Red Hat Ceph Storage 5

File System Guide

Configuring and Mounting Ceph File Systems

Last Updated: 2021-12-13

Red Hat Ceph Storage 5 File System Guide

Configuring and Mounting Ceph File Systems

Legal Notice

Abstract

This guide describes how to configure the Ceph Metadata Server (MDS) and how to create, mount and work the Ceph File System (CephFS). Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

Table of Contents

CHAPTER 1. INTRODUCTION TO THE CEPH FILE SYSTEM

As a storage administrator, you can gain an understanding of the features, system components, and limitations to manage a Ceph File System (CephFS) environment.

1.1. CEPH FILE SYSTEM FEATURES AND ENHANCEMENTS

The Ceph File System (CephFS) is a file system compatible with POSIX standards that is built on top of Ceph's distributed object store, called RADOS (Reliable Autonomic Distributed Object Storage). CephFS provides file access to a Red Hat Ceph Storage cluster, and uses the POSIX semantics wherever possible. For example, in contrast to many other common network file systems like NFS, CephFS maintains strong cache coherency across clients. The goal is for processes using the file system to behave the same when they are on different hosts as when they are on the same host. However, in some cases, CephFS diverges from the strict POSIX semantics.

The Ceph File System has the following features and enhancements:

Scalability

The Ceph File System is highly scalable due to horizontal scaling of metadata servers and direct client reads and writes with individual OSD nodes.

Shared File System

The Ceph File System is a shared file system so multiple clients can work on the same file system at once.

Multiple File Systems

Starting with Red Hat Ceph Storage 5, you can have multiple file systems active on one storage cluster. Each CephFS has its own set of pools and its own set of Metadata Server (MDS) ranks. When deploying multiple file systems this requires more running MDS daemons. This can increase metadata throughput, but also increases operational costs. You can also limit client access to certain file systems.

High Availability

The Ceph File System provides a cluster of Ceph Metadata Servers (MDS). One is active and others are in standby mode. If the active MDS terminates unexpectedly, one of the standby MDS becomes active. As a result, client mounts continue working through a server failure. This behavior makes the Ceph File System highly available. In addition, you can configure multiple active metadata servers.

Configurable File and Directory Layouts

The Ceph File System allows users to configure file and directory layouts to use multiple pools, pool namespaces, and file striping modes across objects.

POSIX Access Control Lists (ACL)

The Ceph File System supports the POSIX Access Control Lists (ACL). ACL are enabled by default with the Ceph File Systems mounted as kernel clients with kernel version **kernel-3.10.0-327.18.2.el7** or newer. To use an ACL with the Ceph File Systems mounted as FUSE clients, you must enable them.

Client Quotas

The Ceph File System supports setting quotas on any directory in a system. The quota can restrict the number of bytes or the number of files stored beneath that point in the directory hierarchy. CephFS client quotas are enabled by default.

Additional Resources

- See the *Management of MDS service using the Ceph Orchestrator* section in the *Operations Guide* to install Ceph Metadata servers.
- See the *Deployment of the Ceph File System* section in the *File System Guide* to create Ceph File Systems.

1.2. CEPH FILE SYSTEM COMPONENTS

The Ceph File System has two primary components:

Clients

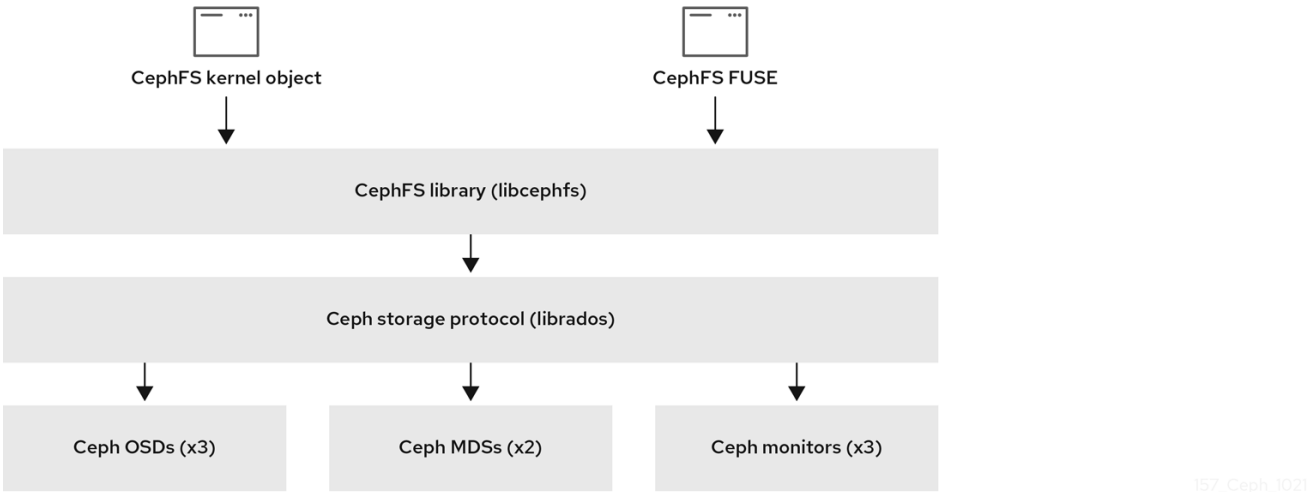
The CephFS clients perform I/O operations on behalf of applications using CephFS, such as, **ceph-fuse** for FUSE clients and **kcephfs** for kernel clients. CephFS clients send metadata requests to an active Metadata Server. In return, the CephFS client learns of the file metadata, and can begin safely caching both metadata and file data.

Metadata Servers (MDS)

The MDS does the following:

- Provides metadata to CephFS clients.
- Manages metadata related to files stored on the Ceph File System.
- Coordinates access to the shared Red Hat Ceph Storage cluster.
- Caches hot metadata to reduce requests to the backing metadata pool store.
- Manages the CephFS clients' caches to maintain cache coherence.
- Replicates hot metadata between active MDS.
- Coalesces metadata mutations to a compact journal with regular flushes to the backing metadata pool.
- CephFS requires at least one Metadata Server daemon (**ceph-mds**) to run.

The diagram below shows the component layers of the Ceph File System.



The bottom layer represents the underlying core storage cluster components:

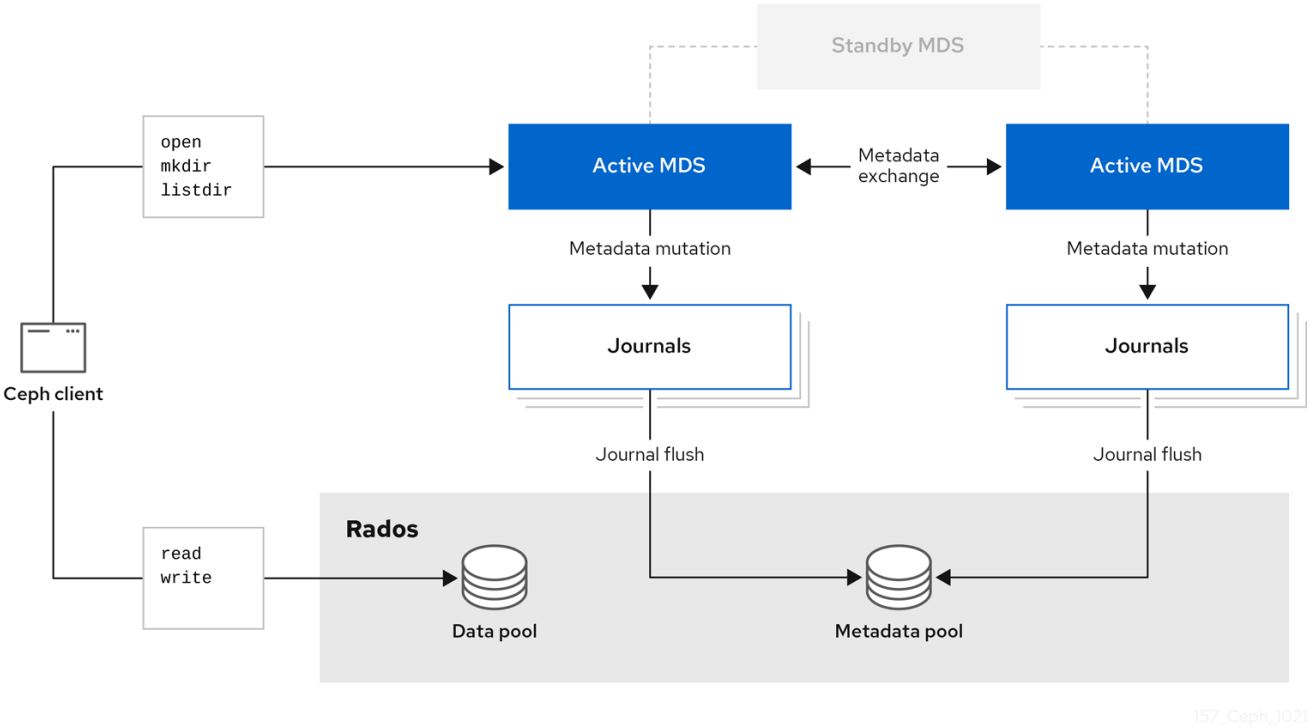
- Ceph OSDs (**ceph-osd**) where the Ceph File System data and metadata are stored.
- Ceph Metadata Servers (**ceph-mds**) that manages Ceph File System metadata.
- Ceph Monitors (**ceph-mon**) that manages the master copy of the cluster map.

The Ceph Storage protocol layer represents the Ceph native **librados** library for interacting with the core storage cluster.

The CephFS library layer includes the CephFS **libcephfs** library that works on top of **librados** and represents the Ceph File System.

The top layer represents two types of Ceph clients that can access the Ceph File Systems.

The diagram below shows more details on how the Ceph File System components interact with each other.



Additional Resources

- See the *Management of MDS service using the Ceph Orchestrator* section in the *File System Guide* to install Ceph Metadata servers.
- See the *Deployment of the Ceph File System* section in the *Red Hat Ceph Storage File System Guide* to create Ceph File Systems.

1.3. CEPH FILE SYSTEM AND SELINUX

Starting with Red Hat Enterprise Linux 8.3 and Red Hat Ceph Storage 4.2, support for using Security-Enhanced Linux (SELinux) on Ceph File Systems (CephFS) environments is available. You can now set any SELinux file type with CephFS, along with assigning a particular SELinux type on individual files. This support applies to the Ceph File System Metadata Server (MDS), the CephFS File System in User Space (FUSE) clients, and the CephFS kernel clients.

Additional Resources

- See the *Using SELinux Guide* on Red Hat Enterprise Linux 8 for more information on SELinux.

1.4. CEPH FILE SYSTEM LIMITATIONS AND THE POSIX STANDARDS

The Ceph File System diverges from the strict POSIX semantics in the following ways:

- If a client's attempt to write a file fails, the write operations are not necessarily atomic. That is, the client might call the **write()** system call on a file opened with the **O_SYNC** flag with an 8MB buffer and then terminates unexpectedly and the write operation can be only partially applied. Almost all file systems, even local file systems, have this behavior.

- In situations when the write operations occur simultaneously, a write operation that exceeds object boundaries is not necessarily atomic. For example, writer *A* writes **"aa|aa"** and writer *B* writes **"bb|bb"** simultaneously, where **"|"** is the object boundary, and **"aa|bb"** is written rather than the proper **"aa|aa"** or **"bb|bb"**.

- POSIX includes the **telldir()** and **seekdir()** system calls that allow you to obtain the current directory offset and seek back to it. Because CephFS can fragment directories at any time, it is difficult to return a stable integer offset for a directory. As such, calling the **seekdir()** system call to a non-zero offset might often work but is not guaranteed to do so. Calling **seekdir()** to offset 0 will always work. This is an equivalent to the **rewinddir()** system call.

- Sparse files propagate incorrectly to the **st_blocks** field of the **stat()** system call. CephFS does not explicitly track parts of a file that are allocated or written to, because the **st_blocks** field is always populated by the quotient of file size divided by block size. This behavior causes utilities, such as **du**, to overestimate used space.

- When the **mmap()** system call maps a file into memory on multiple hosts, write operations are not coherently propagated to caches of other hosts. That is, if a page is cached on host A, and then updated on host B, host A page is not coherently invalidated.

- CephFS clients present a hidden **.snap** directory that is used to access, create, delete, and rename snapshots. Although this directory is excluded from the **readdir()** system call, any process that tries to create a file or directory with the same name returns an error. The name of this hidden directory can be changed at mount time with the **-o snapdirname=.<new_name>** option or by using the **client_snapdir** configuration option.

Additional Resources

- See the *Management of MDS service using the Ceph Orchestrator* section in the *File System Guide* to install Ceph Metadata servers.

- See the *Deployment of the Ceph File System* section in the *Red Hat Ceph Storage File System Guide* to create Ceph File Systems.

1.5. ADDITIONAL RESOURCES

- See the *Management of MDS service using the Ceph Orchestrator* section in the *File System Guide* to install Ceph Metadata servers.

- If you want to use NFS Ganesha as an interface to the Ceph File System with Red Hat OpenStack Platform, see the CephFS with NFS-Ganesha deployment section in the *CephFS via NFS Back End Guide for the Shared File System Service* for instructions on how to deploy such an environment.

CHAPTER 2. THE CEPH FILE SYSTEM METADATA SERVER

As a storage administrator, you can learn about the different states of the Ceph File System (CephFS) Metadata Server (MDS), along with learning about CephFS MDS ranking mechanic, configuring the MDS standby daemon, and cache size limits. Knowing these concepts can enable you to configure the MDS daemons for a storage environment.

2.1. PREREQUISITES

- A running, and healthy Red Hat Ceph Storage cluster.
- Installation of the Ceph Metadata Server daemons (**ceph-mds**).

2.2. METADATA SERVER DAEMON STATES

The Metadata Server (MDS) daemons operate in two states:

- Active – manages metadata for files and directories stores on the Ceph File System.
- Standby – serves as a backup, and becomes active when an active MDS daemon becomes unresponsive.

By default, a Ceph File System uses only one active MDS daemon. However, systems with many clients benefit from multiple active MDS daemons.

You can configure the file system to use multiple active MDS daemons so that you can scale metadata performance for larger workloads. The active MDS daemons dynamically share the metadata workload when metadata load patterns change. Note that systems with multiple active MDS daemons still require standby MDS daemons to remain highly available.

What Happens When the Active MDS Daemon Fails

When the active MDS becomes unresponsive, a Ceph Monitor daemon waits a number of seconds equal to the value specified in the **mds_beacon_grace** option. If the active MDS is still unresponsive after the specified time period has passed, the Ceph Monitor marks the MDS daemon as **laggy**. One of the standby daemons becomes active, depending on the configuration.

NOTE

To change the value of **mds_beacon_grace**, add this option to the Ceph configuration file and specify the new value.

2.3. METADATA SERVER RANKS

Each Ceph File System (CephFS) has a number of ranks, one by default, which starts at zero.

Ranks define the way how the metadata workload is shared between multiple Metadata Server (MDS) daemons. The number of ranks is the maximum number of MDS daemons that can be active at one time. Each MDS daemon handles a subset of the CephFS metadata that is assigned to that rank.

Each MDS daemon initially starts without a rank. The Ceph Monitor assigns a rank to the daemon. The MDS daemon can only hold one rank at a time. Daemons only lose ranks when they are stopped.

The **max_mds** setting controls how many ranks will be created.

The actual number of ranks in the CephFS is only increased if a spare daemon is available to accept the new rank.

Rank States

Ranks can be:

- **Up** - A rank that is assigned to the MDS daemon.
- **Failed** - A rank that is not associated with any MDS daemon.
- **Damaged** - A rank that is damaged; its metadata is corrupted or missing. Damaged ranks are not assigned to any MDS daemons until the operator fixes the problem, and uses the **ceph mds repaired** command on the damaged rank.

2.4. METADATA SERVER CACHE SIZE LIMITS

You can limit the size of the Ceph File System (CephFS) Metadata Server (MDS) cache by:

- **A memory limit** Use the **mds_cache_memory_limit** option. Red Hat recommends a value between 8 GB and 64 GB for **mds_cache_memory_limit**. Setting more cache can cause issues with recovery. This limit is approximately 66% of the desired maximum memory use of the MDS.

IMPORTANT

Red Hat recommends to use memory limits instead of inode count limits.

- **Inode count**: Use the **mds_cache_size** option. By default, limiting the MDS cache by inode count is disabled.

In addition, you can specify a cache reservation by using the **mds_cache_reservation** option for MDS operations. The cache reservation is limited as a percentage of the memory or inode limit and is set to 5% by default. The intent of this parameter is to have the MDS maintain an extra reserve of memory for its cache for new metadata operations to use. As a consequence, the MDS should in general operate below its memory limit because it will recall old state from clients in order to drop unused metadata in its cache.

The **mds_cache_reservation** option replaces the **mds_health_cache_threshold** option in all situations, except when MDS nodes sends a health alert to the Ceph Monitors indicating the cache is too large. By default, **mds_health_cache_threshold** is 150% of the maximum cache size.

Be aware that the cache limit is not a hard limit. Potential bugs in the CephFS client or MDS or misbehaving applications might cause the MDS to exceed its cache size. The **mds_health_cache_threshold** option configures the storage cluster health warning message, so that operators can investigate why the MDS cannot shrink its cache.

Additional Resources

- See the *Metadata Server daemon configuration reference* section in the *Red Hat Ceph Storage File System Guide* for more information.

2.5. FILE SYSTEM AFFINITY

You can configure a Ceph File System (CephFS) to prefer a particular Ceph Metadata Server (MDS) over another Ceph MDS. For example, you have MDS running on newer, faster hardware that you want

to give preference to over a standby MDS running on older, maybe slower hardware. You can specify this preference by setting the **mds_join_fs** option, which enforces this file system affinity. Ceph Monitors give preference to MDS standby daemons with **mds_join_fs** equal to the file system name with the failed rank. The standby-replay daemons are selected before choosing another standby daemons. If no standby daemon exists with the **mds_join_fs** option, then the Ceph Monitors will choose an ordinary standby for replacement or any other available standby as a last resort. The Ceph Monitors will periodically examine Ceph File Systems to see if a standby with a stronger affinity is available to replace the Ceph MDS that has a lower affinity.

Additional Resources

- See the *Configuring file system affinity* section in the *Red Hat Ceph Storage File System Guide* for details.

2.6. MANAGEMENT OF MDS SERVICE USING THE CEPH ORCHESTRATOR

As a storage administrator, you can use 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.

This section covers the following administrative tasks:

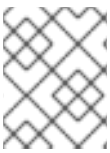
- *Deploying the MDS service using the command line interface .*
- *Deploying the MDS service using the service specification .*
- *Removing the MDS service using the Ceph Orchestrator .*

2.6.1. 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.

2.6.2. Deploying the MDS service using the command line interface

Using the Ceph Orchestrator, you can deploy the Metadata Server (MDS) service using the **placement** specification in the command line interface Ceph File System (CephFS) requires one or more MDS.



NOTE

Ensure you have at least two pools, one for Ceph file system (CephFS) data and one for CephFS metadata.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Hosts are added to the cluster.

- All manager, monitor and OSD daemons are deployed.

Procedure

1. Log into the Cephadm shell:

   Example

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

2. There are two ways of deploying MDS daemons using placement specification:

Method 1

- Use **ceph fs volume** to create the MDS daemons. This creates the CephFS volume, pools associated to the CephFS, and also starts the MDS service on the hosts.

  Syntax

  ```
  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.

  Example

  ```
  [ceph: root@host01 /]# ceph fs volume create test --placement="2 host01 host02"
  ```

Method 2

- Create the pools, CephFS and then deploy MDS service using placement specification:

  a. Create the pools for CephFS:

     Syntax

     ```
     ceph osd pool create DATA_POOL
     ceph osd pool create METADATA_POOL
     ```

     Example

     ```
     [ceph: root@host01 /]# ceph osd pool create cephfs_data
     [ceph: root@host01 /]# ceph osd pool create cephfs_metadata
     ```

  b. Create the file system for the data pools and metadata pools:

     Syntax

     ```
     ceph fs new FILESYSTEM_NAME METADATA_POOL DATA_POOL
     ```

Example

```
[ceph: root@host01 /]# ceph fs new test cephfs_metadata cephfs_data
```

c. Deploy MDS service using the **ceph orch apply** command:

Syntax

```
ceph orch apply mds FILESYSTEM_NAME --placement="NUMBER_OF_DAEMONS HOST_NAME_1 HOST_NAME_2 HOST_NAME_3"
```

Example

```
[ceph: root@host01 /]# ceph orch apply mds test --placement="2 host01 host02"
```

Verification

- List the service:

  Example

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

- Check the CephFS status:

  Example

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

- List the hosts, daemons, and processes:

  Syntax

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ```

  Example

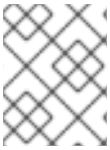
  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=mds
  ```

Additional Resources

- See the *Red Hat Ceph Storage File System guide* for more information on creating the Ceph file system (CephFS).

2.6.3. Deploying the MDS service using the service specification

Using the Ceph Orchestrator, you can deploy the MDS service using the service specification.



NOTE

Ensure you have at least two pools, one for Ceph file system (CephFS) data and one for CephFS metadata.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Hosts are added to the cluster.
- All manager, monitor and OSD daemons are deployed.

Procedure

1. Log into the Cephadm shell:

   Example

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

2. Navigate to the following directory:

   Syntax

   ```
   cd /var/lib/ceph/DAEMON_PATH/
   ```

   Example

   ```
   [ceph: root@host01 mds]# cd /var/lib/ceph/mds/
   ```

   NOTE

   If the directory **mds** does not exist, create the directory.

3. Create the **mds.yml** file:

   Example

   ```
   [ceph: root@host01 mds/]# touch mds.yml
   ```

4. Edit the **mds.yml** file to include the following details:

   Syntax

   ```
   service_type: mds
   service_id: FILESYSTEM_NAME
   placement:
     hosts:
     - HOST_NAME_1
     - HOST_NAME_2
     - HOST_NAME_3
   ```

Example

```
service_type: mds
service_id: fs_name
placement:
  hosts:
  - host01
  - host02
```

5. Deploy MDS service using service specification:

   Syntax

   ```
   ceph orch apply -i FILE_NAME.yml
   ```

   Example

   ```
   [ceph: root@host01 mds]# ceph orch apply -i mds.yml
   ```

6. Once the MDS services is deployed and functional, create the CephFS:

   Syntax

   ```
   ceph fs new CEPHFS_NAME METADATA_POOL DATA_POOL
   ```

   Example

   ```
   [ceph: root@host01 /]# ceph fs new test metadata_pool data_pool
   ```

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=mds
  ```

Additional Resources

- See the *Red Hat Ceph Storage File System guide* for more information on creating the Ceph file system (CephFS).

2.6.4. Removing the MDS service using the Ceph Orchestrator

You can remove the service using the **ceph orch rm** command. Alternatively, you can remove the file system and the associated pools.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to all the nodes.
- Hosts are added to the cluster.
- At least one MDS daemon deployed on the hosts.

Procedure

1. There are two ways of removing MDS daemons from the cluster:

Method 1

- Remove the CephFS volume, associated pools and the services:
  a. Log into the Cephadm shell:

     Example

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

  b. Set the configuration parameter **mon_allow_pool_delete** to **true**:

     Example

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

  c. Remove the file system:

     Syntax

     ```
     ceph fs volume rm FILESYSTEM_NAME --yes-i-really-mean-it
     ```

     Example

     ```
     [ceph: root@host01 /]# ceph fs volume rm cephfs-new --yes-i-really-mean-it
     ```

     This command will remove the file system, its data, and metadata pools. It also tries to remove MDS using the enabled **ceph-mgr** Orchestrator module.

Method 2

- Use the **ceph orch rm** command to remove the MDS service from the entire cluster:
  a. List the service:

Example

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

b. Remove the service

Syntax

```
ceph orch rm SERVICE_NAME
```

Example

```
[ceph: root@host01 /]# ceph orch rm mds.test
```

Verification

- List the hosts, daemons, and processes:

  Syntax

  ```
  ceph orch ps
  ```

  Example

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

Additional Resources

- See *Deploying the MDS service using the command line interface* section in the *Red Hat Ceph Storage Operations Guide* for more information.
- See *Deploying the MDS service using the service specification* section in the *Red Hat Ceph Storage Operations Guide* for more information.

2.7. CONFIGURING FILE SYSTEM AFFINITY

Set the Ceph File System (CephFS) affinity for a particular Ceph Metadata Server (MDS).

Prerequisites

- A healthy, and running Ceph File System.
- Root-level access to a Ceph Monitor node.

Procedure

1. Check the current state of a Ceph File System:

   Example

   ```
   [root@mon ~]# ceph fs dump
   ```

```
dumped fsmap epoch 399
...
Filesystem 'cephfs01' (27)
...
e399
max_mds 1
in      0
up      {0=20384}
failed
damaged
stopped
...
[mds.a{0:20384} state up:active seq 239 addr
[v2:127.0.0.1:6854/966242805,v1:127.0.0.1:6855/966242805]]

Standby daemons:

[mds.b{-1:10420} state up:standby seq 2 addr
[v2:127.0.0.1:6856/2745199145,v1:127.0.0.1:6857/2745199145]]
```

2. Set the file system affinity:

Syntax

```
ceph config set STANDBY_DAEMON mds_join_fs FILE_SYSTEM_NAME
```

Example

```
[root@mon ~]# ceph config set mds.b mds_join_fs cephfs01
```

After a Ceph MDS failover event, the file system favors the standby daemon for which the affinity is set.

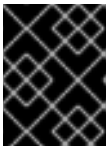
Example

```
[root@mon ~]# ceph fs dump
dumped fsmap epoch 405
e405
...
Filesystem 'cephfs01' (27)
...
max_mds 1
in      0
up      {0=10420}
failed
damaged
stopped
...
[mds.b{0:10420} state up:active seq 274 join_fscid=27 addr
[v2:127.0.0.1:6856/2745199145,v1:127.0.0.1:6857/2745199145]] 1

Standby daemons:
```

```
[mds.a{-1:10720} state up:standby seq 2 addr
[v2:127.0.0.1:6854/1340357658,v1:127.0.0.1:6855/1340357658]]
```

1 The **mds.b** daemon now has the **join_fscid=27** in the file system dump output.



IMPORTANT

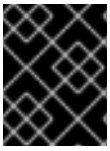
If a file system is in a degraded or undersized state, then no failover will occur to enforce the file system affinity.

Additional Resources

- See the *File system affinity* section in the *Red Hat Ceph Storage File System Guide* for more details.

2.8. CONFIGURING MULTIPLE ACTIVE METADATA SERVER DAEMONS

Configure multiple active Metadata Server (MDS) daemons to scale metadata performance for large systems.



IMPORTANT

Do not convert all standby MDS daemons to active ones. A Ceph File System (CephFS)requires at least one standby MDS daemon to remain highly available.

Prerequisites

- Ceph administration capabilities on the MDS node.

Procedure

1. Set the **max_mds** parameter to the desired number of active MDS daemons:

   Syntax

   ```
   ceph fs set NAME max_mds NUMBER
   ```

   Example

   ```
   [root@mon ~]# ceph fs set cephfs max_mds 2
   ```

   This example increases the number of active MDS daemons to two in the CephFS called **cephfs**

   NOTE

   Ceph only increases the actual number of ranks in the CephFS if a spare MDS daemon is available to take the new rank.

2. Verify the number of active MDS daemons:

Syntax

```
ceph fs status NAME
```

Example

```
[root@mon ~]# ceph fs status cephfs
cephfs - 0 clients
======
+------+--------+-------+---------------+-------+-------+
| Rank | State  |  MDS  |    Activity   |  dns  |  inos |
+------+--------+-------+---------------+-------+-------+
|  0   | active | node1 | Reqs:    0 /s |   10  |   12  |
|  1   | active | node2 | Reqs:    0 /s |   10  |   12  |
+------+--------+-------+---------------+-------+-------+
+-----------------+----------+-------+-------+
|       Pool      |   type   |  used | avail |
+-----------------+----------+-------+-------+
| cephfs_metadata | metadata | 4638  | 26.7G |
|   cephfs_data   |   data   |    0  | 26.7G |
+-----------------+----------+-------+-------+

+-------------+
| Standby MDS |
+-------------+
|    node3    |
+-------------+
```

Additional Resources

- See the *Metadata Server daemons states* section in the *Red Hat Ceph Storage File System Guide* for more details.
- See the *Decreasing the Number of Active MDS Daemons* section in the *Red Hat Ceph Storage File System Guide* for more details.
- See the *Managing Ceph users* section in the *Red Hat Ceph Storage Administration Guide* for more details.

2.9. CONFIGURING THE NUMBER OF STANDBY DAEMONS

Each Ceph File System (CephFS) can specify the required number of standby daemons to be considered healthy. This number also includes the standby-replay daemon waiting for a rank failure.

Prerequisites

- User access to the Ceph Monitor node.

Procedure

1. Set the expected number of standby daemons for a particular CephFS:

   Syntax

```
ceph fs set FS_NAME standby_count_wanted NUMBER
```

NOTE

Setting the *NUMBER* to zero disables the daemon health check.

Example

```
[root@mon]# ceph fs set cephfs standby_count_wanted 2
```

This example sets the expected standby daemon count to two.

2.10. CONFIGURING THE STANDBY-REPLAY METADATA SERVER

Configure each Ceph File System (CephFS) by adding a standby-replay Metadata Server (MDS) daemon. Doing this reduces failover time if the active MDS becomes unavailable.

This specific standby-replay daemon follows the active MDS's metadata journal. The standby-replay daemon is only used by the active MDS of the same rank, and is not available to other ranks.

IMPORTANT

If using standby-replay, then every active MDS must have a standby-replay daemon.

Prerequisites

- User access to the Ceph Monitor node.

Procedure

1. Set the standby-replay for a particular CephFS:

   Syntax

   ```
   ceph fs set FS_NAME allow_standby_replay 1
   ```

   Example

   ```
   [root@mon]# ceph fs set cephfs allow_standby_replay 1
   ```

   In this example, the Boolean value is **1**, which enables the standby-replay daemons to be assigned to the active Ceph MDS daemons.

Additional Resources

- See the *Using the ceph mds fail command* section in the *Red Hat Ceph Storage File System Guide* for details.

2.11. EPHEMERAL PINNING POLICIES

An ephemeral pin is a static partition of subtrees, and can be set with a policy using extended attributes.

A policy can automatically set ephemeral pins to directories. When setting an ephemeral pin to a directory, it is automatically assigned to a particular rank, as to be uniformly distributed across all Ceph MDS ranks. Determining which rank gets assigned is done by a consistent hash and the directory's inode number. Ephemeral pins do not persist when the directory's inode is dropped from file system cache. When failing over a Ceph Metadata Server (MDS), the ephemeral pin is recorded in its journal so the Ceph MDS standby server does not lose this information. There are two types of policies for using ephemeral pins:

Distributed

This policy enforces that all of a directory's immediate children must be ephemerally pinned. For example, use a distributed policy to spread a user's home directory across the entire Ceph File System cluster. Enable this policy by setting the **ceph.dir.pin.distributed** extended attribute.

```
setfattr -n ceph.dir.pin.distributed -v 1 DIRECTORY_PATH
```

Random

This policy enforces a chance that any descendent subdirectory might be ephemerally pinned. You can customize the percent of directories that can be ephemerally pinned. Enable this policy by setting the **ceph.dir.pin.random** and setting a percentage. Red Hat recommends setting this percentage to a value smaller than 1% (**0.01**). Having too many subtree partitions can cause slow performance. You can set the maximum percentage by setting the **mds_export_ephemeral_random_max** Ceph MDS configuration option.

```
setfattr -n ceph.dir.pin.random -v PERCENTAGE DIRECTORY_PATH
```

By default, both of these ephemeral pin policies are disabled. This feature can be enabled by setting either the **mds_export_ephemeral_distributed** or **mds_export_ephemeral_random** Ceph MDS configuration options.

Additional Resources

- See the *Manually pinning directory trees to a particular rank* section in the *Red Hat Ceph Storage File System Guide* for details on manually setting pins.

2.12. MANUALLY PINNING DIRECTORY TREES TO A PARTICULAR RANK

Sometimes it might be desirable to override the dynamic balancer with explicit mappings of metadata to a particular Ceph Metadata Server (MDS) rank. You can do this manually to evenly spread the load of an application or to limit the impact of users' metadata requests on the Ceph File System cluster. Manually pinning directories is also known as an export pin by setting the **ceph.dir.pin** extended attribute.

A directory's export pin is inherited from its closest parent directory, but can be overwritten by setting an export pin on that directory. Setting an export pin on a directory affects all of its sub-directories, for example:

```
[root@client ~]# mkdir -p a/b (1)
[root@client ~]# setfattr -n ceph.dir.pin -v 1 a/ (2)
[root@client ~]# setfattr -n ceph.dir.pin -v 0 a/b (3)
```

(1) Directories **a**/ and **a**/**b** both start without an export pin set.

2 Directories **a/** and **a/b** are now pinned to rank **1**.

3 Directory **a/b** is now pinned to rank **0** and directory **a/** and the rest of its sub-directories are still pinned to rank **1**.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- A running Ceph File System.
- Root-level access to the CephFS client.

Procedure

1. Set the export pin on a directory:

   #### Syntax

   ```
   setfattr -n ceph.dir.pin -v RANK PATH_TO_DIRECTORY
   ```

   #### Example

   ```
   [root@client ~]# setfattr -n ceph.dir.pin -v 2 cephfs/home
   ```

Additional Resources

- See the *Ephemeral pinning policies* section in the *Red Hat Ceph Storage File System Guide* for details automatically setting pins.

2.13. DECREASING THE NUMBER OF ACTIVE METADATA SERVER DAEMONS

How to decrease the number of active Ceph File System (CephFS) Metadata Server (MDS) daemons.

Prerequisites

- The rank that you will remove must be active first, meaning that you must have the same number of MDS daemons as specified by the **max_mds** parameter.

Procedure

1. Set the same number of MDS daemons as specified by the **max_mds** parameter:

   #### Syntax

   ```
   ceph fs status NAME
   ```

   #### Example

   ```
   [root@mon ~]# ceph fs status cephfs
   ```

```
cephfs - 0 clients

+------+--------+-------+--------------+-------+-------+
| Rank | State  |  MDS  |    Activity   |  dns  |  inos |
+------+--------+-------+--------------+-------+-------+
|  0   | active | node1 | Reqs:    0 /s |   10  |   12  |
|  1   | active | node2 | Reqs:    0 /s |   10  |   12  |
+------+--------+-------+--------------+-------+-------+
+-----------------+----------+-------+-------+
|       Pool      |   type   |  used | avail |
+-----------------+----------+-------+-------+
| cephfs_metadata | metadata | 4638  | 26.7G |
|   cephfs_data   |   data   |    0  | 26.7G |
+-----------------+----------+-------+-------+

+-------------+
| Standby MDS |
+-------------+
|    node3    |
+-------------+
```

2. On a node with administration capabilities, change the **max_mds** parameter to the desired number of active MDS daemons:

   Syntax

   ```
   ceph fs set NAME max_mds NUMBER
   ```

   Example

   ```
   [root@mon ~]# ceph fs set cephfs max_mds 1
   ```

3. Wait for the storage cluster to stabilize to the new **max_mds** value by watching the Ceph File System status.

4. Verify the number of active MDS daemons:

   Syntax

   ```
   ceph fs status NAME
   ```

   Example

   ```
   [root@mon ~]# ceph fs status cephfs
   cephfs - 0 clients

   +------+--------+-------+--------------+-------+-------+
   | Rank | State  |  MDS  |    Activity   |  dns  |  inos |
   +------+--------+-------+--------------+-------+-------+
   |  0   | active | node1 | Reqs:    0 /s |   10  |   12  |
   +------+--------+-------+--------------+-------+-------+
   +-----------------+----------+-------+-------+
   |       Pool      |   type   |  used | avail |
   +-----------------+----------+-------+-------+
   ```

```
| cephfs_metadata | metadata | 4638  | 26.7G |
|   cephfs_data   |   data   |    0  | 26.7G |
+-----------------+----------+-------+-------+

+-------------+
| Standby MDS |
+-------------+
|    node3    |
|    node2    |
+-------------+
```

Additional Resources

- See the *Metadata Server daemons states* section in the *Red Hat Ceph Storage File System Guide*.
- See the *Configuring multiple active Metadata Server daemons* section in the *Red Hat Ceph Storage File System Guide*.

2.14. ADDITIONAL RESOURCES

- See the *Red Hat Ceph Storage Installation Guide* for details on installing a Red Hat Ceph Storage cluster.

CHAPTER 3. DEPLOYMENT OF THE CEPH FILE SYSTEM

As a storage administrator, you can deploy Ceph File Systems (CephFS) in a storage environment and have clients mount those Ceph File Systems to meet the storage needs.

Basically, the deployment workflow is three steps:

1. Create Ceph File Systems on a Ceph Monitor node.
2. Create a Ceph client user with the appropriate capabilities, and make the client key available on the node where the Ceph File System will be mounted.
3. Mount CephFS on a dedicated node, using either a kernel client or a File System in User Space (FUSE) client.

3.1. PREREQUISITES

- A running, and healthy Red Hat Ceph Storage cluster.
- Installation and configuration of the Ceph Metadata Server daemon (**ceph-mds**).

3.2. LAYOUT, QUOTA, SNAPSHOT, AND NETWORK RESTRICTIONS

These user capabilities can help you restrict access to a Ceph File System (CephFS) based on the needed requirements.

IMPORTANT

All user capability flags, except **rw**, must be specified in alphabetical order.

Layouts and Quotas

When using layouts or quotas, clients require the **p** flag, in addition to **rw** capabilities. Setting the **p** flag restricts all the attributes being set by special extended attributes, those with a **ceph.** prefix. Also, this restricts other means of setting these fields, such as **openc** operations with layouts.

Example

```
client.0
    key: AQAz7EVWygILFRAAdIcuJ10opU/JKyfFmxhuaw==
    caps: [mds] allow rwp
    caps: [mon] allow r
    caps: [osd] allow rw tag cephfs data=cephfs_a

client.1
    key: AQAz7EVWygILFRAAdIcuJ11opU/JKyfFmxhuaw==
    caps: [mds] allow rw
    caps: [mon] allow r
    caps: [osd] allow rw tag cephfs data=cephfs_a
```

In this example, **client.0** can modify layouts and quotas on the file system **cephfs_a**, but **client.1** cannot.

Snapshots

When creating or deleting snapshots, clients require the **s** flag, in addition to **rw** capabilities. When the capability string also contains the **p** flag, the **s** flag must appear after it.

Example

```
client.0
  key: AQAz7EVWygILFRAAdIcuJ10opU/JKyfFmxhuaw==
  caps: [mds] allow rw, allow rws path=/temp
  caps: [mon] allow r
  caps: [osd] allow rw tag cephfs data=cephfs_a
```

In this example, **client.0** can create or delete snapshots in the **temp** directory of file system **cephfs_a**.

Network

Restricting clients connecting from a particular network.

Example

```
client.0
  key: AQAz7EVWygILFRAAdIcuJ10opU/JKyfFmxhuaw==
  caps: [mds] allow r network 10.0.0.0/8, allow rw path=/bar network 10.0.0.0/8
  caps: [mon] allow r network 10.0.0.0/8
  caps: [osd] allow rw tag cephfs data=cephfs_a network 10.0.0.0/8
```

The optional network and prefix length is in CIDR notation, for example, **10.3.0.0/16**.

Additional Resources

- See the *Creating client users for a Ceph File System* section in the *Red Hat Ceph Storage File System Guide* for details on setting the Ceph user capabilities.

3.3. CREATING CEPH FILE SYSTEMS

You can create multiple Ceph File Systems (CephFS) on a Ceph Monitor node.

Prerequisites

- A running, and healthy Red Hat Ceph Storage cluster.
- Installation and configuration of the Ceph Metadata Server daemon (**ceph-mds**).
- Root-level access to a Ceph Monitor node.
- Root-level access to a Ceph client node.
- On the Ceph client node, installation of the **ceph-fuse** package.

Procedure

1. Create a Ceph File System:

   Syntax

```
ceph fs volume create FILE_SYSTEM_NAME
```

Example

```
[root@mon ~]# ceph fs volume create cephfs01
```

Repeat this step to create additional file systems.

NOTE

By running this command, Ceph automatically creates the new pools, and deploys a new Ceph Metadata Server (MDS) daemon to support the new file system. This also configures the MDS affinity accordingly.

2. Verify access to the new Ceph File System from a Ceph client.

   a. Authorize a Ceph client to access the new file system:

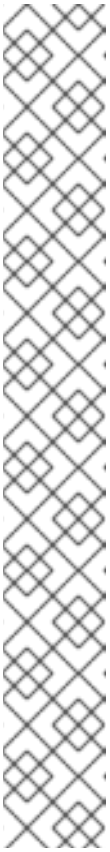
   Syntax

   ```
   ceph fs authorize FILE_SYSTEM_NAME CLIENT_NAME DIRECTORY PERMISSIONS
   ```

   Example

   ```
   [root@mon ~]# ceph fs authorize cephfs01 client.1 / rw
   [client.1]
       key = BQAmthpf81M+JhAAiHDYQkMiCq3x+J0n9e8REK==

   [root@mon ~]# ceph auth get client.1
   exported keyring for client.1
   [client.1]
       key = BQAmthpf81M+JhAAiHDYQkMiCq3x+J0n9e8REK==
       caps mds = "allow rw fsname=cephfs01"
       caps mon = "allow r fsname=cephfs01"
       caps osd = "allow rw tag cephfs data=cephfs01"
   ```



NOTE

Optionally, you can add a safety measure by specifying the **root_squash** option. This prevents accidental deletion scenarios by disallowing clients with a **uid=0** or **gid=0** to do write operations, but still allows read operations.

Example

```
[root@mon ~]# ceph fs authorize cephfs01 client.1 / rw root_squash
/volumes rw
[client.1]
    key = BQAmthpf81M+JhAAiHDYQkMiCq3x+J0n9e8REK==

[root@mon ~]# ceph auth get client.1
[client.1]
    key = BQAmthpf81M+JhAAiHDYQkMiCq3x+J0n9e8REK==
    caps mds = "allow rw fsname=cephfs01 root_squash, allow rw
fsname=cephfs01 path=/volumes"
    caps mon = "allow r fsname=cephfs01"
    caps osd = "allow rw tag cephfs data=cephfs01"
```

In this example, **root_squash** is enabled for the file system **cephfs01**, except within the /**volumes** directory tree.

IMPORTANT

The Ceph client can only see the CephFS it is authorized for.

b. Copy the Ceph user's keyring to the Ceph client node:

Syntax

```
ceph auth get CLIENT_NAME > OUTPUT_FILE_NAME
scp OUTPUT_FILE_NAME TARGET_NODE_NAME:/etc/ceph
```

Example

```
[root@mon ~]# ceph auth get client.1 > ceph.client.1.keyring
exported keyring for client.1
[root@mon ~]# scp ceph.client.1.keyring client:/etc/ceph
root@client's password:
ceph.client.1.keyring                 100%  178   333.0KB/s   00:00
```

c. On the Ceph client node, create a new directory:

Syntax

```
mkdir PATH_TO_NEW_DIRECTORY_NAME
```

Example

```
[root@client ~]# mkdir /mnt/cephfs
```

d. On the Ceph client node, mount the new Ceph File System:

Syntax

```
ceph-fuse PATH_TO_NEW_DIRECTORY_NAME -n CEPH_USER_NAME --client-fs=_FILE_SYSTEM_NAME
```

Example

```
[root@client ~]# ceph-fuse /mnt/cephfs/ -n client.1 --client-fs=cephfs01
ceph-fuse[555001]: starting ceph client
2021-04-07T07:33:27.158+0000 7f11feb81200 -1 init, newargv = 0x55fc4269d5d0
newargc=15
ceph-fuse[555001]: starting fuse
```

e. On the Ceph client node, list the directory contents of the new mount point, or create a file on the new mount point.

Additional Resources

- See the *Creating client users for a Ceph File System* section in the *Red Hat Ceph Storage File System Guide* for more details.
- See the *Mounting the Ceph File System as a kernel client* section in the *Red Hat Ceph Storage File System Guide* for more details.
- See the *Mounting the Ceph File System as a FUSE client* section in the *Red Hat Ceph Storage File System Guide* for more details.
- See the *The Ceph File System* section in the *Red Hat Ceph Storage File System Guide* for more details on the Ceph File System limitations.
- See the *Pools* chapter in the *Red Hat Ceph Storage Storage Strategies Guide* for more details.

3.4. ADDING AN ERASURE-CODED POOL TO A CEPH FILE SYSTEM

By default, Ceph uses replicated pools for data pools. You can also add an additional erasure-coded data pool to the Ceph File System, if needed. Ceph File Systems (CephFS) backed by erasure-coded pools use less overall storage compared to Ceph File Systems backed by replicated pools. While erasure-coded pools use less overall storage, they also use more memory and processor resources than replicated pools.

IMPORTANT

For production environments, Red Hat recommends using the default replicated data pool for CephFS. The creation of inodes in CephFS creates at least one object in the default data pool. It is better to use a replicated pool for the default data to improve small-object write performance, and to improve read performance for updating backtraces.

Prerequisites

- A running Red Hat Ceph Storage cluster.

- An existing Ceph File System.
- Pools using BlueStore OSDs.
- User-level access to a Ceph Monitor node.

Procedure

1. Create an erasure-coded data pool for CephFS:

   Syntax

   ```
   ceph osd pool create DATA_POOL_NAME erasure
   ```

   Example

   ```
   [root@mon ~]# ceph osd pool create cephfs-data-ec01 erasure
   pool 'cephfs-data-ec01' created
   ```

2. Verify the pool was added:

   Example

   ```
   [root@mon ~]# ceph osd lspools
   ```

3. Enable overwrites on the erasure-coded pool:

   Syntax

   ```
   ceph osd pool set DATA_POOL_NAME allow_ec_overwrites true
   ```

   Example

   ```
   [root@mon ~]# ceph osd pool set cephfs-data-ec01 allow_ec_overwrites true
   set pool 15 allow_ec_overwrites to true
   ```

4. Verify the status of the Ceph File System:

   Syntax

   ```
   ceph fs status FILE_SYSTEM_NAME
   ```

   Example

   ```
   [root@mon ~]# ceph fs status cephfs-ec
   cephfs-ec - 14 clients
   =========
   RANK  STATE            MDS              ACTIVITY     DNS    INOS   DIRS   CAPS
    0    active  cephfs-ec.example.ooymyq  Reqs:    0 /s  8231   8233    891    921
          POOL          TYPE     USED  AVAIL
   cephfs-metadata-ec  metadata   787M  8274G
     cephfs-data-ec      data    2360G  12.1T
   ```

```
        STANDBY MDS
cephfs-ec.example.irsrql
cephfs-ec.example.cauuaj
```

5. Add the erasure-coded data pool to the existing CephFS:

   Syntax

   ```
   ceph fs add_data_pool FILE_SYSTEM_NAME DATA_POOL_NAME
   ```

   Example

   ```
   [root@mon ~]# ceph fs add_data_pool cephfs-ec cephfs-data-ec01
   ```

   This example adds the new data pool, **cephfs-data-ec01**, to the existing erasure-coded file system, **cephfs-ec**.

6. Verify that the erasure-coded pool was added to the Ceph File System:

   Syntax

   ```
   ceph fs status FILE_SYSTEM_NAME
   ```

   Example

   ```
   [root@mon ~]# ceph fs status cephfs-ec
   cephfs-ec - 14 clients
   =========
   RANK  STATE            MDS              ACTIVITY     DNS    INOS   DIRS   CAPS
    0    active  cephfs-ec.example.ooymyq  Reqs:    0 /s  8231   8233    891    921
          POOL          TYPE     USED  AVAIL
   cephfs-metadata-ec  metadata   787M  8274G
     cephfs-data-ec      data    2360G  12.1T
    cephfs-data-ec01     data       0   12.1T

          STANDBY MDS
   cephfs-ec.example.irsrql
   cephfs-ec.example.cauuaj
   ```

7. Set the file layout on a new directory:

   Syntax

   ```
   mkdir PATH_TO_DIRECTORY
   setfattr -n ceph.dir.layout.pool -v DATA_POOL_NAME PATH_TO_DIRECTORY
   ```

   Example

   ```
   [root@mon ~]# mkdir /mnt/cephfs/newdir
   [root@mon ~]# setfattr -n ceph.dir.layout.pool -v cephfs-data-ec01 /mnt/cephfs/newdir
   ```

In this example, all new files created in the **/mnt/cephfs/newdir** directory inherits the directory layout and places the data in the newly added erasure-coded pool.

Additional Resources

- See the *The Ceph File System Metadata Server* chapter in the *Red Hat Ceph Storage File System Guide* for more information on the CephFS MDS.
- See the *Creating a Ceph File System* section in the *Red Hat Ceph Storage File System Guide* for more information.
- See the *Erasure-Coded Pools* section in the *Red Hat Ceph Storage Storage Strategies Guide* for more information.
- See the *Erasure Coding with Overwrites* section in the *Red Hat Ceph Storage Storage Strategies Guide* for more information.

3.5. CREATING CLIENT USERS FOR A CEPH FILE SYSTEM

Red Hat Ceph Storage uses **cephx** for authentication, which is enabled by default. To use **cephx** with the Ceph File System, create a user with the correct authorization capabilities on a Ceph Monitor node and make its key available on the node where the Ceph File System will be mounted.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Installation and configuration of the Ceph Metadata Server daemon (ceph-mds).
- Root-level access to a Ceph monitor node.
- Root-level access to a Ceph client node.

Procedure

1. On a Ceph Monitor node, create a client user:

   Syntax

   ```
   ceph fs authorize FILE_SYSTEM_NAME client.CLIENT_NAME /DIRECTORY CAPABILITY [/DIRECTORY CAPABILITY] ...
   ```

   - To restrict the client to only writing in the **temp** directory of filesystem **cephfs_a**:

     Example

     ```
     [root@mon ~]# ceph fs authorize cephfs_a client.1 / r /temp rw

     client.1
       key: AQBSdFhcGZFUDRAAcKhG9Cl2HPiDMMRv4DC43A==
       caps: [mds] allow r, allow rw path=/temp
       caps: [mon] allow r
       caps: [osd] allow rw tag cephfs data=cephfs_a
     ```

- To completely restrict the client to the **temp** directory, remove the root (/) directory:

  Example

  ```
  [root@mon ~]# ceph fs authorize cephfs_a client.1 /temp rw
  ```

NOTE

Supplying **all** or asterisk as the file system name grants access to every file system. Typically, it is necessary to quote the asterisk to protect it from the shell.

2. Verify the created key:

   Syntax

   ```
   ceph auth get client.ID
   ```

   Example

   ```
   [root@mon ~]# ceph auth get client.1
   ```

3. Copy the keyring to the client.

   a. On the Ceph Monitor node, export the keyring to a file:

      Syntax

      ```
      ceph auth get client.ID -o ceph.client.ID.keyring
      ```

      Example

      ```
      [root@mon ~]# ceph auth get client.1 -o ceph.client.1.keyring
      exported keyring for client.1
      ```

   b. Copy the client keyring from the Ceph Monitor node to the **/etc/ceph/** directory on the client node:

      Syntax

      ```
      scp root@MONITOR_NODE_NAME:/root/ceph.client.1.keyring /etc/ceph/
      ```

      Replace_MONITOR_NODE_NAME_with the Ceph Monitor node name or IP.

      Example

      ```
      [root@client ~]# scp root@mon:/root/ceph.client.1.keyring /etc/ceph/ceph.client.1.keyring
      ```

4. Set the appropriate permissions for the keyring file:

   Syntax

```
chmod 644 KEYRING
```

Example

```
[root@client ~]# chmod 644 /etc/ceph/ceph.client.1.keyring
```

Additional Resources

- See the *Ceph user management* chapter in the *Red Hat Ceph Storage Administration Guide* for more details.

3.6. MOUNTING THE CEPH FILE SYSTEM AS A KERNEL CLIENT

You can mount the Ceph File System (CephFS) as a kernel client, either manually or automatically on system boot.

IMPORTANT

Clients running on other Linux distributions, aside from Red Hat Enterprise Linux, are permitted but not supported. If issues are found in the CephFS Metadata Server or other parts of the storage cluster when using these clients, Red Hat will address them. If the cause is found to be on the client side, then the issue will have to be addressed by the kernel vendor of the Linux distribution.

Prerequisites

- Root-level access to a Linux-based client node.
- User-level access to a Ceph Monitor node.
- An existing Ceph File System.

Procedure

1. Configure the client node to use the Ceph storage cluster.
   a. Enable the Red Hat Ceph Storage 5 Tools repository:

   ```
   [root@client ~]# subscription-manager repos --enable=rhceph-5-tools-for-rhel-8-x86_64-rpms
   ```

   b. Install the **ceph-common** package:

   ```
   [root@client ~]# dnf install ceph-common
   ```

   c. Copy the Ceph client keyring from the Ceph Monitor node to the client node:

   Syntax

   ```
   scp root@MONITOR_NODE_NAME:/etc/ceph/KEYRING_FILE /etc/ceph/
   ```

   Replace *MONITOR_NODE_NAME* with the Ceph Monitor host name or IP address.

Example

```
[root@client ~]# scp root@192.168.0.1:/etc/ceph/ceph.client.1.keyring /etc/ceph/
```

d. Copy the Ceph configuration file from a Ceph Monitor node to the client node:

Syntax

```
scp root@MONITOR_NODE_NAME:/etc/ceph/ceph.conf /etc/ceph/ceph.conf
```

Replace *MONITOR_NODE_NAME* with the Ceph Monitor host name or IP address.

Example

```
[root@client ~]# scp root@192.168.0.1:/etc/ceph/ceph.conf /etc/ceph/ceph.conf
```

e. Set the appropriate permissions for the configuration file:

```
[root@client ~]# chmod 644 /etc/ceph/ceph.conf
```

f. Choose either automatically or manually mounting.

Manually Mounting

2. Create a mount directory on the client node:

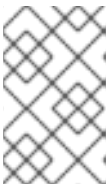
Syntax

```
mkdir -p MOUNT_POINT
```

Example

```
[root@client]# mkdir -p /mnt/cephfs
```

3. Mount the Ceph File System. To specify multiple Ceph Monitor addresses, separate them with commas in the **mount** command, specify the mount point, and set the client name:



NOTE

As of Red Hat Ceph Storage 4.1, **mount.ceph** can read keyring files directly. As such, a secret file is no longer necessary. Just specify the client ID with **name=*CLIENT_ID***, and **mount.ceph** will find the right keyring file.

Syntax

```
mount -t ceph MONITOR-1_NAME:6789,MONITOR-2_NAME:6789,MONITOR-
3_NAME:6789:/ MOUNT_POINT -o name=CLIENT_ID,fs=FILE_SYSTEM_NAME
```

Example

```
[root@client ~]# mount -t ceph mon1:6789,mon2:6789,mon3:6789:/ /mnt/cephfs -o
name=1,fs=cephfs01
```

NOTE

You can configure a DNS server so that a single host name resolves to multiple IP addresses. Then you can use that single host name with the **mount** command, instead of supplying a comma-separated list.

NOTE

You can also replace the Monitor host names with the string **:/** and **mount.ceph** will read the Ceph configuration file to determine which Monitors to connect to.

4. Verify that the file system is successfully mounted:

Syntax

```
stat -f MOUNT_POINT
```

Example

```
[root@client ~]# stat -f /mnt/cephfs
```

Automatically Mounting

2. On the client host, create a new directory for mounting the Ceph File System.

Syntax

```
mkdir -p MOUNT_POINT
```

Example

```
[root@client ~]# mkdir -p /mnt/cephfs
```

3. Edit the **/etc/fstab** file as follows:

Syntax

```
#DEVICE                PATH          TYPE     OPTIONS                DUMP  FSCK
HOST_NAME:PORT,        MOUNT_POINT   ceph     name=CLIENT_ID,        0     0
HOST_NAME:PORT,
ceph.client_mountpoint=/VOL/SUB_VOL_GROUP/SUB_VOL/UID_SUB_VOL,
HOST_NAME:PORT:/                     fs=FILE_SYSTEM_NAME,
                                     [ADDITIONAL_OPTIONS]
```

The **first column** sets the Ceph Monitor host names and the port number.

The **second column** sets the mount point

The **third column** sets the file system type, in this case, **ceph**, for CephFS.

The **fourth column** sets the various options, such as, the user name and the secret file using the **name** and **secretfile** options, respectively. You can also set specific volumes, sub-volume groups, and sub-volumes using the **ceph.client_mountpoint** option.

Set the **_netdev** option to ensure that the file system is mounted after the networking subsystem starts to prevent hanging and networking issues. If you do not need access time information, then setting the **noatime** option can increase performance.

Set the **fifth and sixth columns** to zero.

Example

```
#DEVICE        PATH                TYPE    OPTIONS        DUMP  FSCK
mon1:6789,     /mnt/cephfs          ceph    name=1,          0     0
mon2:6789,
ceph.client_mountpoint=/my_vol/my_sub_vol_group/my_sub_vol/0,
mon3:6789:/                                 fs=cephfs01,
                                  _netdev,noatime
```

The Ceph File System will be mounted on the next system boot.

NOTE

As of Red Hat Ceph Storage 4.1, **mount.ceph** can read keyring files directly. As such, a secret file is no longer necessary. Just specify the client ID with **name=*CLIENT_ID***, and **mount.ceph** will find the right keyring file.

NOTE

You can also replace the Monitor host names with the string **:/** and **mount.ceph** will read the Ceph configuration file to determine which Monitors to connect to.

Additional Resources

- See the **mount(8)** manual page.
- See the *Ceph user management* chapter in the *Red Hat Ceph Storage Administration Guide* for more details on creating a Ceph user.
- See the *Creating a Ceph File System* section of the *Red Hat Ceph Storage File System Guide* for details.

3.7. MOUNTING THE CEPH FILE SYSTEM AS A FUSE CLIENT

You can mount the Ceph File System (CephFS) as a File System in User Space (FUSE) client, either manually or automatically on system boot.

Prerequisites

- Root-level access to a Linux-based client node.
- User-level access to a Ceph Monitor node.

- An existing Ceph File System.

Procedure

1. Configure the client node to use the Ceph storage cluster.
   a. Enable the Red Hat Ceph Storage 5 Tools repository:

```
[root@client ~]# subscription-manager repos --enable=rhceph-5-tools-for-rhel-8-x86_64-rpms
```

   b. Install the **ceph-fuse** package:

```
[root@client ~]# dnf install ceph-fuse
```

   c. Copy the Ceph client keyring from the Ceph Monitor node to the client node:

   Syntax

```
scp root@MONITOR_NODE_NAME:/etc/ceph/KEYRING_FILE /etc/ceph/
```

   Replace *MONITOR_NODE_NAME* with the Ceph Monitor host name or IP address.

   Example

```
[root@client ~]# scp root@192.168.0.1:/etc/ceph/ceph.client.1.keyring /etc/ceph/
```

   d. Copy the Ceph configuration file from a Ceph Monitor node to the client node:

   Syntax

```
scp root@MONITOR_NODE_NAME:/etc/ceph/ceph.conf /etc/ceph/ceph.conf
```

   Replace *MONITOR_NODE_NAME* with the Ceph Monitor host name or IP address.

   Example

```
[root@client ~]# scp root@192.168.0.1:/etc/ceph/ceph.conf /etc/ceph/ceph.conf
```

   e. Set the appropriate permissions for the configuration file:

```
[root@client ~]# chmod 644 /etc/ceph/ceph.conf
```

   f. Choose either automatically or manually mounting.

Manually Mounting

2. On the client node, create a directory for the mount point:

   Syntax

```
mkdir PATH_TO_MOUNT_POINT
```

Example

```
[root@client ~]# mkdir /mnt/mycephfs
```

NOTE

If you used the **path** option with MDS capabilities, then the mount point must be within what is specified by **path**.

3. Use the **ceph-fuse** utility to mount the Ceph File System.

Syntax

```
ceph-fuse -n client.CLIENT_ID --client_fs FILE_SYSTEM_NAME MOUNT_POINT
```

Example

```
[root@client ~]# ceph-fuse -n client.1 --client_fs cephfs01 /mnt/mycephfs
```

NOTE

If you do not use the default name and location of the user keyring, that is **/etc/ceph/ceph.client.*CLIENT_ID*.keyring**, then use the **--keyring** option to specify the path to the user keyring, for example:

Example

```
[root@client ~]# ceph-fuse -n client.1 --keyring=/etc/ceph/client.1.keyring
/mnt/mycephfs
```

NOTE

Use the **-r** option to instruct the client to treat that path as its root:

Syntax

```
ceph-fuse -n client.CLIENT_ID MOUNT_POINT -r PATH
```

Example

```
[root@client ~]# ceph-fuse -n client.1 /mnt/cephfs -r /home/cephfs
```

NOTE

If you want to automatically reconnect an evicted Ceph client, then add the **--client_reconnect_stale=true** option.

Example

```
[root@client ~]# ceph-fuse -n client.1 /mnt/cephfs --
client_reconnect_stale=true
```

4. Verify that the file system is successfully mounted:

Syntax

```
stat -f MOUNT_POINT
```

Example

```
[user@client ~]$ stat -f /mnt/cephfs
```

Automatically Mounting

2. On the client node, create a directory for the mount point:

Syntax

```
mkdir PATH_TO_MOUNT_POINT
```

Example

```
[root@client ~]# mkdir /mnt/mycephfs
```

NOTE

If you used the **path** option with MDS capabilities, then the mount point must be within what is specified by **path**.

3. Edit the **/etc/fstab** file as follows:

Syntax

```
#DEVICE PATH TYPE OPTIONS DUMP FSCK
HOST_NAME:PORT, MOUNT_POINT fuse.ceph ceph.id=CLIENT_ID, 0 0
HOST_NAME:PORT,
ceph.client_mountpoint=/VOL/SUB_VOL_GROUP/SUB_VOL/UID_SUB_VOL,
HOST_NAME:PORT:/ ceph.client_fs=FILE_SYSTEM_NAME,
[ADDITIONAL_OPTIONS]
```

The **first column** sets the Ceph Monitor host names and the port number.

The **second column** sets the mount point

The **third column** sets the file system type, in this case, **fuse.ceph**, for CephFS.

The **fourth column** sets the various options, such as, the user name and the secret file using the **name** and **secretfile** options, respectively. You can also set specific volumes, sub-volume groups, and sub-volumes using the **ceph.client_mountpoint** option. To specify which Ceph File System to access, use the **ceph.client_fs** option. Set the **_netdev** option to ensure that the file system is mounted after the networking subsystem starts to prevent hanging and networking issues. If you do not need access time information, then setting the **noatime** option can increase performance. If you want to automatically reconnect after an eviction, then set the **client_reconnect_stale=true** option.

Set the **fifth and sixth columns** to zero.

Example

```
#DEVICE        PATH            TYPE        OPTIONS        DUMP  FSCK
mon1:6789,     /mnt/cephfs     fuse.ceph   ceph.id=1,       0     0
mon2:6789,
ceph.client_mountpoint=/my_vol/my_sub_vol_group/my_sub_vol/0,
mon3:6789:/                                ceph.client_fs=cephfs01,
                                           _netdev,defaults
```

The Ceph File System will be mounted on the next system boot.

Additional Resources

- The **ceph-fuse(8)** manual page.
- See the *Ceph user management* chapter in the *Red Hat Ceph Storage Administration Guide* for more details on creating a Ceph user.
- See the *Creating a Ceph File System* section of the *Red Hat Ceph Storage File System Guide* for details.

3.8. ADDITIONAL RESOURCES

- See Section 2.6, "Management of MDS service using the Ceph Orchestrator" to install Ceph Metadata servers.
- See Section 3.3, "Creating Ceph File Systems" for details.
- See Section 3.5, "Creating client users for a Ceph File System" for details.
- See Section 3.6, "Mounting the Ceph File System as a kernel client" for details.
- See Section 3.7, "Mounting the Ceph File System as a FUSE client" for details.
- See Chapter 2, *The Ceph File System Metadata Server* for details on configuring the CephFS Metadata Server daemon.

CHAPTER 4. MANAGEMENT OF CEPH FILE SYSTEM VOLUMES, SUB-VOLUME GROUPS, AND SUB-VOLUMES

As a storage administrator, you can use Red Hat's Ceph Container Storage Interface (CSI) to manage Ceph File System (CephFS) exports. This also allows you to use other services, such as OpenStack's file system service (Manila) by having a common command-line interface to interact with. The **volumes** module for the Ceph Manager daemon (**ceph-mgr**) implements the ability to export Ceph File Systems (CephFS).

The Ceph Manager volumes module implements the following file system export abstractions:

- CephFS volumes
- CephFS subvolume groups
- CephFS subvolumes

This chapter describes how to work with:

- Ceph File System volumes
- Ceph File System subvolume groups
- Ceph File System subvolumes

4.1. CEPH FILE SYSTEM VOLUMES

As a storage administrator, you can create, list, and remove Ceph File System (CephFS) volumes. CephFS volumes are an abstraction for Ceph File Systems.

This section describes how to:

- Create a file system volume.
- List file system volume.
- Remove a file system volume.

4.1.1. Creating a file system volume

Ceph Manager's orchestrator module creates a Meta Data Server (MDS) for the Ceph File System (CephFS). This section describes how to create CephFS volume.

NOTE

This creates the Ceph File System, along with the data and metadata pools.

Prerequisites

- A working Red Hat Ceph Storage cluster with Ceph File System deployed.
- At least read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.

Procedure

1. Create a CephFS volume:

   Syntax

   ```
   ceph fs volume create VOLUME_NAME
   ```

   Example

   ```
   [root@mon ~]# ceph fs volume create cephfs
   ```

4.1.2. Listing file system volume

This section describes the step to list the Ceph File system (CephFS) volumes.

Prerequisites

- A working Red Hat Ceph Storage cluster with Ceph File System deployed.
- At least read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- A CephFS volume.

Procedure

1. List the CephFS volume:

   Example

   ```
   [root@mon ~]# ceph fs volume ls
   ```

4.1.3. Removing a file system volume

Ceph Manager's orchestrator module removes the Meta Data Server (MDS) for the Ceph File System (CephFS). This section shows how to remove the Ceph File system (CephFS) volume.

Prerequisites

- A working Red Hat Ceph Storage cluster with Ceph File System deployed.
- At least read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- A CephFS volume.

Procedure

1. If the **mon_allow_pool_delete** option is not set to **true**, then set to **true** before removing the CephFS volume:

Example

```
[root@mon ~]# ceph config set mon mon_allow_pool_delete true
```

2. Remove the CephFS volume:

Syntax

```
ceph fs volume rm VOLUME_NAME [--yes-i-really-mean-it]
```

Example

```
[root@mon ~]# ceph fs volume rm cephfs --yes-i-really-mean-it
```

4.2. CEPH FILE SYSTEM SUBVOLUME GROUPS

As a storage administrator, you can create, list, fetch absolute path, and remove Ceph File System (CephFS) subvolume groups. CephFS subvolume groups are abstractions at a directory level which effects policies, for example, file layouts, across a set of subvolumes.

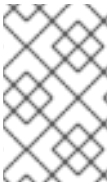
Starting Red Hat Ceph Storage 5.0, the subvolume group snapshot feature is not supported. You can only list and remove the existing snapshots of these subvolume groups.

This section describes how to:

- Create a file system subvolume group.
- List file system subvolume groups.
- Fetch absolute path of a file system subvolume group.
- List snapshots of a file system subvolume group.
- Remove snapshot of a file system subvolume group.
- Remove a file system subvolume group.

4.2.1. Creating a file system subvolume group

This section describes how to create Ceph File system (CephFS) subvolume group.



NOTE

When creating a subvolume group you can specify its data pool layout, uid, gid, and file mode in octal numerals. By default, the subvolume group is created with an octal file mode '755', uid '0', gid '0' and data pool layout of its parent directory.

Prerequisites

- A working Red Hat Ceph Storage cluster with Ceph File System deployed.
- At least read access on the Ceph Monitor.

- Read and write capability on the Ceph Manager nodes.

Procedure

1. Create a CephFS subvolume group:

   Syntax

   ```
   ceph fs subvolumegroup create VOLUME_NAME GROUP_NAME [--pool_layout
   DATA_POOL_NAME --uid UID --gid GID --mode OCTAL_MODE]
   ```

   Example

   ```
   [root@mon ~]# ceph fs subvolumegroup create cephfs subgroup0
   ```

   The command succeeds even if the subvolume group already exists.

4.2.2. Listing file system subvolume groups

This section describes the step to list the Ceph File system (CephFS) subvolume groups.

Prerequisites

- A working Red Hat Ceph Storage cluster with Ceph File System deployed.
- At least read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- A CephFS subvolume group.

Procedure

1. List the CephFS subvolume groups:

   Syntax

   ```
   ceph fs subvolumegroup ls VOLUME_NAME
   ```

   Example

   ```
   [root@mon ~]# ceph fs subvolumegroup ls cephfs
   ```

4.2.3. Fetching absolute path of a file system subvolume group

This section shows how to fetch the absolute path of a Ceph File system (CephFS) subvolume group.

Prerequisites

- A working Red Hat Ceph Storage cluster with Ceph File System deployed.
- At least read access on the Ceph Monitor.

- Read and write capability on the Ceph Manager nodes.
- A CephFS subvolume group.

Procedure

1. Fetch the absolute path of the CephFS subvolume group:

   Syntax

   ```
   ceph fs subvolumegroup getpath VOLUME_NAME GROUP_NAME
   ```

   Example

   ```
   [root@mon ~]# ceph fs subvolumegroup getpath cephfs subgroup0
   ```

4.2.4. Listing snapshots of a file system subvolume group

This section provides the steps to list the snapshots of a Ceph File system (CephFS) subvolume group.

Prerequisites

- A working Red Hat Ceph Storage cluster with Ceph File System deployed.
- At least read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- A CephFS subvolume group.
- Snapshots of the subvolume group.

Procedure

1. List the snapshots of a CephFS subvolume group:

   Syntax

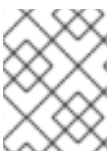
   ```
   ceph fs subvolumegroup snapshot ls VOLUME_NAME GROUP_NAME
   ```

   Example

   ```
   [root@mon ~]# ceph fs subvolumegroup snapshot ls cephfs subgroup0
   ```

4.2.5. Removing snapshot of a file system subvolume group

This section provides the step to remove snapshots of a Ceph File system (CephFS) subvolume group.



NOTE

Using the **--force** flag allows the command to succeed that would otherwise fail if the snapshot did not exist.

Prerequisites

- A working Red Hat Ceph Storage cluster with Ceph File System deployed.
- At least read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- A Ceph File System volume.
- A snapshot of the subvolume group.

Procedure

1. Remove the snapshot of the CephFS subvolume group:

   Syntax

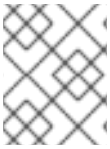
   ```
   ceph fs subvolumegroup snapshot rm VOLUME_NAME GROUP_NAME SNAP_NAME [--force]
   ```

   Example

   ```
   [root@mon ~]# ceph fs subvolumegroup snapshot rm cephfs subgroup0 snap0 --force
   ```

4.2.6. Removing a file system subvolume group

This section shows how to remove the Ceph File system (CephFS) subvolume group.



NOTE

The removal of a subvolume group fails if it is not empty or non-existent. The **--force** flag allows the non-existent subvolume group to be removed.

Prerequisites

- A working Red Hat Ceph Storage cluster with Ceph File System deployed.
- At least read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- A CephFS subvolume group.

Procedure

1. Remove the CephFS subvolume group:

   Syntax

   ```
   ceph fs subvolumegroup rm VOLUME_NAME GROUP_NAME [--force]
   ```

   Example

```
[root@mon ~]# ceph fs subvolumegroup rm cephfs subgroup0 --force
```

4.3. CEPH FILE SYSTEM SUBVOLUMES

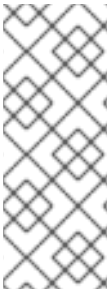
As a storage administrator, you can create, list, fetch absolute path, fetch metadata, and remove Ceph File System (CephFS) subvolumes. Additionally, you can also create, list and remove snapshots of these subvolumes. CephFS subvolumes are an abstraction for independent Ceph File Systems directory trees.

This section describes how to:

- Create a file system subvolume.
- List file system subvolume.
- Resizing a file system subvolume.
- Fetch absolute path of a file system subvolume.
- Fetch metadata of a file system subvolume.
- Create snapshot of a file system subvolume.
- List snapshots of a file system subvolume.
- Fetching metadata of the snapshots of a file system subvolume.
- Remove a file system subvolume.
- Remove snapshot of a file system subvolume.

4.3.1. Creating a file system subvolume

This section describes how to create Ceph File system (CephFS) subvolume.



NOTE

When creating a subvolume you can specify its subvolume group, data pool layout, uid, gid, file mode in octal numerals, and size in bytes. The subvolume can be created in a separate RADOS namespace by specifying `--namespace-isolated` option. By default a subvolume is created within the default subvolume group, and with an octal file mode '755', uid of its subvolume group, gid of its subvolume group, data pool layout of its parent directory and no size limit.

Prerequisites

- A working Red Hat Ceph Storage cluster with Ceph File System deployed.
- At least read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.

Procedure

1. Create a CephFS subvolume:

Syntax

```
ceph fs subvolume create VOLUME_NAME SUBVOLUME_NAME [--size SIZE_IN_BYTES --group_name SUBVOLUME_GROUP_NAME --pool_layout DATA_POOL_NAME --uid _UID --gid GID --mode OCTAL_MODE] [--namespace-isolated]
```

Example

```
[root@mon ~]# ceph fs subvolume create cephfs sub0 --group_name subgroup0 --namespace-isolated
```

The command succeeds even if the subvolume already exists.

4.3.2. Listing file system subvolume

This section describes the step to list the Ceph File system (CephFS) subvolume.

Prerequisites

- A working Red Hat Ceph Storage cluster with Ceph File System deployed.
- At least read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- A CephFS subvolume.

Procedure

1. List the CephFS subvolume:

Syntax

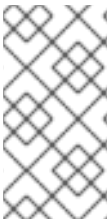
```
ceph fs subvolume ls VOLUME_NAME [--group_name SUBVOLUME_GROUP_NAME]
```

Example

```
[root@mon ~]# ceph fs subvolume ls cephfs --group_name subgroup0
```

4.3.3. Resizing a file system subvolume

This section describes the step to resize the Ceph File system (CephFS) subvolume.



NOTE

The **ceph fs subvolume resize** command resizes the subvolume quota using the size specified by **new_size**. The **--no_shrink** flag prevents the subvolume to shrink below the current used size of the subvolume. The subvolume can be resized to an infinite size by passing **inf** or **infinite** as the **new_size**.

Prerequisites

- A working Red Hat Ceph Storage cluster with Ceph File System deployed.
- At least read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- A CephFS subvolume.

Procedure

1. Resize a CephFS subvolume:

   Syntax

   ```
   ceph fs subvolume resize VOLUME_NAME SUBVOLUME_NAME_ NEW_SIZE [--group_name SUBVOLUME_GROUP_NAME] [--no_shrink]
   ```

   Example

   ```
   [root@mon ~]# ceph fs subvolume resize cephfs sub0 1024000000 --group_name subgroup0 --no_shrink
   ```

4.3.4. Fetching absolute path of a file system subvolume

This section shows how to fetch the absolute path of a Ceph File system (CephFS) subvolume.

Prerequisites

- A working Red Hat Ceph Storage cluster with Ceph File System deployed.
- At least read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- A CephFS subvolume.

Procedure

1. Fetch the absolute path of the CephFS subvolume:

   Syntax

   ```
   ceph fs subvolume getpath VOLUME_NAME SUBVOLUME_NAME [--group_name SUBVOLUME_GROUP_NAME]
   ```

   Example

   ```
   [root@mon ~]# ceph fs subvolume getpath cephfs sub0 --group_name subgroup0
   ```

4.3.5. Fetching metadata of a file system subvolume

This section shows how to fetch metadata of a Ceph File system (CephFS) subvolume.

Prerequisites

- A working Red Hat Ceph Storage cluster with Ceph File System deployed.
- At least read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- A CephFS subvolume.

Procedure

1. Fetch the metadata of a CephFS subvolume:

   Syntax

   ```
   ceph fs subvolume info VOLUME_NAME SUBVOLUME_NAME [--group_name SUBVOLUME_GROUP_NAME]
   ```

   Example

   ```
   [root@mon ~]# ceph fs subvolume info cephfs sub0 --group_name subgroup0
   ```

   Example output

   ```
   {
       "atime": "2020-09-08 09:27:15",
       "bytes_pcent": "undefined",
       "bytes_quota": "infinite",
       "bytes_used": 0,
       "created_at": "2020-09-08 09:27:15",
       "ctime": "2020-09-08 09:27:15",
       "data_pool": "cephfs_data",
       "features": [
           "snapshot-clone",
           "snapshot-autoprotect",
           "snapshot-retention"
       ],
       "gid": 0,
       "mode": 16877,
       "mon_addrs": [
           "10.8.128.22:6789",
           "10.8.128.23:6789",
           "10.8.128.24:6789"
       ],
       "mtime": "2020-09-08 09:27:15",
       "path": "/volumes/subgroup0/sub0/6d01a68a-e981-4ebe-84ca-96b660879173",
       "pool_namespace": "",
       "state": "complete",
       "type": "subvolume",
       "uid": 0
   }
   ```

The output format is a json and contains the following fields:

- **atime**: access time of subvolume path in the format "YYYY-MM-DD HH:MM:SS".
- **mtime**: modification time of subvolume path in the format "YYYY-MM-DD HH:MM:SS".
- **ctime**: change time of subvolume path in the format "YYYY-MM-DD HH:MM:SS".
- **uid**: uid of subvolume path.
- **gid**: gid of subvolume path.
- **mode**: mode of subvolume path.
- **mon_addrs**: list of monitor addresses.
- **bytes_pcent**: quota used in percentage if quota is set, else displays "undefined".
- **bytes_quota**: quota size in bytes if quota is set, else displays "infinite".
- **bytes_used**: current used size of the subvolume in bytes.
- **created_at**: time of creation of subvolume in the format "YYYY-MM-DD HH:MM:SS".
- **data_pool**: data pool the subvolume belongs to.
- **path**: absolute path of a subvolume.
- **type**: subvolume type indicating whether it's clone or subvolume.
- **pool_namespace**: RADOS namespace of the subvolume.
- **features**: features supported by the subvolume, such as , "snapshot-clone", "snapshot-autoprotect", or "snapshot-retention".
- **state**: current state of the subvolume, such as, "complete" or "snapshot-retained"

4.3.6. Creating snapshot of a file system subvolume

This section shows how to create snapshots of Ceph File System (CephFS) subvolume.

Prerequisites

- A working Red Hat Ceph Storage cluster with Ceph File System deployed.
- At least read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- A CephFS subvolume.
- In addition to read (**r**) and write (**w**) capabilities, clients also require **s** flag on a directory path within the file system.

Procedure

1. Verify that the **s** flag is set on the directory:

   Syntax

```
ceph auth get CLIENT_NAME
```

Example

```
client.0
    key: AQAz7EVWygILFRAAdIcuJ12opU/JKyfFmxhuaw==
    caps: [mds] allow rw, allow rws path=/bar (1)
    caps: [mon] allow r
    caps: [osd] allow rw tag cephfs data=cephfs_a (2)
```

(1) (2) In the example, **client.0** can create or delete snapshots in the **bar** directory of file system **cephfs_a**.

2. Create a snapshot of the Ceph File System subvolume:

Syntax

```
ceph fs subvolume snapshot create VOLUME_NAME SUBVOLUME_NAME _SNAP_NAME
[--group_name GROUP_NAME]
```

Example

```
[root@mon ~]# ceph fs subvolume snapshot create cephfs sub0 snap0 --group_name
subgroup0
```

4.3.7. Cloning subvolumes from snapshots

Subvolumes can be created by cloning subvolume snapshots. It is an asynchronous operation involving copying data from a snapshot to a subvolume.

Prerequisites

- A working Red Hat Ceph Storage cluster with Ceph File System deployed.
- At least read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- To create or delete snapshots, in addition to read and write capability, clients require **s** flag on a directory path within the filesystem.

Syntax

```
CLIENT_NAME
    key: AQAz7EVWygILFRAAdIcuJ12opU/JKyfFmxhuaw==
    caps: [mds] allow rw, allow rws path=DIRECTORY_PATH
    caps: [mon] allow r
    caps: [osd] allow rw tag cephfs data=DIRECTORY_NAME
```

In the following example, **client.0** can create or delete snapshots in the **bar** directory of filesystem **cephfs_a**.

Example

```
client.0
    key: AQAz7EVWygILFRAAdIcuJ12opU/JKyfFmxhuaw==
    caps: [mds] allow rw, allow rws path=/bar
    caps: [mon] allow r
    caps: [osd] allow rw tag cephfs data=cephfs_a
```

Procedure

1. Create a Ceph File System (CephFS) volume:

   ### Syntax

   ```
   ceph fs volume create VOLUME_NAME
   ```

   ### Example

   ```
   [root@mon ~]# ceph fs volume create cephfs
   ```

   This creates the CephFS file system, its data and metadata pools.

2. Create a subvolume group. By default, the subvolume group is created with an octal file mode '755', and data pool layout of its parent directory.

   ### Syntax

   ```
   ceph fs subvolumegroup create VOLUME_NAME GROUP_NAME [--pool_layout
   DATA_POOL_NAME --uid UID --gid GID --mode OCTAL_MODE]
   ```

   ### Example

   ```
   [root@mon ~]# ceph fs subvolumegroup create cephfs subgroup0
   ```

3. Create a subvolume. By default, a subvolume is created within the default subvolume group, and with an octal file mode '755', uid of its subvolume group, gid of its subvolume group, data pool layout of its parent directory and no size limit.

   ### Syntax

   ```
   ceph fs subvolume create VOLUME_NAME SUBVOLUME_NAME [--size SIZE_IN_BYTES -
   -group_name SUBVOLUME_GROUP_NAME --pool_layout DATA_POOL_NAME --uid _UID
   --gid GID --mode OCTAL_MODE]
   ```

   ### Example

   ```
   [root@mon ~]# ceph fs subvolume create cephfs sub0 --group_name subgroup0
   ```

4. Create a snapshot of a subvolume:

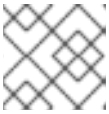
   ### Syntax

```
ceph fs subvolume snapshot create VOLUME_NAME _SUBVOLUME_NAME SNAP_NAME
[--group_name SUBVOLUME_GROUP_NAME]
```

Example

```
[root@mon ~]# ceph fs subvolume snapshot create cephfs sub0 snap0  --group_name
subgroup0
```

5. Initiate a clone operation:



NOTE

By default, cloned subvolumes are created in the default group.

a. If the source subvolume and the target clone are in the default group, run the following command:

Syntax

```
ceph fs subvolume snapshot clone VOLUME_NAME SUBVOLUME_NAME
SNAP_NAME TARGET_SUBVOLUME_NAME
```

Example

```
[root@mon ~]# ceph fs subvolume snapshot clone cephfs sub0 snap0 clone0
```

b. If the source subvolume is in the non-default group, then specify the source subvolume group in the following command:

Syntax

```
ceph fs subvolume snapshot clone VOLUME_NAME SUBVOLUME_NAME
SNAP_NAME TARGET_SUBVOLUME_NAME --group_name
SUBVOLUME_GROUP_NAME
```

Example

```
[root@mon ~]# ceph fs subvolume snapshot clone cephfs sub0 snap0 clone0 --
group_name subgroup0
```

c. If the target clone is to a non-default group, then specify the target group in the following command:

Syntax

```
ceph fs subvolume snapshot clone VOLUME_NAME SUBVOLUME_NAME
SNAP_NAME TARGET_SUBVOLUME_NAME --target_group_name
_SUBVOLUME_GROUP_NAME
```

Example

```
[root@mon ~]# ceph fs subvolume snapshot clone cephfs sub0 snap0 clone0 --
target_group_name subgroup1
```

6. Check the status of the clone operation:

Syntax

```
ceph fs clone status VOLUME_NAME CLONE_NAME [--group_name
TARGET_GROUP_NAME]
```

Example

```
[root@mon ~]# ceph fs clone status cephfs clone0 --group_name subgroup1

{
  "status": {
    "state": "complete"
  }
}
```

Additional Resources

- See the *Managing Ceph Users* section in the *Red Hat Ceph Storage Administration Guide*.

4.3.8. Listing snapshots of a file system subvolume

This section provides the step to list the snapshots of a Ceph File system (CephFS) subvolume.

Prerequisites

- A working Red Hat Ceph Storage cluster with Ceph File System deployed.
- At least read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- A CephFS subvolume.
- Snapshots of the subvolume.

Procedure

1. List the snapshots of a CephFS subvolume:

Syntax

```
ceph fs subvolume snapshot ls VOLUME_NAME SUBVOLUME_NAME [--group_name
SUBVOLUME_GROUP_NAME]
```

Example

```
[root@mon ~]# ceph fs subvolume snapshot ls cephfs sub0 --group_name subgroup0
```

4.3.9. Fetching metadata of the snapshots of a file system subvolume

This section provides the step to fetch the metadata of the snapshots of a Ceph File system (CephFS) subvolume.

Prerequisites

- A working Red Hat Ceph Storage cluster with CephFS deployed.
- At least read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- A CephFS subvolume.
- Snapshots of the subvolume.

Procedure

1. Fetch the metadata of the snapshots of a CephFS subvolume:

   Syntax

   ```
   ceph fs subvolume snapshot info VOLUME_NAME SUBVOLUME_NAME SNAP_NAME --
   group_name SUBVOLUME_GROUP_NAME]
   ```

   Example

   ```
   [root@mon ~]# ceph fs subvolume snapshot info cephfs sub0 snap0 --group_name
   subgroup0
   ```

   Example output

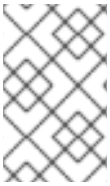
   ```
   {
       "created_at": "2021-09-08 06:18:47.330682",
       "data_pool": "cephfs_data",
       "has_pending_clones": "no",
       "size": 0
   }
   ```

The output format is json and contains the following fields:

- **created_at**: time of creation of snapshot in the format "YYYY-MM-DD HH:MM:SS:ffffff".
- **data_pool**: data pool the snapshot belongs to.
- **has_pending_clones**: "yes" if snapshot clone is in progress otherwise "no".
- **size**: snapshot size in bytes.

4.3.10. Removing a file system subvolume

This section describes the step to remove the Ceph File system (CephFS) subvolume.



NOTE

The **ceph fs subvolume rm** command removes the subvolume and its contents in two steps. First, it moves the subvolume to a trash folder, and then asynchronously purges its contents.

A subvolume can be removed retaining existing snapshots of the subvolume using the **--retain-snapshots** option. If snapshots are retained, the subvolume is considered empty for all operations not involving the retained snapshots. Retained snapshots can be used as a clone source to recreate the subvolume, or cloned to a newer subvolume.

Prerequisites

- A working Red Hat Ceph Storage cluster with Ceph File System deployed.
- At least read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- A CephFS subvolume.

Procedure

1. Remove a CephFS subvolume:

   Syntax

   ```
   ceph fs subvolume rm VOLUME_NAME SUBVOLUME_NAME [--group_name SUBVOLUME_GROUP_NAME] [--force] [--retain-snapshots]
   ```

   Example

   ```
   [root@mon ~]# ceph fs subvolume rm cephfs sub0 --group_name subgroup0 --retain-snapshots
   ```

2. To recreate a subvolume from a retained snapshot:

   Syntax

   ```
   ceph fs subvolume snapshot clone VOLUME_NAME DELETED_SUBVOLUME RETAINED_SNAPSHOT NEW_SUBVOLUME --group_name SUBVOLUME_GROUP_NAME --target_group_name SUBVOLUME_TARGET_GROUP_NAME
   ```

   **NEW_SUBVOLUME* - can either be the same subvolume which was deleted earlier or clone it to a new subvolume.

   Example

   ```
   ceph fs subvolume snapshot clone cephfs sub0 snap0 sub1 --group_name subgroup0 --target_group_name subgroup0
   ```

4.3.11. Removing snapshot of a file system subvolume

This section provides the step to remove snapshots of a Ceph File system (CephFS) subvolume group.

NOTE

Using the **--force** flag allows the command to succeed that would otherwise fail if the snapshot did not exist.

Prerequisites

- A working Red Hat Ceph Storage cluster with Ceph File System deployed.
- At least read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- A Ceph File System volume.
- A snapshot of the subvolume group.

Procedure

1. Remove the snapshot of the CephFS subvolume:

   Syntax

   ```
   ceph fs subvolume snapshot rm VOLUME_NAME SUBVOLUME_NAME _SNAP_NAME [--group_name GROUP_NAME --force]
   ```

   Example

   ```
   [root@mon ~]# ceph fs subvolume snapshot rm cephfs sub0 snap0 --group_name subgroup0 --force
   ```

4.4. ADDITIONAL RESOURCES

- See the *Managing Ceph Users* section in the *Red Hat Ceph Storage Administration Guide*.

CHAPTER 5. CEPH FILE SYSTEM ADMINISTRATION

As a storage administrator, you can perform common Ceph File System (CephFS) administrative tasks, such as:

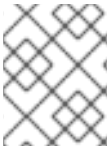
- Monitor CephFS metrics in real-time, see Section 5.2, "Using the **cephfs-top** utility"
- Mapping a directory to a particular MDS rank, see Section 5.6, "Mapping directory trees to Metadata Server daemon ranks".
- Disassociating a directory from a MDS rank, see Section 5.7, "Disassociating directory trees from Metadata Server daemon ranks".
- Adding a new data pool, see Section 5.8, "Adding data pools" .
- Working with quotas, see Section 5.18, "Ceph File System quotas" .
- Working with files and directory layouts, see Section 5.19, "File and directory layouts".
- Removing a Ceph File System, see Section 5.10, "Removing a Ceph File System" .
- Client features, see Section 5.12, "Client features".
- Use the **ceph mds fail** command, see Section 5.11, "Using the **ceph mds fail** command".
- Manually evict a CephFS client, see Section 5.15, "Manually evicting a Ceph File System client"

5.1. PREREQUISITES

- A running, and healthy Red Hat Ceph Storage cluster.
- Installation and configuration of the Ceph Metadata Server daemons (**ceph-mds**).
- Create and mount a Ceph File System.

5.2. USING THE CEPHFS-TOP UTILITY

The Ceph File System (CephFS) provides a **top**-like utility to display metrics on Ceph File Systems in realtime. The **cephfs-top** utility is a **curses**-based Python script that uses the Ceph Manager **stats** module to fetch and displays client performance metrics.



NOTE

Currently, not all of the performance stats are available in the Red Hat Enterprise Linux 8 kernel.

Prerequisites

- A healthy and running Red Hat Ceph Storage cluster.
- Deployment of a Ceph File System.
- Root-level access to a Ceph Monitor node.

Procedure

1. Enable the Red Hat Ceph Storage 5 tools repository, if it is not already enabled:

   Example

   ```
   [root@mon ~]# subscription-manager repos --enable=rhceph-5-tools-for-rhel-8-x86_64-rpms
   ```

2. Install the **cephfs-top** package:

   Example

   ```
   [root@mon ~]# dnf install cephfs-top
   ```

3. Enable the Ceph Manager **stats** plugin:

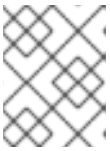
   Example

   ```
   [root@mon ~]# ceph mgr module enable stats
   ```

4. Create the **client.fstop** Ceph user:

   Example

   ```
   [root@mon ~]# ceph auth get-or-create client.fstop mon 'allow r' mds 'allow r' osd 'allow r'
   mgr 'allow r' > /etc/ceph/ceph.client.fstop.keyring
   ```

   NOTE

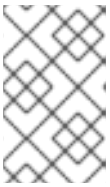
   Optionally, use the **--id** argument to specify a different Ceph user, other than **client.fstop**.

5. Start the **cephfs-top** utility:

   Example

   ```
   [root@mon ~]# cephfs-top
   cephfs-top 0.0.1 - Tue Feb 17 16:54:04 2021
   Client(s): 2 - 1 FUSE, 0 kclient, 1 libcephfs

     CLIENT_ID  MOUNT_POINT  CAP_HIT(%)  READ_LATENCY(s)  WRITE_LATENCY(s)
   METADATA_LATENCY(s)  DENTRY_LEASE(%)  MOUNT_POINT@HOST/ADDR
     1244133    /          100.0       0.0              0.0              0.0                0.0
   /mnt/cephfs@example/192.168.1.4
     1244143    /          100.0       0.0              0.0              0.01               0.0
   N/A@example/192.168.1.4
   ```

   NOTE

   By default, **cephfs-top** connects to the storage cluster name *ceph*. To use a non-default storage cluster name, you can use the **--cluster *NAME*** option with the **cephfs-top** utility.

5.3. USING THE MDS AUTOSCALER MODULE

The MDS Autoscaler Module monitors the Ceph file systems (CephFS) to ensure sufficient MDS daemons are available. It works by adjusting the placement specification for the Orchestrator backend of the MDS service.

The module monitors the following file system settings to inform placement count adjustments:

- **max_mds** file system setting
- **standby_count_wanted** file system setting

The Ceph monitor daemons are still responsible for promoting or stopping MDS according to these settings. The **mds_autoscaler** simply adjusts the number of MDS which are spawned by the orchestrator.

Prerequisites

- A healthy and running Red Hat Ceph Storage cluster.
- Deployment of a Ceph File System.
- Root-level access to a Ceph Monitor node.

Procedure

- Enable the MDS autoscaler module:

  #### Example

  ```
  [ceph: root@host01 /]# ceph mgr module enable mds_autoscaler
  ```

5.4. UNMOUNTING CEPH FILE SYSTEMS MOUNTED AS KERNEL CLIENTS

How to unmount a Ceph File System that is mounted as a kernel client.

Prerequisites

- Root-level access to the node doing the mounting.

Procedure

1. To unmount a Ceph File System mounted as a kernel client:

   #### Syntax

   ```
   umount MOUNT_POINT
   ```

   #### Example

   ```
   [root@client ~]# umount /mnt/cephfs
   ```

Additional Resources

- The **umount(8)** manual page

5.5. UNMOUNTING CEPH FILE SYSTEMS MOUNTED AS FUSE CLIENTS

Unmounting a Ceph File System that is mounted as a File System in User Space (FUSE) client.

Prerequisites

- Root-level access to the FUSE client node.

Procedure

1. To unmount a Ceph File System mounted in FUSE:

   #### Syntax

   ```
   fusermount -u MOUNT_POINT
   ```

   #### Example

   ```
   [root@client ~]# fusermount -u /mnt/cephfs
   ```

Additional Resources

- The **ceph-fuse(8)** manual page

5.6. MAPPING DIRECTORY TREES TO METADATA SERVER DAEMON RANKS

To map a directory and its subdirectories to a particular active Metadata Server (MDS) rank so that its metadata is only managed by the MDS daemon holding that rank. This approach enables you to evenly spread application load or limit impact of users' metadata requests to the entire storage cluster.

IMPORTANT

An internal balancer already dynamically spreads the application load. Therefore, only map directory trees to ranks for certain carefully chosen applications.

In addition, when a directory is mapped to a rank, the balancer cannot split it. Consequently, a large number of operations within the mapped directory can overload the rank and the MDS daemon that manages it.

Prerequisites

- At least two active MDS daemons.
- User access to the CephFS client node.
- Verify that the **attr** package is installed on the CephFS client node with a mounted Ceph File System.

Procedure

1. Add the **p** flag to the Ceph user's capabilities:

   Syntax

   ```
   ceph fs authorize FILE_SYSTEM_NAME client.CLIENT_NAME /DIRECTORY CAPABILITY
   [/DIRECTORY CAPABILITY] ...
   ```

   Example

   ```
   [user@client ~]$ ceph fs authorize cephfs_a client.1 /temp rwp

   client.1
     key: AQBSdFhcGZFUDRAAcKhG9Cl2HPiDMMRv4DC43A==
     caps: [mds] allow r, allow rwp path=/temp
     caps: [mon] allow r
     caps: [osd] allow rw tag cephfs data=cephfs_a
   ```

2. Set the **ceph.dir.pin** extended attribute on a directory:

   Syntax

   ```
   setfattr -n ceph.dir.pin -v RANK DIRECTORY
   ```

   Example

   ```
   [user@client ~]$ setfattr -n ceph.dir.pin -v 2 /temp
   ```

   This example assigns the **/temp** directory and all of its subdirectories to rank 2.

Additional Resources

- See the *Layout, quota, snapshot, and network restrictions* section in the *Red Hat Ceph Storage File System Guide* for more details about the **p** flag.
- See the *Manually pinning directory trees to a particular rank* section in the *Red Hat Ceph Storage File System Guide* for more details.
- See the *Configuring multiple active Metadata Server daemons* section in the *Red Hat Ceph Storage File System Guide* for more details.

5.7. DISASSOCIATING DIRECTORY TREES FROM METADATA SERVER DAEMON RANKS

Disassociate a directory from a particular active Metadata Server (MDS) rank.

Prerequisites

- User access to the Ceph File System (CephFS) client node.
- Ensure that the **attr** package is installed on the client node with a mounted CephFS.

Procedure

- Set the **ceph.dir.pin** extended attribute to -1 on a directory:

  Syntax

  ```
  setfattr -n ceph.dir.pin -v -1 DIRECTORY
  ```

  Example

  ```
  [user@client ~]$ setfattr -n ceph.dir.pin -v -1 /home/ceph-user
  ```

  NOTE

  Any separately mapped subdirectories of **/home/ceph-user/** are not affected.

Additional Resources

- See the *Mapping Directory Trees to MDS Ranks* section in *Red Hat Ceph Storage File System Guide* for more details.

5.8. ADDING DATA POOLS

The Ceph File System (CephFS) supports adding more than one pool to be used for storing data. This can be useful for:

- Storing log data on reduced redundancy pools
- Storing user home directories on an SSD or NVMe pool
- Basic data segregation.

Before using another data pool in the Ceph File System, you must add it as described in this section.

By default, for storing file data, CephFS uses the initial data pool that was specified during its creation. To use a secondary data pool, you must also configure a part of the file system hierarchy to store file data in that pool or optionally within a namespace of that pool, using file and directory layouts.

Prerequisites

- Root-level access to the Ceph Monitor node.

Procedure

1. Create a new data pool:

   Syntax

   ```
   ceph osd pool create POOL_NAME
   ```

   Replace:

   - ***POOL_NAME*** with the name of the pool.

Example

```
[root@mon ~]# ceph osd pool create cephfs_data_ssd
pool 'cephfs_data_ssd' created
```

2. Add the newly created pool under the control of the Metadata Servers:

   Syntax

   ```
   ceph fs add_data_pool FS_NAME POOL_NAME
   ```

   Replace:

   - ***FS_NAME*** with the name of the file system.
   - ***POOL_NAME*** with the name of the pool.

   Example:

   ```
   [root@mon ~]# ceph fs add_data_pool cephfs cephfs_data_ssd
   added data pool 6 to fsmap
   ```

3. Verify that the pool was successfully added:

   Example

   ```
   [root@mon ~]# ceph fs ls
   name: cephfs, metadata pool: cephfs_metadata, data pools: [cephfs_data cephfs_data_ssd]
   ```

4. If you use the **cephx** authentication, make sure that clients can access the new pool.

Additional Resources

- See the *File and directory layouts* for details.
- See the *Creating client users for a Ceph File System* for details.

5.9. TAKING DOWN A CEPH FILE SYSTEM CLUSTER

You can take down Ceph File System (CephFS) cluster by simply setting the **down** flag **true**. Doing this gracefully shuts down the Metadata Server (MDS) daemons by flushing journals to the metadata pool and all client I/O is stopped.

You can also take the CephFS cluster down quickly for testing the deletion of a file system and bring the Metadata Server (MDS) daemons down, for example, practicing a disaster recovery scenario. Doing this sets the **jointable** flag to prevent the MDS standby daemons from activating the file system.

Prerequisites

- User access to the Ceph Monitor node.

Procedure

1. To mark the CephFS cluster down:

   Syntax

   ```
   ceph fs set FS_NAME down true
   ```

   Example

   ```
   [root@mon]# ceph fs set cephfs down true
   ```

   a. To bring the CephFS cluster back up:

      Syntax

      ```
      ceph fs set FS_NAME down false
      ```

      Example

      ```
      [root@mon]# ceph fs set cephfs down false
      ```

or

1. To quickly take down a CephFS cluster:

   Syntax

   ```
   ceph fs fail FS_NAME
   ```

   Exmaple

   ```
   [root@mon]# ceph fs fail cephfs
   ```

NOTE

To get the CephFS cluster back up, set **cephfs** to **joinable**:

Syntax

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

Example

```
[root@mon]# ceph fs set cephfs joinable true
cephfs marked joinable; MDS may join as newly active.
```

5.10. REMOVING A CEPH FILE SYSTEM

You can remove a Ceph File System (CephFS). Before doing so, consider backing up all the data and verifying that all clients have unmounted the file system locally.

WARNING

This operation is destructive and will make the data stored on the Ceph File System permanently inaccessible.

Prerequisites

- Back up your data.
- Root-level access to a Ceph Monitor node.

Procedure

1. Mark the storage cluster as down:

   Syntax

   ```
   ceph fs set FS_NAME down true
   ```

   Replace

   - *FS_NAME* with the name of the Ceph File System you want to remove.

   Example

   ```
   [root@mon]# ceph fs set cephfs down true
   marked down
   ```

2. Display the status of the Ceph File System:

   ```
   ceph fs status
   ```

   Example

   ```
   [root@mon]# ceph fs status
   cephfs - 0 clients
   ======
   +------------------+----------+-------+-------+
   |       Pool       |   type   |  used | avail |
   +-----------------+-----------+-------+-------+
   |cephfs.cephfs.meta | metadata | 31.5M |  52.6G|
   |cephfs.cephfs.data |   data   |    0  |  52.6G|
   +-----------------+----------+-------+---------+
                STANDBY MDS
   cephfs.ceph-host01
   cephfs.ceph-host02
   cephfs.ceph-host03
   ```

3. Remove the Ceph File System:

   Syntax

   ```
   ceph fs rm FS_NAME --yes-i-really-mean-it
   ```

   Replace

   - *FS_NAME* with the name of the Ceph File System you want to remove.

   Example

   ```
   [root@mon]# ceph fs rm cephfs --yes-i-really-mean-it
   ```

4. Verify that the file system has been successfully removed:

   ```
   [root@mon]# ceph fs ls
   ```

5. Optional. Remove data and metadata pools associated with the removed file system.

Additional Resources

- See the *Delete a pool* section in the *Red Hat Ceph Storage Storage Strategies Guide*.

5.11. USING THE CEPH MDS FAIL COMMAND

Use the **ceph mds fail** command to:

- Mark an MDS daemon as failed. If the daemon was active and a suitable standby daemon was available, and if the standby daemon was active after disabling the **standby-replay** configuration, using this command forces a failover to the standby daemon. By disabling the **standby-replay** daemon, this prevents new **standby-replay** daemons from being assigned.
- Restart a running MDS daemon. If the daemon was active and a suitable standby daemon was available, the "failed" daemon becomes a standby daemon.

Prerequisites

- Installation and configuration of the Ceph MDS daemons.

Procedure

1. To fail a daemon:

   Syntax

   ```
   ceph mds fail MDS_NAME
   ```

   Where *MDS_NAME* is the name of the **standby-replay** MDS node.

   Example

```
[root@mds ~]# ceph mds fail example01
```

NOTE

You can find the Ceph MDS name from the **ceph fs status** command.

Additional Resources

- See the *Decreasing the number of active Metadata Server daemons* in the *Red Hat Ceph Storage File System Guide*.
- See the *Configuring the number of standby daemons* in the *Red Hat Ceph Storage File System Guide*.
- See the *Metadata Server ranks* in the *Red Hat Ceph Storage File System Guide*.

5.12. CLIENT FEATURES

At times you might want to set Ceph File System (CephFS) features that clients must support to enable them to use Ceph File Systems. Clients without these features might disrupt other CephFS clients, or behave in unexpected ways. Also, you might want to require new features as to prevent older, and possibly buggy clients from connecting to a Ceph File System.

IMPORTANT

CephFS clients missing newly added features are evicted automatically.

You can list all the CephFS features by using the **fs features ls** command. You can add or remove requirements by using the **fs required_client_features** command.

Syntax

```
fs required_client_features FILE_SYSTEM_NAME add FEATURE_NAME
fs required_client_features FILE_SYSTEM_NAME rm FEATURE_NAME
```

Feature Descriptions

reply_encoding

Description

The Ceph Metadata Server (MDS) encodes reply requests in extensible format, if the client supports this feature.

reclaim_client

Description

The Ceph MDS allows a new client to reclaim another, perhaps a dead, client's state. This feature is used by NFS Ganesha.

lazy_caps_wanted

Description

When a stale client resumes the Ceph MDS only needs to re-issue the capabilities that are explicitly wanted, if the client supports this feature.

multi_reconnect

Description

After a Ceph MDS failover event, the client sends a reconnect message to the MDS to reestablish cache states. A client can split large reconnect messages into multiple messages.

deleg_ino

Description

A Ceph MDS delegates inode numbers to a client, if the client supports this feature. Delegating inode numbers is a prerequisite for a client to do async file creation.

metric_collect

Description

CephFS clients can send performance metrics to a Ceph MDS.

alternate_name

Description

CephFS clients can set and understand alternate names for directory entries. This feature allows for encrypted file names.

5.13. CEPH FILE SYSTEM CLIENT EVICTIONS

When a Ceph File System (CephFS) client is unresponsive or misbehaving, it might be necessary to forcibly terminate, or evict it from accessing the CephFS. Evicting a CephFS client prevents it from communicating further with Metadata Server (MDS) daemons and Ceph OSD daemons. If a CephFS client is buffering I/O to the CephFS at the time of eviction, then any un-flushed data will be lost. The CephFS client eviction process applies to all client types: FUSE mounts, kernel mounts, NFS gateways, and any process using **libcephfs** API library.

You can evict CephFS clients automatically, if they fail to communicate promptly with the MDS daemon, or manually.

Automatically Evictions

These scenarios cause an automatic CephFS client eviction:

- If a CephFS client has not communicated with the active MDS daemon for over the default 300 seconds, or as set by the **session_autoclose** option.
- If the **mds_cap_revoke_eviction_timeout** option is set, and a CephFS client has not responded to the cap revoke messages for over the set amount of seconds. The **mds_cap_revoke_eviction_timeout** option is disabled by default.
- During MDS startup or failover, the MDS daemon goes through a reconnect phase waiting for all the CephFS clients to connect to the new MDS daemon. If any CephFS clients fails to reconnect within the default time window of 45 seconds, or as set by the **mds_reconnect_timeout** option.

Additional Resources

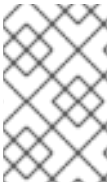
- See the *Manually evicting a Ceph File System client* section in the *Red Hat Ceph Storage File System Guide* for more details.

5.14. BLOCKLIST CEPH FILE SYSTEM CLIENTS

Ceph File System (CephFS) client blocklisting is enabled by default. When you send an eviction command to a single Metadata Server (MDS) daemon, it propagates the blocklist to the other MDS daemons. This is to prevent the CephFS client from accessing any data objects, so it is necessary to update the other CephFS clients, and MDS daemons with the latest Ceph OSD map, which includes the blocklisted client entries.

An internal "osdmap epoch barrier" mechanism is used when updating the Ceph OSD map. The purpose of the barrier is to verify the CephFS clients receiving the capabilities have a sufficiently recent Ceph OSD map, before any capabilities are assigned that might allow access to the same RADOS objects, as to not race with cancelled operations, such as, from ENOSPC or blocklisted clients from evictions.

If you are experiencing frequent CephFS client evictions due to slow nodes or an unreliable network, and you cannot fix the underlying issue, then you can ask the MDS to be less strict. It is possible to respond to slow CephFS clients by simply dropping their MDS sessions, but permit the CephFS client to re-open sessions and to continue talking to Ceph OSDs. By setting the **mds_session_blocklist_on_timeout** and **mds_session_blocklist_on_evict** options to **false** enables this mode.



NOTE

When blocklisting is disabled, the evicted CephFS client has only an effect on the MDS daemon you send the command to. On a system with multiple active MDS daemons, you need to send an eviction command to each active daemon.

5.15. MANUALLY EVICTING A CEPH FILE SYSTEM CLIENT

You might want to manually evict a Ceph File System (CephFS) client, if the client is misbehaving and you do not have access to the client node, or if a client dies, and you do not want to wait for the client session to time out.

Prerequisites

- User access to the Ceph Monitor node.

Procedure

1. Review the client list:

   Syntax

   ```
   ceph tell DAEMON_NAME client ls
   ```

   Exmaple

   ```
   [root@mon]# ceph tell mds.0 client ls
   [
       {
           "id": 4305,
           "num_leases": 0,
   ```

```
        "num_caps": 3,
        "state": "open",
        "replay_requests": 0,
        "completed_requests": 0,
        "reconnecting": false,
        "inst": "client.4305 172.21.9.34:0/422650892",
        "client_metadata": {
            "ceph_sha1": "ae81e49d369875ac8b569ff3e3c456a31b8f3af5",
            "ceph_version": "ceph version 12.0.0-1934-gae81e49
(ae81e49d369875ac8b569ff3e3c456a31b8f3af5)",
            "entity_id": "0",
            "hostname": "senta04",
            "mount_point": "/tmp/tmpcMpF1b/mnt.0",
            "pid": "29377",
            "root": "/"
        }
    }
]
```

2. Evict the specified CephFS client:

Syntax

```
ceph tell DAEMON_NAME client evict id=ID_NUMBER
```

Exmaple

```
[root@mon]# ceph tell mds.0 client evict id=4305
```

5.16. REMOVING A CEPH FILE SYSTEM CLIENT FROM THE BLOCKLIST

In some situations, it can be useful to allow a previous blocklisted Ceph File System (CephFS) client to reconnect to the storage cluster.

IMPORTANT

Removing a CephFS client from the blocklist puts data integrity at risk, and does not guarantee a fully healthy, and functional CephFS client as a result. The best way to get a fully healthy CephFS client back after an eviction, is to unmount the CephFS client and do a fresh mount. If other CephFS clients are accessing files that the blocklisted CephFS client was doing buffered I/O to can result in data corruption.

Prerequisites

- User access to the Ceph Monitor node.

Procedure

1. Review the blocklist:

   Exmaple

```
[root@mon]# ceph osd blocklist ls
listed 1 entries
127.0.0.1:0/3710147553 2020-03-19 11:32:24.716146
```

2. Remove the CephFS client from the blocklist:

   Syntax

   ```
   ceph osd blocklist rm CLIENT_NAME_OR_IP_ADDR
   ```

   Exmaple

   ```
   [root@mon]# ceph osd blocklist rm 127.0.0.1:0/3710147553
   un-blocklisting 127.0.0.1:0/3710147553
   ```

3. Optionally, you can have kernel-based CephFS clients automatically reconnect when removing them from the blocklist. On the kernel-based CephFS client, set the following option to **clean** either when doing a manual mount, or automatically mounting with an entry in the **/etc/fstab** file:

   ```
   recover_session=clean
   ```

4. Optionally, you can have FUSE-based CephFS clients automatically reconnect when removing them from the blocklist. On the FUSE client, set the following option to **true** either when doing a manual mount, or automatically mounting with an entry in the **/etc/fstab** file:

   ```
   client_reconnect_stale=true
   ```

Additional Resources

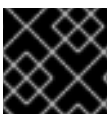
- See the *Mounting the Ceph File System as a FUSE client* in the *Red Hat Ceph Storage File System Guide* for more information.

5.17. EXPORTING CEPH FILE SYSTEM NAMESPACES OVER THE NFS PROTOCOL

Ceph File Systems (CephFS) namespaces can be exported over the NFS protocol using a NFS Ganesha file server. To export a CephFS namespace, you must first have a running NFS Ganesha cluster.

IMPORTANT

Red Hat supports only NFS version 4.0 or higher.



IMPORTANT

Red Hat does not support the creation of CephFS snapshots on NFS exports.

Prerequisites

- A running, and healthy Red Hat Ceph Storage cluster.
- An existing Ceph File System.

- Root-level access to a Ceph Monitor node.
- Installation of the **nfs-ganesha**, **nfs-ganesha-ceph**, **nfs-ganesha-rados-grace** and **nfs-ganesha-rados-urls** packages on the Ceph Manager nodes.
- Root-level access to a client node.

Procedure

1. Enable the Ceph Manager NFS module:

   Example

   ```
   [root@mon ~]# ceph mgr module enable nfs
   ```

2. Create a NFS Ganesha cluster:

   Syntax

   ```
   ceph nfs cluster create CLUSTER_NAME "NODE_PLACEMENT_LIST"
   ```

   Example

   ```
   [root@mon ~]# ceph nfs cluster create nfs-cephfs "node1.example.com,node2.example.com"
   NFS Cluster Created Successfully
   ```

   In this example, the NFS Ganesha cluster name is **nfs-cephfs** and the daemon containers are deployed to **node1.example.com**, and **node2.example.com**.

   IMPORTANT

   Red Hat only supports one NFS Ganesha daemon running per node.

   a. To delete a NFS Ganesha cluster:

      Syntax

      ```
      ceph nfs cluster delete CLUSTER_NAME
      ```

      Example

      ```
      [root@mon ~]# ceph nfs cluster delete nfs-cephfs
      NFS Cluster Deleted Successfully
      ```

3. Verify the NFS Ganesha cluster information:

   Syntax

   ```
   ceph nfs cluster info [CLUSTER_NAME]
   ```

   Example

```
[root@mon ~]# ceph nfs cluster info nfs-cephfs
{
    "nfs-cephfs": [
        {
            "hostname": "node1.example.com",
            "ip": [
                "2620:52:0:880:225:90ff:fefc:2a2e",
                "10.8.128.21"
            ],
            "port": 2049
        },
        {
            "hostname": "node2.example.com",
            "ip": [
                "2620:52:0:880:225:90ff:fefc:2c3c",
                "10.8.128.22"
            ],
            "port": 2049
        }
    ]
}
```

NOTE

Specifying the *CLUSTER_NAME* is optional.

4. Create a CephFS export:

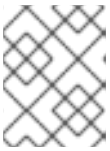
Syntax

```
ceph nfs export create cephfs FILE_SYSTEM_NAME CLUSTER_NAME BINDING [--readonly] [--path=PATH_WITHIN_CEPHFS]
```

Example

```
[root@mon ~]# ceph nfs export create cephfs cephfs01 nfs-cephfs /ceph --path=/
```

In this example, the *BINDING* (/**ceph**) is the pseudo root path, which must be unique, and an absolute path.



NOTE

The **--readonly** option exports a path with the read-only permission, the default being read, and write permissions.

NOTE

The *PATH_WITHIN_CEPHFS* can be a subvolume. You can get the absolute subvolume path by using the following command:

Syntax

```
ceph fs subvolume getpath VOLUME_NAME SUBVOLUME_NAME [--group_name SUBVOLUME_GROUP_NAME]
```

Example

```
[root@mon ~]# ceph fs subvolume getpath cephfs sub0
```

a. To view the export block based on the pseudo root name:

Syntax

```
ceph nfs export get CLUSTER_NAME BINDING
```

Example

```
[root@mon ~]# ceph nfs export get nfs-cephfs /ceph
{
  "export_id": 1,
  "path": "/",
  "cluster_id": "nfs-cephfs",
  "pseudo": "/ceph",
  "access_type": "RW",
  "squash": "no_root_squash",
  "security_label": true,
  "protocols": [
    4
  ],
  "transports": [
    "TCP"
  ],
  "fsal": {
    "name": "CEPH",
    "user_id": "cephnfs11",
    "fs_name": "garbage",
    "sec_label_xattr": ""
  },
  "clients": []
}
```

b. To delete a CephFS export:

Syntax

```
ceph nfs export delete CLUSTER_NAME BINDING
```

Example

```
[root@mon ~]# ceph nfs export delete nfs-cephfs /ceph
```

5. Customize a NFS Ganesha configuration.

   a. Create a new Ceph user:

   Syntax

   ```
   ceph auth get-or-create client.USER_NAME mon 'allow r' osd 'allow rw pool=nfs-ganesha namespace=CLUSTER_NAME, allow rw tag cephfs data=FILE_SYSTEM_NAME' mds 'allow rw path=PATH_WITHIN_CEPHFS'
   ```

   Example

   ```
   [root@mon ~]# ceph auth get-or-create client.nfstest01 mon 'allow r' osd 'allow rw pool=nfs-ganesha namespace=nfs-cephfs, allow rw tag cephfs data=cephfs01' mds 'allow rw path=/'
   ```

   b. Create a new configuration file:

   Syntax

   ```
   touch PATH_TO_CONFIG_FILE
   ```

   Example

   ```
   [root@mon ~]# touch /root/nfs-cephfs.conf
   ```

   c. Open, and edit the new configuration file by adding a custom export block:

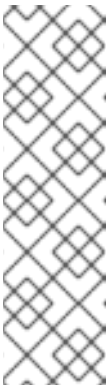
   Syntax

   ```
   EXPORT {
     Export_Id = NUMERICAL_ID;
     Transports = TCP;
     Path = PATH_WITHIN_CEPHFS;
     Pseudo = BINDING;
     Protocols = 4;
     Access_Type = PERMISSIONS;
     Attr_Expiration_Time = 0;
     Squash = None;
     FSAL {
       Name = CEPH;
       Filesystem = "FILE_SYSTEM_NAME";
       User_Id = "USER_NAME";
       Secret_Access_Key = "USER_SECRET_KEY";
     }
   }
   ```

   Example

   ```
   EXPORT {
   ```

```
  Export_Id = 100;
  Transports = TCP;
  Path = /;
  Pseudo = /ceph/;
  Protocols = 4;
  Access_Type = RW;
  Attr_Expiration_Time = 0;
  Squash = None;
  FSAL {
    Name = CEPH;
    Filesystem = "cephfs01";
    User_Id = "nfstest01";
    Secret_Access_Key = "AQD9aW1g1UWmCxAAxZTW8YKeVi4sF5X+5ehH4Q==";
  }
}
```



NOTE

Optionally, you can also enable logging in the configuration file with the following block:

```
LOG {
 COMPONENTS {
     ALL = FULL_DEBUG;
 }
}
```

d. Set the new configuration:

Syntax

```
ceph nfs cluster config set CLUSTER_NAME -i PATH_TO_CONFIG_FILE
```

Example

```
[root@mon ~]# ceph nfs cluster config set nfs-cephfs -i /root/nfs-cephfs.conf
```

e. List the NFS exports:

Syntax

```
ceph nfs export ls CLUSTER_NAME [--detailed]
```

Example

```
[root@mon ~]# ceph nfs export ls nfs-cephfs
[
  "/ceph/"
]
[root@mon ~]#
[root@mon ~]# ceph nfs export ls nfs-cephfs --detailed
[
```

```
 {
  "export_id": 100,
  "path": "/",
  "cluster_id": "nfs-cephfs",
  "pseudo": "/ceph/",
  "access_type": "RW",
  "squash": "no_root_squash",
  "security_label": true,
  "protocols": [
   4
  ],
  "transports": [
   "TCP"
  ],
  "fsal": {
   "name": "CEPH",
   "user_id": "nfstest01",
   "fs_name": "cephfs",
   "sec_label_xattr": ""
  },
  "clients": []
 }
]
```

f. To delete the custom configuration:

Syntax

```
ceph nfs cluster config reset CLUSTER_NAME -i PATH_TO_CONFIG_FILE
```

Example

```
[root@mon ~]# ceph nfs cluster config reset nfs-cephfs -i /root/nfs-cephfs.conf
```

6. On a client node, mount the exported Ceph File System:

Syntax

```
mount -t nfs -o port=GANESHA_PORT NODE_NAME:_BINDING_ LOCAL_MOUNT_POINT
```

Example

```
[root@client ~]# mount -t nfs -o port=2049 node1:/ceph/ /mnt/nfs-cephfs
```

a. To automatically mount on boot, open and edit the **/etc/fstab** file by adding a new line:

Syntax

```
NODE_NAME:_BINDING_   LOCAL_MOUNT_POINT   nfs4
defaults,seclabel,vers=4.2,proto=tcp,port=2049  0  0
```

Example

```
node1.example.com:/ceph/   /mnt/nfs-cephfs   nfs4
defaults,seclabel,vers=4.2,proto=tcp,port=2049  0  0
```

Additional Resources

- See the *Mounting the Ceph File System as a kernel client* section in the *Red Hat Ceph Storage File System Guide* for more details.
- See the *Mounting the Ceph File System as a FUSE client* section in the *Red Hat Ceph Storage File System Guide* for more details.

5.18. CEPH FILE SYSTEM QUOTAS

As a storage administrator, you can view, set, and remove quotas on any directory in the file system. You can place quota restrictions on the number of bytes or the number of files within the directory.

5.18.1. Prerequisites

- A running, and healthy Red Hat Ceph Storage cluster.
- Deployment of a Ceph File System.
- Make sure that the **attr** package is installed.

5.18.2. Ceph File System quotas

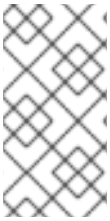
The Ceph File System (CephFS) quotas allow you to restrict the number of bytes or the number of files stored in the directory structure. Ceph File System quotas are fully supported using a FUSE client or using Kernel clients, version 4.17 or newer.

Limitations

- CephFS quotas rely on the cooperation of the client mounting the file system to stop writing data when it reaches the configured limit. However, quotas alone cannot prevent an adversarial, untrusted client from filling the file system.
- Once processes that write data to the file system reach the configured limit, a short period of time elapses between when the amount of data reaches the quota limit, and when the processes stop writing data. The time period generally measures in the tenths of seconds. However, processes continue to write data during that time. The amount of additional data that the processes write depends on the amount of time elapsed before they stop.
- When using path-based access restrictions, be sure to configure the quota on the directory to which the client is restricted, or to a directory nested beneath it. If the client has restricted access to a specific path based on the MDS capability, and the quota is configured on an ancestor directory that the client cannot access, the client will not enforce the quota. For example, if the client cannot access the **/home/** directory and the quota is configured on **/home/**, the client cannot enforce that quota on the directory **/home/user/**.
- Snapshot file data that has been deleted or changed does not count towards the quota.
- No support for quotas with NFS clients when using **setxattr**, and no support for file-level quotas on NFS. To use quotas on NFS shares, you can export them using subvolumes and setting the **--size** option.

5.18.3. Viewing quotas

Use the **getfattr** command and the **ceph.quota** extended attributes to view the quota settings for a directory.



NOTE

If the attributes appear on a directory inode, then that directory has a configured quota. If the attributes do not appear on the inode, then the directory does not have a quota set, although its parent directory might have a quota configured. If the value of the extended attribute is **0**, the quota is not set.

Prerequisites

- Root-level access to the Ceph client node.
- Make sure that the **attr** package is installed.

Procedure

1. To view CephFS quotas.
   a. Using a byte-limit quota:

      Syntax

      ```
      getfattr -n ceph.quota.max_bytes DIRECTORY
      ```

      Example

      ```
      [root@client ~]# getfattr -n ceph.quota.max_bytes /mnt/cephfs/
      getfattr: Removing leading '/' from absolute path names
      # file: mnt/cephfs/
      ceph.quota.max_bytes="100000000"
      ```

      In this example, **100000000** equals 100 MB.

   b. Using a file-limit quota:

      Syntax

      ```
      getfattr -n ceph.quota.max_files DIRECTORY
      ```

      Example

      ```
      [root@client ~]# getfattr -n ceph.quota.max_files /mnt/cephfs/
      getfattr: Removing leading '/' from absolute path names
      # file: mnt/cephfs/
      ceph.quota.max_files="10000"
      ```

      In this example, **10000** equals 10,000 files.

Additional Resources

- See the **getfattr(1)** manual page for more information.

5.18.4. Setting quotas

This section describes how to use the **setfattr** command and the **ceph.quota** extended attributes to set the quota for a directory.

Prerequisites

- Root-level access to the Ceph client node.
- Make sure that the **attr** package is installed.

Procedure

1. To set CephFS quotas.
   a. Using a byte-limit quota:

      Syntax

      ```
      setfattr -n ceph.quota.max_bytes -v 100000000 DIRECTORY
      ```

      Example

      ```
      [root@client ~]# setfattr -n ceph.quota.max_bytes -v 100000000 /cephfs/
      ```

      In this example, **100000000** bytes equals 100 MB.

   b. Using a file-limit quota:

      Syntax

      ```
      setfattr -n ceph.quota.max_files -v 10000 DIRECTORY
      ```

      Example

      ```
      [root@client ~]# setfattr -n ceph.quota.max_files -v 10000 /cephfs/
      ```

      In this example, **10000** equals 10,000 files.

Additional Resources

- See the **setfattr(1)** manual page for more information.

5.18.5. Removing quotas

This section describes how to use the **setfattr** command and the **ceph.quota** extended attributes to remove a quota from a directory.

Prerequisites

- Root-level access to the Ceph client node.
- Make sure that the **attr** package is installed.

Procedure

1. To remove CephFS quotas.
   a. Using a byte-limit quota:

      Syntax

      ```
      setfattr -n ceph.quota.max_bytes -v 0 DIRECTORY
      ```

      Example

      ```
      [root@client ~]# setfattr -n ceph.quota.max_bytes -v 0 /mnt/cephfs/
      ```

   b. Using a file-limit quota:

      Syntax

      ```
      setfattr -n ceph.quota.max_files -v 0 DIRECTORY
      ```

      Example

      ```
      [root@client ~]# setfattr -n ceph.quota.max_files -v 0 /mnt/cephfs/
      ```

Additional Resources

- See the **setfattr(1)** manual page for more information.

5.18.6. Additional Resources

- See the *Deployment of the Ceph File System* section in the *Red Hat Ceph Storage File System Guide*.
- See the **getfattr(1)** manual page for more information.
- See the **setfattr(1)** manual page for more information.

5.19. FILE AND DIRECTORY LAYOUTS

As a storage administrator, you can control how file or directory data is mapped to objects.

This section describes how to:

- Understand file and directory layouts
- Set file and directory layouts
- View file and directory layout fields

- View individual layout fields
- Remove the directory layouts

5.19.1. Prerequisites

- A running, and healthy Red Hat Ceph Storage cluster.
- Deployment of a Ceph File System.
- The installation of the **attr** package.

5.19.2. Overview of file and directory layouts

This section explains what file and directory layouts are in the context for the Ceph File System.

A layout of a file or directory controls how its content is mapped to Ceph RADOS objects. The directory layouts serves primarily for setting an inherited layout for new files in that directory.

To view and set a file or directory layout, use virtual extended attributes or extended file attributes (**xattrs**). The name of the layout attributes depends on whether a file is a regular file or a directory:

- Regular files layout attributes are called **ceph.file.layout**.
- Directories layout attributes are called **ceph.dir.layout**.

Layouts Inheritance

Files inherit the layout of their parent directory when you create them. However, subsequent changes to the parent directory layout do not affect children. If a directory does not have any layouts set, files inherit the layout from the closest directory with layout in the directory structure.

5.19.3. Setting file and directory layout fields

Use the **setfattr** command to set layout fields on a file or directory.

IMPORTANT

When you modify the layout fields of a file, the file must be empty, otherwise an error occurs.

Prerequisites

- Root-level access to the node.

Procedure

1. To modify layout fields on a file or directory:

   Syntax

   ```
   setfattr -n ceph.TYPE.layout.FIELD -v VALUE PATH
   ```

   Replace:

- *TYPE* with **file** or **dir**.
- *FIELD* with the name of the field.
- *VALUE* with the new value of the field.
- *PATH* with the path to the file or directory.

Example

```
[root@fs ~]# setfattr -n ceph.file.layout.stripe_unit -v 1048576 test
```

Additional Resources

- See the table in the *Overview of the file and directory layouts* section of the *Red Hat Ceph Storage File System Guide* for more details.
- See the **setfattr(1)** manual page.

5.19.4. Viewing file and directory layout fields

To use the **getfattr** command to view layout fields on a file or directory.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to all nodes in the storage cluster.

Procedure

1. To view layout fields on a file or directory as a single string:

Syntax

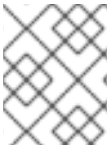
```
getfattr -n ceph.TYPE.layout PATH
```

Replace

- *PATH* with the path to the file or directory.
- *TYPE* with **file** or **dir**.

Example

```
[root@mon ~] getfattr -n ceph.dir.layout /home/test
ceph.dir.layout="stripe_unit=4194304 stripe_count=2 object_size=4194304
pool=cephfs_data"
```



NOTE

A directory does not have an explicit layout until you set it. Consequently, attempting to view the layout without first setting it fails because there are no changes to display.

Additional Resources

- The **getfattr(1)** manual page.
- For more information, see *Setting file and directory layouts* section in the *Red Hat Ceph Storage File System Guide*.

5.19.5. Viewing individual layout fields

Use the **getfattr** command to view individual layout fields for a file or directory.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to all nodes in the storage cluster.

Procedure

1. To view individual layout fields on a file or directory:

   #### Syntax

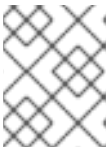
   ```
   getfattr -n ceph.TYPE.layout.FIELD _PATH
   ```

   #### Replace

   - *TYPE* with **file** or **dir**.
   - *FIELD* with the name of the field.
   - *PATH* with the path to the file or directory.

   #### Example

   ```
   [root@mon ~] getfattr -n ceph.file.layout.pool test
   ceph.file.layout.pool="cephfs_data"
   ```

   NOTE

   Pools in the **pool** field are indicated by name. However, newly created pools can be indicated by ID.

Additional Resources

- The **getfattr(1)** manual page.
- For more information, see *File and directory layouts* section in the *Red Hat Ceph Storage File System Guide*.

5.19.6. Removing directory layouts

Use the **setfattr** command to remove layouts from a directory.

NOTE

When you set a file layout, you cannot change or remove it.

Prerequisites

- A directory with a layout.

Procedure

1. To remove a layout from a directory:

   Syntax

   ```
   setfattr -x ceph.dir.layout DIRECTORY_PATH
   ```

   Example

   ```
   [user@client ~]$ setfattr -x ceph.dir.layout /home/cephfs
   ```

2. To remove the **pool_namespace** field:

   Syntax

   ```
   setfattr -x ceph.dir.layout.pool_namespace DIRECTORY_PATH
   ```

   Example

   ```
   [user@client ~]$ setfattr -x ceph.dir.layout.pool_namespace /home/cephfs
   ```

   NOTE

   The **pool_namespace** field is the only field you can remove separately.

Additional Resources

- The **setfattr(1)** manual page

5.19.7. Additional Resources

- See the *Deployment of the Ceph File System* section in the *Red Hat Ceph Storage File System Guide*.
- See the **getfattr(1)** manual page for more information.
- See the **setfattr(1)** manual page for more information.

5.20. CEPH FILE SYSTEM SNAPSHOTS

As a storage administrator, you can take a point-in-time snapshot of a Ceph File System (CephFS) directory. CephFS snapshots are asynchronous, and you can choose which directory snapshots are created in.

5.20.1. Prerequisites

- A running, and healthy Red Hat Ceph Storage cluster.
- Deployment of a Ceph File System.

5.20.2. Ceph File System snapshots

The Ceph File System (CephFS) snapshotting feature is enabled by default on new Ceph File Systems, but must be manually enabled on existing Ceph File Systems. CephFS snapshots creates an immutable, point-in-time view of a Ceph File System. CephFS snapshots are asynchronous, and are kept in a special hidden directory in the CephFS directory, named **.snap**. You can specify snapshot creation for any directory within a Ceph File System. When specifying a directory, the snapshot also includes all the sub directories beneath it.

WARNING

Each Ceph Metadata Server (MDS) cluster allocates the snap identifiers independently. Using snapshots for multiple Ceph File Systems that are sharing a single pool causes snapshot collisions, and results in missing file data.

Additional Resources

- See the *Creating a snapshot for a Ceph File System* section in the *Red Hat Ceph Storage File System Guide* for more details.
- See the *Creating a snapshot schedule for a Ceph File System* section in the *Red Hat Ceph Storage File System Guide* for more details.

5.20.3. Creating a snapshot for a Ceph File System

You can create an immutable, point-in-time view of a Ceph File System (CephFS) by creating a snapshot.

NOTE

For a new Ceph File System, snapshots are enabled by default.

Prerequisites

- A running, and healthy Red Hat Ceph Storage cluster.
- Deployment of a Ceph File System.
- Root-level access to a Ceph Metadata Server (MDS) node.

Procedure

1. For existing Ceph File Systems, enable the snapshotting feature:

Syntax

```
ceph fs set FILE_SYSTEM_NAME allow_new_snaps true
```

Example

```
[ceph: root@mds ~]# ceph fs set cephfs01 allow_new_snaps true
```

2. Create a new snapshot subdirectory under the **.snap** directory:

Syntax

```
mkdir NEW_DIRECTORY_PATH
```

Example

```
[ceph: root@mds ~]# mkdir /.snap/new-snaps
```

This example creates the **new-snaps** subdirectory, and this informs the Ceph Metadata Server (MDS) to start making snapshots.

a. To delete snapshots:

Syntax

```
rmdir NEW_DIRECTORY_PATH
```

Example

```
[ceph: root@mds ~]# rmdir /.snap/new-snaps
```

IMPORTANT

Attempting to delete root-level snapshots, which might contain underlying snapshots, will fail.

Additional Resources

- See the *Ceph File System snapshot schedules* section in the *Red Hat Ceph Storage File System Guide* for more details.
- See the *Ceph File System snapshots* section in the *Red Hat Ceph Storage File System Guide* for more details.

5.20.4. Ceph File System snapshot schedules

A Ceph File System (CephFS) can schedule snapshots of a file system directory. The scheduling of snapshots is managed by the Ceph Manager, and relies on Python Timers. The snapshot schedule data is stored as an object in the CephFS metadata pool, and at runtime, all the schedule data lives in a serialized SQLite database.

IMPORTANT

The scheduler is precise based on the specified time to keep snapshots apart, when a storage cluster is under normal load. When the Ceph Manager is under a heavy load, it's possible that a snapshot might not get scheduled right away, resulting in a slightly delayed snapshot. If this happens, then the next scheduled snapshot acts as if there was no delay. Scheduled snapshots that are delayed do not cause drift in the overall schedule.

Usage

Scheduling snapshots for a Ceph File System (CephFS) is managed by the **snap_schedule** Ceph Manager module. This module provides an interface to add, query, and delete snapshot schedules, and to manage the retention policies. This module also implements the **ceph fs snap-schedule** command, with several subcommands to manage schedules, and retention policies. All of the subcommands take the CephFS volume path and subvolume path arguments to specify the file system path when using multiple Ceph File Systems. Not specifying the CephFS volume path, the argument defaults to the first file system listed in the **fs_map**, and not specifying the subvolume path argument defaults to nothing.

Snapshot schedules are identified by the file system path, the repeat interval, and the start time. The repeat interval defines the time between two subsequent snapshots. The interval format is a number plus a time designator: **h**(our), **d**(ay), or **w**(eek). For example, having an interval of **4h**, means one snapshot every four hours. The start time is a string value in the ISO format, **%Y-%m-%dT%H:%M:%S**, and if not specified, the start time uses a default value of last midnight. For example, you schedule a snapshot at **14:45**, using the default start time value, with a repeat interval of **1h**, the first snapshot will be taken at 15:00.

Retention policies are identified by the file system path, and the retention policy specifications. Defining a retention policy consist of either a number plus a time designator or a concatenated pairs in the format of ***COUNT TIME_PERIOD***. The policy ensures a number of snapshots are kept, and the snapshots are at least for a specified time period apart. The time period designators are: **h**(our), **d**(ay), **w**(eek), **m**(onth), **y**(ear), and **n**. The **n** time period designator is a special modifier, which means keep the last number of snapshots regardless of timing. For example, **4d** means keep four snapshots that are at least one day, or longer apart from each other.

Additional Resources

- See the *Creating a snapshot for a Ceph File System* section in the *Red Hat Ceph Storage File System Guide* for more details.
- See the *Creating a snapshot schedule for a Ceph File System* section in the *Red Hat Ceph Storage File System Guide* for more details.

5.20.5. Creating a snapshot schedule for a Ceph File System

You can add, query, and delete snapshot schedules, and manage the retention policies for Ceph File System (CephFS) snapshots. You can have different schedules for a single path. Schedules are considered different, if their repeat interval and start times are different. A snapshot schedule can be added for a CephFS path that do not exist yet. A CephFS path can only have one retention policy, but a retention policy can have multiple count-time period pairs.

NOTE

Once the scheduler module is enabled, running the **ceph fs snap-schedule** command displays the available subcommands and their usage format.

Prerequisites

- A running, and healthy Red Hat Ceph Storage cluster.
- Deployment of a Ceph File System.
- Root-level access to a Ceph Manager and Metadata Server (MDS) nodes.
- Enable CephFS snapshots on the file system.

Procedure

1. Enable the **snap_schedule** module on a Ceph Manager node:

   Example

   ```
   [ceph: root@mon ~]# ceph mgr module enable snap_schedule
   ```

2. Add a new schedule for a Ceph File System:

   Syntax

   ```
   ceph fs snap-schedule add FILE_SYSTEM_VOLUME_PATH REPEAT_INTERVAL
   [START_TIME]
   ```

   Example

   ```
   [ceph: root@mds ~]# ceph fs snap-schedule add /cephfs 4h 14:00
   ```

   This example creates a snapshot schedule for the CephFS /**cephfs** volume, snapshotting every four hours, and starts at **14:00**.

   a. To remove a specific snapshot schedule:

      Syntax

      ```
      ceph fs snap-schedule remove FILE_SYSTEM_VOLUME_PATH [REPEAT_INTERVAL]
      [START_TIME]
      ```

      Example

      ```
      [ceph: root@mds ~]# ceph fs snap-schedule remove /cephfs 4h 14:00
      ```

      This example removes the specific snapshot schedule for the CephFS /**cephfs** volume, that is snapshotting every four hours, and started at **14:00**.

   b. To remove all snapshot schedule for a specific CephFS volume:

      Syntax

      ```
      ceph fs snap-schedule remove FILE_SYSTEM_VOLUME_PATH
      ```

      Example

```
[ceph: root@mds ~]# ceph fs snap-schedule remove /cephfs
```

This example removes all the snapshot schedules for the CephFS /**cephfs** volume.

3. Add a new retention policy for snapshots of a CephFS volume path:

Syntax

```
ceph fs snap-schedule retention add FILE_SYSTEM_VOLUME_PATH
[COUNT_TIME_PERIOD_PAIR] TIME_PERIOD COUNT
```

Example

```
[ceph: root@mds ~]# ceph fs snap-schedule retention add /cephfs h 14 (1)
[ceph: root@mds ~]# ceph fs snap-schedule retention add /cephfs d 4 (2)
[ceph: root@mds ~]# ceph fs snap-schedule retention add /cephfs 14h4w (3)
```

(1) This example keeps 14 snapshots at least one hour apart.

(2) This example keeps 4 snapshots at least one day apart.

(3) This example keeps 14 hourly, and 4 weekly snapshots.

a. To remove a retention policy on a CephFS path:

Syntax

```
ceph fs snap-schedule retention remove FILE_SYSTEM_VOLUME_PATH
[COUNT_TIME_PERIOD_PAIR] TIME_PERIOD COUNT
```

Example

```
[ceph: root@mds ~]# ceph fs snap-schedule retention remove /cephfs h 4 (1)
[ceph: root@mds ~]# ceph fs snap-schedule retention remove /cephfs 14d4w (2)
```

This example removes 4 hourly snapshots.

This example removes 14 daily, and 4 weekly snapshots.

4. Activate the new snapshot schedule for a CephFS path:

Syntax

```
ceph fs snap-schedule activate FILE_SYSTEM_VOLUME_PATH [REPEAT_INTERVAL]
```

Example

```
[ceph: root@mds ~]# ceph fs snap-schedule activate /cephfs
```

This example activates all schedules for the CephFS /**cephfs** path.

5. To deactivate a snapshot schedule for a CephFS path:

   Syntax

   ```
   ceph fs snap-schedule deactivate FILE_SYSTEM_VOLUME_PATH [REPEAT_INTERVAL]
   ```

   Example

   ```
   [ceph: root@mds ~]# ceph fs snap-schedule deactivate /cephfs 1d
   ```

   This example deactivates the daily snapshots for the CephFS /**cephfs** path, thereby pausing any further snapshot creation.

6. Check the status of snapshot schedules.

   a. List the snapshot schedules:

      Syntax

      ```
      ceph fs snap-schedule list FILE_SYSTEM_VOLUME_PATH [--format=plain|json] [--recursive=true]
      ```

      Example

      ```
      [ceph: root@mds ~]# ceph fs snap-schedule list /cephfs --recursive=true
      ```

      This example lists all schedules in the directory tree.

   b. Check the status of a snapshot schedule:

      Syntax

      ```
      ceph fs snap-schedule status FILE_SYSTEM_VOLUME_PATH [--format=plain|json]
      ```

      Example

      ```
      [ceph: root@mds ~]# ceph fs snap-schedule status /cephfs --format=json
      ```

      This example displays the status of the snapshot schedule for the CephFS /**cephfs** path in a JSON format. The default format is plain text, if not specified.

Additional Resources

- See the *Ceph File System snapshot schedules* section in the *Red Hat Ceph Storage File System Guide* for more details.
- See the *Ceph File System snapshots* section in the *Red Hat Ceph Storage File System Guide* for more details.

5.20.6. Additional Resources

- See the *Deployment of the Ceph File System* section in the *Red Hat Ceph Storage File System Guide*.

5.21. CEPH FILE SYSTEM MIRRORS

As a storage administrator, you can replicate a Ceph File System (CephFS) to a remote Ceph File System on another Red Hat Ceph Storage cluster. A Ceph File System supports asynchronous replication of snapshots directories.

5.21.1. Prerequisites

- The source and the target storage clusters must be running Red Hat Ceph Storage 5.0 or later.

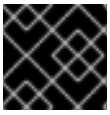
5.21.2. Ceph File System mirroring

The Ceph File System (CephFS) supports asynchronous replication of snapshots to a remote Ceph File System on another Red Hat Ceph Storage cluster. Snapshot synchronization copies snapshot data to a remote Ceph File System, and creates a new snapshot on the remote target with the same name. You can configure specific directories for snapshot synchronization.

Management of CephFS mirrors is done by the CephFS mirroring daemon (**cephfs-mirror**). This snapshot data is synchronized by doing a bulk copy to the remote CephFS. The chosen order of synchronizing snapshot pairs is based on the creation using the **snap-id**.

IMPORTANT

Hard linked files get synchronized as separate files.



IMPORTANT

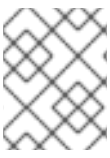
Red Hat supports running only one **cephfs-mirror** daemon per storage cluster.

Ceph Manager Module

The Ceph Manager **mirroring** module is disabled by default. It provides interfaces for managing directory snapshot mirroring, and is responsible for assigning directories to the **cephfs-mirror** daemon for synchronization. The Ceph Manager **mirroring** module also provides a family of commands to control mirroring of directory snapshots. The **mirroring** module does not manage the **cephfs-mirror** daemon. The stopping, starting, restarting, and enabling of the **cephfs-mirror** daemon is controlled by **systemctl**, but managed by **cephadm**.

5.21.3. Configuring a snapshot mirror for a Ceph File System

You can configure a Ceph File System (CephFS) for mirroring to replicate snapshots to another CephFS on a remote Red Hat Ceph Storage cluster.



NOTE

The time taken for synchronizing to remote storage cluster depends on the file size and the total number of files in the mirroring path.

Prerequisites

- The source and the target storage clusters must be healthy and running Red Hat Ceph Storage 5.0 or later.
- Root-level access to a Ceph Monitor node in the source and the target storage clusters.
- At least one deployment of a Ceph File System.

Procedure

1. On the source storage cluster, deploy the CephFS mirroring daemon:

   Syntax

   ```
   ceph orch apply cephfs-mirror ["NODE_NAME"]
   ```

   Example

   ```
   [root@mon ~]# ceph orch apply cephfs-mirror "node1.example.com"
   Scheduled cephfs-mirror update...
   ```

   This command creates a Ceph user called, **cephfs-mirror**, and deploys the **cephfs-mirror** daemon on the given node.

2. On the target storage cluster, create a user for each CephFS peer:

   Syntax

   ```
   ceph fs authorize FILE_SYSTEM_NAME CLIENT_NAME / rwps
   ```

   Example

   ```
   [root@mon ~]# ceph fs authorize cephfs client.mirror_remote / rwps
   [client.mirror_remote]
         key = AQCjZ5Jg739AAxAAxduIKoTZbiFJ0lgose8luQ==
   ```

3. On the source storage cluster, enable the CephFS mirroring module:

   Example

   ```
   [root@mon ~]# ceph mgr module enable mirroring
   ```

4. On the source storage cluster, enable mirroring on a Ceph File System:

   Syntax

   ```
   ceph fs snapshot mirror enable FILE_SYSTEM_NAME
   ```

   Example

   ```
   [root@mon ~]# ceph fs snapshot mirror enable cephfs
   ```

   a. Optional. To disable snapshot mirroring, use the following command:

Syntax

```
ceph fs snapshot mirror disable FILE_SYSTEM_NAME
```

Example

```
[root@mon ~]# ceph fs snapshot mirror disable cephfs
```

WARNING

Disabling snapshot mirroring on a file system removes the configured peers. You have to import the peers again by bootstrapping them.

5. Prepare the target peer storage cluster.

   a. On a target node, enable the **mirroring** Ceph Manager module:

   Example

   ```
   [root@mon ~]# ceph mgr module enable mirroring
   ```

   b. On the same target node, create the peer bootstrap:

   Syntax

   ```
   ceph fs snapshot mirror peer_bootstrap create FILE_SYSTEM_NAME CLIENT_NAME SITE_NAME
   ```

   The *SITE_NAME* is a user-defined string to identify the target storage cluster.

   Example

   ```
   [root@mon ~]# ceph fs snapshot mirror peer_bootstrap create cephfs client.mirror_remote remote-site
   {"token": "eyJmc2lkIjogIjBkZjE3MjE3LWRmY2QtNDAzMC05MDc5LTM2Nzk4NTVkNDJlZiIsICJmaWxlc3lzdGVtIjogImJhY2t1cF9mcyIsICJ1c2VyIjogImNsaWVudC5taXJyb3JfcGVlcl9ib290c3RyYXAiLCAic2l0ZV9uYW1lIjogInNpdGUtcmVtb3RlIiwgImtleSI6ICJBUUFhcDBCZ0xtRmpOeEFBVnNyZXozai9YYUV0T2UrbUJEZlJDZz09IiwgIm1vbl9ob3N0IjogIlt2MjoxOTIuMTY4LjAuNTo0MDkxOCx2MToxOTIuMTY4LjAuNTo0MDkxOV0ifQ=="}
   ```

   Copy the token string between the double quotes for use in the next step.

6. On the source storage cluster, import the bootstrap token from the target storage cluster:

   Syntax

   ```
   ceph fs snapshot mirror peer_bootstrap import FILE_SYSTEM_NAME TOKEN
   ```

–

Example

```
[root@mon ~]# ceph fs snapshot mirror peer_bootstrap import cephfs
eyJmc2lkIjogIjBkZjE3MjE3LWRmY2QtNDAzMC05MDc5LTM2Nzk4NTVkNDJlZiIsICJmaWxlc3lz
dGVtIjogImJhY2t1cF9mcyIsICJ1c2VyIjogImNsaWVudC5taXJyb3JfcGVlcl9ib290c3RyYXAiLCAic
2l0ZV9uYW1lIjogInNpdGUtcmVtb3RlIiwgImtleSI6ICJBUUFhcDBCZ0xtRmpOeEFBVnNyZXoza
i9YYUV0T2UrbUJEZlJDZz09IiwgIm1vbl9ob3N0IjogIlt2MjoxOTIuMTY4LjAuNTo0MDkxOCx2M
ToxOTIuMTY4LjAuNTo0MDkxOV0ifQ==
```

7. On the source storage cluster, list the CephFS mirror peers:

Syntax

```
ceph fs snapshot mirror peer_list FILE_SYSTEM_NAME
```

Example

```
[root@mon ~]# ceph fs snapshot mirror peer_list cephfs
```

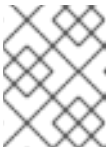
a. Optional. To remove a snapshot peer, use the following command:

Syntax

```
ceph fs snapshot mirror peer_remove FILE_SYSTEM_NAME PEER_UUID
```

Example

```
[root@mon ~]# ceph fs snapshot mirror peer_remove cephfs a2dc7784-e7a1-4723-
b103-03ee8d8768f8
```



NOTE

See the *Viewing the mirror status for a Ceph File System* link in the *Additional Resources* section of this procedure on how to find the peer UUID value.

8. On the source storage cluster, configure a directory for snapshot mirroring:

Syntax

```
ceph fs snapshot mirror add FILE_SYSTEM_NAME PATH
```

Example

```
[root@mon ~]# ceph fs snapshot mirror add cephfs /home/user1
```

IMPORTANT

Only absolute paths are valid.

NOTE

The Ceph Manager **mirroring** module normalizes the path. For example, the **/d1/d2/../dN** directories are equivalent to **/d1/d2**. Once a directory has been added for mirroring, its ancestor directories and subdirectories are prevented from being added for mirroring.

a. Optional. To stop snapshot mirroring for a directory, use the following command:

Syntax

```
ceph fs snapshot mirror remove FILE_SYSTEM_NAME PATH
```

Example

```
[root@mon ~]# ceph fs snapshot mirror remove cephfs /home/user1
```

Additional Resources

- See the *Viewing the mirror status for a Ceph File System* section in the *Red Hat Ceph Storage File System Guide* for more information.
- See the *Ceph File System mirroring* section in the *Red Hat Ceph Storage File System Guide* for more information.

5.21.4. Viewing the mirror status for a Ceph File System

The Ceph File System (CephFS) mirror daemon (**cephfs-mirror**) gets asynchronous notifications about changes in the CephFS mirroring status, along with peer updates. You can query the **cephfs-mirror** admin socket with commands to retrieve the mirror status and peer status.

Prerequisites

- At least one deployment of a Ceph File System with mirroring enabled.
- Root-level access to the node running the CephFS mirroring daemon.

Procedure

1. Find the Ceph File System ID:

Syntax

```
ceph --admin-daemon PATH_TO_THE_ASOK_FILE help
```

Example

```
[root@mon ~]# ceph --admin-daemon /var/run/ceph/ceph-client.cephfs-
mirror.node1.bndvox.asok help
{
    ...
    "fs mirror peer status cephfs@11 1011435c-9e30-4db6-b720-5bf482006e0e": "get peer
mirror status",
```

```
    "fs mirror status cephfs@11": "get filesystem mirror status",
    ...
}
```

The Ceph File System ID in this example is **cephfs@11**.

2. To view the mirror status:

   Syntax

   ```
   ceph --admin-daemon PATH_TO_THE_ASOK_FILE fs mirror status
   FILE_SYSTEM_NAME@_FILE_SYSTEM_ID
   ```

   Example

   ```
   [root@mon ~]# ceph --admin-daemon /var/run/ceph/ceph-client.cephfs-
   mirror.node1.bndvox.asok fs mirror status cephfs@11
   {
    "rados_inst": "192.168.0.5:0/1476644347",
    "peers": {
       "1011435c-9e30-4db6-b720-5bf482006e0e": { (1)
          "remote": {
             "client_name": "client.mirror_remote",
             "cluster_name": "remote-site",
             "fs_name": "cephfs"
          }
       }
    },
    "snap_dirs": {
       "dir_count": 1
    }
   }
   ```

   (1) This is the unique peer UUID.

3. To view the peer status:

   Syntax

   ```
   ceph --admin-daemon PATH_TO_ADMIN_SOCKET fs mirror status
   FILE_SYSTEM_NAME@FILE_SYSTEM_ID _PEER_UUID
   ```

   Example

   ```
   [root@mon ~]# ceph --admin-daemon /var/run/ceph/cephfs-mirror.asok fs mirror peer status
   cephfs@11 1011435c-9e30-4db6-b720-5bf482006e0e
   {
    "/home/user1": {
       "state": "idle", (1)
       "last_synced_snap": {
          "id": 120,
          "name": "snap1",
          "sync_duration": 0.079997898999999997,
   ```

```
            "sync_time_stamp": "274900.558797s"
        },
        "snaps_synced": 2, (2)
        "snaps_deleted": 0, (3)
        "snaps_renamed": 0 (4)
    }
}
```

(1) The **state** can be one of these three values:

- **idle** means the directory is currently not being synchronized.
- **syncing** means the directory is currently being synchronized.
- **failed** means the directory has hit the upper limit of consecutive failures. The default number of consecutive failures is 10, and the default retry interval is 60 seconds.

(2) (3) (4) The synchronization stats: **snaps_synced**, **snaps_deleted**, and **snaps_renamed** are reset when the **cephfs-mirror** daemon restarts.

Additional Resources

- See the *Ceph File System mirrors* section in the *Red Hat Ceph Storage File System Guide* for more information.

5.22. ADDITIONAL RESOURCES

- For details, see Chapter 3, *Deployment of the Ceph File System*.
- For details, see the *Red Hat Ceph Storage Installation Guide*.
- For details, see the *The Ceph File System Metadata Server* in the *Red Hat Ceph Storage File System Guide*.

APPENDIX A. HEALTH MESSAGES FOR THE CEPH FILE SYSTEM

Cluster health checks

The Ceph Monitor daemons generate health messages in response to certain states of the Metadata Server (MDS). Below is the list of the health messages and their explanation:

mds rank(s) <ranks> have failed

One or more MDS ranks are not currently assigned to any MDS daemon. The storage cluster will not recover until a suitable replacement daemon starts.

mds rank(s) <ranks> are damaged

One or more MDS ranks has encountered severe damage to its stored metadata, and cannot start again until the metadata is repaired.

mds cluster is degraded

One or more MDS ranks are not currently up and running, clients might pause metadata I/O until this situation is resolved. This includes ranks being failed or damaged, and includes ranks which are running on an MDS but are not in the **active** state yet – for example, ranks in the **replay** state.

mds <names> are laggy

The MDS daemons are supposed to send beacon messages to the monitor in an interval specified by the **mds_beacon_interval** option, the default is 4 seconds. If an MDS daemon fails to send a message within the time specified by the **mds_beacon_grace** option, the default is 15 seconds. The Ceph Monitor marks the MDS daemon as **laggy** and automatically replaces it with a standby daemon if any is available.

Daemon-reported health checks

The MDS daemons can identify a variety of unwanted conditions, and return them in the output of the **ceph status** command. These conditions have human readable messages, and also have a unique code starting **MDS_HEALTH**, which appears in JSON output. Below is the list of the daemon messages, their codes, and explanation.

"Behind on trimming..."

Code: MDS_HEALTH_TRIM
CephFS maintains a metadata journal that is divided into log segments. The length of journal (in number of segments) is controlled by the **mds_log_max_segments** setting. When the number of segments exceeds that setting, the MDS starts writing back metadata so that it can remove (trim) the oldest segments. If this process is too slow, or a software bug is preventing trimming, then this health message appears. The threshold for this message to appear is for the number of segments to be double **mds_log_max_segments**.

"Client <name> failing to respond to capability release"

Code: MDS_HEALTH_CLIENT_LATE_RELEASE, MDS_HEALTH_CLIENT_LATE_RELEASE_MANY
CephFS clients are issued capabilities by the MDS. The capabilities work like locks. Sometimes, for example when another client needs access, the MDS requests clients to release their capabilities. If the client is unresponsive, it might fail to do so promptly, or fail to do so at all. This message appears if a client has taken a longer time to comply than the time specified by the **mds_revoke_cap_timeout** option (default is 60 seconds).

"Client <name> failing to respond to cache pressure"

Code: MDS_HEALTH_CLIENT_RECALL, MDS_HEALTH_CLIENT_RECALL_MANY
Clients maintain a metadata cache. Items, such as inodes, in the client cache are also pinned in the

MDS cache. When the MDS needs to shrink its cache to stay within its own cache size limits, the MDS sends messages to clients to shrink their caches too. If a client is unresponsive, it can prevent the MDS from properly staying within its cache size, and the MDS might eventually run out of memory and terminate unexpectedly. This message appears if a client has taken more time to comply than the time specified by the **mds_recall_state_timeout** option (default is 60 seconds). See Metadata Server cache size limits section for details.

"Client name failing to advance its oldest client/flush tid"

Code: MDS_HEALTH_CLIENT_OLDEST_TID, MDS_HEALTH_CLIENT_OLDEST_TID_MANY
The CephFS protocol for communicating between clients and MDS servers uses a field called **oldest tid** to inform the MDS of which client requests are fully complete so that the MDS can forget about them. If an unresponsive client is failing to advance this field, the MDS might be prevented from properly cleaning up resources used by client requests. This message appears if a client has more requests than the number specified by the **max_completed_requests** option (default is 100000) that are complete on the MDS side but have not yet been accounted for in the client's **oldest tid** value.

"Metadata damage detected"

Code: MDS_HEALTH_DAMAGE
Corrupt or missing metadata was encountered when reading from the metadata pool. This message indicates that the damage was sufficiently isolated for the MDS to continue operating, although client accesses to the damaged subtree return I/O errors. Use the **damage ls** administration socket command to view details on the damage. This message appears as soon as any damage is encountered.

"MDS in read-only mode"

Code: MDS_HEALTH_READ_ONLY
The MDS has entered into read-only mode and will return the **EROFS** error codes to client operations that attempt to modify any metadata. The MDS enters into read-only mode:

- If it encounters a write error while writing to the metadata pool.
- If the administrator forces the MDS to enter into read-only mode by using the **force_readonly** administration socket command.

"<N> slow requests are blocked"

Code: MDS_HEALTH_SLOW_REQUEST
One or more client requests have not been completed promptly, indicating that the MDS is either running very slowly, or encountering a bug. Use the **ops** administration socket command to list outstanding metadata operations. This message appears if any client requests have taken more time than the value specified by the **mds_op_complaint_time** option (default is 30 seconds).

"Too many inodes in cache"

Code: MDS_HEALTH_CACHE_OVERSIZED

The MDS has failed to trim its cache to comply with the limit set by the administrator. If the MDS cache becomes too large, the daemon might exhaust available memory and terminate unexpectedly. By default, this message appears if the MDS cache size is 50% greater than its limit.

Additional Resources

- See the *Metadata Server cache size limits* section in the *Red Hat Ceph Storage File System Guide* for details.

APPENDIX B. METADATA SERVER DAEMON CONFIGURATION REFERENCE

Refer the list commands that can be used for Metadata Server (MDS) daemon configuration.

mon_force_standby_active

Description

If set to **true**, monitors force MDS in standby replay mode to be active. Set under the **[mon]** or **[global]** section in the Ceph configuration file.

Type

Boolean

Default

true

max_mds

Description

The number of active MDS daemons during cluster creation. Set under the **[mon]** or **[global]** section in the Ceph configuration file.

Type

32-bit Integer

Default

1

mds_cache_memory_limit

Description

The memory limit the MDS enforces for its cache. Red Hat recommends to use this parameter instead of the **mds cache size** parameter.

Type

64-bit Integer Unsigned

Default

1073741824

mds_cache_reservation

Description

The cache reservation, memory or inodes, for the MDS cache to maintain. The value is a percentage of the maximum cache configured. Once the MDS begins dipping into its reservation, it recalls client state until its cache size shrinks to restore the reservation.

Type

Float

Default

0.05

mds_cache_size

Description

The number of inodes to cache. A value of 0 indicates an unlimited number. Red Hat recommends to use the **mds_cache_memory_limit** to limit the amount of memory the MDS cache uses.

Type

32-bit Integer

Default

0

mds_cache_mid

Description

The insertion point for new items in the cache LRU, from the top.

Type

Float

Default

0.7

mds_dir_commit_ratio

Description

The fraction of directory contains erroneous information before Ceph commits using a full update, instead of partial update.

Type

Float

Default

0.5

mds_dir_max_commit_size

Description

The maximum size of a directory update before Ceph breaks the directory into smaller transactions, in MB.

Type

32-bit Integer

Default

90

mds_decay_halflife

Description

The half-life of MDS cache temperature.

Type

Float

Default

5

mds_beacon_interval

Description

The frequency, in seconds, of beacon messages sent to the monitor.

Type

Float

Default

4

mds_beacon_grace

Description

The interval without beacons before Ceph declares an MDS **laggy**, and possibly replace it.

Type

Float

Default

15

mds_blacklist_interval

Description

The blacklist duration for failed MDS daemons in the OSD map.

Type

Float

Default

24.0*60.0

mds_session_timeout

Description

The interval, in seconds, of client inactivity before Ceph times out capabilities and leases.

Type

Float

Default

60

mds_session_autoclose

Description

The interval, in seconds, before Ceph closes a **laggy** client's session.

Type

Float

Default

300

mds_reconnect_timeout

Description

The interval, in seconds, to wait for clients to reconnect during MDS restart.

Type

Float

Default

45

mds_tick_interval

Description

How frequently the MDS performs internal periodic tasks.

Type

Float

Default

5

mds_dirstat_min_interval

Description

The minimum interval, in seconds, to try to avoid propagating recursive statistics up the tree.

Type

Float

Default

1

mds_scatter_nudge_interval

Description

How quickly changes in directory statistics propagate up.

Type

Float

Default

5

mds_client_prealloc_inos

Description

The number of inode numbers to preallocate per client session.

Type

32-bit Integer

Default

1000

mds_early_reply

Description

Determines whether the MDS allows clients to see request results before they commit to the journal.

Type

Boolean

Default

true

mds_use_tmap

Description

Use **trivialmap** for directory updates.

Type

Boolean

Default

true

mds_default_dir_hash

Description

The function to use for hashing files across directory fragments.

Type

32-bit Integer

Default

2,that is, **rjenkins**

mds_log

Description

Set to **true** if the MDS should journal metadata updates. Disable for benchmarking only.

Type

Boolean

Default

true

mds_log_skip_corrupt_events

Description

Determines whether the MDS tries to skip corrupt journal events during journal replay.

Type

Boolean

Default

false

mds_log_max_events

Description

The maximum events in the journal before Ceph initiates trimming. Set to **-1** to disable limits.

Type

32-bit Integer

Default

-1

mds_log_max_segments

Description

The maximum number of segments or objects, in the journal before Ceph initiates trimming. Set to **-1** to disable limits.

Type

32-bit Integer

Default

30

mds_log_max_expiring

Description

The maximum number of segments to expire in parallels.

Type

32-bit Integer

Default

20

mds_log_eopen_size

Description

The maximum number of inodes in an **EOpen** event.

Type

32-bit Integer

Default

100

mds_bal_sample_interval

Description

Determines how frequently to sample directory temperature, when making fragmentation decisions.

Type

Float

Default

3

mds_bal_replicate_threshold

Description

The maximum temperature before Ceph attempts to replicate metadata to other nodes.

Type

Float

Default

8000

mds_bal_unreplicate_threshold

Description

The minimum temperature before Ceph stops replicating metadata to other nodes.

Type

Float

Default

0

mds_bal_frag

Description

Determines whether the MDS fragments directories.

Type

Boolean

Default

false

mds_bal_split_size

Description

The maximum directory size before the MDS splits a directory fragment into smaller bits. The root directory has a default fragment size limit of 10000.

Type

32-bit Integer

Default

10000

mds_bal_split_rd

Description

The maximum directory read temperature before Ceph splits a directory fragment.

Type

Float

Default

25000

mds_bal_split_wr

Description

The maximum directory write temperature before Ceph splits a directory fragment.

Type

Float

Default

10000

mds_bal_split_bits

Description

The number of bits by which to split a directory fragment.

Type

32-bit Integer

Default

3

mds_bal_merge_size

Description

The minimum directory size before Ceph tries to merge adjacent directory fragments.

Type

32-bit Integer

Default

50

mds_bal_merge_rd

Description

The minimum read temperature before Ceph merges adjacent directory fragments.

Type

Float

Default

1000

mds_bal_merge_wr

Description

The minimum write temperature before Ceph merges adjacent directory fragments.

Type

Float

Default

1000

mds_bal_interval

Description

The frequency, in seconds, of workload exchanges between MDS nodes.

Type

32-bit Integer

Default

10

mds_bal_fragment_interval

Description

The frequency, in seconds, of adjusting directory fragmentation.

Type

32-bit Integer

Default

5

mds_bal_idle_threshold

Description

The minimum temperature before Ceph migrates a subtree back to its parent.

Type

Float

Default

0

mds_bal_max

Description

The number of iterations to run balancer before Ceph stops. For testing purposes only.

Type

32-bit Integer

Default

-1

mds_bal_max_until

Description

The number of seconds to run balancer before Ceph stops. For testing purposes only.

Type

32-bit Integer

Default

-1

mds_bal_mode

Description

The method for calculating MDS load:

- **1** = Hybrid.
- **2** = Request rate and latency.
- **3** = CPU load.

Type

32-bit Integer

Default

0

mds_bal_min_rebalance

Description

The minimum subtree temperature before Ceph migrates.

Type

Float

Default

0.1

mds_bal_min_start

Description

The minimum subtree temperature before Ceph searches a subtree.

Type

Float

Default

0.2

mds_bal_need_min

Description

The minimum fraction of target subtree size to accept.

Type

Float

Default

0.8

mds_bal_need_max

Description

The maximum fraction of target subtree size to accept.

Type

Float

Default

1.2

mds_bal_midchunk

Description

Ceph migrates any subtree that is larger than this fraction of the target subtree size.

Type

Float

Default

0.3

mds_bal_minchunk

Description

Ceph ignores any subtree that is smaller than this fraction of the target subtree size.

Type

Float

Default

0.001

mds_bal_target_removal_min

Description

The minimum number of balancer iterations before Ceph removes an old MDS target from the MDS map.

Type

32-bit Integer

Default

5

mds_bal_target_removal_max

Description

The maximum number of balancer iterations before Ceph removes an old MDS target from the MDS map.

Type

32-bit Integer

Default

10

mds_replay_interval

Description

The journal poll interval when in **standby-replay** mode for a **hot standby**.

Type

Float

Default

1

mds_shutdown_check

Description

The interval for polling the cache during MDS shutdown.

Type

32-bit Integer

Default

0

mds_thrash_exports

Description

Ceph randomly exports subtrees between nodes. For testing purposes only.

Type

32-bit Integer

Default

0

mds_thrash_fragments

Description

Ceph randomly fragments or merges directories.

Type

32-bit Integer

Default

0

mds_dump_cache_on_map

Description

Ceph dumps the MDS cache contents to a file on each MDS map.

Type

Boolean

Default

false

mds_dump_cache_after_rejoin

Description

Ceph dumps MDS cache contents to a file after rejoining the cache during recovery.

Type

Boolean

Default

false

mds_verify_scatter

Description

Ceph asserts that various scatter/gather invariants are **true**. For developer use only.

Type

Boolean

Default

false

mds_debug_scatterstat

Description

Ceph asserts that various recursive statistics invariants are **true**. For developer use only.

Type

Boolean

Default

false

mds_debug_frag

Description

Ceph verifies directory fragmentation invariants when convenient. For developer use only.

Type

Boolean

Default

false

mds_debug_auth_pins

Description

The debug authentication pin invariants. For developer use only.

Type

Boolean

Default

false

mds_debug_subtrees

Description

Debugging subtree invariants. For developer use only.

Type

Boolean

Default

false

mds_kill_mdstable_at

Description

Ceph injects MDS failure in MDS Table code. For developer use only.

Type

32-bit Integer

Default

0

mds_kill_export_at

Description

Ceph injects MDS failure in the subtree export code. For developer use only.

Type

32-bit Integer

Default

0

mds_kill_import_at

Description

Ceph injects MDS failure in the subtree import code. For developer use only.

Type

32-bit Integer

Default

0

mds_kill_link_at

Description

Ceph injects MDS failure in hard link code. For developer use only.

Type

32-bit Integer

Default

0

mds_kill_rename_at

Description

Ceph injects MDS failure in the rename code. For developer use only.

Type

32-bit Integer

Default

0

mds_wipe_sessions

Description

Ceph deletes all client sessions on startup. For testing purposes only.

Type

Boolean

Default

0

mds_wipe_ino_prealloc

Description

Ceph deletea inode preallocation metadata on startup. For testing purposes only.

Type

Boolean

Default

0

mds_skip_ino

Description

The number of inode numbers to skip on startup. For testing purposes only.

Type

32-bit Integer

Default

0

mds_standby_for_name

Description

The MDS daemon is a standby for another MDS daemon of the name specified in this setting.

Type

String

Default

N/A

mds_standby_for_rank

Description

An instance of the MDS daemon is a standby for another MDS daemon instance of this rank.

Type

32-bit Integer

Default

-1

mds_standby_replay

Description

Determines whether the MDS daemon polls and replays the log of an active MDS when used as a **hot standby**.

Type

Boolean

Default

false

APPENDIX C. JOURNALER CONFIGURATION REFERENCE

Reference of the list commands that can be used for journaler configuration.

journaler_write_head_interval

Description

How frequently to update the journal head object.

Type

Integer

Required

No

Default

15

journaler_prefetch_periods

Description

How many stripe periods to read ahead on journal replay.

Type

Integer

Required

No

Default

10

journal_prezero_periods

Description

How many stripe periods to zero ahead of write position.

Type

Integer

Required

No

Default

10

journaler_batch_interval

Description

Maximum additional latency in seconds to incur artificially.

Type

Double

Required

No

Default

.001

journaler_batch_max

Description

Maximum bytes that will be delayed flushing.

Type

64-bit Unsigned Integer

Required

No

Default

0

APPENDIX D. CEPH FILE SYSTEM CLIENT CONFIGURATION REFERENCE

This section lists configuration options for Ceph File System (CephFS) FUSE clients. Set them in the Ceph configuration file under the **[client]** section.

client_acl_type

Description

Set the ACL type. Currently, only possible value is **posix_acl** to enable POSIX ACL, or an empty string. This option only takes effect when the **fuse_default_permissions** is set to **false**.

Type

String

Default

"" (no ACL enforcement)

client_cache_mid

Description

Set the client cache midpoint. The midpoint splits the least recently used lists into a hot and warm list.

Type

Float

Default

0.75

client_cache size

Description

Set the number of inodes that the client keeps in the metadata cache.

Type

Integer

Default

16384 (16 MB)

client_caps_release_delay

Description

Set the delay between capability releases in seconds. The delay sets how many seconds a client waits to release capabilities that it no longer needs in case the capabilities are needed for another user space operation.

Type

Integer

Default

5 (seconds)

client_debug_force_sync_read

Description

If set to **true**, clients read data directly from OSDs instead of using a local page cache.

Type

Boolean

Default

false

client_dirsize_rbytes

Description

If set to **true**, use the recursive size of a directory (that is, total of all descendants).

Type

Boolean

Default

true

client_max_inline_size

Description

Set the maximum size of inlined data stored in a file inode rather than in a separate data object in RADOS. This setting only applies if the **inline_data** flag is set on the MDS map.

Type

Integer

Default

4096

client_metadata

Description

Comma-delimited strings for client metadata sent to each MDS, in addition to the automatically generated version, host name, and other metadata.

Type

String

Default

"" (no additional metadata)

client_mount_gid

Description

Set the group ID of CephFS mount.

Type

Integer

Default

-1

client_mount_timeout

Description

Set the timeout for CephFS mount in seconds.

Type

Float

Default

300.0

client_mount_uid

Description

Set the user ID of CephFS mount.

Type

Integer

Default

-1

client_mountpoint

Description

An alternative to the **-r** option of the **ceph-fuse** command.

Type

String

Default

/

client_oc

Description

Enable object caching.

Type

Boolean

Default

true

client_oc_max_dirty

Description

Set the maximum number of dirty bytes in the object cache.

Type

Integer

Default

104857600 (100MB)

client_oc_max_dirty_age

Description

Set the maximum age in seconds of dirty data in the object cache before writeback.

Type

Float

Default

5.0 (seconds)

client_oc_max_objects

Description

Set the maximum number of objects in the object cache.

Type

Integer

Default

1000

client_oc_size

Description

Set how many bytes of data will the client cache.

Type

Integer

Default

209715200 (200 MB)

client_oc_target_dirty

Description

Set the target size of dirty data. Red Hat recommends to keep this number low.

Type

Integer

Default

8388608 (8MB)

client_permissions

Description

Check client permissions on all I/O operations.

Type

Boolean

Default

true

client_quota_df

Description

Report root directory quota for the **statfs** operation.

Type

Boolean

Default

true

client_readahead_max_bytes

Description

Set the maximum number of bytes that the kernel reads ahead for future read operations. Overridden by the **client_readahead_max_periods** setting.

Type

Integer

Default

0 (unlimited)

client_readahead_max_periods

Description

Set the number of file layout periods (object size * number of stripes) that the kernel reads ahead. Overrides the **client_readahead_max_bytes** setting.

Type

Integer

Default

4

client_readahead_min

Description

Set the minimum number bytes that the kernel reads ahead.

Type

Integer

Default

131072 (128KB)

client_snapdir

Description

Set the snapshot directory name.

Type

String

Default

".snap"

client_tick_interval

Description

Set the interval in seconds between capability renewal and other upkeep.

Type

Float

Default

1.0

client_use_random_mds

Description

Choose random MDS for each request.

Type

Boolean

Default

false

fuse_default_permissions

Description

When set to **false**, the **ceph-fuse** utility checks does its own permissions checking, instead of relying on the permissions enforcement in FUSE. Set to false together with the **client acl type=posix_acl** option to enable POSIX ACL.

Type

Boolean

Default

true

DEVELOPER OPTIONS

These options are internal. They are listed here only to complete the list of options.

client_debug_getattr_caps

Description

Check if the reply from the MDS contains required capabilities.

Type

Boolean

Default

false

client_debug_inject_tick_delay

Description

Add artificial delay between client ticks.

Type

Integer

Default

0

client_inject_fixed_oldest_tid

Description, Type

Boolean

Default

false

client_inject_release_failure

Description, Type

Boolean

Default

false

client_trace

Description

The path to the trace file for all file operations. The output is designed to be used by the Ceph synthetic client. See the **ceph-syn(8)** manual page for details.

Type

String

Default

"" (disabled)