 225 . Set the mode to either `upmap-read` or `read`.

Using the Ceph Manager alerts module

You can use the Ceph Manager alerts module to send simple alert messages about the Red Hat Ceph Storage cluster’s health by email.

> **Note:** This module is not intended to be a robust monitoring solution. The fact that it is run as part of the Ceph cluster itself is fundamentally limiting in that a failure of the `ceph-mgr` daemon prevents alerts from being sent. This module can, however, be useful for standalone clusters that exist in environments where existing monitoring infrastructure does not exist.

Prerequisites

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

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. Enable the alerts module:

   Example

   ```
   [ceph: root@host01 /]# ceph mgr module enable alerts
   ```

3. Ensure the alerts module is enabled:

   Example

   ```
   [ceph: root@host01 /]# ceph mgr module ls | more
   {
       "always_on_modules": [
           "balancer",
           "crash",
           "devicehealth",
           "orchestrator",
           "pg_autoscaler",
           "progress",
           "rbd_support",
           "status",
           "telemetry",
           "volumes"
       ],
       "enabled_modules": [
           "alerts",
           "cephadm",
           "dashboard",
           "iostat",
           "prometheus",
           "restful"
       ]
   ```

4. Configure the Simple Mail Transfer Protocol (SMTP):

   Syntax

   ```
   ceph config set mgr mgr/alerts/smtp_host SMTP_SERVER
   ceph config set mgr mgr/alerts/smtp_destination RECEIVER_EMAIL_ADDRESS
   ceph config set mgr mgr/alerts/smtp_sender SENDER_EMAIL_ADDRESS
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph config set mgr mgr/alerts/smtp_host smtp.example.com
   [ceph: root@host01 /]# ceph config set mgr mgr/alerts/smtp_destination
   example@example.com
   [ceph: root@host01 /]# ceph config set mgr mgr/alerts/smtp_sender example2@example.com
   ```

5. Optional: By default, the alerts module uses SSL and port 465.

   Syntax

   ```
   ceph config set mgr mgr/alerts/smtp_port PORT_NUMBER
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph config set mgr mgr/alerts/smtp_port 587
   ```

Do not set the `smtp_ssl` parameter while configuring alerts.

6. Authenticate to the SMTP server:

   Syntax

   ```
   ceph config set mgr mgr/alerts/smtp_user USERNAME
   ceph config set mgr mgr/alerts/smtp_password PASSWORD
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph config set mgr mgr/alerts/smtp_user admin1234
   [ceph: root@host01 /]# ceph config set mgr mgr/alerts/smtp_password admin1234
   ```

7. Optional: By default, SMTP `From` name is `Ceph`. To change that, set the `smtp_from_name` parameter:

   Syntax

   ```
   ceph config set mgr mgr/alerts/smtp_from_name CLUSTER_NAME
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph config set mgr mgr/alerts/smtp_from_name 'Ceph Cluster
   Test'
   ```

8. Optional: By default, the alerts module checks the storage cluster's health every minute, and sends a message when there is a change in the cluster health status. To change the frequency, set the `interval` parameter:

   Syntax

   ```
   ceph config set mgr mgr/alerts/interval INTERVAL
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph config set mgr mgr/alerts/interval "5m"
   ```

   In this example, the interval is set to 5 minutes.

9. Optional: Send an alert immediately:

   Example

   ```
   [ceph: root@host01 /]# ceph alerts send
   ```

Reference

- *Health messages of a Ceph cluster*

Using the Ceph manager crash module

By default, daemon crashdumps are dumped in `/var/lib/ceph/crash`. You can configure it with the option `crash dir`. Crash directories are named by time, date, and a randomly-generated UUID, and contain a metadata file `meta` and a recent log file, with a `crash_id` that is the same.

You can use `ceph-crash.service` to submit these crashes automatically and persist in the Ceph Monitors. The `ceph-crash.service` watches the `crashdump` directory and uploads them with `ceph crash post`.

The *RECENT_CRASH* heath message is one of the most common health messages in a Ceph cluster. This health message means that one or more Ceph daemons has crashed recently, and the crash has not yet been archived or acknowledged by the administrator. This might indicate a software bug, a hardware problem like a failing disk, or some other problem. The option `mgr/crash/warn_recent_interval` controls the time period of what recent means, which is two weeks by default. You can disable the warnings by running the following command:

Example

```
[ceph: root@host01 /]# ceph config set mgr/crash/warn_recent_interval 0
```

The option `mgr/crash/retain_interval` controls the period for which you want to retain the crash reports before they are automatically purged. The default for this option is one year.

Prerequisites

- A running Red Hat Ceph Storage cluster.

Procedure

1. Ensure the crash module is enabled:

   Example

   ```
   [ceph: root@host01 /]# ceph mgr module ls | more
   {
       "always_on_modules": [
           "balancer",
           "crash",
           "devicehealth",
           "orchestrator_cli",
           "progress",
           "rbd_support",
           "status",
           "volumes"
       ],
       "enabled_modules": [
           "dashboard",
           "pg_autoscaler",
           "prometheus"
       ]
   ```

2. Save a crash dump: The metadata file is a JSON blob stored in the crash dir as `meta`. You can invoke the ceph command `-i -` option, which reads from stdin.

   Example

   ```
   [ceph: root@host01 /]# ceph crash post -i meta
   ```

3. List the timestamp or the UUID crash IDs for all the new and archived crash info:

   Example

   ```
   [ceph: root@host01 /]# ceph crash ls
   ```

4. List the timestamp or the UUID crash IDs for all the new crash information:

   Example

   ```
   [ceph: root@host01 /]# ceph crash ls-new
   ```

5. List the timestamp or the UUID crash IDs for all the new crash information:

   Example

   ```
   [ceph: root@host01 /]# ceph crash ls-new
   ```

6. List the summary of saved crash information grouped by age:

   Example

   ```
   [ceph: root@host01 /]# ceph crash stat
   8 crashes recorded
   8 older than 1 days old:
   2022-05-20T08:30:14.533316Z_4ea88673-8db6-4959-a8c6-0eea22d305c2
   2022-05-20T08:30:14.590789Z_30a8bb92-2147-4e0f-a58b-a12c2c73d4f5
   2022-05-20T08:34:42.278648Z_6a91a778-bce6-4ef3-a3fb-84c4276c8297
   2022-05-20T08:34:42.801268Z_e5f25c74-c381-46b1-bee3-63d891f9fc2d
   2022-05-20T08:34:42.803141Z_96adfc59-be3a-4a38-9981-e71ad3d55e47
   ```

```
2022-05-20T08:34:42.830416Z_e45ed474-550c-44b3-b9bb-283e3f4cc1fe
2022-05-24T19:58:42.549073Z_b2382865-ea89-4be2-b46f-9a59af7b7a2d
2022-05-24T19:58:44.315282Z_1847afbc-f8a9-45da-94e8-5aef0738954e
```

7. View the details of the saved crash:

   Syntax

   ```
   ceph crash info CRASH_ID
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph crash info 2022-05-24T19:58:42.549073Z_b2382865-ea89-4be2-
   b46f-9a59af7b7a2d
   {
       "assert_condition": "session_map.sessions.empty()",
       "assert_file": "/builddir/build/BUILD/ceph-16.1.0-486-g324d7073/src/mon/
   Monitor.cc",
       "assert_func": "virtual Monitor::~Monitor()",
       "assert_line": 287,
       "assert_msg": "/builddir/build/BUILD/ceph-16.1.0-486-g324d7073/src/mon/Monitor.cc:
   In function 'virtual Monitor::~Monitor()' thread 7f67a1aeb700 time
   2022-05-24T19:58:42.545485+0000\n/builddir/build/BUILD/ceph-16.1.0-486-g324d7073/src/
   mon/Monitor.cc: 287: FAILED ceph_assert(session_map.sessions.empty())\n",
       "assert_thread_name": "ceph-mon",
       "backtrace": [
           "/lib64/libpthread.so.0(+0x12b30) [0x7f679678bb30]",
           "gsignal()",
           "abort()",
           "(ceph::__ceph_assert_fail(char const*, char const*, int, char const*)+0x1a9)
   [0x7f6798c8d37b]",
           "/usr/lib64/ceph/libceph-common.so.2(+0x276544) [0x7f6798c8d544]",
           "(Monitor::~Monitor()+0xe30) [0x561152ed3c80]",
           "(Monitor::~Monitor()+0xd) [0x561152ed3cdd]",
           "main()",
           "__libc_start_main()",
           "_start()"
       ],
       "ceph_version": "16.2.8-65.el8cp",
       "crash_id": "2022-07-06T19:58:42.549073Z_b2382865-ea89-4be2-b46f-9a59af7b7a2d",
       "entity_name": "mon.ceph-adm4",
       "os_id": "rhel",
       "os_name": "Red Hat Enterprise Linux",
       "os_version": "8.5 (Ootpa)",
       "os_version_id": "8.5",
       "process_name": "ceph-mon",
       "stack_sig": "957c21d558d0cba4cee9e8aaf9227b3b1b09738b8a4d2c9f4dc26d9233b0d511",
       "timestamp": "2022-07-06T19:58:42.549073Z",
       "utsname_hostname": "host02",
       "utsname_machine": "x86_64",
       "utsname_release": "4.18.0-240.15.1.el8_3.x86_64",
       "utsname_sysname": "Linux",
       "utsname_version": "#1 SMP Wed Jul 06 03:12:15 EDT 2022"
   }
   ```

8. Remove saved crashes older than *KEEP* days: Here, *KEEP* must be an integer.

   Syntax

   ```
   ceph crash prune KEEP
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph crash prune 60
   ```

9. Archive a crash report so that it is no longer considered for the `RECENT_CRASH` health check and does not appear in the `crash ls-new` output. It appears in the `crash ls`.

   Syntax

   ```
   ceph crash archive CRASH_ID
   ```

   Example

```
[ceph: root@host01 /]# ceph crash archive 2022-05-24T19:58:42.549073Z_b2382865-
ea89-4be2-b46f-9a59af7b7a2d
```

10. Archive all crash reports:

    Example

    ```
    [ceph: root@host01 /]# ceph crash archive-all
    ```

11. Remove the crash dump:

    Syntax

    ```
    ceph crash rm CRASH_ID
    ```

    Example

    ```
    [ceph: root@host01 /]# ceph crash rm 2022-05-24T19:58:42.549073Z_b2382865-ea89-4be2-
    b46f-9a59af7b7a2d
    ```

Reference

- *Health messages of a Ceph cluster*

Telemetry module

The telemetry module sends data about the storage cluster to help understand how Ceph is used and what problems are encountered during operations. The data is visualized on the public dashboard to view the summary statistics on how many clusters are reporting, their total capacity and OSD count, and version distribution trends.

The telemetry report is broken down into different channels, each with a different type of information. After the telemetry is enabled, you can turn on or turn off the individual channels.

The following are the available channels:

- `basic`
- `crash`
- `device`
- `ident`
- `perf`

`basic`

Default is `on`. This channel provides the basic information about the clusters, which includes the following information:

- The capacity of the cluster.
- The number of monitors, managers, OSDs, MDSs, Ceph Object Gateways, or other daemons.
- The software version that is currently being used.
- The number and types of RADOS pools and Ceph File Systems.
- The names of configuration options that are changed from their default (but not their values).

`crash`

Default is `on`. This channel provides information about the daemon crashes, which includes the following information:

- The type of daemon.
- The version of the daemon.
- The operating system, the OS distribution, and the kernel version.
- The stack trace that identifies where in the Ceph code the crash occurred.

device

Default is on. This channel provides information about the device metrics, which includes anonymized SMART metrics.

ident

Default is off. This channel provides the user-provided identifying information about the cluster such as cluster description, and contact email address.

perf

Default is off. This channel provides the various performance metrics of the cluster, which can be used for the following:

- Reveal overall cluster health.
- Identify workload patterns.
- Troubleshoot issues with latency, throttling, memory management, and other similar issues.
- Monitor cluster performance by daemon.

The data that is reported does not contain any sensitive data such as pool names, object names, object contents, hostnames, or device serial numbers.

It contains counters and statistics on how the cluster is deployed, Ceph version, host distribution, and other parameters that help the project to gain a better understanding of the way Ceph is used.

Data is secure and is sent to https://telemetry.ceph.com.

Enabling telemetry

Before enabling channels, ensure that the telemetry is on.

Enable telemetry by using the following command:

```
ceph telemetry on
```

Enabling and disabling channels

You can enable and disable either individual channels, multiple channels simultaneously, or all channels.

- Enable an individual channel by using the following command:

  ```
  ceph telemetry enable channel CHANNEL
  ```

  For example,

  ```
  [ceph: root@host01 /]# ceph telemetry enable channel basic
  ```

- Disable an individual channel by using the following command:

  ```
  ceph telemetry disable channel CHANNEL
  ```

  For example,

  ```
  [ceph: root@host01 /]# ceph telemetry disable channel basic
  ```

- Enable multiple individual channels with a single command by using the following command:

  ```
  ceph telemetry enable channel CHANNEL1 CHANNEL2 CHANNEL3
  ```

  For example,

  ```
  [ceph: root@host01 /]# ceph telemetry enable channel basic crash device
  ```

- Disable multiple individual channels with a single command by using the following command:

  ```
  ceph telemetry disable channel CHANNEL1 CHANNEL2 CHANNEL3
  ```

  For example,

```
[ceph: root@host01 /]# ceph telemetry disable channel basic crash device
```

– Enable all channels simultaneously by using the following command:

```
ceph telemetry enable channel all
```

For example,

```
[ceph: root@host01 /]# ceph telemetry enable channel all
```

– Disable all channels simultaneously by using the following command:

```
ceph telemetry disable channel all
```

For example,

```
[ceph: root@host01 /]# ceph telemetry disable channel all
```

Sample reports

Use the following commands to generate and view sample reports.

– Review the data report at any time by generating a sample report.

```
ceph telemetry show
```

▪ If telemetry is `off`, you can preview the sample report.

```
ceph telemetry preview
```

It takes longer to generate a sample report for storage clusters with hundreds of OSDs or more.

– Device reports are generated separately to protect your privacy. Device reports output with data such as the host name and device serial number as anonymous. The device telemetry is sent to a different endpoint and does not associate the device data with a particular cluster.

```
ceph telemetry show-device
```

▪ If telemetry is `off`, you can preview the sample device report.

```
ceph telemetry preview-device
```

– Generate a single output for both the data and device.

```
ceph telemetry show-all
```

▪ If telemetry is `off`, you can preview both reports.

```
ceph telemetry preview-all
```

– Generate a sample report per channel.

```
ceph telemetry show CHANNEL_NAME
```

▪ If telemetry is `off`, you can preview a sample report per channel.

```
ceph telemetry preview CHANNEL_NAME
```

Collections

Collections are different aspects of data that is collected within a channel.

– List the collections.

```
ceph telemetry collection ls
```

– See the difference between the collections that you are enrolled in, and the new, available collections.

```
ceph telemetry diff
```

– Enroll to the most recent collections.

```
ceph telemetry on
ceph telemetry enable channel CHANNEL_NAME
```

interval

The `interval` module compiles and sends a new report every 24 hours by default.

Use the **mgr/telemetry/interval** option to adjust the report interval.

```
ceph config set mgr mgr/telemetry/interval INTERVAL
```

The following example sets the report generation for every three days (72 hours).

```
[ceph: root@host01 /]# ceph config set mgr mgr/telemetry/interval 72
```

status

Use the `status` option to view the current configuration.

```
ceph telemetry status
```

Manually sending telemetry

Send an asynchronous, one-time set of telemetry data, by running the `send` command.

```
ceph telemetry send
```

If telemetry is `off`, include `--license sharing-1-0` to the command.

```
ceph telemetry send --license sharing-1-0
```

Sending telemetry through a proxy

If the cluster cannot connect directly to the configured telemetry endpoint, you can configure a HTTP/HTTPs proxy server.

```
ceph config set mgr mgr/telemetry/proxy PROXY_URL
```

For example,

```
[ceph: root@host01 /]# ceph config set mgr mgr/telemetry/proxy https://10.0.0.1:8080
```

Contact and description

A contact and description can be added to the report. This is optional and is disabled by default.

> **Note:** If `ident` flag is enabled, its details are not displayed in the leaderboard.

```
ceph config set mgr mgr/telemetry/contact 'CONTACT_NAME'
ceph config set mgr mgr/telemetry/description 'DESCRIPTION'
ceph config set mgr mgr/telemetry/channel_ident true
```

For example,

```
[ceph: root@host01 /]# ceph config set mgr mgr/telemetry/contact 'John Doe
<john.doe@example.com>'
[ceph: root@host01 /]# ceph config set mgr mgr/telemetry/description 'My first Ceph
cluster'
[ceph: root@host01 /]# ceph config set mgr mgr/telemetry/channel_ident true
```

Leaderboard

You can participate in a leaderboard on the public Telemetry dashboard.

The leaderboard displays basic information about the storage cluster. This board includes the total storage capacity and the number of OSDs.

```
ceph config set mgr mgr/telemetry/leaderboard true
```

Managing OSDs

As a storage administrator, you can use the Ceph Orchestrators to manage Red Hat Ceph Storage OSDs.

Deploy one OSD per physical storage device, using the entire device. This recommendation applies to all supported device types, including HDDs, SSDs, and NVMe drives.

- Do not deploy multiple OSDs per device for NVMe SSDs.
- This guidance concerns the main OSD block device, not SSDs used for WAL+DB metadata offload from HDD OSDs.

Note: In some cases, deploying multiple OSDs per device may be appropriate. In the following cases, contact Red Hat Support for configuration guidance.

- Ceph Object Gateway workloads with a large number of very small S3 objects or many object versions.
- Very large NVMe SSDs (30 TB or larger).

Check the capacity of a cluster regularly to see whether it is reaching the upper end of its storage capacity. As a storage cluster reaches its `near full` ratio, add one or more OSDs to expand the storage cluster's capacity. If the node has multiple storage drives, you might also need to remove one of the `ceph-osd` daemons for that drive.

Important:

- Ensure that when you remove an OSD, the storage cluster is not at its `near full` ratio.
- Do not allow a storage cluster to reach the `full` ratio before adding an OSD. OSD failures that occur after the storage cluster reaches the `near full` ratio can cause the storage cluster to exceed the `full` ratio. Ceph blocks write access to protect the data until you resolve the storage capacity issues. When removing an OSD, be sure to consider the impact on the `full` ratio first.

Configure Ceph OSDs and their supporting hardware in a similar manner as a storage strategy for any pools that will use the OSDs. Ceph prefers uniform hardware across pools for a consistent performance profile. For best performance, consider a CRUSH hierarchy with drives of the same type or size.

In cases where you add drives of dissimilar size, adjust their weights accordingly. When you add the OSD to the CRUSH map, consider the weight for the new OSD. Hard disk drive capacity grows approximately 40% per year, so newer OSD nodes might have larger hard disk drives than older nodes in the storage cluster, that is, they might have a greater weight.

Automatically tuning OSD memory

The OSD daemons adjust the memory consumption based on the `osd_memory_target` configuration option.

The option `osd_memory_target` sets OSD memory based upon the available RAM in the system.

Syntax

```
ceph config set osd osd_memory_target_autotune true
```

cephadm starts with a fraction mgr/cephadm/autotune_memory_target_ratio, which defaults to 0.7 of the total RAM in the system, subtract off any memory consumed by non-autotuned daemons such as non-OSDS and for OSDs for which osd_memory_target_autotune is false, and then divide by the remaining OSDs.

By default, autotune_memory_target_ratio is 0.2 for hyper-converged infrastructure and 0.7 for other environments.

The osd_memory_target parameter is calculated as follows:

Syntax

```
osd_memory_target = TOTAL_RAM_OF_THE_OSD_NODE (in Bytes) * (autotune_memory_target_ratio) /
NUMBER_OF_OSDS_IN_THE_OSD_NODE - (SPACE_ALLOCATED_FOR_OTHER_DAEMONS (in Bytes))
```

SPACE_ALLOCATED_FOR_OTHER_DAEMONS may optionally include the following daemon space allocations:

- Alertmanager: 1 GB
- Grafana: 1 GB
- Ceph Manager: 4 GB
- Ceph Monitor: 2 GB
- Node-exporter: 1 GB
- Prometheus: 1 GB

For example, if a node has 24 OSDs and has 251 GB RAM space, then osd_memory_target is 7860684936.

The final targets are reflected in the configuration database with options. You can view the limits and the current memory consumed by each daemon from the ceph orch ps output under MEM LIMIT column.

Note:
The default setting of osd_memory_target_autotune true is unsuitable for hyperconverged infrastructures where compute and Ceph storage services are colocated. In a hyperconverged infrastructure, the autotune_memory_target_ratio can be set to 0.2 to reduce the memory consumption of Ceph.

```
[ceph: root@host01 /]# ceph config set mgr mgr/cephadm/autotune_memory_target_ratio
0.2
```

You can manually set a specific memory target for an OSD in the storage cluster.

Example

```
[ceph: root@host01 /]# ceph config set osd.123 osd_memory_target 7860684936
```

You can manually set a specific memory target for an OSD host in the storage cluster.

Syntax

```
ceph config set osd/host:_HOSTNAME_ osd_memory_target _TARGET_BYTES_
```

Example

```
[ceph: root@host01 /]# ceph config set osd/host:host01 osd_memory_target 1000000000
```

Note: Enabling osd_memory_target_autotune overwrites existing manual OSD memory target settings. To prevent daemon memory from being tuned even when the osd_memory_target_autotune option or other similar options are enabled, set the _no_autotune_memory label on the host.

Syntax

```
ceph orch host label add HOSTNAME _no_autotune_memory
```

You can exclude an OSD from memory autotuning by disabling the autotune option and setting a specific memory target.

Example

```
[ceph: root@host01 /]# ceph config set osd.123 osd_memory_target_autotune false
[ceph: root@host01 /]# ceph config set osd.123 osd_memory_target 16G
```

Listing devices for Ceph OSD deployment

You can check the list of available devices before deploying OSDs using the Ceph Orchestrator.

The commands are used to print a list of devices discoverable by `cephadm`. A storage device is considered available if all of the following conditions are met:

- The device must have no partitions.
- The device must not have any LVM state.
- The device must not be mounted.
- The device must not contain a file system.
- The device must not contain a Ceph BlueStore OSD.
- The device must be larger than 5 GB.

> **Note:** Ceph will not provision an OSD on a device that is not available.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Hosts are added to the cluster.
- All manager and monitor daemons are deployed.

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. List the available devices to deploy OSDs:

   Syntax

   ```
   ceph orch device ls [--hostname=HOSTNAME_1 HOSTNAME_2] [--wide] [--refresh]
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch device ls --wide --refresh
   ```

   Using the `--wide` option provides all details relating to the device, including any reasons that the device might not be eligible for use as an OSD. This option does not support NVMe devices.

3. Optional: To enable *Health*, *Ident*, and *Fault* fields in the output of `ceph orch device ls`, run the following commands:

   NOTE: These fields are supported by `libstoragemgmt` library and currently supports SCSI, SAS, and SATA devices.

a. As root user outside the `cephadm` shell, check your hardware's compatibility with `libstoragemgmt` library to avoid unplanned interruption to services:

   Example

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

   In the output, you see the *Health Status* as *Good* with the respective *SCSI VPD 0x83* ID.

   NOTE: If you do not get this information, then enabling the fields might cause erratic behavior of devices.

b. Log back into the `cephadm` shell and enable `libstoragemgmt` support:

   Example

   ```
   [root@host01 ~]# cephadm shell
   [ceph: root@host01 /]# ceph config set mgr mgr/cephadm/device_enhanced_scan true
   ```

   Once this is enabled, `ceph orch device ls` gives the output of *Health* field as *Good*.

Verification

- List the devices:

  Example

  ```
  [ceph: root@host01 /]# ceph orch device ls
  ```

Zapping devices for Ceph OSD deployment

Before deploying OSDs, it is important to check the list of available devices. If there is no space available on the devices, you can clear the data on the devices by zapping them.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Hosts are added to the cluster.
- All manager and monitor daemons are deployed.

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. List the available devices to deploy OSDs:

   Syntax

   ```
   ceph orch device ls [--hostname=HOSTNAME_1 HOSTNAME_2] [--wide] [--refresh]
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch device ls --wide --refresh
   ```

3. Clear the data of a device:

   Syntax

   ```
   ceph orch device zap HOSTNAME FILE_PATH --force
   ```

Example

```
[ceph: root@host01 /]# ceph orch device zap host02 /dev/sdb --force
```

Verification

- Verify the space is available on the device:

  Example

  ```
  [ceph: root@host01 /]# ceph orch device ls
  ```

  You will see that the field under *Available* is *Yes*.

Reference

- *Listing devices for Ceph OSD deployment*

Deploying Ceph OSDs on all available devices

You can deploy all OSDS on all the available devices. `cephadm` allows the Ceph Orchestrator to discover and deploy the OSDs on any available and unused storage device.

To deploy OSDs all available devices, run the command without the **unmanaged** parameter and then re-run the command with the parameter to prevent from creating future OSDs.

> **Note:** The deployment of OSDs with `--all-available-devices` is generally used for smaller clusters. For larger clusters, use the OSD specification file.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Hosts are added to the cluster.
- All manager and monitor daemons are deployed.

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. List the available devices to deploy OSDs:

   Syntax

   ```
   ceph orch device ls [--hostname=HOSTNAME_1 HOSTNAME_2] [--wide] [--refresh]
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch device ls --wide --refresh
   ```

3. Deploy OSDs on all available devices:

   Example

   ```
   [ceph: root@host01 /]# ceph orch apply osd --all-available-devices
   ```

   The effect of `ceph orch apply` is persistent which means that the Orchestrator automatically finds the device, adds it to the cluster, and creates new OSDs. This occurs under the following conditions:

- New disks or drives are added to the system.
- Existing disks or drives are zapped.
- An OSD is removed and the devices are zapped.

  You can disable automatic creation of OSDs on all the available devices by using the `--unmanaged` parameter.

  Example

  ```
  [ceph: root@host01 /]# ceph orch apply osd --all-available-devices --
  unmanaged=true
  ```

  Setting the parameter `--unmanaged` to `true` disables the creation of OSDs and also there is no change if you apply a new OSD service.

  > **Note:** The command `ceph orch daemon add` creates new OSDs, but does not add an OSD service.

Verification

- List the service:

  Example

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

- View the details of the node and devices:

  Example

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

Reference

- *Listing devices for Ceph OSD deployment*

Deploying Ceph OSDs on specific devices and hosts

You can deploy all the Ceph OSDs on specific devices and hosts using the Ceph Orchestrator.

For more information, see “Listing devices for Ceph OSD deployment” on page 240.

Before you begin

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Hosts are added to the cluster.
- All manager and monitor daemons are deployed.

Procedure

1. Log into the `cephadm` shell:
   Example

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

2. List the available devices to deploy OSDs:

   ```
   ceph orch device ls [--hostname=HOSTNAME_1 HOSTNAME_2] [--wide] [--refresh]
   ```

Example

```
[ceph: root@host01 /]# ceph orch device ls --wide --refresh
```

3. Deploy OSDs on specific devices and hosts:

```
ceph orch daemon add osd _HOSTNAME_:_DEVICE_PATH_
```

Example

```
[ceph: root@host01 /]# ceph orch daemon add osd host02:/dev/sdb
```

To deploy ODSs on a raw physical device, without an LVM layer, use the `--method raw` option.

```
ceph orch daemon add osd --method raw _HOSTNAME_:_DEVICE_PATH_
```

Example

```
[ceph: root@host01 /]# ceph orch daemon add osd --method raw host02:/dev/sdb
```

> **Note:** If you have separate DB or WAL devices, the ratio of block to DB or WAL devices **MUST** be 1:1.

Verification

- List the service:
  Example

```
[ceph: root@host01 /]# ceph orch ls osd
```

- View the details of the node and devices:
  Example

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

- List the hosts, daemons, and processes:

```
ceph orch ps --service_name=SERVICE_NAME
```

Example

```
[ceph: root@host01 /]# ceph orch ps --service_name=osd
```

Advanced service specifications and filters for deploying OSDs

Use OSD service specifications to describe a cluster layout using the properties of storage devices. Service specifications give the user an abstract way to tell Ceph which storage devices should turn into an OSD with the required configuration without knowing the specifics of device names and paths.

For each device and each host, define a YAML or JSON file. For more deployment information, see “Deploying Ceph OSDs using advanced service specifications” on page 249.

“General settings for OSD specifications” on page 244 provides the supported OSD settings and information.

General settings for OSD specifications		
OSD setting	**Additional options**	**Description**
`service_type:osd`	None	Mandatory to create OSDs.

OSD setting	Additional options	Description
`service_id`	None	Use the service name or identification you prefer. A set of OSDs is created using the specification file. This name is used to manage all the OSDs together and represent an Orchestrator service.
`placement`	**`host_pattern:`** A hostname pattern to limit the hosts to which this service should be applied. Set to * to apply to all hosts. The host pattern is matched by using `fnmatch`, which supports UNIX shell-style wildcards. **`label: OSD_HOST`** Apply only to hosts with the specified ceph `orch host` label. **`hosts`: HOST01, HOST02** An explicit list of hostnames to which the service is to be applied.	This is used to define the hosts on which the OSDs need to be deployed.
`selection of devices`	None	Allows the user to select specific devices for OSD services. Fine-grained selection allows the use of multiple OSD services for multiple classes of OSD devices. BlueStore OSDs have three components, which can be deployed together on one storage device, or across multiple devices. Only create BlueStore OSDs which have three components: **OSD data** Contains all the OSD data. **BlueStore Write-Ahead Log (WAL)** By default this is colocated with the DB device. It is rarely appropriate to specify a different device just for the WAL. **DB** BlueStore internal metadata, including omaps used for Ceph Object Gateway index pools. By default the WAL is colocated on this device. Use `block` for OSD payload data. By default this is the device used for all three components.

OSD setting	Additional options	Description
`data_devices`	None	Define the devices to deploy OSD. In this case, OSDs are created in a collocated schema. You can use filters to select devices and folders.
`wal_devices`	None	Define the devices used for WAL offloading. You can use filters to select devices and folders. If not specified the WAL is colocated on `data_devices`.
`db_devices`	None	Define the devices for DB offload. You can use the filters to select devices and folders. If not specified, the DB is colocated on `data_devices`.
`encrypted`	None	An optional parameter to encrypt information on the OSD. Input options: `True` or `False`.
`unmanaged`	None	An optional parameter, set to `False` by default. You can set it to `True` if you do not want the Orchestrator to manage the OSD service. Input options:`True` or `False`.
`block_wal_size`	None	User-defined value, in bytes.
`block_db_size`	None	User-defined value, in bytes.
`osds_per_device`	None	User-defined value for deploying more than one OSD per device. Most deployments should let this default to `1`. Contact Red Hat Support for additional guidance if your OSD devices are 30 TB or larger or if your workload includes large numbers of very small S3 objects or many object versions.
`method`	None	An optional parameter to specify if an OSD is created with an LVM layer or not. Set to `raw` if you want to create OSDs on raw physical devices that do not include an LVM layer. If you have separate DB or WAL devices, the ratio of block to DB or WAL devices **MUST** be 1:1.

"Filters for specifying devices" on page 247 defines the filters that are used in conjunction with the `data_devices`, `wal_devices`, and `db_devices` settings that are defined in "General settings for OSD specifications" on page 244.

Filters for specifying devices			
Name of the filter	**Description**	**Syntax**	**Example**
Model	Target specific SKUs. You can get details of the model by running one of the following commands: – **`lsblk -o NAME,FSTYPE,LABEL,MOUNTPOINT,SIZE,MODEL`** – **`smartctl -i /DEVICE_PATH`** – **`lsscsi`**	Model: *`STORAGE_MODEL_NAME`*	Model: MC-55-44-XZ
Vendor	Target only storage devices from a specific manufacturer or vendor.	Vendor: *`STORAGE_VENDOR_NAME`*	Vendor: Vendor Cs
Size Specification	Match only storage devices that are an exact specific size. **Note:** Using an exact size is not recommended. Using a size range is usually the better strategy.	size: *`EXACT`*	size: 10G
Size Specification	Match storage devices within a size range. This is the recommended filter. For example, when deploying nominal 20TB HDDs, one might specify `size: 18:22`.	size: *`LOW:HIGH`*	size: 10G:40G
Size Specification	Match storage devices less than or equal to in size.	size: *`:HIGH`*	size: :10G
Size Specification	Match storage devices greater than or equal in size. For example, if your OSDs are currently nominal 20 TB HDDs but in the future you might use 24 TB replacement or expansion drives, you could specify `18T:` so that both the current and future storage devices are matched.	size: *`LOW:`*	size: 40G:

Name of the filter	Description	Syntax	Example
Rotational	Rotational attribute of the storage devices. – `1` matches rotating HDDs – `0` matches SSDs. This matches against the kernel's determination of the device type, which in most cases is accurate. There are certain storage layers that may require a `udev` rule to override the kernel's determination. For more information, contact Red Hat Support .	`rotational`:`0`or`1`	rotational: 0
All	Matches all the available storage devices. **Note:** Storage devices with existing partitions, even if not in active use, are considered `unavailable` by the orchestrator and must be zapped to be considered `available`.	all: true	all: true
Limiter	Optional. This is an optional limit to the number of storage devices that another filter will match. For example, if there are 4 unused SSDs on each server, but one of them is to be reserved for non-Ceph purposes. Most deployments do not benefit from this directive. For more information, contact Red Hat Support.	limit: *NUMBER*	limit: 2

Note:

– To create an OSD with non-collocated components in the same host, specify the different types of devices used and the devices should be on the same host.

- The devices used for deploying OSDs must be supported by `libstoragemgmt`. For more information, see “Listing devices for Ceph OSD deployment” on page 240.

Deploying Ceph OSDs using advanced service specifications

The service specification of type OSD is a way to describe a cluster layout using the properties of disks. It gives the user an abstract way to tell Ceph which disks should turn into an OSD with the required configuration without knowing the specifics of device names and paths.

You can deploy the OSD for each device and each host by defining a YAML or JSON file.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Hosts are added to the cluster.
- All manager and monitor daemons are deployed.

Procedure

1. On the monitor node, create the `osd_spec.yaml` file:
   Example

   ```
   [root@host01 ~]# touch osd_spec.yaml
   ```

2. Edit the `osd_spec.yaml` file to include the following details:

   ```
   service_type: osd
   service_id: SERVICE_ID
   placement:
     host_pattern: * # optional
   data_devices: # optional
     model: DISK_MODEL_NAME # optional
     paths:
     - /DEVICE_PATH
   osds_per_device: NUMBER_OF_DEVICES # optional
   db_devices: # optional
     size: # optional
     all: true # optional
     paths:
      - /DEVICE_PATH
   encrypted: true
   ```

   a. Simple scenarios: In these cases, all the nodes have the same set-up.
      Example

      ```
      service_type: osd
      service_id: osd_spec_default
      placement:
        host_pattern: '*'
      data_devices:
        all: true
        paths:
        - /dev/sdb
      encrypted: true
      ```

      Example

      ```
      service_type: osd
      service_id: osd_spec_default
      placement:
        host_pattern: '*'
      data_devices:
        size: '80G'
      db_devices:
        size: '40G:'
      ```

```
  paths:
   - /dev/sdc
```

b. Simple scenario: In this case, all the nodes have the same setup with OSD devices created in raw mode, without an LVM layer.
Example

```
service_type: osd
service_id: all-available-devices
encrypted: "true"
method: raw
placement:
  host_pattern: "*"
data_devices:
  all: "true"
```

c. Advanced scenario: This would create the desired layout by using all HDDs as `data_devices` with two SSD assigned as dedicated DB or WAL devices. The remaining SSDs are `data_devices` that have the NVMe vendors assigned as dedicated DB or WAL devices.
Example

```
service_type: osd
service_id: osd_spec_hdd
placement:
  host_pattern: '*'
data_devices:
  rotational: 0
db_devices:
  model: Model-name
  limit: 2
---
service_type: osd
service_id: osd_spec_ssd
placement:
  host_pattern: '*'
data_devices:
  model: Model-name
db_devices:
  vendor: Vendor-name
```

d. Advanced scenario with non-uniform nodes: This applies different OSD specs to different hosts depending on the host_pattern key.
Example

```
service_type: osd
service_id: osd_spec_node_one_to_five
placement:
  host_pattern: 'node[1-5]'
data_devices:
  rotational: 1
db_devices:
  rotational: 0
---
service_type: osd
service_id: osd_spec_six_to_ten
placement:
  host_pattern: 'node[6-10]'
data_devices:
  model: Model-name
db_devices:
  model: Model-name
```

e. Advanced scenario with dedicated WAL and DB devices:
Example

```
service_type: osd
service_id: osd_using_paths
placement:
  hosts:
    - host01
    - host02
data_devices:
  paths:
```

```
    - /dev/sdb
db_devices:
  paths:
    - /dev/sdc
wal_devices:
  paths:
    - /dev/sdd
```

f. Advanced scenario with multiple OSDs per device:
Example

```
service_type: osd
service_id: multiple_osds
placement:
  hosts:
    - host01
    - host02
osds_per_device: 4
data_devices:
  paths:
    - /dev/sdb
```

g. For pre-created volumes, edit the `osd_spec.yaml` file to include the following details:

```
service_type: osd
service_id: SERVICE_ID
placement:
  hosts:
    - HOSTNAME
data_devices: # optional
  model: DISK_MODEL_NAME # optional
  paths:
  - /DEVICE_PATH
db_devices: # optional
  size: # optional
  all: true # optional
  paths:
  - /DEVICE_PATH
```

Example

```
service_type: osd
service_id: osd_spec
placement:
  hosts:
    - machine1
data_devices:
  paths:
    - /dev/vg_hdd/lv_hdd
db_devices:
  paths:
    - /dev/vg_nvme/lv_nvmes
```

3. Mount the YAML file under a directory in the container:
Example

```
[root@host01 ~]# cephadm shell --mount osd_spec.yaml:/var/lib/ceph/osd/osd_spec.yaml
```

4. Navigate to the directory:
Example

```
[ceph: root@host01 /]# cd /var/lib/ceph/osd/
```

5. Before deploying OSDs, do a dry run:

> **Note:** This step gives a preview of the deployment, without deploying the daemons.

Example

```
[ceph: root@host01 osd]# ceph orch apply -i osd_spec.yaml --dry-run
```

6. Deploy OSDs using service specification:

```
ceph orch apply -i _FILE_NAME_.yml
```

Example

```
[ceph: root@host01 osd]# ceph orch apply -i osd_spec.yaml
```

Verification

- List the service:
  Example

```
[ceph: root@host01 /]# ceph orch ls osd
```

- View the details of the node and devices:
  Example

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

Reference

- *Advanced service specifications and filters for deploying OSDs*

Removing OSDs

You can remove the OSD from a cluster by using `cephadm`.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage.
- Hosts are added to the cluster.
- Ceph Monitor, Ceph Manager, and Ceph OSD daemons are deployed on the storage cluster.

Removing an OSD from a cluster involves two steps:

1. Evacuates all placement groups (PGs) from the cluster.
2. Removes the PG-free OSDs from the cluster.

The **`--zap`** option removed the volume groups, logical volumes, and the LVM metadata.

1. Log into the `cephadm` shell.
   For example,

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

2. Check the device and the node from which the OSD has to be removed.

```
ceph osd tree
```

For example,

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

3. Remove the OSD.
   You can either remove a specific OSD from the storage cluster or multiple OSDs from a specific node.
   - Remove the OSD from the storage cluster.

> **Important:** Removing the OSD from the storage cluster without an option, such as `--replace`, removes the device completely from the storage cluster completely. If you want to use the same device for deploying OSDs, first zap the device before adding it to the storage cluster.

```
ceph orch osd rm OSD_ID [--replace] [--force] --zap
```

For example,

```
[ceph: root@host01 /]# ceph orch osd rm 0 --zap
```

- Remove multiple OSDs from a specific node.

```
ceph orch osd rm OSD_ID OSD_ID --zap
```

For example,

```
[ceph: root@host01 /]# ceph orch osd rm 2 5 --zap
```

4. Use the **`osd rm status`** command to check the removal status.
For example,

```
[ceph: root@host01 /]# ceph orch osd rm status
OSD  HOST   STATE                    PGS  REPLACE  FORCE  ZAP   DRAIN STARTED AT

9    host01 done, waiting for purge    0  False    False  True  2023-06-06
17:50:50.525690
10   host03 done, waiting for purge    0  False    False  True  2023-06-06
17:49:38.731533
11   host02 done, waiting for purge    0  False    False  True  2023-06-06
17:48:36.641105
```

When no PGs are left on the OSD, it is decommissioned and removed from the cluster.

AFTER COMPLETING THE TASK

Verify the details of the devices and the nodes from which the Ceph OSDs are removed.

```
ceph osd tree
```

Related information

Deploying Ceph OSDs on all available devices
Deploying Ceph OSDs on specific devices and hosts
Zapping devices for Ceph OSD deployment

Replacing the OSDs

When disks fail, you can replace the physical storage device and reuse the same OSD ID to avoid having to reconfigure the CRUSH map.

You can replace the OSDs from the cluster by preserving the OSD ID using the **`ceph orch rm`** command.

> **Note:** If you want to replace a single OSD, Deploying Ceph OSDs on specific devices and hosts . If you want to deploy OSDs on all available devices, Deploying Ceph OSDs on all available devices.

The OSD is not permanently removed from the CRUSH hierarchy, but is assigned the `destroyed` flag. This flag is used to determine the OSD IDs that can be reused in the next OSD deployment. The `destroyed` flag is used to determine which OSD id is reused in the next OSD deployment.

If you use OSD specification for deployment, your newly added disk is assigned the OSD ID of their replaced counterparts.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Hosts are added to the cluster.
- Monitor, Manager, and OSD daemons are deployed on the storage cluster.

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. Check the device and the node from which the OSD has to be replaced:

   Example

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

3. Set the OSD service specification to **unmanaged** state:

   > **Note:** Set the **unmanaged** parameter to `true` in the OSD service specification before removal. If you skip this step, `cephadm` may automatically redeploy OSDs, causing conflicts between orchestrator-driven deployment and OSD removal.

   Example

   ```
   service_type:osd
   service_id:<service_id>
   unmanaged:true
   ```

4. Apply the updated specification:

   ```
   ceph orch apply -i <osds.yaml>
   ```

5. Run the **`orch ls ceph`** command to confirm that the OSD service is set to the unmanaged state.
6. Replace the OSD:

   > **Important:** If the storage cluster has `health_warn` or other errors associated with it, check and try to fix any errors before replacing the OSD to avoid data loss.

   Syntax

   ```
   ceph orch osd rm OSD_ID --replace
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch osd rm 0 --replace
   ```

7. Check the status of the OSD replacement:

   Example

   ```
   [ceph: root@host01 /]# ceph orch osd rm status
   ```

8. Set the OSD service specification back to managed state. After you replace the disk, set the OSD service back to a **managed** state so that `cephadm` can resume orchestration.

a. Update the OSD service specification.

```
unmanaged: false
```

b. Apply the updated specification. This step allows `cephadm` to redeploy the OSD on the new device while reusing the existing OSD ID.

```
ceph orch apply -i <osds.yaml>
```

Verification

- Verify the details of the devices and the nodes from which the Ceph OSDs are replaced:

  Example

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

  You will see an OSD with the same id as the one you replaced running on the same host.

Reference

- *Deploying Ceph OSDs on all available devices*
- *Deploying Ceph OSDs on specific devices and hosts*

Replacing the OSDs with pre-created LVM

After purging the OSD with the **`ceph-volume lvm zap`** command, if the directory is not present, then you can replace the OSDs with the OSD service specification file with the pre-created LVM.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Failed OSD

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. Set the OSD service to an **unmanaged** state to prevent `cephadm` from automatically redeploying OSDs during removal.

   > **Note:** If you skip this step, `cephadm` may redeploy OSDs automatically, causing conflicts with the removal operation.

   Example

   ```
   service_type: osd
   service_id: <service_id>
   unmanaged: true
   ```

3. Apply the specification:

   Example

   ```
   ceph orch apply -i <osds.yaml>
   ```

4. Run the **`orch ls ceph`** command to confirm that the OSD service is set to the unmanaged state.
5. Remove the OSD:

   Syntax

   ```
   ceph orch osd rm OSD_ID [--replace]
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch osd rm 8 --replace
   Scheduled OSD(s) for removal
   ```

6. Verify the OSD is destroyed:

   Example

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

   ID   CLASS  WEIGHT   TYPE NAME        STATUS     REWEIGHT  PRI-AFF
   -1         0.32297  root default
   -9         0.05177      host host10
    3    hdd  0.01520          osd.3           up   1.00000  1.00000
   13    hdd  0.02489          osd.13          up   1.00000  1.00000
   17    hdd  0.01169          osd.17          up   1.00000  1.00000
   -13         0.05177      host host11
    2    hdd  0.01520          osd.2           up   1.00000  1.00000
   15    hdd  0.02489          osd.15          up   1.00000  1.00000
   19    hdd  0.01169          osd.19          up   1.00000  1.00000
   -7         0.05835      host host12
   20    hdd  0.01459          osd.20          up   1.00000  1.00000
   21    hdd  0.01459          osd.21          up   1.00000  1.00000
   22    hdd  0.01459          osd.22          up   1.00000  1.00000
   23    hdd  0.01459          osd.23          up   1.00000  1.00000
   -5         0.03827      host host04
    1    hdd  0.01169          osd.1           up   1.00000  1.00000
    6    hdd  0.01129          osd.6           up   1.00000  1.00000
    7    hdd  0.00749          osd.7           up   1.00000  1.00000
    9    hdd  0.00780          osd.9           up   1.00000  1.00000
   -3         0.03816      host host05
    0    hdd  0.01169          osd.0           up   1.00000  1.00000
    8    hdd  0.01129          osd.8    destroyed         0  1.00000
   12    hdd  0.00749          osd.12          up   1.00000  1.00000
   16    hdd  0.00769          osd.16          up   1.00000  1.00000
   -15         0.04237      host host06
    5    hdd  0.01239          osd.5           up   1.00000  1.00000
   10    hdd  0.01540          osd.10          up   1.00000  1.00000
   11    hdd  0.01459          osd.11          up   1.00000  1.00000
   -11         0.04227      host host07
    4    hdd  0.01239          osd.4           up   1.00000  1.00000
   14    hdd  0.01529          osd.14          up   1.00000  1.00000
   18    hdd  0.01459          osd.18          up   1.00000  1.00000
   ```

7. Zap and remove the OSD using the `ceph-volume` command:

   Syntax

   ```
   ceph-volume lvm zap --osd-id OSD_ID
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph-volume lvm zap --osd-id 8

   Zapping: /dev/vg1/data-lv2
   Closing encrypted path /dev/mapper/l4D6ql-Prji-IzH4-dfhF-xzuf-5ETl-jNRcXC
   Running command: /usr/sbin/cryptsetup remove /dev/mapper/l4D6ql-Prji-IzH4-dfhF-
  xzuf-5ETl-jNRcXC
   Running command: /usr/bin/dd if=/dev/zero of=/dev/vg1/data-lv2 bs=1M count=10
  conv=fsync
   stderr: 10+0 records in
   10+0 records out
   stderr: 10485760 bytes (10 MB, 10 MiB) copied, 0.034742 s, 302 MB/s
   Zapping successful for OSD: 8
   ```

8. Check the OSD topology:

Example

```
[ceph: root@host01 /]# ceph-volume lvm list
```

9. Recreate the OSD with a specification file corresponding to that specific OSD topology:

   Example

```
[ceph: root@host01 /]# cat osd.yml
service_type: osd
service_id: osd_service
placement:
  hosts:
  - host03
data_devices:
  paths:
  - /dev/vg1/data-lv2
db_devices:
  paths:
  - /dev/vg1/db-lv1
```

10. Apply the updated specification file:

    Example

```
ceph orch apply -i osd.yml
```

11. Set the OSD service specification back to managed state. After you replace the disk, set the OSD service back to a **managed** state so that `cephadm` can resume orchestration.

    a. Update the OSD service specification.

```
unmanaged: false
```

    b. Apply the updated specification. This step allows `cephadm` to redeploy the OSD on the new device while reusing the existing OSD ID.

```
ceph orch apply -i <osds.yaml>
```

12. Verify the OSD is back:

    Example

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

Replacing the OSDs in a non-colocated scenario

When the an OSD fails in a non-colocated scenario, you can replace the WAL/DB devices. The procedure is the same for DB and WAL devices. You need to edit the `paths` under `db_devices` for DB devices and `paths` under `wal_devices` for WAL devices.

Prerequisites

- A running Red Hat Ceph Storage.
- Daemons are non-colocated.
- Failed OSD

Procedure

1. Identify the devices in the cluster.

   Example

```
[root@host01 ~]# lsblk
```

```
NAME
                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda
                  8:0    0   20G  0 disk
├─sda1
                   8:1    0    1G  0 part /boot
└─sda2
                   8:2    0   19G  0 part
  ├─rhel-root
                253:0    0   17G  0 lvm  /
  └─rhel-swap
                253:1    0    2G  0 lvm  [SWAP]
sdb
                  8:16   0   10G  0 disk
└─ceph--5726d3e9--4fdb--4eda--b56a--3e0df88d663f-osd--
block--3ceb89ec--87ef--46b4--99c6--2a56bac09ff0 253:2    0   10G  0 lvm
sdc
                  8:32   0   10G  0 disk
└─ceph--d7c9ab50--f5c0--4be0--a8fd--e0313115f65c-osd--block--37c370df--1263--487f--
a476--08e28bdbcd3c 253:4    0   10G  0 lvm
sdd
                  8:48   0   10G  0 disk
├─ceph--1774f992--44f9--4e78--be7b--b403057cf5c3-osd--
db--31b20150--4cbc--4c2c--9c8f--6f624f3bfd89    253:7    0  2.5G  0 lvm
└─ceph--1774f992--44f9--4e78--be7b--b403057cf5c3-osd--db--1bee5101--dbab--4155--
a02c--e5a747d38a56    253:9    0  2.5G  0 lvm
sde
                  8:64   0   10G  0 disk
sdf
                  8:80   0   10G  0 disk
└─ceph--412ee99b--4303--4199--930a--0d976e1599a2-osd--
block--3a99af02--7c73--4236--9879--1fad1fe6203d 253:6    0   10G  0 lvm
sdg
                  8:96   0   10G  0 disk
└─ceph--316ca066--aeb6--46e1--8c57--f12f279467b4-osd--
block--58475365--51e7--42f2--9681--e0c921947ae6 253:8    0   10G  0 lvm
sdh
                  8:112  0   10G  0 disk
├─ceph--d7064874--66cb--4a77--a7c2--8aa0b0125c3c-osd--db--0dfe6eca--
ba58--438a--9510--d96e6814d853    253:3    0    5G  0 lvm
└─ceph--d7064874--66cb--4a77--a7c2--8aa0b0125c3c-osd--
db--26b70c30--8817--45de--8843--4c0932ad2429    253:5    0    5G  0 lvm
sr0
```

2. Log into the `cephadm` shell.

   Example

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

3. Identify the OSDs and their DB device.

   Example

```
[ceph: root@host01 /]# ceph-volume lvm list /dev/sdh

====== osd.2 =======

  [db]          /dev/ceph-d7064874-66cb-4a77-a7c2-8aa0b0125c3c/osd-db-0dfe6eca-
ba58-438a-9510-d96e6814d853

      block device              /dev/ceph-5726d3e9-4fdb-4eda-b56a-3e0df88d663f/osd-
block-3ceb89ec-87ef-46b4-99c6-2a56bac09ff0
      block uuid                GkWLoo-f0jd-Apj2-Zmwj-ce0h-OY6J-UuW8aD
      cephx lockbox secret
      cluster fsid              fa0bd9dc-e4c4-11ed-8db4-001a4a00046e
      cluster name              ceph
      crush device class
      db device                 /dev/ceph-d7064874-66cb-4a77-a7c2-8aa0b0125c3c/osd-
db-0dfe6eca-ba58-438a-9510-d96e6814d853
      db uuid                   6gSPoc-L39h-afN3-rDl6-kozT-AX9S-XR20xM
      encrypted                 0
      osd fsid                  3ceb89ec-87ef-46b4-99c6-2a56bac09ff0
      osd id                    2
      osdspec affinity          non-colocated
      type                      db
      vdo                       0
      devices                   /dev/sdh
```

```
 ====== osd.5 =======

   [db]          /dev/ceph-d7064874-66cb-4a77-a7c2-8aa0b0125c3c/osd-
db-26b70c30-8817-45de-8843-4c0932ad2429

       block device              /dev/ceph-d7c9ab50-f5c0-4be0-a8fd-e0313115f65c/osd-
block-37c370df-1263-487f-a476-08e28bdbcd3c
       block uuid                Eay3I7-fcz5-AWvp-kRcI-mJaH-n03V-Zr0wmJ
       cephx lockbox secret
       cluster fsid              fa0bd9dc-e4c4-11ed-8db4-001a4a00046e
       cluster name              ceph
       crush device class
       db device                 /dev/ceph-d7064874-66cb-4a77-a7c2-8aa0b0125c3c/osd-
db-26b70c30-8817-45de-8843-4c0932ad2429
       db uuid                   mwSohP-u72r-DHcT-BPka-piwA-lSwx-w24N0M
       encrypted                 0
       osd fsid                  37c370df-1263-487f-a476-08e28bdbcd3c
       osd id                    5
       osdspec affinity          non-colocated
       type                      db
       vdo                       0
       devices                   /dev/sdh
```

4. In the `osds.yaml` file, set `unmanaged` parameter to `true`, else `cephadm` redeploys the OSDs.

   Example

```
[ceph: root@host01 /]# cat osds.yml
service_type: osd
service_id: non-colocated
unmanaged: true
placement:
  host_pattern: 'ceph*'
data_devices:
  paths:
  - /dev/sdb
  - /dev/sdc
  - /dev/sdf
  - /dev/sdg
db_devices:
  paths:
  - /dev/sdd
  - /dev/sdh
```

5. Apply the updated specification file.

   Example

```
[ceph: root@host01 /]# ceph orch apply -i osds.yml

Scheduled osd.non-colocated update...
```

6. Check the status.

   Example

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

NAME           PORTS        RUNNING  REFRESHED  AGE  PLACEMENT
alertmanager   ?:9093,9094      1/1  9m ago     4d   count:1
crash                           3/4  4d ago     4d   *
grafana        ?:3000           1/1  9m ago     4d   count:1
mgr                             1/2  4d ago     4d   count:2
mon                             3/5  4d ago     4d   count:5
node-exporter  ?:9100           3/4  4d ago     4d   *
osd.non-colocated                 8  4d ago     5s   <unmanaged>
prometheus     ?:9095           1/1  9m ago     4d   count:1
```

7. Remove the OSDs. Ensure to use the `--zap` option to remove hte backend services and the `--replace` option to retain the OSD IDs.

   Example

```
[ceph: root@host01 /]# ceph orch osd rm 2 5 --zap --replace
Scheduled OSD(s) for removal
```

8. Check the status.

   Example

```
[ceph: root@host01 /]# ceph osd df tree | egrep -i "ID|host02|osd.2|osd.5"

ID  CLASS  WEIGHT   REWEIGHT  SIZE     RAW USE  DATA     OMAP  META     AVAIL   %USE
 VAR   PGS  STATUS     TYPE NAME
-5         0.04877         -   55 GiB   15 GiB  4.1 MiB   0 B   60 MiB  40 GiB  27.27
1.17    -                 host02
2    hdd  0.01219   1.00000   15 GiB  5.0 GiB  996 KiB   0 B   15 MiB  10 GiB  33.33
1.43    0  destroyed          osd.2
5    hdd  0.01219   1.00000   15 GiB  5.0 GiB  1.0 MiB   0 B   15 MiB  10 GiB  33.33
1.43    0  destroyed          osd.5
```

9. Edit the `osds.yaml` specification file to change `unmanaged` parameter to `false` and replace the path to the DB device if it has changed after the device got physically replaced.

   Example

```
[ceph: root@host01 /]# cat osds.yml
service_type: osd
service_id: non-colocated
unmanaged: false
placement:
  host_pattern: 'ceph01*'
data_devices:
  paths:
  - /dev/sdb
  - /dev/sdc
  - /dev/sdf
  - /dev/sdg
db_devices:
  paths:
  - /dev/sdd
  - /dev/sde
```

   In the above example, `/dev/sdh` is replaced with `/dev/sde`.

> **Important:** If you use the same host specification file to replace the faulty DB device on a single OSD node, modify the `host_pattern` option to specify only the OSD node, else the deployment fails and you cannot find the new DB device on other hosts.

10. Reapply the specification file with the `--dry-run` option to ensure the OSDs shall be deployed with the new DB device.

    Example

```
[ceph: root@host01 /]# ceph orch apply -i osds.yml --dry-run
WARNING! Dry-Runs are snapshots of a certain point in time and are bound
to the current inventory setup. If any of these conditions change, the
preview will be invalid. Please make sure to have a minimal
timeframe between planning and applying the specs.
####################
SERVICESPEC PREVIEWS
####################
+---------+------+--------+-------------+
|SERVICE  |NAME  |ADD_TO  |REMOVE_FROM  |
+---------+------+--------+-------------+
+---------+------+--------+-------------+
################
OSDSPEC PREVIEWS
################
+---------+-------+-------+----------+----------+-----+
|SERVICE  |NAME   |HOST   |DATA      |DB        |WAL  |
+---------+-------+-------+----------+----------+-----+
|osd      |non-colocated  |host02  |/dev/sdb  |/dev/sde  |-    |
|osd      |non-colocated  |host02  |/dev/sdc  |/dev/sde  |-    |
+---------+-------+-------+----------+----------+-----+
```

11. Apply the specification file:

    Example

    ```
    [ceph: root@host01 /]# ceph orch apply -i osds.yml
    Scheduled osd.non-colocated update...
    ```

12. Check the OSDs are redeployed.

    Example

    ```
    [ceph: root@host01 /]# ceph osd df tree | egrep -i "ID|host02|osd.2|osd.5"

    ID  CLASS  WEIGHT   REWEIGHT  SIZE     RAW USE  DATA     OMAP  META     AVAIL   %USE
    VAR   PGS  STATUS  TYPE NAME
    -5 0.04877 - 55 GiB 15 GiB 4.5 MiB 0 B 60 MiB 40 GiB 27.27 1.17 - host host02
    2 hdd 0.01219 1.00000 15 GiB 5.0 GiB 1.1 MiB 0 B 15 MiB 10 GiB 33.33 1.43 0 up osd.2
    5 hdd 0.01219 1.00000 15 GiB 5.0 GiB 1.1 MiB 0 B 15 MiB 10 GiB 33.33 1.43 0 up osd.5
    ```

    -

Verification

– From the OSD host where the OSDS are redeployed, verify if they are on the new DB device.

  Example

  ```
   [ceph: root@host01 /]# ceph-volume lvm list /dev/sde

   ====== osd.2 =======

     [db]          /dev/ceph-15ce813a-8a4c-46d9-ad99-7e0845baf15e/osd-
  db-1998a02e-5e67-42a9-b057-e02c22bbf461

         block device              /dev/ceph-a4afcb78-c804-4daf-b78f-3c7ad1ed0379/osd-
  block-564b3d2f-0f85-4289-899a-9f98a2641979
         block uuid                ITPVPa-CCQ5-BbFa-FZCn-FeYt-c5N4-ssdU41
         cephx lockbox secret
         cluster fsid              fa0bd9dc-e4c4-11ed-8db4-001a4a00046e
         cluster name              ceph
         crush device class
         db device                 /dev/ceph-15ce813a-8a4c-46d9-ad99-7e0845baf15e/osd-
  db-1998a02e-5e67-42a9-b057-e02c22bbf461
         db uuid                   HF1bYb-fTK7-0dcB-CHzW-xvNn-dCym-KKdU5e
         encrypted                 0
         osd fsid                  564b3d2f-0f85-4289-899a-9f98a2641979
         osd id                    2
         osdspec affinity          non-colocated
         type                      db
         vdo                       0
         devices                   /dev/sde

   ====== osd.5 =======

     [db]          /dev/ceph-15ce813a-8a4c-46d9-ad99-7e0845baf15e/osd-
  db-6c154191-846d-4e63-8c57-fc4b99e182bd

         block device              /dev/ceph-b37c8310-77f9-4163-964b-f17b4c29c537/osd-
  block-b42a4f1f-8e19-4416-a874-6ff5d305d97f
         block uuid                0LuPoz-ao7S-UL2t-BDIs-C9pl-ct8J-xh5ep4
         cephx lockbox secret
         cluster fsid              fa0bd9dc-e4c4-11ed-8db4-001a4a00046e
         cluster name              ceph
         crush device class
         db device                 /dev/ceph-15ce813a-8a4c-46d9-ad99-7e0845baf15e/osd-
  db-6c154191-846d-4e63-8c57-fc4b99e182bd
         db uuid                   SvmXms-iWkj-MTG7-VnJj-r5Mo-Moiw-MsbqVD
         encrypted                 0
         osd fsid                  b42a4f1f-8e19-4416-a874-6ff5d305d97f
         osd id                    5
         osdspec affinity          non-colocated
         type                      db
         vdo                       0
         devices                   /dev/sde
  ```

Stopping the removal of the OSDs

You can stop the removal of only the OSDs that are queued for removal. This resets the initial state of the OSD and takes it off the removal queue.

If the OSD is in the process of removal, then you cannot stop the process.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Hosts are added to the cluster.
- Monitor, Manager and OSD daemons are deployed on the cluster.
- Remove OSD process initiated.

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. Check the device and the node from which the OSD was initiated to be removed:

   Example

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

3. Stop the removal of the queued OSD:

   Syntax

   ```
   ceph orch osd rm stop OSD_ID
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch osd rm stop 0
   ```

4. Check the status of the OSD removal:

   Example

   ```
   [ceph: root@host01 /]# ceph orch osd rm status
   ```

Verification

- Verify the details of the devices and the nodes from which the Ceph OSDs were queued for removal:

  Example

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

Reference

- *Removing the OSD daemons*

Activating the OSDs

You can activate the OSDs in the cluster in cases where the operating system of the host was reinstalled.

Prerequisites

- A running Red Hat Ceph Storage cluster.

- Hosts are added to the cluster.
- Monitor, Manager and OSD daemons are deployed on the storage cluster.

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. After the operating system of the host is reinstalled, activate the OSDs:

   Syntax

   ```
   ceph cephadm osd activate HOSTNAME
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph cephadm osd activate host03
   ```

Verification

- List the service:

  Example

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

- List the hosts, daemons, and processes:

  Syntax

  ```
  ceph orch ps --service_name=SERVICE_NAME
  ```

  Example

  ```
  [ceph: root@host01 /]# ceph orch ps --service_name=osd
  ```

Observing the data migration

When you add or remove an OSD to the CRUSH map, Ceph begins rebalancing the data by migrating placement groups to the new or existing OSD(s).

You can observe the data migration by using the **ceph -w** command.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Recently added or removed an OSD.

Procedure

1. To observe the data migration:

   Example

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

2. Watch as the placement group states change from `active+clean` to `active, some degraded objects`, and finally `active+clean` when migration completes.
3. To exit the utility, press `Ctrl + C`.

Recalculating the placement groups

Adjusting the number of placement groups can be done online. Recalculating is not only a recalculation of the PG numbers, but will involve data relocation, which will be a lengthy process. However, the data availability will be maintained at any time.

Placement groups (PGs) define the spread of any pool data across the available OSDs. A placement group is built upon the given redundancy algorithm to be used. For a 3-way replication, the redundancy is defined to use three different OSDs. For erasure-coded pools, the number of OSDs to use is defined by the number of chunks.

When defining a pool the number of placement groups defines the grade of granularity the data is spread with across all available OSDs. The higher the number the better the equalization of capacity load can be. However, since handling the placement groups is also important in case of reconstruction of data, the number is significant to be carefully chosen upfront. To support calculation a tool is available to produce agile environments.

During the lifetime of a storage cluster, a pool can grow above the initially anticipated limits. With the growing number of drives a recalculation is recommended. The number of placement groups per OSD should be around 100. When adding more OSDs to the storage cluster the number of PGs per OSD will lower over time. Starting with 120 drives initially in the storage cluster and setting the `pg_num` of the pool to 4000 will end up in 100 PGs per OSD, given with the replication factor of three. Over time, when growing to ten times the number of OSDs, the number of PGs per OSD will go down to ten only. Because a small number of PGs per OSD will tend to an unevenly distributed capacity, consider adjusting the PGs per pool.

Very high numbers of PGs per OSD should be avoided, because reconstruction of all PGs on a failed OSD will start at once. A high number of IOPS is required to perform reconstruction in a timely manner, which might not be available. This would lead to deep I/O queues and high latency rendering the storage cluster unusable or will result in long healing times.

Reference

- To calculate the values given by use case, see the *PG calculator*.
- *Erasure Code Pools*

Checking whether OSDs are safe to upgrade

Determines whether Object Storage Daemons (OSDs) can be safely upgraded using the `ceph osd ok-to-upgrade` command. This command is primarily used by the Ceph Orchestrator to automate upgrade batching.

The `ceph osd ok-to-upgrade` command replaces the older `ok-to-stop` command, improving upgrade efficiency and reducing upgrade times.

The `mgr_osd_upgrade_check_convergence_factor` configuration option lets you tune the balance between response time and the size of the safe set. Adjust this value based on observed command performance.

How the command works

Before upgrading Object Storage Daemons (OSDs), you can use the `ceph osd ok-to-upgrade` command to determine how many OSDs can be upgraded at the same time without impacting access to data.

The command checks the current cluster state and identifies a set of OSDs within a specified CRUSH bucket that can be upgraded simultaneously. When a safe set is found, all data remains readable and writeable. However, data redundancy might be temporarily reduced and some placement groups (PGs) might appear as degraded but remain active.

This behavior is used when scoping upgrades to a CRUSH failure domain, such as a rack, to determine which OSDs in that domain can be upgraded in parallel. For more information, see “Upgrading OSDs by rack” on page 0.

Only OSDs that still require an upgrade to the specified Ceph version are considered. If a safe set cannot be determined, the command returns an error message explaining the reason.

> **Note:** Use this command as part of upgrade planning. Ensure that the cluster is healthy before upgrading OSDs.

During orchestrated cluster upgrades, the Ceph Orchestrator uses this command to determine how many OSDs can be upgraded concurrently. For more information, see “Upgrading the Red Hat Ceph Storage cluster” on page 0.

Usage

Use the following syntax and parameters to use the **ceph osd ok-to-upgrade** command.

```
ceph osd ok-to-upgrade CRUSH_BUCKET_NAME NEW_CEPH_VERSION_SHORT [--max NUM]
```

CRUSH_BUCKET_NAME

The CRUSH bucket that contains the OSDs to evaluate. Supported bucket types are `rack`, `chassis`, `host`, and `osd`.

NEW_CEPH_VERSION_SHORT

The target Ceph version in short format, for example `20.3.0-3803-g63ca1ffb5a2`, `20.1.0-144.el9cp`. This format matches the `NEW_CEPH_VERSION_SHORT` value shown by the `ceph osd metadata id` command.

--max NUM

Optional. Limits the number of OSDs returned in the safe set to the specified value. Use this option to upgrade OSDs in smaller batches.

Example

```
# ceph osd ok-to-upgrade my_crush_bucket my_ceph_version
{"ok_to_upgrade":true,"all_osds_upgraded":false,"osds_in_crush_bucket":
[9,21,2,15,17,10,4,22],"osds_ok_to_upgrade":[9,21,2,15,17,10,4,22],"osds_upgraded":
[],"bad_no_version":[]}
```

The command returns a JSON object with the following fields.

ok_to_upgrade

Indicates whether one or more OSDs in the specified CRUSH bucket can be safely upgraded to the target Ceph version. Values are `true` or `false`.

osds_in_crush_bucket

Lists all OSD IDs that belong to the specified CRUSH bucket.

osds_ok_to_upgrade

Lists the OSD IDs within the specified CRUSH bucket that are currently eligible for upgrade to the target Ceph version.

osds_upgraded

Lists the OSD IDs in the specified CRUSH bucket that have already been upgraded to the target Ceph version.

bad_no_version

Lists OSD IDs for which the current Ceph version could not be determined.

Managing the monitoring stack

As a storage administrator, you can use the Ceph Orchestrator with `cephadm` in the backend to deploy monitoring and alerting stack. The monitoring stack consists of Prometheus, Prometheus exporters, Prometheus Alertmanager, and Grafana. Users need to either define these services with `cephadm` in a YAML configuration file, or they can use the command line interface to deploy them. When multiple services of the same type are deployed, a highly-available setup is deployed. The node exporter is an exception to this rule.

> **Note:** Red Hat Ceph Storage does not support custom images for deploying monitoring services such as Prometheus, Grafana, Alertmanager, and node-exporter.

The following monitoring services can be deployed with `cephadm`:

- Prometheus is the monitoring and alerting toolkit. It collects the data provided by Prometheus exporters and fires preconfigured alerts if predefined thresholds have been reached. The Prometheus manager module provides a Prometheus exporter to pass on Ceph performance counters from the collection point in `ceph-mgr`.

The Prometheus configuration, including scrape targets, such as metrics providing daemons, is set up automatically by `cephadm`. `cephadm` also deploys a list of default alerts, for example, health error, 10% OSDs down, or pgs inactive.

- Alertmanager handles alerts sent by the Prometheus server. It deduplicates, groups, and routes the alerts to the correct receiver. By default, the Ceph dashboard is automatically configured as the receiver. The Alertmanager handles alerts sent by the Prometheus server. Alerts can be silenced using the Alertmanager, but silences can also be managed using the Ceph Dashboard.
- Grafana is a visualization and alerting software. The alerting functionality of Grafana is not used by this monitoring stack. For alerting, the Alertmanager is used.

By default, traffic to Grafana is encrypted with TLS. You can either supply your own TLS certificate or use a self-signed one. If no custom certificate has been configured before Grafana has been deployed, then a self-signed certificate is automatically created and configured for Grafana. Custom certificates for Grafana can be configured using the following commands:

```
ceph config-key set mgr/cephadm/HOSTNAME/grafana_key -i PRESENT_WORKING_DIRECTORY/key.pem
ceph config-key set mgr/cephadm/HOSTNAME/grafana_crt -i PRESENT_WORKING_DIRECTORY/
certificate.pem
```

Node exporter is an exporter for Prometheus which provides data about the node on which it is installed. It is recommended to install the node exporter on all nodes. This can be done using the `monitoring.yml` file with the node-exporter service type.

Deploying the monitoring stack

You can deploy the monitoring stack using the service specification in YAML file format. All the monitoring services can have the network and port they bind to configured in the YAML file.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

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

Deploy the monitoring stack using the Ceph orchestrator. The monitoring stack consists of Prometheus, Prometheus exporters, Prometheus Alertmanager, Grafana, and Ceph Exporter. Ceph Dashboard makes use of these components to store and visualize detailed metrics on cluster usage and performance.

1. Enable the Prometheus module in the Ceph Manager daemon. This exposes the internal Ceph metrics so that Prometheus can read them.

   ```
   ceph mgr module enable prometheus
   ```

   > **Important:** Ensure this command is run before Prometheus is deployed. If the command was not run before the deployment, you must redeploy Prometheus to update the configuration.
   >
   > ```
   > ceph orch redeploy prometheus
   > ```

2. Navigate to the following directory.

   ```
   cd /var/lib/ceph/DAEMON_PATH/
   ```

   For example,

   ```
   [ceph: root@host01 mds/]# cd /var/lib/ceph/monitoring/
   ```

Note:

Note: If the directory `monitoring` does not exist, create it.

3. Create the `monitoring.yml` file.

```
touch monitoring.yml
```

4. Edit the specification file with a content similar to the following example.
For example,

```
service_type: prometheus
service_name: prometheus
placement:
  hosts:
  - host01
networks:
- 192.169.142.0/24
---
service_type: node-exporter
---
service_type: alertmanager
service_name: alertmanager
placement:
  hosts:
  - host01
networks:
- 192.169.142.0/24
---
service_type: grafana
service_name: grafana
placement:
  hosts:
  - host01
networks:
- 192.169.142.0/24
---
service_type: ceph-exporter
```

Important: Ensure the monitoring stack components **`alertmanager`**, **`prometheus`**, and **`grafana`** are deployed on the same host. The **`node-exporter`** and **`ceph-exporter`** components should be deployed on all the hosts.

5. Apply monitoring services.

```
ceph orch apply -i MONITORING_STACK_FILE.yml
```

For example,

```
[ceph: root@host01 monitoring]# ceph orch apply -i monitoring.yml
```

AFTER COMPLETING THE TASK

Important: Prometheus, Grafana, and the Ceph dashboard are all automatically configured to talk to each other, resulting in a fully functional Grafana integration in the Ceph dashboard.

– List the service.

```
ceph orch ls
```

- List the hosts, daemons, and processes.

  ```
  ceph orch ps --service_name=SERVICE_NAME
  ```

  For example,

  ```
  [ceph: root@host01 /]# ceph orch ps --service_name=prometheus
  ```

Removing the monitoring stack

You can remove the monitoring stack using the **ceph orch rm** command.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Monitoring stack deployed on the cluster.

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. Use the `ceph orch rm` command to remove the monitoring stack:

   Syntax

   ```
   ceph orch rm SERVICE_NAME --force
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch rm grafana
   [ceph: root@host01 /]# ceph orch rm prometheus
   [ceph: root@host01 /]# ceph orch rm node-exporter
   [ceph: root@host01 /]# ceph orch rm ceph-exporter
   [ceph: root@host01 /]# ceph orch rm alertmanager
   [ceph: root@host01 /]# ceph mgr module disable prometheus
   ```

3. Check the status of the process:

   Example

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

Verification

- List the service:

  Example

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

- List the hosts, daemons, and processes:

  Syntax

  ```
  ceph orch ps
  ```

  Example

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

Reference

- *Deploying the monitoring stack*

Basic client setup

As a storage administrator, you have to set up Red Hat Ceph Storage client machines with basic configuration to interact with the storage cluster.

Most client machines only need the `ceph-common package` and its dependencies installed. It will supply the basic `ceph` and `rados` commands, as well as other commands like `mount.ceph` and `rbd`.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to all the nodes.
- Hosts are added to the cluster.
- All manager, monitor and OSD daemons are deployed.

Configuring file setup on client machines

Client machines generally need a smaller configuration file than a full-fledged storage cluster member. You can generate a minimal configuration file which can give details to clients to reach the Ceph monitors.

Prerequisites

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

Procedure

1. On the node where you want to set up the files, create a directory `ceph` in the `/etc` folder:

   Example

   ```
   [root@host01 ~]# mkdir /etc/ceph/
   ```

2. Navigate to `/etc/ceph` directory:

   Example

   ```
   [root@host01 ~]# cd /etc/ceph/
   ```

3. Generate the configuration file in the `ceph` directory:

   Example

   ```
   [root@host01 ceph]# ceph config generate-minimal-conf

   # minimal ceph.conf for 417b1d7a-a0e6-11eb-b940-001a4a000740
   [global]
           fsid = 417b1d7a-a0e6-11eb-b940-001a4a000740
           mon_host = [v2:10.74.249.41:3300/0,v1:10.74.249.41:6789/0]
   ```

   The contents of this file should be installed in `/etc/ceph/ceph.conf` path. You can use this configuration file to reach the Ceph monitors.

Setting-up keyring on client machines

Most Ceph clusters are run with the authentication enabled, and the client needs the keys in order to communicate with cluster machines. You can generate the keyring which can give details to clients to reach the Ceph monitors.

Prerequisites

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

Procedure

1. On the node where you want to set up the keyring, create a directory `ceph` in the `/etc` folder:

   Example

   ```
   [root@host01 ~]# mkdir /etc/ceph/
   ```

2. Navigate to `/etc/ceph` directory in the `ceph` directory:

   Example

   ```
   [root@host01 ~]# cd /etc/ceph/
   ```

3. Generate the keyring for the client:

   Syntax

   ```
   ceph auth get-or-create client.CLIENT_NAME -o /etc/ceph/NAME_OF_THE_FILE
   ```

   Example

   ```
   [root@host01 ceph]# ceph auth get-or-create client.fs -o /etc/ceph/ceph.keyring
   ```

4. Verify the output in the `ceph.keyring` file:

   Example

   ```
   [root@host01 ceph]# cat ceph.keyring

   [client.fs]
           key = AQAvoH5gkUCsExAATz3xCBLd4n6B6jRv+Z7CVQ==
   ```

   The resulting output should be put into a keyring file, for example `/etc/ceph/ceph.keyring`.

Managing the MDS service using the Ceph Orchestrator

Use the Ceph Orchestrator with `cephadm` in the backend to deploy the MDS service.

By default, a Ceph File System (CephFS) uses only one active MDS daemon. However, systems with many clients benefit from multiple active MDS daemons.

When managing the MDS service by using the Ceph Orchestrator, be sure to have:

- A running Red Hat Ceph Storage cluster.
- Root-level access to all the nodes.
- Hosts added to the cluster.
- All manager, monitor, and OSD daemons deployed.

Deploying the MDS service with the command-line interface

Using the Ceph Orchestrator, you can deploy the Metadata Server (MDS) service by using the `placement` specification in the command-line interface.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.

- Hosts added to the cluster.
- All manager, monitor, and OSD daemons deployed.

Ceph File System (CephFS) requires one or more MDS.

> **Note:** Ensure that you have at least two pools, one for Ceph file system (CephFS) data and one for CephFS metadata.

For more information about creating Ceph File Systems, see Creating Ceph File Systems.

To deploy the MDS service with the command-line interface, log in to the `cephadm` shell.

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

Use one of the following options to deploy MDS daemons using placement specification.

Creating the MDS daemons through the command-line

- Use **`ceph fs volume`** to create the MDS daemons.
  Creating the MDS daemons creates the CephFS volume and pools associated with the CephFS, and also starts the MDS service on the hosts.

  ```
  ceph fs volume create FILESYSTEM_NAME --
  placement="NUMBER_OF_DAEMONS HOST_NAME_1 HOST_NAME_2 HOST_NAME_3"
  ```

  > **Note:** By default, replicated pools are created for this command.

  For example,

  ```
  [ceph: root@host01 /]# ceph fs volume create test --placement="2 host01 host02"
  ```

Creating the pools, CephFS, and then deploying MDS service using placement specification

1. Create the pools for CephFS.

   ```
   ceph osd pool create DATA_POOL PG_NUM
   ceph osd pool create METADATA_POOL PG_NUM
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph osd pool create cephfs_data 64
   [ceph: root@host01 /]# ceph osd pool create cephfs_metadata 64
   ```

2. Create the file system for the data pools and metadata pools.

   ```
   ceph fs new FILESYSTEM_NAME METADATA_POOL DATA_POOL
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph fs new test cephfs_metadata cephfs_data
   ```

3. Deploy MDS service using the **`ceph orch apply`** command.

   ```
   ceph orch apply mds FILESYSTEM_NAME --
   placement="NUMBER_OF_DAEMONS HOST_NAME_1 HOST_NAME_2 HOST_NAME_3"
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph orch apply mds test --placement="2 host01 host02"
   ```

Verifying the MDS service deployment

1. List the service.
   For example,

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

2. Check the CephFS status.
   For example,

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

3. List the hosts, daemons, and processes.

   ```
   ceph orch ps --daemon_type=DAEMON_NAME
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph orch ps --daemon_type=mds
   ```

Deploying the MDS service using the service specification

Using the Ceph Orchestrator, you can deploy the MDS service using the service specification.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Hosts added to the cluster.
- All manager, monitor, and OSD daemons deployed.

> **Note:** Ensure that you have at least two pools, one for the Ceph File System (CephFS) data and one for the CephFS metadata.

1. Create the `mds.yaml` file.

   ```
   touch mds.yaml
   ```

2. Edit the `mds.yaml` file.

   ```
   service_type: mds
   service_id: FILESYSTEM_NAME
   placement:
     hosts:
     - HOST_NAME_1
     - HOST_NAME_2
     - HOST_NAME_3
   ```

   For example,

   ```
   service_type: mds
   service_id: fs_name
   placement:
     hosts:
     - host01
     - host02
   ```

3. Mount the YAML file under a directory in the container.
   For example,

   ```
   [root@host01 ~]# cephadm shell --mount mds.yaml:/var/lib/ceph/mds/mds.yaml
   ```

4. Navigate to the ceph/mds/ directory.
   For example,

   ```
   [ceph: root@host01 /]# cd /var/lib/ceph/mds/
   ```

5. Log into the cephadm shell.

   ```
   cephadm shell
   ```

6. Navigate to the /ceph/mds/ directory.
   For example,

   ```
   [ceph: root@host01 /]# cd /var/lib/ceph/mds/
   ```

7. Deploy MDS service using service specification.

   ```
   ceph orch apply -i FILE_NAME.yaml
   ```

   For example,

   ```
   [ceph: root@host01 mds]# ceph orch apply -i mds.yaml
   ```

8. Once the MDS services is deployed and functional, create the CephFS.

   ```
   ceph fs new CEPHFS_NAME METADATA_POOL DATA_POOL
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph fs new test metadata_pool data_pool
   ```

AFTER COMPLETING THE TASK

Verify the MDS service deployment.

1. List the service.

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

2. List the hosts, daemons, and processes.

   ```
   ceph orch ps --daemon_type=DAEMON_NAME
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph orch ps --daemon_type=mds
   ```

Removing the MDS service using the Ceph Orchestrator

Remove the service either by using the **ceph orch rm** command or by removing the file system and the associated pools.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Hosts added to the cluster.
- Root-level access to all the nodes.
- At least one MDS daemon deployed on the hosts.

To remove the MDS service with the command-line interface, log in to the cephadm shell.

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

Use one of the following options to remove MDS daemons.

Removing the MDS service from the entire cluster through the command line

Use the **ceph orch rm** command to remove the MDS service from the entire cluster.

1. List the service.

```
ceph orch ls
```

For example,

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

2. Remove the service.

```
ceph orch rm SERVICE_NAME
```

For example,

```
[ceph: root@host01 /]# ceph orch rm mds.test
```

Removing the CephFS volume, associated pools, and the services

1. Set the configuration parameter **mon_allow_pool_delete** to `true`.

```
ceph config set mon mon_allow_pool_delete true
```

For example,

```
[ceph: root@host01 /]# ceph config set mon mon_allow_pool_delete true
```

2. Remove the file system, its data, and metadata pools.

```
ceph fs volume rm FILESYSTEM_NAME --yes-i-really-mean-it
```

This command also tries to remove the MDS using the enabled `ceph-mgr` Orchestrator module.

```
ceph fs volume rm cephfs-new --yes-i-really-mean-it
```

For example,

```
[ceph: root@host01 /]# ceph fs volume rm cephfs-new --yes-i-really-mean-it
```

Verifying the MDS service removal

– List the hosts, daemons, and processes, by using the **ceph orch ps** command.
For example,

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

Managing the Ceph Object Gateway

As a storage administrator, you can deploy Ceph Object Gateway using the command line interface or by using the service specification.

You can also configure multi-site Ceph Object Gateways, and remove the Ceph Object Gateway.

`cephadm` deploys Ceph Object Gateway as a collection of daemons that manages a single-cluster deployment or a particular realm and zone in a multi-site deployment.

> **Note:**

- With `cephadm`, the Ceph Object Gateway daemons are configured using the monitor configuration database instead of a `ceph.conf` or the command line. If that configuration is not already in the `client.rgw` section, then the Ceph Object Gateway will start up with default settings and bind to the port `80`.
- The `.default.rgw.buckets.index` pool is created only after the bucket is created in Ceph Object Gateway, while the `.default.rgw.buckets.data` pool is created after the data is uploaded to the bucket.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to all the nodes.
- Hosts are added to the cluster.
- All the managers, monitors, and OSDs are deployed in the storage cluster.

Deploying the Ceph Object Gateway using the command line interface

Deploy the Ceph Object Gateway by using the Ceph Orchestrator with the **`ceph orch`** command in the command line interface.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Root-level access to all the nodes.
- Hosts are added to the cluster.
- All manager, monitor, and OSD daemons are deployed.

Log in to the `cephadm` shell by using the **`cephadm shell`** to deploy Ceph Object Gateway daemons.
Deploy using one of the following methods:

- "Creating a realm, zone group, and zone" on page 275
- "Using an arbitrary service name for a single cluster deployment" on page 276
- "Using an arbitrary service name on a labeled set of hosts" on page 276

Creating a realm, zone group, and zone

Create a realm, zone group, zone, and then use the placement specification with the hostname.

1. Create a realm.

```
radosgw-admin realm create --rgw-realm=REALM_NAME --default
```

For example,

```
[ceph: root@host01 /]# radosgw-admin realm create --rgw-realm=test_realm --default
```

2. Create a zone group.

```
radosgw-admin zonegroup create --rgw-zonegroup=ZONE_GROUP_NAME  --master --default
```

For example,

```
[ceph: root@host01 /]# radosgw-admin zonegroup create --rgw-zonegroup=default  --
master --default
```

3. Create a zone.

```
radosgw-admin zone create --rgw-zonegroup=ZONE_GROUP_NAME --rgw-zone=ZONE_NAME --
master --default
```

For example,

```
[ceph: root@host01 /]# radosgw-admin zone create --rgw-zonegroup=default --rgw-
zone=test_zone --master --default
```

4. Commit the changes.

```
radosgw-admin period update --rgw-realm=REALM_NAME --commit
```

For example,

```
[ceph: root@host01 /]# radosgw-admin period update --rgw-realm=test_realm --commit
```

5. Apply the changes by using the **ceph orch apply** command.

```
ceph orch apply rgw NAME [--realm=REALM_NAME] [--zone=ZONE_NAME] [--
zonegroup=ZONE_GROUP_NAME] --placement="NUMBER_OF_DAEMONS [HOST_NAME_1 HOST_NAME_2]"
```

For example,

```
[ceph: root@host01 /]# ceph orch apply rgw test --realm=test_realm --zone=test_zone --
zonegroup=default --placement="2 host01 host02"
```

Using an arbitrary service name for a single cluster deployment

Use an arbitrary service name to deploy two Ceph Object Gateway daemons for a single cluster deployment.

```
ceph orch apply rgw SERVICE_NAME
```

For example,

```
[ceph: root@host01 /]# ceph orch apply rgw foo
```

Using an arbitrary service name on a labeled set of hosts

Use an arbitrary service name on a labeled set of hosts.

```
ceph orch host label add HOST_NAME_1 LABEL_NAME
ceph orch host label add HOSTNAME_2 LABEL_NAME
ceph orch apply rgw SERVICE_NAME --placement="label:LABEL_NAME count-per-
host:NUMBER_OF_DAEMONS" --port=8000
```

NUMBER_OF_DAEMONS controls the number of Ceph Object Gateways that are deployed on each host. To achieve the highest performance without incurring any extra cost, set this value to 2.
For example,

```
[ceph: root@host01 /]# ceph orch host label add host01 rgw  # the 'rgw' label can be
anything
[ceph: root@host01 /]# ceph orch host label add host02 rgw
[ceph: root@host01 /]# ceph orch apply rgw foo --placement="label:rgw count-per-host:2" --
port=8000
```

AFTER COMPLETING THE TASK

Verify the Ceph Object Gateway deployment.

- List the service, by using the **ceph orch ls** command.
  For example,

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

- List the hosts, daemons, and processes.

```
ceph orch ps --daemon_type=DAEMON_NAME
```

For example,

```
[ceph: root@host01 /]# ceph orch ps --daemon_type=rgw
```

Deploying the Ceph Object Gateway using the service specification

Deploy the Ceph Object Gateway using the service specification with either the default or the custom realms, zones, and zone groups.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the bootstrapped host.
- Hosts are added to the cluster.
- All manager, monitor, and OSD daemons are deployed.

Procedure

1. As a root user, create a specification file.

   ```
   touch rgw_spec.yaml
   ```

   For example,

   ```
   [root@host01 ~]# touch rgw_spec.yaml
   ```

2. Configure S3 requests to wait for the duration defined in the **`rgw_exit_timeout_secs`** parameter for all outstanding requests to complete by setting **`rgw_graceful_stop`** to *true* during Ceph Object gateway shutdown/restart.

   ```
   ceph config set client.rgw rgw_graceful_stop true

   ceph config set client.rgw rgw_exit_timeout_secs 120
   ```

   Note: In containerized deployments, an additional `extra_container_args` configuration of `--stop-timeout=120` (or the value of **`rgw_exit_timeout_secs`** configuration, if not default) is also necessary in order for it to work as expected with `ceph orch stop/restart` commands.
   For example,

   ```
   [root@host1 ~]$ cat rgw_spec.yaml
   service_type: rgw
   service_id: foo
   placement:
     count_per_host: 1
     hosts:
       - rgw_node
   spec:
     rgw_frontend_port: 8081
   extra_container_args:
     - "--stop-timeout=120"
   ```

3. Edit the `rgw_spec.yaml` file to include the following details for the default realm, zone, and zone group.

   ```
   service_type: rgw
   service_id: REALM_NAME.ZONE_NAME
   placement:
     hosts:
     - HOST_NAME_1
     - HOST_NAME_2
     count_per_host: NUMBER_OF_DAEMONS
   spec:
     rgw_realm: REALM_NAME
   ```

```
  rgw_zone: ZONE_NAME
  rgw_zonegroup: ZONE_GROUP_NAME
  rgw_frontend_port: FRONT_END_PORT
networks:
  -  NETWORK_CIDR # Ceph Object Gateway service binds to a specific network
```

NUMBER_OF_DAEMONS controls the number of Ceph Object Gateways deployed on each host. To achieve the highest performance without incurring an additional cost, set this value to 2.

For example,

```
service_type: rgw
service_id: default
placement:
  hosts:
  - host01
  - host02
  - host03
  count_per_host: 2
spec:
  rgw_realm: default
  rgw_zone: default
  rgw_zonegroup: default
  rgw_frontend_port: 1234
networks:
  - 192.169.142.0/24
```

4. Optional: For custom realm, zone, and zone group, create the resources and then create the `rgw_spec.yaml` file.

   a. Create the custom realm, zone, and zone group.

      For example,

      ```
      [root@host01 ~]# radosgw-admin realm create --rgw-realm=test_realm --default
      [root@host01 ~]# radosgw-admin zonegroup create --rgw-zonegroup=test_zonegroup --
      default
      [root@host01 ~]# radosgw-admin zone create --rgw-zonegroup=test_zonegroup --rgw-
      zone=test_zone --default
      [root@host01 ~]# radosgw-admin period update --rgw-realm=test_realm --commit
      ```

   b. Create the `rgw_spec.yaml` file with the following details.

      For example,

      ```
      service_type: rgw
      service_id: test_realm.test_zone
      placement:
        hosts:
        - host01
        - host02
        - host03
        count_per_host: 2
      spec:
        rgw_realm: test_realm
        rgw_zone: test_zone
        rgw_zonegroup: test_zonegroup
        rgw_frontend_port: 1234
      networks:
        - 192.169.142.0/24
      ```

5. Optional: Deploy HA Proxy concentrator for local RGW load balancing. To load balance requests across RGW daemons on the same host, configure the RGW service with a local HA Proxy concentrator.

   Note: HA Proxy concentrator does not currently support SSL.

   Note: Use this only when `count_per_host > 1`.

   For example,

```
service_type: rgw
service_id: foo
service_name: rgw.foo
placement:
  count_per_host: 2
  hosts:
    - host1
    - host2
spec:
  concentrator: haproxy
  concentrator_frontend_port: 8080
  concentrator_monitor_port: 1967
  concentrator_monitor_user: admin
```

6. Mount the `rgw_spec.yaml` file under a directory in the container.

   For example,

   ```
   [root@host01 ~]# cephadm shell --mount rgw_spec.yaml:/var/lib/ceph/radosgw/
   rgw_spec.yaml
   ```

   Note: Every time you exit the shell, you have to mount the file in the container before deploying the daemon.

7. Deploy the Ceph Object Gateway using the service specification.

   ```
   ceph orch apply -i FILE_NAME.yaml
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph orch apply -i /var/lib/ceph/radosgw/rgw_spec.yaml
   ```

Note: In containerized deployments, an additional `extra_container_args` configuration of `--stop-timeout=120` (or the value of **`rgw_exit_timeout_secs`** configuration, if not default) is also necessary. For example,

```
[root@host1 ~]$ cat rgw_spec.yaml
service_type: rgw
service_id: foo
placement:
  count_per_host: 1
  hosts:
    - rgw_node
spec:
  rgw_frontend_port: 8081
extra_container_args:
  - "--stop-timeout=120"
```

Verification

- List the service.

  ```
  ceph orch ls
  ```

- List the hosts, daemons, and processes.

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ```

  For example,

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=rgw
  ```

Deploying a multi-site Ceph Object Gateway

Ceph Orchestrator supports multi-site configuration options for the Ceph Object Gateway.

You can configure each object gateway to work in an active-active zone configuration allowing writes to a non-primary zone. The multi-site configuration is stored within a container called a realm.

The realm stores zone groups, zones, and a time period. The `rgw` daemons handle the synchronization eliminating the need for a separate synchronization agent, thereby operating with an active-active configuration.

You can also deploy multi-site zones using the command line interface (CLI).

> **Note:** The following configuration assumes at least two Red Hat Ceph Storage clusters are in geographically separate locations. However, the configuration also works on the same site.

Prerequisites

- At least two running Red Hat Ceph Storage clusters.
- At least two Ceph Object Gateway instances, one for each Red Hat Ceph Storage cluster.
- Root-level access to all the nodes.
- Nodes or containers are added to the storage cluster.
- All Ceph Manager, Monitor and OSD daemons are deployed.

Procedure

1. In the `cephadm` shell, configure the primary zone:
   a. Create a realm:

      Syntax

      ```
      radosgw-admin realm create --rgw-realm=REALM_NAME --default
      ```

      Example

      ```
      [ceph: root@host01 /]# radosgw-admin realm create --rgw-realm=test_realm --
      default
      ```

      If the storage cluster has a single realm, then specify the `--default` flag.

   b. Create a primary zone group:

      Syntax

      ```
      radosgw-admin zonegroup create --rgw-zonegroup=ZONE_GROUP_NAME --
      endpoints=http://RGW_PRIMARY_HOSTNAME:RGW_PRIMARY_PORT_NUMBER_1 --master --
      default
      ```

      Example

      ```
      [ceph: root@host01 /]# radosgw-admin zonegroup create --rgw-zonegroup=us --
      endpoints=http://rgw1:80 --master --default
      ```

   c. Create a primary zone:

      Syntax

      ```
      radosgw-admin zone create --rgw-zonegroup=PRIMARY_ZONE_GROUP_NAME --rgw-
      zone=PRIMARY_ZONE_NAME --endpoints=http://
      RGW_PRIMARY_HOSTNAME:RGW_PRIMARY_PORT_NUMBER_1 --access-key=SYSTEM_ACCESS_KEY --
      secret=SYSTEM_SECRET_KEY
      ```

      Example

```
[ceph: root@host01 /]# radosgw-admin zone create --rgw-zonegroup=us --rgw-
zone=us-east-1 --endpoints=http://rgw1:80 --access-key=LIPEYZJLTWXRKXS9LPJC --
secret-key=IsAje0AVDNXNw48LjMAimpCpI7VaxJYSnfD0FFKQ
```

d. Optional: Delete the default zone, zone group, and the associated pools.

> **Important:** Do not delete the default zone and its pools if you are using the default zone and zone group to store data. Also, removing the default zone group deletes the system user.

To access old data in the `default` zone and zonegroup, use `--rgw-zone default` and `--rgw-zonegroup default` in `radosgw-admin` commands.

Example

```
[ceph: root@host01 /]# radosgw-admin zonegroup delete --rgw-zonegroup=default
[ceph: root@host01 /]# ceph osd pool rm default.rgw.log default.rgw.log --yes-i-
really-really-mean-it
[ceph: root@host01 /]# ceph osd pool rm default.rgw.meta default.rgw.meta --yes-
i-really-really-mean-it
[ceph: root@host01 /]# ceph osd pool rm default.rgw.control default.rgw.control
--yes-i-really-really-mean-it
[ceph: root@host01 /]# ceph osd pool rm default.rgw.data.root
default.rgw.data.root --yes-i-really-really-mean-it
[ceph: root@host01 /]# ceph osd pool rm default.rgw.gc default.rgw.gc --yes-i-
really-really-mean-it
```

e. Create a system user:

Syntax

```
radosgw-admin user create --uid=USER_NAME --display-name="USER_NAME" --access-
key=SYSTEM_ACCESS_KEY --secret=SYSTEM_SECRET_KEY --system
```

Example

```
[ceph: root@host01 /]# radosgw-admin user create --uid=zone.user --display-
name="Zone user" --system
```

Make a note of the `access_key` and `secret_key`.

f. Add the access key and system key to the primary zone:

Syntax

```
radosgw-admin zone modify --rgw-zone=PRIMARY_ZONE_NAME --access-key=ACCESS_KEY --
secret=SECRET_KEY
```

Example

```
[ceph: root@host01 /]# radosgw-admin zone modify --rgw-zone=us-east-1 --access-
key=LIPEYZJLTWXRKXS9LPJC --secret-key=IsAje0AVDNXNw48LjMAimpCpI7VaxJYSnfD0FFKQ
```

g. Commit the changes:

Syntax

```
radosgw-admin period update --commit
```

Example

```
[ceph: root@host01 /]# radosgw-admin period update --commit
```

h. Outside the `cephadm` shell, fetch the `FSID` of the storage cluster and the processes:

Example

```
[root@host01 ~]#  systemctl list-units | grep ceph
```

i. Start the Ceph Object Gateway daemon:

Syntax

```
systemctl start ceph-FSID@DAEMON_NAME
systemctl enable ceph-FSID@DAEMON_NAME
```

Example

```
[root@host01 ~]# systemctl start ceph-62a081a6-88aa-11eb-
a367-001a4a000672@rgw.test_realm.us-east-1.host01.ahdtsw.service
[root@host01 ~]# systemctl enable ceph-62a081a6-88aa-11eb-
a367-001a4a000672@rgw.test_realm.us-east-1.host01.ahdtsw.service
```

2. In the `cephadm` shell, configure the secondary zone.

a. Pull the primary realm configuration from the host:

Syntax

```
radosgw-admin realm pull --url=URL_TO_PRIMARY_ZONE_GATEWAY --access-
key=ACCESS_KEY --secret-key=SECRET_KEY
```

Example

```
[ceph: root@host04 /]# radosgw-admin realm pull --url=http://10.74.249.26:80 --
access-key=LIPEYZJLTWXRKXS9LPJC --secret-
key=IsAje0AVDNXNw48LjMAimpCpI7VaxJYSnfD0FFKQ
```

b. Pull the primary period configuration from the host:

Syntax

```
radosgw-admin period pull --url=URL_TO_PRIMARY_ZONE_GATEWAY --access-
key=ACCESS_KEY --secret-key=SECRET_KEY
```

Example

```
[ceph: root@host04 /]# radosgw-admin period pull --url=http://10.74.249.26:80 --
access-key=LIPEYZJLTWXRKXS9LPJC --secret-
key=IsAje0AVDNXNw48LjMAimpCpI7VaxJYSnfD0FFKQ
```

c. Configure a secondary zone:

Syntax

```
radosgw-admin zone create --rgw-zonegroup=ZONE_GROUP_NAME
              --rgw-zone=_SECONDARY_ZONE_NAME_
              --access-key=_SYSTEM_ACCESS_KEY_ --secret=SYSTEM_SECRET_KEY
              --endpoints=http://RGW_SECONDARY_HOSTNAME:PORT_NUMBER"
              [--read-only]
```

Example

```
[ceph: root@host04 /]# radosgw-admin zone create --rgw-zonegroup=us --rgw-
zone=us-east-2 --access-key=LIPEYZJLTWXRKXS9LPJC --secret-
key=IsAje0AVDNXNw48LjMAimpCpI7VaxJYSnfD0FFKQ --endpoints=http://
rgw.example.com:80
```

d. Optional: Delete the default zone.

> **Important:** Do not delete the default zone and its pools if you are using the default zone and zone group to store data.

To access old data in the `default` zone and zonegroup, use `--rgw-zone default` and `--rgw-zonegroup default` in `radosgw-admin` commands.

Example

```
[ceph: root@host04 /]# radosgw-admin zone rm --rgw-zone=default
[ceph: root@host04 /]# ceph osd pool rm default.rgw.log default.rgw.log --yes-i-
really-really-mean-it
[ceph: root@host04 /]# ceph osd pool rm default.rgw.meta default.rgw.meta --yes-
i-really-really-mean-it
[ceph: root@host04 /]# ceph osd pool rm default.rgw.control default.rgw.control
--yes-i-really-really-mean-it
[ceph: root@host04 /]# ceph osd pool rm default.rgw.data.root
default.rgw.data.root --yes-i-really-really-mean-it
[ceph: root@host04 /]# ceph osd pool rm default.rgw.gc default.rgw.gc --yes-i-
really-really-mean-it
```

e. Update the Ceph configuration database:

Syntax

```
ceph config set SERVICE_NAME rgw_zone SECONDARY_ZONE_NAME
```

Example

```
[ceph: root@host04 /]# ceph config set rgw rgw_zone us-east-2
```

f. Commit the changes:

Syntax

```
radosgw-admin period update --commit
```

Example

```
[ceph: root@host04 /]# radosgw-admin period update --commit
```

g. Outside the `cephadm` shell, fetch the FSID of the storage cluster and the processes:

Example

```
[root@host04 ~]#  systemctl list-units | grep ceph
```

h. Start the Ceph Object Gateway daemon:

Syntax

```
systemctl start ceph-FSID@DAEMON_NAME
systemctl enable ceph-FSID@DAEMON_NAME
```

Example

```
[root@host04 ~]# systemctl start ceph-62a081a6-88aa-11eb-
a367-001a4a000672@rgw.test_realm.us-east-2.host04.ahdtsw.service
[root@host04 ~]# systemctl enable ceph-62a081a6-88aa-11eb-
a367-001a4a000672@rgw.test_realm.us-east-2.host04.ahdtsw.service
```

3. Optional: Deploy multi-site Ceph Object Gateways using the placement specification:

Syntax

```
ceph orch apply rgw NAME --realm=REALM_NAME --zone=PRIMARY_ZONE_NAME --
placement="NUMBER_OF_DAEMONS HOST_NAME_1 HOST_NAME_2"
```

Example

```
[ceph: root@host04 /]# ceph orch apply rgw east --realm=test_realm --zone=us-east-1 --
placement="2 host01 host02"
```

Verification

- Check the synchronization status to verify the deployment:

  Example

  ```
  [ceph: root@host04 /]# radosgw-admin sync status
  ```

Removing the Ceph Object Gateway

Use the **ceph orch rm** command to remove Ceph Object Gateway daemons.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to all the nodes.
- Hosts are added to the cluster.
- At least one Ceph Object Gateway daemon deployed on the hosts.

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. List the service:

   Example

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

3. Remove the service:

   Syntax

   ```
   ceph orch rm SERVICE_NAME
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch rm rgw.test_realm.test_zone_bb
   ```

Verification

- List the hosts, daemons, and processes:

  Syntax

  ```
  ceph orch ps
  ```

  Example

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

Reference

- Deploying the Ceph Object Gateway using the command line interface
- Deploying the Ceph Object Gateway using the service specification

Handling a node failure

As a storage administrator, you can experience a whole node failing within the storage cluster, and handling a node failure is similar to handling a disk failure. With a node failure, instead of Ceph recovering placement groups (PGs) for only one disk, all PGs on the disks within that node must be recovered. Ceph will detect that the OSDs are all down and automatically start the recovery process, known as self-healing.

There are three node failure scenarios.

- Replacing the node by using the root and Ceph OSD disks from the failed node.
- Replacing the node by reinstalling the operating system and using the Ceph OSD disks from the failed node.
- Replacing the node by reinstalling the operating system and using all new Ceph OSD disks.

For a high level workflow for each node replacement scenario, see Workflow for replacing a node.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- A failed node.

Considerations before adding or removing a node

Ceph OSD nodes can be added or removed at run time. This means that you can resize the storage cluster capacity or replace hardware without taking down the storage cluster. Understand these considerations before adding or removing a node.

The ability to serve Ceph clients while the storage cluster is in a `degraded` state also has operational benefits. For example, you can add or remove or replace hardware during regular business hours, rather than working overtime or on weekends. However, adding and removing Ceph OSD nodes can have a significant impact on performance.

Before you add or remove Ceph OSD nodes, consider the effects on storage cluster performance:

- Whether you are expanding or reducing the storage cluster capacity, adding or removing Ceph OSD nodes induces backfilling as the storage cluster rebalances. During that rebalancing time period, Ceph uses additional resources, which can impact storage cluster performance.
- In a production Ceph storage cluster, a Ceph OSD node has a particular hardware configuration that facilitates a particular type of storage strategy.
- Since a Ceph OSD node is part of a CRUSH hierarchy, the performance impact of adding or removing a node typically affects the performance of pools that use the CRUSH ruleset.

> **Important:** In director-deployed Red Hat Ceph Storage environments, replacing Controller nodes can affect the Ceph Monitor service and lead to storage outages if Monitor IP addresses change. For more information, see “Managing the Ceph Monitor service with Red Hat OpenStack Platform” on page 215.

Workflow for replacing a node

There are three node failure scenarios. Use these high-level workflows for each scenario when replacing a node.

There are three node failure scenarios.

- “Replacing the node by using the root and Ceph OSD disks from the failed node” on page 286
- “Replacing the node by reinstalling the operating system and using the Ceph OSD disks from the failed node” on page 286
- “Replacing the node by reinstalling the operating system and using all new Ceph OSD disks” on page 286

Prerequisites

- A running Red Hat Ceph Storage cluster.
- A failed node.

Replacing the node by using the root and Ceph OSD disks from the failed node

Use the root and Ceph OSD disks from the failed node to replace the node.

1. Disable backfilling.

   ```
   ceph osd set noout
   ceph osd set noscrub
   ceph osd set nodeep-scrub
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph osd set noout
   [ceph: root@host01 /]# ceph osd set noscrub
   [ceph: root@host01 /]# ceph osd set nodeep-scrub
   ```

2. Replace the node, taking the disks from the old node, and adding them to the new node.
3. Enable backfilling.

   ```
   ceph osd unset noout
   ceph osd unset noscrub
   ceph osd unset nodeep-scrub
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph osd unset noout
   [ceph: root@host01 /]# ceph osd unset noscrub
   [ceph: root@host01 /]# ceph osd unset nodeep-scrub
   ```

Replacing the node by reinstalling the operating system and using the Ceph OSD disks from the failed node

Reinstall the operating system and use the Ceph OSD disks from the failed node to replace the node.

1. Disable backfilling.

   ```
   ceph osd set noout
   ceph osd set noscrub
   ceph osd set nodeep-scrub
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph osd set noout
   [ceph: root@host01 /]# ceph osd set noscrub
   [ceph: root@host01 /]# ceph osd set nodeep-scrub
   ```

2. Create a backup of the Ceph configuration.

   ```
   cp /etc/ceph/ceph.conf /PATH_TO_BACKUP_LOCATION/ceph.conf
   ```

   For example,

   ```
   [ceph: root@host01 /]# cp /etc/ceph/ceph.conf /some/backup/location/ceph.conf
   ```

3. Replace the node and add the Ceph OSD disks from the failed node.
4. Configure disks as JBOD.

   > **Note:** This should be done by the storage administrator.

5. Install the operating system.
   For more information about operating system requirements, see Operating system requirements. For ore information about installing the operating system, see the Red Hat Enterprise Linux product documentation.

> **Note:** This should be done by the system administrator.

6. Restore the Ceph configuration.

```
cp /PATH_TO_BACKUP_LOCATION/ceph.conf /etc/ceph/ceph.conf
```

   For example,

```
[ceph: root@host01 /]# cp /some/backup/location/ceph.conf /etc/ceph/ceph.conf
```

7. Add the new node to the storage cluster using the Ceph Orchestrator commands.
   Ceph daemons are placed automatically on the respective node. For more information, see “Adding a Ceph OSD node” on page 289.

8. Enable backfilling.

```
ceph osd unset noout
ceph osd unset noscrub
ceph osd unset nodeep-scrub
```

   For example,

```
[ceph: root@host01 /]# ceph osd unset noout
[ceph: root@host01 /]# ceph osd unset noscrub
[ceph: root@host01 /]# ceph osd unset nodeep-scrub
```

Replacing the node by reinstalling the operating system and using all new Ceph OSD disks

Reinstall the operating system and use all new Ceph OSD disks to replace the node.

1. Disable backfilling.

```
ceph osd set noout
ceph osd set noscrub
ceph osd set nodeep-scrub
```

   For example,

```
[ceph: root@host01 /]# ceph osd set noout
[ceph: root@host01 /]# ceph osd set noscrub
[ceph: root@host01 /]# ceph osd set nodeep-scrub
```

2. Remove all OSDs on the failed node from the storage cluster.
   For more information, see “Removing a Ceph OSD node” on page 291.

3. Create a backup of the Ceph configuration.

```
cp /etc/ceph/ceph.conf /PATH_TO_BACKUP_LOCATION/ceph.conf
```

   For example,

```
[ceph: root@host01 /]# cp /etc/ceph/ceph.conf /some/backup/location/ceph.conf
```

4. Replace the node and add the Ceph OSD disks from the failed node.

5. Configure disks as JBOD.

> **Note:** This should be done by the storage administrator.

6. Install the operating system.

For more information about operating system requirements, see Operating system requirements. For ore information about installing the operating system, see the Red Hat Enterprise Linux product documentation.

> **Note:** This should be done by the system administrator.

7. Add the new node to the storage cluster using the Ceph Orchestrator commands.
   Ceph daemons are placed automatically on the respective node. For more information, see “Adding a Ceph OSD node” on page 289.
8. Enable backfilling.

```
ceph osd unset noout
ceph osd unset noscrub
ceph osd unset nodeep-scrub
```

For example,

```
[ceph: root@host01 /]# ceph osd unset noout
[ceph: root@host01 /]# ceph osd unset noscrub
[ceph: root@host01 /]# ceph osd unset nodeep-scrub
```

Performance considerations

Understand the performance considerations that affect a storage cluster's performance when adding or removing Ceph OSD nodes.

The following factors typically affect a storage cluster’s performance when adding or removing Ceph OSD nodes:

- Ceph clients place load on the I/O interface to Ceph; that is, the clients place load on a pool. A pool maps to a CRUSH ruleset. The underlying CRUSH hierarchy allows Ceph to place data across failure domains. If the underlying Ceph OSD node involves a pool that is experiencing high client load, the client load could significantly affect recovery time and reduce performance. Because write operations require data replication for durability, write-intensive client loads in particular can increase the time for the storage cluster to recover.
- Generally, the capacity you are adding or removing affects the storage cluster’s time to recover. In addition, the storage density of the node you add or remove might also affect recovery times. For example, a node with 36 OSDs typically takes longer to recover than a node with 12 OSDs.
- When removing nodes, you MUST ensure that you have sufficient spare capacity so that you will not reach `full ratio` or `near full ratio`. If the storage cluster reaches `full ratio`, Ceph will suspend write operations to prevent data loss.
- A Ceph OSD node maps to at least one Ceph CRUSH hierarchy, and the hierarchy maps to at least one pool. Each pool that uses a CRUSH ruleset experiences a performance impact when Ceph OSD nodes are added or removed.
- Replication pools tend to use more network bandwidth to replicate deep copies of the data, whereas erasure coded pools tend to use more CPU to calculate `k+m` coding chunks. The more copies that exist of the data, the longer it takes for the storage cluster to recover. For example, a larger pool or one that has a greater number of `k+m` chunks will take longer to recover than a replication pool with fewer copies of the same data.
- Drives, controllers and network interface cards all have throughput characteristics that might impact the recovery time. Generally, nodes with higher throughput characteristics, such as 10 Gbps and SSDs, recover more quickly than nodes with lower throughput characteristics, such as 1 Gbps and SATA drives.

Recommendations for adding or removing nodes

Understand these recommendations for adding or removing nodes.

Red Hat recommends adding or removing one OSD at a time within a node and allowing the storage cluster to recover before proceeding to the next OSD. This helps to minimize the impact on storage cluster performance. Note that if a node fails, you might need to change the entire node at once, rather than one OSD at a time.

To remove an OSD:

- Using *Removing the OSD daemons*.

To add an OSD:

- Using *Deploying Ceph OSDs on all available devices*.
- Using *Deploying Ceph OSDs using advanced service specification*.
- Using *Deploying Ceph OSDs on specific devices and hosts*.

When adding or removing Ceph OSD nodes, consider that other ongoing processes also affect storage cluster performance. To reduce the impact on client I/O, Red Hat recommends the following:

Calculate capacity

Before removing a Ceph OSD node, ensure that the storage cluster can backfill the contents of all its OSDs without reaching the `full ratio`. Reaching the `full ratio` will cause the storage cluster to refuse write operations.

Temporarily disable scrubbing

Scrubbing is essential to ensuring the durability of the storage cluster's data; however, it is resource intensive. Before adding or removing a Ceph OSD node, disable scrubbing and deep-scrubbing and let the current scrubbing operations complete before proceeding.

```
ceph osd set noscrub
ceph osd set nodeep-scrub
```

Once you have added or removed a Ceph OSD node and the storage cluster has returned to an `active+clean` state, unset the `noscrub` and `nodeep-scrub` settings.

```
ceph osd unset noscrub
ceph osd unset nodeep-scrub
```

Limit backfill and recovery

If you have reasonable data durability, there is nothing wrong with operating in a `degraded` state. For example, you can operate the storage cluster with `osd_pool_default_size = 3` and `osd_pool_default_min_size = 2`. You can tune the storage cluster for the fastest possible recovery time, but doing so significantly affects Ceph client I/O performance. To maintain the highest Ceph client I/O performance, limit the backfill and recovery operations and allow them to take longer.

```
osd_max_backfills = 1
osd_recovery_max_active = 1
osd_recovery_op_priority = 1
```

You can also consider setting the sleep and delay parameters such as, `osd_recovery_sleep`.

Increase the number of placement groups

Finally, if you are expanding the size of the storage cluster, you may need to increase the number of placement groups. If you determine that you need to expand the number of placement groups, Red Hat recommends making incremental increases in the number of placement groups. Increasing the number of placement groups by a significant amount will cause a considerable degradation in performance.

Adding a Ceph OSD node

To expand the capacity of the Red Hat Ceph Storage cluster, you can add an OSD node.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- A provisioned node with a network connection.

Procedure

1. Verify that other nodes in the storage cluster can reach the new node by its short host name.

2. Temporarily disable scrubbing:

   Example

   ```
   [ceph: root@host01 /]# ceph osd set noscrub
   [ceph: root@host01 /]# ceph osd set nodeep-scrub
   ```

3. Limit the backfill and recovery features:

   Syntax

   ```
   ceph tell DAEMON_TYPE.* injectargs --OPTION_NAME VALUE [--OPTION_NAME VALUE]
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph tell osd.* injectargs --osd-max-backfills 1 --osd-
   recovery-max-active 1 --osd-recovery-op-priority 1
   ```

4. Extract the cluster's public SSH keys to a folder:

   Syntax

   ```
   ceph cephadm get-pub-key > ~/PATH
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph cephadm get-pub-key > ~/ceph.pub
   ```

5. Copy ceph cluster's public SSH keys to the root user's `authorized_keys` file on the new host:

   Syntax

   ```
   ssh-copy-id -f -i ~/PATH root@HOST_NAME_2
   ```

   Example

   ```
   [ceph: root@host01 /]# ssh-copy-id -f -i ~/ceph.pub root@host02
   ```

6. Add the new node to the CRUSH map:

   Syntax

   ```
   ceph orch host add NODE_NAME IP_ADDRESS
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch host add host02 10.10.128.70
   ```

7. Add an OSD for each disk on the node to the storage cluster.
   - Using *Deploying Ceph OSDs on all available devices*.
   - Using *Deploying Ceph OSDs using advanced service specification*.
   - Using *Deploying Ceph OSDs on specific devices and hosts*.

Important: When adding an OSD node to a Red Hat Ceph Storage cluster, Red Hat recommends adding one OSD daemon at a time and allowing the cluster to recover to an `active+clean` state before proceeding to the next OSD.

Reference

- Setting a specific configuration setting at runtime.

- Adding a bucket
- Moving a bucket

Removing a Ceph OSD node

To reduce the capacity of a storage cluster, remove an OSD node.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster.
- Root-level access to all nodes in the storage cluster.

> **Warning:** Before removing a Ceph OSD node, ensure that the storage cluster can backfill the contents of all OSDs without reaching the `full ratio`. Reaching the `full ratio` will cause the storage cluster to refuse write operations.

1. Check the storage cluster's capacity.

```
ceph df
rados df
ceph osd df
```

2. Temporarily disable scrubbing.

```
ceph osd set noscrub
ceph osd set nodeep-scrub
```

3. Limit the backfill and recovery features.

```
ceph tell DAEMON_TYPE.* injectargs --OPTION_NAME VALUE [--OPTION_NAME VALUE]
```

For example,

```
[ceph: root@host01 /]# ceph tell osd.* injectargs --osd-max-backfills 1 --osd-
recovery-max-active 1 --osd-recovery-op-priority 1
```

4. Remove each OSD on the node from the storage cluster.

> **Important:** When removing an OSD node from the storage cluster, Red Hat recommends removing one OSD at a time within the node and allowing the cluster to recover to an `active+clean` state before proceeding to remove the next OSD.

5. Verify that the storage cluster is not getting to the `near-full ratio`.

```
ceph -s
ceph df
```

Repeat this step until all OSDs on the node are removed from the storage cluster.
Only continue after verifying that all OSDs are removed.

6. Remove the host.
For more information, see Removing hosts.

Related information

Setting a specific configuration at runtime

Simulating a node failure

To simulate a hard node failure, power off the node and reinstall the operating system.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to all nodes on the storage cluster.

Procedure

1. Check the storage cluster's capacity to understand the impact of removing the node:

   Example

   ```
   [ceph: root@host01 /]# ceph df
   [ceph: root@host01 /]# rados df
   [ceph: root@host01 /]# ceph osd df
   ```

2. Optionally, disable recovery and backfilling:

   Example

   ```
   [ceph: root@host01 /]# ceph osd set noout
   [ceph: root@host01 /]# ceph osd set noscrub
   [ceph: root@host01 /]# ceph osd set nodeep-scrub
   ```

3. Shut down the node.
4. If you are changing the host name, remove the node from CRUSH map:

   Example

   ```
   [ceph: root@host01 /]# ceph osd crush rm host03
   ```

5. Check the status of the storage cluster:

   Example

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

6. Reinstall the operating system on the node.
7. Add the new node:
   - For more information, see Adding hosts.
8. Optionally, enable recovery and backfilling:

   Example

   ```
   [ceph: root@host01 /]# ceph osd unset noout
   [ceph: root@host01 /]# ceph osd unset noscrub
   [ceph: root@host01 /]# ceph osd unset nodeep-scrub
   ```

9. Check Ceph's health:

   Example

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

Configuring SNMP traps

As a storage administrator, you can deploy and configure the simple network management protocol (SNMP) gateway in a Red Hat Ceph Storage cluster to receive alerts from the Prometheus Alertmanager and route them as SNMP traps to the cluster.

Simple network management protocol

Simple network management protocol (SNMP) is one of the most widely used open protocols, to monitor distributed systems and devices across a variety of hardware and software platforms. Ceph's SNMP integration focuses on forwarding alerts from its Prometheus Alertmanager cluster to a gateway daemon. The gateway daemon transforms the alert into an SNMP Notification and sends it on to a designated SNMP management platform. The gateway daemon is from the `snmp_notifier_project`, which provides SNMP V2c and V3 support with authentication and encryption.

The Red Hat Ceph Storage SNMP gateway service deploys one instance of the gateway by default. You can increase this by providing placement information. However, if you enable multiple SNMP gateway daemons, your SNMP management platform receives multiple notifications for the same event.

The SNMP traps are alert messages and the Prometheus Alertmanager sends these alerts to the SNMP notifier which then looks for object identifier (OID) in the given alerts' labels. Each SNMP trap has a unique ID which allows it to send additional traps with updated status to a given SNMP poller. SNMP hooks into the Ceph health checks so that every health warning generates a specific SNMP trap.

In order to work correctly and transfer information on device status to the user to monitor, SNMP relies on several components. There are four main components that makeup SNMP:

SNMP Manager

The SNMP manager, also called a management station, is a computer that runs network monitoring platforms. A platform that has the job of polling SNMP-enabled devices and retrieving data from them. An SNMP Manager queries agents, receives responses from agents and acknowledges asynchronous events from agents.

SNMP Agent

- An SNMP agent is a program that runs on a system to be managed and contains the MIB database for the system. These collect data like bandwidth and disk space, aggregates it, and sends it to the management information base (MIB).

Management information base (MIB)

These are components contained within the SNMP agents. The SNMP manager uses this as a database and asks the agent for access to particular information. This information is needed for the network management systems (NMS). The NMS polls the agent to take information from these files and then proceeds to translate it into graphs and displays that can be viewed by the user. MIBs contain statistical and control values that are determined by the network device.

SNMP Devices

These are devices used for SNMP.

"SNMP supported versions" on page 293 lists the compatible and supported SNMP versions for gateway implementation.

SNMP supported versions

SNMP version	Description
V2c	Uses a community string without any authentication and is vulnerable to outside attacks.
V3 authNoPriv	Uses the username and password authentication without encryption.
V3 authPriv	Uses the username and password authentication with encryption to the SNMP management platform.

Important: When using SNMP traps, ensure that you have the correct security configuration for your version number to minimize the vulnerabilities that are inherent to SNMP and keep your network protected from unauthorized users.

Configuring `snmptrapd`

It is important to configure the simple network management protocol (SNMP) target before deploying the `snmp-gateway` because the `snmptrapd` daemon contains the auth settings that you need to specify when creating the `snmp-gateway` service.

The SNMP gateway feature provides a means of exposing the alerts that are generated in the Prometheus stack to an SNMP management platform. You can configure the SNMP traps to the destination based on the `snmptrapd` tool. This tool allows you to establish one or more SNMP trap listeners.

The following parameters are important for configuration:

- The `engine-id` is a unique identifier for the device, in hex, and required for SNMPV3 gateway. Red Hat recommends using `8000C53F_CLUSTER_FSID_WITHOUT_DASHES`for this parameter.
- The `snmp-community`, which is the *`SNMP_COMMUNITY_FOR_SNMPV2`* parameter, is `public` for SNMPV2c gateway.
- The `auth-protocol` which is the *`AUTH_PROTOCOL`*, is mandatory for SNMPV3 gateway and is `SHA` by default.
- The `privacy-protocol`, which is the *`PRIVACY_PROTOCOL`*, is mandatory for SNMPV3 gateway.
- The *`PRIVACY_PASSWORD`* is mandatory for SNMPV3 gateway with encryption.
- The *`SNMP_V3_AUTH_USER_NAME`* is the user name and is mandatory for SNMPV3 gateway.
- The *`SNMP_V3_AUTH_PASSWORD`* is the password and is mandatory for SNMPV3 gateway.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the nodes.
- Install `firewalld` on Red Hat Enterprise Linux system.

Procedure

1. On the SNMP management host, install the SNMP packages.

   Example

   ```
   [root@host01 ~]# dnf install -y net-snmp-utils net-snmp
   ```

2. Open the port 162 for SNMP to receive alerts.

   Example

   ```
   [root@host01 ~]# firewall-cmd --zone=public --add-port=162/udp
   [root@host01 ~]# firewall-cmd --zone=public --add-port=162/udp --permanent
   ```

3. Implement the management information base (MIB) to make sense of the SNMP notification and enhance SNMP support on the destination host. Copy the raw file from the main repository. https://github.com/ceph/ceph/blob/master/monitoring/snmp/CEPH-MIB.txt

   Example

   ```
   [root@host01 ~]# curl -o CEPH_MIB.txt -L https://raw.githubusercontent.com/ceph/ceph/
   master/monitoring/snmp/CEPH-MIB.txt
   [root@host01 ~]# scp CEPH_MIB.txt root@host02:/usr/share/snmp/mibs
   ```

4. Create the `snmptrapd` directory.

   Example

   ```
   [root@host01 ~]# mkdir /root/snmptrapd/
   ```

5. Create the configuration files in `snmptrapd` directory for each protocol based on the SNMP version.

   Syntax

```
format2 %V\n% Agent Address: %A \n Agent Hostname: %B \n Date: %H - %J - %K - %L - %M
- %Y \n Enterprise OID: %N \n Trap Type: %W \n Trap Sub-Type: %q \n Community/Infosec
Context: %P \n Uptime: %T \n Description: %W \n PDU Attribute/Value Pair Array:\n%v \n
-------------- \n
createuser -e 0x_ENGINE_ID_ SNMPV3_AUTH_USER_NAME AUTH_PROTOCOL SNMP_V3_AUTH_PASSWORD
PRIVACY_PROTOCOL PRIVACY_PASSWORD
authuser log,execute SNMP_V3_AUTH_USER_NAME
authCommunity   log,execute,net SNMP_COMMUNITY_FOR_SNMPV2
```

- For SNMPV2c, create the `snmptrapd_public.conf` file as follows:

  Example

  ```
  format2 %V\n% Agent Address: %A \n Agent Hostname: %B \n Date: %H - %J - %K - %L
  - %M - %Y \n Enterprise OID: %N \n Trap Type: %W \n Trap Sub-Type: %q \n
  Community/Infosec Context: %P \n Uptime: %T \n Description: %W \n PDU Attribute/
  Value Pair Array:\n%v \n -------------- \n

  authCommunity   log,execute,net public
  ```

  The `public` setting here must match the `snmp_community` setting used when deploying the `snmp-gateway` service.

- For SNMPV3 with authentication only, create the `snmptrapd_auth.conf` file.

  Example

  ```
  format2 %V\n% Agent Address: %A \n Agent Hostname: %B \n Date: %H - %J - %K - %L
  - %M - %Y \n Enterprise OID: %N \n Trap Type: %W \n Trap Sub-Type: %q \n
  Community/Infosec Context: %P \n Uptime: %T \n Description: %W \n PDU Attribute/
  Value Pair Array:\n%v \n -------------- \n
  createuser -e 0x8000C53Ff64f341c655d11eb8778fa163e914bcc myuser SHA mypassword
  authuser log,execute myuser
  ```

  The `0x8000C53Ff64f341c655d11eb8778fa163e914bcc` string is the `engine_id`, and `myuser` and `mypassword` are the credentials. The password security is defined by the `SHA` algorithm.

  This corresponds to the settings for deploying the `snmp-gateway` daemon.

  Example

  ```
  snmp_v3_auth_username: myuser
  snmp_v3_auth_password: mypassword
  ```

- For SNMPV3 with authentication and encryption, create the `snmptrapd_authpriv.conf` file.

  Example

  ```
  format2 %V\n% Agent Address: %A \n Agent Hostname: %B \n Date: %H - %J - %K - %L
  - %M - %Y \n Enterprise OID: %N \n Trap Type: %W \n Trap Sub-Type: %q \n
  Community/Infosec Context: %P \n Uptime: %T \n Description: %W \n PDU Attribute/
  Value Pair Array:\n%v \n -------------- \n
  createuser -e 0x8000C53Ff64f341c655d11eb8778fa163e914bcc myuser SHA mypassword
  DES mysecret
  authuser log,execute myuser
  ```

  The `0x8000C53Ff64f341c655d11eb8778fa163e914bcc` string is the `engine_id`, and `myuser` and `mypassword` are the credentials. The password security is defined by the `SHA` algorithm and `DES` is the type of privacy encryption.

  This corresponds to the settings for deploying the `snmp-gateway` daemon.

  Example

  ```
  snmp_v3_auth_username: myuser
  snmp_v3_auth_password: mypassword
  snmp_v3_priv_password: mysecret
  ```

6. Run the daemon on the SNMP management host.

   Syntax

```
/usr/sbin/snmptrapd -M /usr/share/snmp/mibs -m CEPH-MIB.txt -f -C -c /root/snmptrapd/
CONFIGURATION_FILE -Of -Lo :162
```

Example

```
[root@host01 ~]# /usr/sbin/snmptrapd -M /usr/share/snmp/mibs -m CEPH-MIB.txt -f -C
-c /root/snmptrapd/snmptrapd_auth.conf -Of -Lo :162
```

7. If any alert is triggered on the storage cluster, you can monitor the output on the SNMP management host. Verify the SNMP traps and also the traps decoded by MIB.

   Example

```
NET-SNMP version 5.8
 Agent Address: 0.0.0.0
 Agent Hostname: <UNKNOWN>
 Date: 15 - 5 - 12 - 8 - 10 - 4461391
 Enterprise OID: .
 Trap Type: Cold Start
 Trap Sub-Type: 0
 Community/Infosec Context: TRAP2, SNMP v3, user myuser, context
 Uptime: 0
 Description: Cold Start
 PDU Attribute/Value Pair Array:
.iso.org.dod.internet.mgmt.mib-2.1.3.0 = Timeticks: (292276100) 3 days, 19:52:41.00
.iso.org.dod.internet.snmpV2.snmpModules.1.1.4.1.0 =
OID: .iso.org.dod.internet.private.enterprises.ceph.cephCluster.cephNotifications.prom
etheus.promMgr.promMgrPrometheusInactive
.iso.org.dod.internet.private.enterprises.ceph.cephCluster.cephNotifications.prometheu
s.promMgr.promMgrPrometheusInactive.1 = STRING:
"1.3.6.1.4.1.50495.1.2.1.6.2[alertname=CephMgrPrometheusModuleInactive]"
.iso.org.dod.internet.private.enterprises.ceph.cephCluster.cephNotifications.prometheu
s.promMgr.promMgrPrometheusInactive.2 = STRING: "critical"
.iso.org.dod.internet.private.enterprises.ceph.cephCluster.cephNotifications.prometheu
s.promMgr.promMgrPrometheusInactive.3 = STRING: "Status: critical
- Alert: CephMgrPrometheusModuleInactive
  Summary: Ceph's mgr/prometheus module is not available
  Description: The mgr/prometheus module at 10.70.39.243:9283 is unreachable. This
could mean that the module has been disabled or the mgr itself is down.
Without the mgr/prometheus module metrics and alerts will no longer function. Open a
shell to ceph and use 'ceph -s' to determine whether the mgr is active. If the mgr is
not active, restart it, otherwise you can check the mgr/prometheus module is loaded
with 'ceph mgr module ls'  and if it's not listed as enabled, enable it with 'ceph mgr
module enable prometheus'"
```

   In the above example, an alert is generated after the Prometheus module is disabled.

Reference

- *Deploying the SNMP gateway*

Deploying the SNMP gateway

You can deploy the simple network management protocol (SNMP) gateway using either SNMPV2c or SNMPV3.

There are two methods to deploy the SNMP gateway:

1. By creating a credentials file.
2. By creating one service configuration yaml file with all the details.

You can use the following parameters to deploy the SNMP gateway based on the versions:

- The `service_type` is the `snmp-gateway`.
- The `service_name` is any user-defined string.
- The `count` is the number of SNMP gateways to be deployed in a storage cluster.
- The `snmp_destination` parameter must be of the format hostname:port.
- The `engine-id` is a unique identifier for the device, in hex, and required for SNMPV3 gateway. Red Hat recommends to use `8000C53F_CLUSTER_FSID_WITHOUT_DASHES_`for this parameter.

- The `snmp_community` parameter is `public` for SNMPV2c gateway.
- The `auth-protocol` is mandatory for SNMPV3 gateway and is `SHA` by default.
- The `privacy-protocol` is mandatory for SNMPV3 gateway with authentication and encryption.
- The port is `9464` by default.
- You must provide a `-i _FILENAME_` to pass the secrets and passwords to the orchestrator.

Once the SNMP gateway service is deployed or updated, the Prometheus Alertmanager configuration is automatically updated to forward any alert that has an `objectidentifier` to the SNMP gateway daemon for further processing.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the nodes.
- Configuring `snmptrapd` on the destination host, which is the SNMP management host.

Procedure

1. Log into the `cephadm` shell:

   Example

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

2. Create a label for the host on which SNMP gateway needs to be deployed:

   Syntax

   ```
   ceph orch host label add HOSTNAME snmp-gateway
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph orch host label add host02 snmp-gateway
   ```

3. Create a credentials file or a service configuration file based on the SNMP version:
   - For SNMPV2c, create the file as follows:

     Example

     ```
     [ceph: root@host01 /]# cat snmp_creds.yml

     snmp_community: public
     ```

     OR

     Example

     ```
     [ceph: root@host01 /]# cat snmp-gateway.yml

     service_type: snmp-gateway
     service_name: snmp-gateway
     placement:
       count: 1
     spec:
       credentials:
         snmp_community: public
       port: 9464
       snmp_destination: 192.168.122.73:162
       snmp_version: V2c
     ```

   - For SNMPV3 with authentication only, create the file as follows:

     Example

```
[ceph: root@host01 /]# cat snmp_creds.yml

snmp_v3_auth_username: myuser
snmp_v3_auth_password: mypassword
```

OR

Example

```
[ceph: root@host01 /]# cat snmp-gateway.yml

service_type: snmp-gateway
service_name: snmp-gateway
placement:
  count: 1
spec:
  credentials:
    snmp_v3_auth_password: mypassword
    snmp_v3_auth_username: myuser
  engine_id: 8000C53Ff64f341c655d11eb8778fa163e914bcc
  port: 9464
  snmp_destination: 192.168.122.1:162
  snmp_version: V3
```

– For SNMPV3 with authentication and encryption, create the file.

Example

```
[ceph: root@host01 /]# cat snmp_creds.yml

snmp_v3_auth_username: myuser
snmp_v3_auth_password: mypassword
snmp_v3_priv_password: mysecret
```

OR

Example

```
[ceph: root@host01 /]# cat snmp-gateway.yml

service_type: snmp-gateway
service_name: snmp-gateway
placement:
  count: 1
spec:
  credentials:
    snmp_v3_auth_password: mypassword
    snmp_v3_auth_username: myuser
    snmp_v3_priv_password: mysecret
  engine_id: 8000C53Ff64f341c655d11eb8778fa163e914bcc
  port: 9464
  snmp_destination: 192.168.122.1:162
  snmp_version: V3
```

4. Run the `ceph orch` command:

Syntax

```
ceph orch apply snmp-gateway --snmp_version=V2c_OR_V3 --destination=SNMP_DESTINATION
[--port=PORT_NUMBER]\
[--engine-id=8000C53FCLUSTER_FSID_WITHOUT_DASHES] [--auth-protocol=MDS_OR_SHA] [--
privacy_protocol=DES_OR_AES] -i FILENAME
```

OR

Syntax

```
ceph orch apply -i FILENAME.yml
```

– For SNMPV2c, with the `snmp_creds` file, run the `ceph orch` command with the `snmp-version` as `V2c`:

Example

```
[ceph: root@host01 /]# ceph orch apply snmp-gateway --snmp-version=V2c --
destination=192.168.122.73:162 --port=9464 -i snmp_creds.yml
```

- For SNMPV3 with authentication only, with the `snmp_creds` file, run the `ceph orch` command with the `snmp-version` as `V3` and `engine-id`:

  Example

  ```
  [ceph: root@host01 /]# ceph orch apply snmp-gateway --snmp-version=V3 --engine-
  id=8000C53Ff64f341c655d11eb8778fa163e914bcc--destination=192.168.122.73:162 -i
  snmp_creds.yml
  ```

- For SNMPV3 with authentication and encryption, with the `snmp_creds` file, run the `ceph orch` command with the `snmp-version` as `V3`, `privacy-protocol`, and `engine-id`:

  Example

  ```
  [ceph: root@host01 /]# ceph orch apply snmp-gateway --snmp-version=V3 --engine-
  id=8000C53Ff64f341c655d11eb8778fa163e914bcc--destination=192.168.122.73:162 --privacy-
  protocol=AES -i snmp_creds.yml
  ```

  OR

- For all the SNMP versions, with the `snmp-gateway` file, run the following command:

  Example

  ```
  [ceph: root@host01 /]# ceph orch apply -i snmp-gateway.yml
  ```

Reference

- *Configuring `snmptrapd`*

Handling a data center failure

As a storage administrator, you can take preventive measures to avoid a data center failure.

These preventive measures include:

- Configuring the data center infrastructure.
- Setting up failure domains within the CRUSH map hierarchy.
- Designating failure nodes within the domains.

Prerequisites

- A healthy running Red Hat Ceph Storage cluster.
- Root-level access to all nodes in the storage cluster.

Avoiding a data center failure

Use this information to understand how to avoid a data center failure.

Configuring the data center infrastructure

Each data center within a stretch cluster can have a different storage cluster configuration to reflect local capabilities and dependencies. Set up replication between the data centers to help preserve the data. If one data center fails, the other data centers in the storage cluster contain copies of the data.

Setting up failure domains within the CRUSH map hierarchy

Failure, or failover, domains are redundant copies of domains within the storage cluster. If an active domain fails, the failure domain becomes the active domain.

By default, the CRUSH map lists all nodes in a storage cluster within a flat hierarchy. However, for best results, create a logical hierarchical structure within the CRUSH map. The hierarchy designates the domains to which each node belongs and the relationships among those domains within the storage cluster, including the failure

domains. Defining the failure domains for each domain within the hierarchy improves the reliability of the storage cluster.

When planning a storage cluster that contains multiple data centers, place the nodes within the CRUSH map hierarchy so that if one data center goes down, the rest of the storage cluster stays up and running.

Designating failure nodes within the domains

If you plan to use three-way replication for data within the storage cluster, consider the location of the nodes within the failure domain. If an outage occurs within a data center, it is possible that some data might reside in only one copy. When this scenario happens, there are two options:

- Leave the data in read-only status with the standard settings.
- Live with only one copy for the duration of the outage.

With the standard settings, and because of the randomness of data placement across the nodes, not all the data will be affected, but some data can have only one copy and the storage cluster would revert to read-only mode. However, if some data exist in only one copy, the storage cluster reverts to read-only mode.

Handling a data center failure

Red Hat Ceph Storage can withstand catastrophic failures to the infrastructure, such as losing one of the data centers in a stretch cluster. For the standard object store use case, configuring all three data centers can be done independently with replication set up between them. In this scenario, the storage cluster configuration in each of the data centers might be different, reflecting the local capabilities and dependencies.

A logical structure of the placement hierarchy should be considered. A proper CRUSH map can be used, reflecting the hierarchical structure of the failure domains within the infrastructure. Using logical hierarchical definitions improves the reliability of the storage cluster, versus using the standard hierarchical definitions. Failure domains are defined in the CRUSH map. The default CRUSH map contains all nodes in a flat hierarchy. In a three data center environment, such as a stretch cluster, the placement of nodes should be managed in a way that one data center can go down, but the storage cluster stays up and running. Consider which failure domain a node resides in when using 3-way replication for the data.

In the following example, the resulting map is derived from the initial setup of the storage cluster with 6 OSD nodes. In this example, all nodes have only one disk and hence one OSD. All of the nodes are arranged under the default *root*, that is the standard *root* of the hierarchy tree. Because there is a weight assigned to two of the OSDs, these OSDs receive fewer chunks of data than the other OSDs. These nodes were introduced later with bigger disks than the initial OSD disks. This does not affect the data placement to withstand a failure of a group of nodes.

For example:

```
[ceph: root@host01 /]# ceph osd tree
ID WEIGHT  TYPE NAME          UP/DOWN REWEIGHT PRIMARY-AFFINITY
-1 0.33554 root default
-2 0.04779     host host03
 0 0.04779         osd.0           up  1.00000          1.00000
-3 0.04779     host host02
 1 0.04779         osd.1           up  1.00000          1.00000
-4 0.04779     host host01
 2 0.04779         osd.2           up  1.00000          1.00000
-5 0.04779     host host04
 3 0.04779         osd.3           up  1.00000          1.00000
-6 0.07219     host host06
 4 0.07219         osd.4           up  0.79999          1.00000
-7 0.07219     host host05
 5 0.07219         osd.5           up  0.79999          1.00000
```

Using logical hierarchical definitions to group the nodes into same data center can achieve data placement maturity. Possible definition types of *root*, *datacenter*, *rack*, *row*, and *host* allow the reflection of the failure domains for the three data center (DC) stretch cluster:

- Nodes host01 and host02 reside in data center 1 (DC1)
- Nodes host03 and host05 reside in data center 2 (DC2)
- Nodes host04 and host06 reside in data center 3 (DC3)
- All data centers belong to the same structure (allDC)

Since all OSDs in a host belong to the host definition there is no change needed. All the other assignments can be adjusted during runtime of the storage cluster by:

- Defining the *bucket* structure with the following commands:

```
ceph osd crush add-bucket allDC root
ceph osd crush add-bucket DC1 datacenter
ceph osd crush add-bucket DC2 datacenter
ceph osd crush add-bucket DC3 datacenter
```

- Moving the nodes into the appropriate place within this structure by modifying the CRUSH map:

```
ceph osd crush move DC1 root=allDC
ceph osd crush move DC2 root=allDC
ceph osd crush move DC3 root=allDC
ceph osd crush move host01 datacenter=DC1
ceph osd crush move host02 datacenter=DC1
ceph osd crush move host03 datacenter=DC2
ceph osd crush move host05 datacenter=DC2
ceph osd crush move host04 datacenter=DC3
ceph osd crush move host06 datacenter=DC3
```

Within this structure any new hosts can be added too, as well as new disks. By placing the OSDs in the correct place in the hierarchy the CRUSH algorithm is changed to place redundant pieces into different failure domains within the structure. This results in the following:

```
[ceph: root@host01 /]# ceph osd tree
ID  WEIGHT  TYPE NAME              UP/DOWN REWEIGHT PRIMARY-AFFINITY
 -8 6.00000 root allDC
 -9 2.00000     datacenter DC1
 -4 1.00000         host host01
  2 1.00000             osd.2            up  1.00000          1.00000
 -3 1.00000         host host02
  1 1.00000             osd.1            up  1.00000          1.00000
-10 2.00000     datacenter DC2
 -2 1.00000         host host03
  0 1.00000             osd.0            up  1.00000          1.00000
 -7 1.00000         host host05
  5 1.00000             osd.5            up  0.79999          1.00000
-11 2.00000     datacenter DC3
 -6 1.00000         host host06
  4 1.00000             osd.4            up  0.79999          1.00000
 -5 1.00000         host host04
  3 1.00000             osd.3            up  1.00000          1.00000
 -1       0 root default
```

The listing shows the resulting CRUSH map by displaying the osd tree. Easy to see is now how the hosts belong to a data center and all data centers belong to the same top level structure but clearly distinguishing between locations.

> **Note:** Placing the data in the proper locations according to the map works only properly within the healthy cluster. Misplacement might happen under circumstances, when some OSDs are not available. Those misplacements will be corrected automatically once it is possible to do so.

Reference

- CRUSH administration

Certificate management

Manage TLS certificates for Red Hat Ceph Storage services with `cephadm` certificate manager (`certmgr`). Supports self-signed and bring-your-own certificates.

`cephadm` `certmgr` acts as the Root Certificate Authority (CA) for all self-signed certificates generated by `cephadm`. For services that require SSL, admins have the option to either bring their own certificate or allow `cephadm` to generate a self-signed certificate. This ensures secure communication while offering flexibility for deployment preferences.

`cephadm certmgr` automatically detects whether a certificate is self-signed (generated by `cephadm`) or user-provided as an embedded value in the spec, or referenced externally. This distinction determines how it handles expirations and renewals.

Self-signed certificates

With self-signed certificates:

- `certmgr` fully automates renewal, helping to ensure seamless service operation.
- Automation is controlled by configuration parameters defining certificate duration, renewal thresholds, and whether automated rotation is enabled.

User-provided certificates

There are two types of user-provided certificates, inline and reference.

Inline certificates

Certificates are embedded in the service specification. The certificates cannot be modified directly with `certmgr`.

Reference certificates

Certificates are set and managed with `certmgr`. These certificates must be manually uploaded.

For more information about how `certmgr` handles user-provided certificate expiration, see "Certificate health monitoring" on page 303.

Lifecycle configuration

To manage certificate lifecycles, `certmgr` continuously monitors certificates and applies renewal policies based on the certificate type and configured parameters. `cephadm` provides several configuration options to manage certificate lifecycle and renewal.
"certmgr certificate lifecycle configuration options" on page 302 details the different configuration options, value options, and descriptions.

`certmgr` certificate lifecycle configuration options

Configuration	Values	Description

`mgr/cephadm/certificate_automated_rotation_enabled`	**Default**
	`True`
	Additional values
	`False`
		Enabled by default, this configuration option controls whether `cephadm` automatically rotates certificates upon expiration. This helps ensure continuity and security without manual intervention. When disabled `cephadm` will still check periodically the certificates but instead of automatically renewing self-signed expired ones it will issue a health error/warning when an issue is detected.

`mgr/cephadm/certificate_duration_days`	**Default**
	`3 * 365`
	Minimum
	`90`
	Maximum
	`10*365`
		Specifies the duration (in days) of self-signed certificates generated and signed by the `cephadm` root CA. This determines the validity period before renewal is required.

Configuration	Values	Description

`mgr/cephadm/certificate_renewal_threshold_days`	**Default**
	`30`
	Minimum
	`10`
	Maximum
	`90`
		Defines the number of days before a certificate's expiration when `cephadm` should initiate renewal. This ensures timely replacement before expiration occurs. This applies to both self-signed and user-provided certificates. In the case of user-provided certificates, `cephadm` will issue a health error or warning alerting administrators about the upcoming renewal period proximity.

`mgr/cephadm/certificate_check_period`	**Default**
	`1`
	Minimum
	`0`
	Maximum
	`30`
		Specifies how often (in days) the certificate should be checked for validity. This ensures timely detection of any issues related to certificate expiration. Setting this to `0` disables the certificate check functionality.

Certificate health monitoring

`cephadm` continuously monitors the status of all managed certificates, both self-signed and user-provided.

- If a certificate is invalid or has already expired, `cephadm` issues a health error (`CEPHADM_CERT_ERROR`) to alert administrators.
- If a certificate is approaching its expiration date, as determined by the configured `mgr/cephadm/certificate_renewal_threshold_days`, `cephadm` issues a health warning.

Self-signed certificates are automatically renewed by `cephadm` if automation is enabled, but user-provided certificates cannot be renewed automatically. In either case, `cephadm` alerts administrators so they can take timely action. This proactive monitoring helps ensure uninterrupted service operation while giving users control over their certificate policies.

SSL fields in service specifications

Service specifications supporting SSL/TLS can define certain fields, as needed. "Optional SSL configuration fields" on page 303 defines the optional configuration fields and their values.

> **Note:** If you set `ssl` to `true` without additional configuration, the certificate manager issues `cephadm`-signed certificates for the service by default.

Optional SSL configuration fields

Field	Values
`ssl`	Boolean to enable/disable SSL
`ssl_cert`	Use for `inline`: certificate content
`ssl_key`	Use for `inline`: key content
`certificate_source`	One of the following: – `inline` – `reference` – `cephadm_signed`

Note: If `certificate_source` is set to `reference` and the certificate is missing, you must upload both the certificate and the key. Use the names shown in the output of the **`ceph orch certmgr bindings ls`** command, which lists the certificate and key names available for each service.

```
ceph orch certmgr cert set --cert-name CERT_NAME --service-name SERVICE_NAME -i
CERT_FILE
ceph orch certmgr key set --key-name KEY_NAME --service-name SERIVCE_NAME -i KEY_FILE
```

Certificate scopes

`cephadm certmgr` supports three different scopes for certificate management: global, per-host, and per-service.

Certificate scopes

Scope	Description	Example services
Global	Certificates in this scope are shared across all service daemons, regardless of which host they are running on.	`mgmt-gateway` certificate is a globally shared certificate used by all service daemons.
Per-host	Certificates are assigned per host, meaning each host has its own unique certificate. When configuring a custom certificate, the user must specify the host for which the certificate applies.	`grafana` service certificates are configured at the host level and apply specifically to a single machine.
Per-service	Certificates are configured per service name, where each service instance can have its own certificate. When specifying a custom certificate, the user must define the service to which it belongs	An `rgw` service certificate is assigned specifically and only to a Ceph Object Gateway service.

Related information

Using the Ceph Manager alerts module

Manage certificates with `certmgr`

Use `certmgr` to list, check, retrieve, upload, generate, and remove TLS certificates and keys for Red Hat Ceph Storage services.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

- A running Red Hat Ceph Storage cluster with `cephadm`.
- Administrator access to the `cephadm` shell.

List certificates

List all certificates that `certmgr` manages.

- List all certificates, by using the **`certmgr cert ls`** command.

> **Note:** By default, the command omits `cephadm`-signed certificates. To include the `cephadm`-signed certificates, use the **`--include-cephadm-signed`** option.

```
ceph orch certmgr cert ls
```

The **`certmgr cert ls`** command includes the following filtering options:

`--show-details`

The basic command displays an overview of all certificates currently managed by `cephadm`. Use the **`--show-details`** option to include additional information such as issuing authorities and certificate extensions.

`--include-cephadm-signed`

Use this option to include `cephadm`-signed certificates.

`--filter-by`

Filter by name, status, and scope.

```
ceph orch certmgr cert ls [--show-details] [--include-cephadm-signed] [--filter-by
EXPRESSION]
```

For example,

```
[ceph: root@host01 /]# ceph orch certmgr cert ls --include-cephadm-signed
[ceph: root@host01 /]# ceph orch certmgr cert ls --filter-by "status=expired"
[ceph: root@host01 /]# ceph orch certmgr cert ls --filter-by
"scope=service,status=expiring"
[ceph: root@host01 /]# ceph orch certmgr cert ls --include-cephadm-signed --filter-by
"name=rgw*,status=valid"
```

List certificate bindings

Show bindings between services (consumers) and the certificates/keys managed by `certmgr`.

You will reuse these names when retrieving or setting scoped material.

- Display all entities that are associated with managed certificates.

```
ceph orch certmgr bindings ls
```

Check certificate status

Validate integrity and expiration across all managed certificates.

- Check the validity and expiration for certificates that `certmgr` manages.

```
ceph orch certmgr cert check
```

List certificate keys

List private keys managed by `certmgr`.

- List the private keys.

> **Note:** By default, the command omits `cephadm`-signed keys. To include the `cephadm`-signed keys, use the **`--include-cephadm-signed`** option.

```
ceph orch certmgr key ls [--include-cephadm-signed]
```

Retrieve a certificate

Retrieve a certificate for inspection or export.

PREREQUISITE

Before you begin, identify the certificate name and relevant `service_name` or `hostname`. Use these when the certificate scope is per-service or per-host. For more information, see "List certificate bindings" on page 305.

– Get the PEM content of a specific certificate for inspection or export.

```
ceph orch certmgr cert get CERTIFICATE_NAME [--service_name SERVICE_NAME] [--hostname
HOSTNAME] [--no-exception-when-missing]
```

Replace *CERTIFICATE_NAME* with the certificate name.

- Use the **--service_name** argument for certificates with service scope.
- Use the **--hostname** argument for certificates with host scope.

Retrieve a certificate key

Retrieve a private key associated with a specific certificate.

PREREQUISITE

Before you begin, identify the key name and relevant `service_name` or `hostname`. Use these when the certificate scope is per-service or per-host. For more information, see "List certificate bindings" on page 305.

– Get the private key.

```
ceph orch certmgr key get KEY_NAME [--service_name SERVICE_NAME] [--hostname HOSTNAME]
[--no-exception-when-missing]
```

Replace *KEY_NAME* with the certificate key name.

- Use the **--service_name** argument for certificates with service scope.
- Use the **--hostname** argument for certificates with host scope.

Set a certificate-key pair

Set a certificate-key pair to upload or replace an existing certificate-key pair for a certain service.

PREREQUISITE

Before you begin, make sure that you have the following prerequisites in place:

– Identify the certificate name to use as the *SERVICE_TYPE*.

– Identify other relevant fields, as needed, such as the `service_name` or `hostname`.

For more information, see "List certificate bindings" on page 305.

– Set the certificate-key pair for a service.

```
ceph orch certmgr cert-key set SERVICE_TYPE [--cert CERTIFICATE] [--key KEY] [--
service_name SERVICE_NAME] [--hostname HOSTNAME] [-i CERT_KEY_PATH] [--force]
```

Use the **-i** option to specify a file containing a combined certificate and key in PEM format.

> **Note:** When specifying a combined certificate and key be sure that the file contains both the certificate and private key placed together in a single PEM file in sequence.

Set a certificate

Add or replace an existing certificate.

- Set or update the certificate.

```
ceph orch certmgr cert set CERTIFICATE_NAME [--cert CERTIFICATE] [--service_name
SERVICE_NAME] [--hostname HOSTNAME] [-i CERT_KEY_PATH]
```

Set a certificate key

Administrators can provide new private keys for services.

- Set or update the key.

```
ceph orch certmgr key set KEY_NAME [--key KEY] [--service_name SERVICE_NAME] [--
hostname HOSTNAME] [-i CERT_KEY_PATH]
```

Remove a certificate

Use this information to remove an existing certificate.

PREREQUISITE

Removing a certificate requires a valid certificate name. To get the certificate name, use the **`certmgr cert ls`** command. for more information, see “List certificates” on page 304.

- Remove an existing certificate.

```
ceph orch certmgr cert rm CERTIFICATE_NAME [--service_name SERVICE_NAME] [--hostname
HOSTNAME]
```

  - Use the **`--service_name`** argument for certificates with service scope.
  - Use the **`--hostname`** argument for certificates with host scope.

Remove a certificate key

Use this information to remove an existing private keys.

PREREQUISITE

Removing a private key requires a valid key name. To get the key name, use the **`certmgr key ls`** command. for more information, see “List certificate keys” on page 305.

- Remove an existing key.

```
ceph orch certmgr key rm KEY_NAME [--key KEY] [--service_name SERVICE_NAME] [--hostname
HOSTNAME]
```

  - Use the **`--service_name`** argument for certificates with service scope.
  - Use the **`--hostname`** argument for certificates with host scope.

Generate certificates for a module

Automatically provision certificates for a Manager module.

Generating certificates for a specified Manager module is typically used for specified modules that require automatic TLS provisioning, such as the Ceph Dashboard.

- Generate a new certificate.

```
ceph orch certmgr generate-certificates MODULE_NAME
```