Red Hat Ceph Storage 3

Ceph File System Guide

Configuring and Mounting Ceph File Systems

Last Updated: 2018-09-07

Red Hat Ceph Storage 3 Ceph 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).

Table of Contents

CHAPTER 1. INTRODUCTION TO CEPH FILE SYSTEM

This chapter explains what the Ceph File System (CephFS) is and how it works.

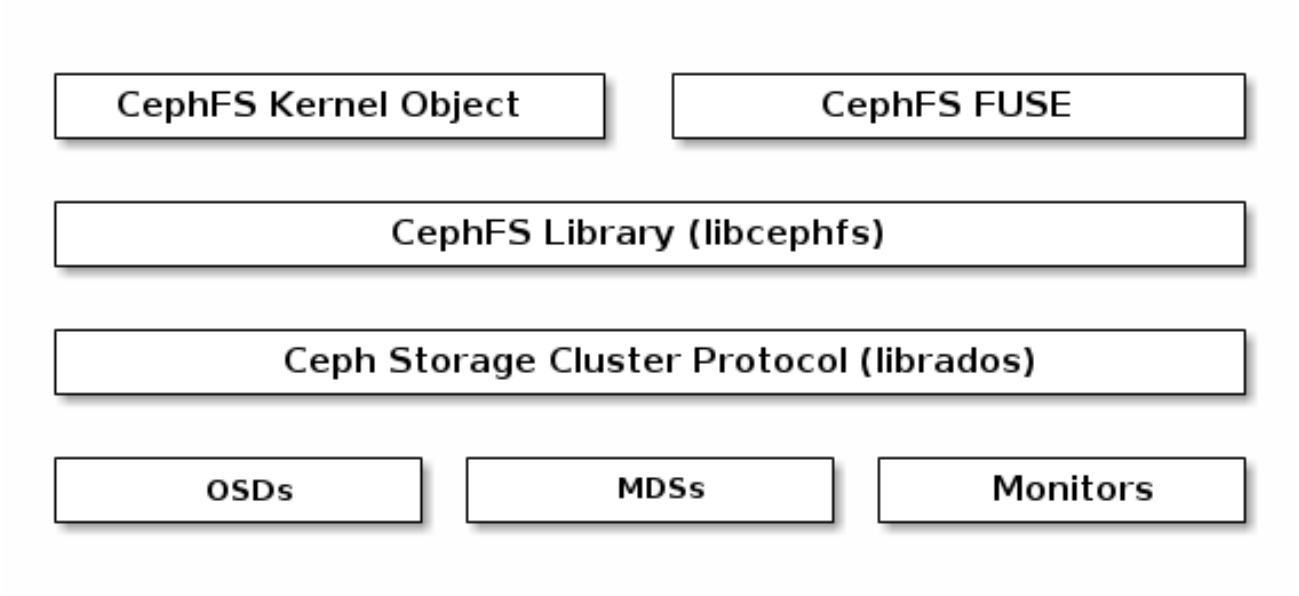
1.1. ABOUT THE CEPH FILE SYSTEM

The Ceph File System (CephFS) is a file system compatible with POSIX standards that provides a file access to a Ceph Storage Cluster.

The CephFS requires at least one Metadata Server (MDS) daemon (**ceph-mds**) to run. The MDS daemon manages metadata related to files stored on the Ceph File System and also coordinates access to the shared Ceph Storage Cluster.

CephFS 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. For details, see Section 1.4, “Differences from POSIX Compliance in the Ceph File System”.

The Ceph File System Components
This picture shows various layers of the Ceph File System.



The bottom layer represents the underlying core cluster that includes:

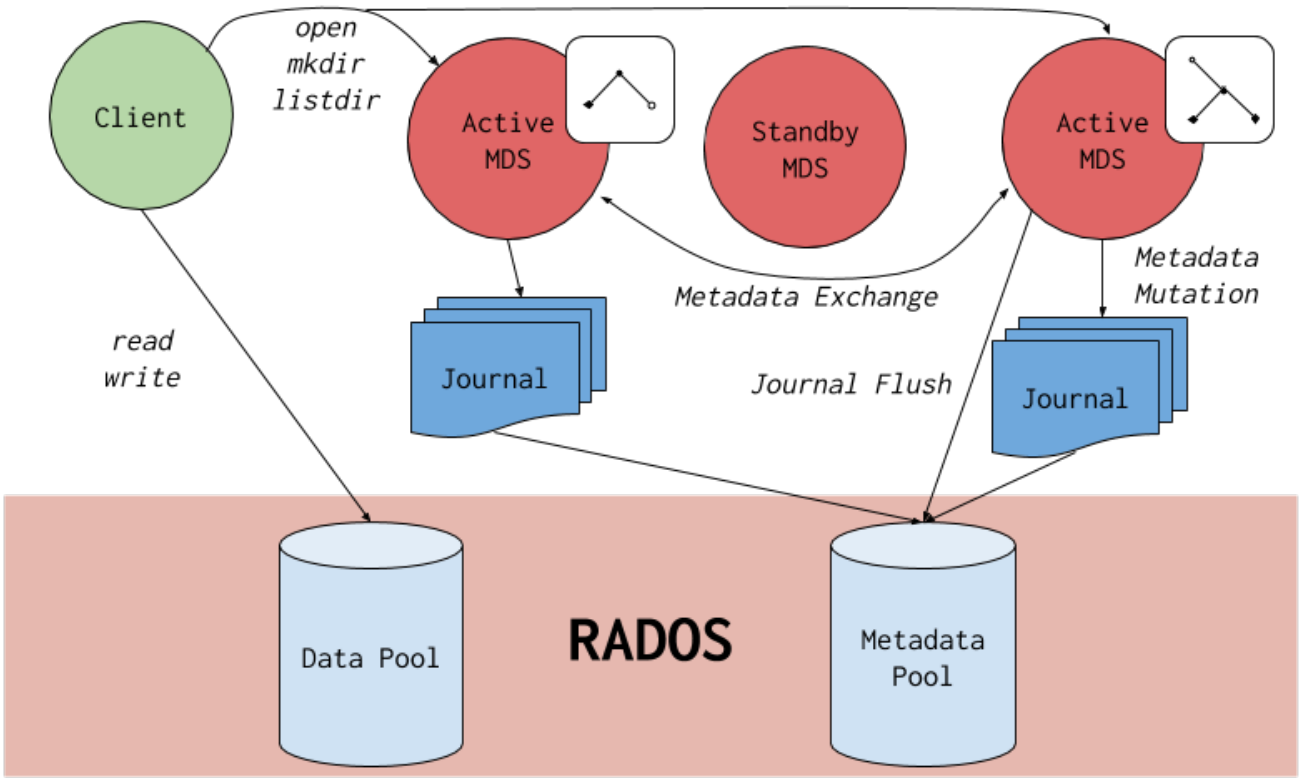
- OSDs (**ceph-osd**) where the Ceph File System data and metadata are stored
- Metadata Servers (**ceph-mds**) that manages Ceph File System metadata
- Monitors (**ceph-mon**) that manages the master copy of the cluster map

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

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

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

This picture shows in more detail how the Ceph File System components interact with each other.



The Ceph File System has the following primary components:

- *Clients* represent the entities performing I/O operations on behalf of applications using CephFS (**`ceph-fuse`** for FUSE clients and **`kcephfs`** for kernel clients). Clients send metadata requests to active MDS. In return, the client learns of file metadata and can begin safely caching both metadata and file data.

- *Metadata Servers* serves metadata to clients, caches hot metadata to reduce requests to the backing metadata pool store, manages client caches to maintain cache coherency, replicates hot metadata between active MDS, and coalesces metadata mutations to a compact journal with regular flushes to the backing metadata pool.

1.2. MAIN CEPHFS FEATURES

The Ceph File System introduces 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.

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. See Section 2.6, “Configuring Multiple Active Metadata Server Daemons” for details.

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. See Section 4.4, “Working with File and Directory Layouts” for details.

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`**.

To use ACL with the Ceph File Systems mounted as FUSE clients, you must enabled them. See Section 1.3, “CephFS Limitations” for details.

Client Quotas

The Ceph File System FUSE client 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. Client quotas are enabled by default.

1.3. CEPHFS LIMITATIONS

Access Control Lists (ACL) support in FUSE clients

To use the ACL feature with the Ceph File System mounted as a FUSE client, you must enable it. To do so, add the following options to the Ceph configuration file:

```
[client]

client_acl_type=posix_acl
```

Then restart the Ceph client.

Snapshots

Creating snapshots is not enabled by default because this feature is still experimental and it can cause the MDS or client nodes to terminate unexpectedly.

If you understand the risks and still wish to enable snapshots, use:

```
ceph mds set allow_new_snaps true --yes-i-really-mean-it
```

Multiple Ceph File Systems

By default, creation of multiple Ceph File Systems in one cluster is disabled. An attempt to create an additional Ceph File System fails with the following error:

```
Error EINVAL: Creation of multiple filesystems is disabled.
```

Creating multiple Ceph File Systems in one cluster is not fully supported yet and can cause the MDS or client nodes to terminate unexpectedly.

If you understand the risks and still wish to enable multiple Ceph file systems, use:

```
ceph fs flag set enable_multiple true  --yes-i-really-mean-it
```

1.4. DIFFERENCES FROM POSIX COMPLIANCE IN THE CEPH FILE SYSTEM

This section lists situations where the Ceph File System (CephFS) diverges from the strict POSIX semantics.

- 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. Because CephFS does not explicitly track which parts of a file are allocated or written, the **`st_blocks`** field is always populated by the file size divided by the block size. This behavior causes utilities, such as **`du`**, to overestimate consumed 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. You can change the name of this hidden directory at mount time with the **`-o snapdirname=.<new_name>`** option or by using the **`client_snapdir`** configuration option.

CHAPTER 2. CONFIGURING METADATA SERVER DAEMONS

This chapter explains how to configure Ceph Metadata Server MDS daemons.

- To understand different states of MDS daemons, see Section 2.2, “States of Metadata Server Daemons”.
- To understand what a "rank" mean in MDS configuration, see Section 2.3, “Explanation of Ranks in Metadata Server Configuration”.
- To learn about various configuration types of standby MDS daemons, see Section 2.4, “Types of Standby Configuration”.
- To configure standby MDS daemons, see Section 2.5, “Configuring Standby Metadata Server Daemons”.
- To configure multiple active MDS daemons, see Section 2.6, “Configuring Multiple Active Metadata Server Daemons”.
- To decrease the number of active MDS daemons, see Section 2.7, “Decreasing the Number of Active MDS Daemons”.
- To learn about MDS cache size limits, see Section 2.8, “Understanding MDS Cache Size Limits”.

2.1. PREREQUISITES

- Deploy a Ceph Storage Cluster if you do not have one. For details, see the *Installation Guide for Red Hat Enterprise Linux* or *Ubuntu*.
- Install Ceph Metadata Server daemons (**ceph-mds**). For details, see the *Installation Guide for Red Hat Enterprise Linux* or *Ubuntu*.

2.2. STATES OF METADATA SERVER DAEMONS

This section explains two different modes of Metadata Server (MDS) daemons and how a daemon in one mode starts operating in the other mode.

The MDS daemons can be:

- Active
- Standby

The **active** MDS daemon manages the metadata for files and directories stored on the Ceph File System. The **standby** MDS daemons serves as backup daemons and become active when an active MDS daemon becomes unresponsive.

By default, a Ceph File System uses only one active MDS daemon. However, you can configure the file system to use multiple active MDS daemons to scale metadata performance for larger workloads. The active MDS daemons will share the metadata workload with one another dynamically when metadata load patterns change. Typically, systems with many clients benefit from multiple active MDS daemons. 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 Monitor will wait the number of seconds specified by the **mds_beacon_grace** option. Then the Monitor marks the MDS daemon as **laggy** and one of the standby daemons becomes active depending on the configuration.

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

2.3. EXPLANATION OF RANKS IN METADATA SERVER CONFIGURATION

Each Ceph File System 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 Ceph File System metadata that is assigned to that rank.

Each MDS daemon initially starts without a rank. The Monitor assigns a rank to the daemon. An 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 Ceph File System 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 an 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 will not be assigned to any MDS daemons until the operators fixes the problem and uses the **ceph mds repaired** command on the damaged rank.

2.4. TYPES OF STANDBY CONFIGURATION

This section describes various types of standby daemons configuration.

Prerequisites

- Familiarize yourself with the meaning of *rank* in Ceph File System context. See Section 2.3, “Explanation of Ranks in Metadata Server Configuration” for details.

Configuration Parameters

By default, all Metadata Server daemons that do not hold a rank are standby daemons for any active daemon. However, you can configure how the MDS daemons behave in standby mode by using the following parameters in the Ceph configuration file.

- **mds_standby_replay** (Standby Replay)
- **mds_standby_for_name** (Standby for Name)
- **mds_standby_for_rank** (Standby for Rank)

- **mds_standby_for_fscid** (Standby for FSCID)

You can set these parameters in the Ceph configuration file on the host where the MDS daemon runs as opposed to the one on the Monitor node. The MDS daemon loads these settings when it starts and sends them to the Monitor node.

Standby Replay
When the **mds_standby_replay** option is set to **true** for a daemon, this daemon will continuously read the metadata journal of a rank associated with another MDS daemon (the **up** rank). This behavior gives the standby replay daemon a more recent metadata cache and makes the failover process faster if the daemon serving the rank fails.

An **up** rank can only have one standby replay daemon assigned to it. If two daemons are both set to be standby replay then one of them becomes a normal non-replay standby daemon.

If the **mon_force_standby_active** option is set to **false**, a standby replay daemon is only used as a standby for the rank that it is following. If another rank fails, the standby replay daemon will not be used as a replacement, even if no other standby daemons are available. By default, **mon_force_standby_active** is set to **true**.

Standby for Name
Each daemon has a static name that is set by the administrator when configuring the daemon for the first time. Usually, the host name of the host where the daemon runs is used as the daemon name.

When setting the **mds_standby_for_name** option, the standby daemon only takes over a failed rank if the name of the daemon that previously held the rank matches the given name.

Standby for Rank
Set the **mds_standby_for_rank** option to configure the standby daemon to only take over the specified rank. If another rank fails, this daemon will not replace it.

If you have multiple file systems, use this option in conjunction with the **mds_standby_for_fscid** option to specify which file system rank you target.

Standby for FSCID
The File System Cluster ID (FSCID) is an integer ID specific to a Ceph File System.

If the **mds_standby_for_fscid** option is used in conjunction with **mds_standby_for_rank** it only specifies which file system rank is referred to.

If **mds_standby_for_rank** is not set, then setting **mds_standby_for_fscid** causes the standby daemon to target any rank in the specified FSCID. Use **mds_standby_for_fscid** if you want to use the standby daemon for any rank, but only within a particular file system.

2.5. CONFIGURING STANDBY METADATA SERVER DAEMONS

This section describes how to configure Metadata Sever (MDS) daemons in standby mode to better manage a failure of the active MDS daemon.

Procedure

- Edit the Ceph configuration file. You can edit the main Ceph configuration file present on all nodes, or you can use different configuration files on each MDS node that contain just configuration related to that node. Use parameters described in Section 2.4, “Types of Standby Configuration”.
    - For example, to configure two MDS daemons **a** and **b** acting as a pair, where whichever one

- For example, to configure two MDS daemons **a** and **b** acting, as a pair, where whichever one has not currently assigned a rank will be the standby replay follower of the other:

```
[mds.a]
mds_standby_replay = true
mds_standby_for_rank = 0

[mds.b]
mds_standby_replay = true
mds_standby_for_rank = 0
```

- For example, to configure four MDS daemons (**a**, **b**, **c**, and **d**) on two Ceph File Systems, where each File System has a pair of daemons:

```
[mds.a]
mds_standby_for_fscid = 1

[mds.b]
mds_standby_for_fscid = 1

[mds.c]
mds_standby_for_fscid = 2

[mds.d]
mds_standby_for_fscid = 2
```

Additional Resources

- Section 2.2, “States of Metadata Server Daemons”

2.6. CONFIGURING MULTIPLE ACTIVE METADATA SERVER DAEMONS

This section describes how to 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 requires at least one standby MDS daemon to remain highly available.

Procedure

1. On a node with administration capabilities, set the **max_mds** parameter to the desired number of active MDS daemons. Note that Ceph only increases the actual number of ranks in the Ceph File Systems if a spare MDS daemon is available to take the new rank.

```
ceph fs set <name> max_mds <number>
```

   For example, to increase the number of active MDS daemons to two in the Ceph File System called **cephfs**:

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

2. Verify the number of active MDS daemons.

```
ceph fs status <name>
```

Specify the name of the Ceph File System, for example:

```
[root@monitor ~]# 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

- Section 2.2, “States of Metadata Server Daemons”
- Section 2.7, “Decreasing the Number of Active MDS Daemons”

2.7. DECREASING THE NUMBER OF ACTIVE MDS DAEMONS

This section describes how to decrease the number of active 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.

```
ceph fs status <name>
```

Specify the name of the Ceph File System, for example:

```
[root@monitor ~]# 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    |
+-------------+
```

Procedure

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

   ```
   ceph fs set <name> max_mds <number>
   ```

   For example, to decrease the number of active MDS daemons to one in the Ceph File System called **cephfs**:

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

2. Deactivate the active MDS daemon:

   ```
   ceph mds deactivate <role>
   ```

   Replace **<role>** with "name of the Ceph File System:rank", "FSID:rank", or just rank. For example, to deactivate the MDS daemon with rank 1 on the Ceph File System named **cephfs**:

   ```
   [root@monitor ~]# ceph mds deactivate cephfs:1
   telling mds.1:1 127.0.0.1:6800/3187061458 to deactivate
   ```

3. Verify the number of active MDS daemons.

   ```
   ceph fs status <name>
   ```

   Specify the name of the Ceph File System, for example:

   ```
   [root@monitor ~]# 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

- Section 2.2, “States of Metadata Server Daemons”
- Section 2.6, “Configuring Multiple Active Metadata Server Daemons”

2.8. UNDERSTANDING MDS CACHE SIZE LIMITS

This section describes ways to limit MDS cache size.

You can limit the size of the Metadata Server (MDS) cache by:

- **A memory limit**: A new behavior introduced in the Red Hat Ceph Storage 3. Use the **`mds_cache_memory_limit`** parameters. Red Hat recommends to use memory limits instead of inode count limits.
- **Inode count**: Use the **`mds_cache_size`** parameter. 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`** parameter 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`** parameter replaces the **`mds_health_cache_threshold`** in all situations except when MDS nodes sends a health alert to the 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`** configures the cluster health warning message so that operators can investigate why the MDS cannot shrink its cache.

Additional Resources

- MDS Configuration Reference

2.9. ADDITIONAL RESOURCES

- The *Installation Guide for Red Hat Enterprise Linux*
- The *Installation Guide for Ubuntu*

CHAPTER 3. DEPLOYING CEPH FILE SYSTEMS

This chapter describes how to create and mount Ceph File Systems.

To deploy a Ceph File System:

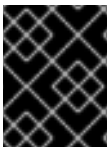
1. Create a Ceph file system on a Monitor node. See Section 3.2, “Creating the Ceph File Systems” for details.
2. If you use the **cephx** authentication, create a client keyring with capabilities that specify client access rights and permissions and copy the keyring to the node where the Ceph File System will be mounted. See Section 3.3, “Creating Ceph File System Client Users” for details.
3. Mount CephFS on a dedicated node. Choose one of the following methods:
   a. Mounting CephFS as a kernel client. See Section 3.4, “Mounting the Ceph File System as a kernel client”
   b. Mounting CephFS as a FUSE client. See Section 3.5, “Mounting the Ceph File System as a FUSE Client”

3.1. PREREQUISITES

- Deploy a Ceph Storage Cluster if you do not have one. For details, see the *Installation Guide for Red Hat Enterprise Linux* or *Ubuntu*.
- Install and configure Ceph Metadata Server daemons (**`ceph-mds`**). For details, see the the *Installation Guide for Red Hat Enterprise Linux* or *Ubuntu* and Chapter 2, *Configuring Metadata Server Daemons*.

3.2. CREATING THE CEPH FILE SYSTEMS

This section describes how to create a Ceph File System on a Monitor node.



IMPORTANT

By default, you can create only one Ceph File System in the Ceph Storage Cluster. See Section 1.3, “CephFS Limitations” for details.

Prerequisites

- On the Monitor node, enable the Red Hat Ceph Storage 3 Tools repository.
  - On Red Hat Enterprise Linux, use:

```
[root@monitor ~]# subscription-manager repos enable rhel-7-
server-rhceph-3-tools-rpms
```

  - On Ubuntu, use:

```
[user@monitor ~]$ sudo bash -c 'umask 0077; echo deb
https://customername:customerpasswd@rhcs.download.redhat.com/3-
updates/Tools $(lsb_release -sc) main | tee
/etc/apt/sources.list.d/Tools.list'
```

```
[user@monitor ~]$ sudo bash -c 'wget -O -
https://www.redhat.com/security/fd431d51.txt | apt-key add -'
[user@monitor ~]$ sudo apt-get update
```

Procedure

Use the following commands from a Monitor host and as the **root** user.

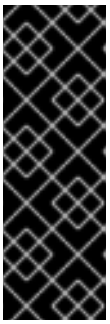
1. Create two pools, one for storing data and one for storing metadata:

   ```
   ceph osd pool create <name> <pg_num>
   ```

   Specify the pool name and the number of placement groups (PGs), for example:

   ```
   [root@monitor ~]# ceph osd pool create cephfs-data 64
   [root@monitor ~]# ceph osd pool create cephfs-metadata 64
   ```

   Typically, the metadata pool can start with a conservative number of PGs as it will generally have far fewer objects than the data pool. It is possible to increase the number of PGs if needed. Size the data pool proportional to the number and sizes of files you expect in the file system.

   IMPORTANT

   For the metadata pool, consider to use

   - A higher replication level because any data loss to this pool can make the whole file system inaccessible
   - Storage with lower latency such as Solid-state Drive (SSD) disks because this directly affects the observed latency of file system operations on clients

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

   - On Red Hat Enterprise Linux, use:

     ```
     [root@monitor ~]# yum install ceph-common
     ```

   - On Ubuntu, use:

     ```
     [user@monitor ~]$ sudo apt-get install ceph-common
     ```

3. Create the Ceph File System:

   ```
   ceph fs new <name> <metadata-pool> <data-pool>
   ```

   Specify the name of the Ceph File System, the metadata and data pool, for example:

   ```
   [root@monitor ~]# ceph fs new cephfs cephfs-metadata cephfs-data
   ```

4. Verify that one or more MDSs enter to the active state based on you configuration.

   ```
   ceph fs status <name>
   ```

   Specify the name of the Ceph File System, for example:

```
[root@monitor ~]# 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

- The *Enabling the Red Hat Ceph Storage Repositories* section in Red Hat Ceph Storage 3 *Installation Guide for Red Hat Enterprise Linux*
- The *Enabling the Red Hat Ceph Storage Repositories* Red Hat Ceph Storage 3 *Installation Guide for Ubuntu*
- The Pools chapter in the *Storage Strategies* guide for Red Hat Ceph Storage 3

3.3. CREATING CEPH FILE SYSTEM CLIENT USERS

If you use the **cephx** authentication, you must create a user for Ceph File System clients with correct authentication capabilities on a Monitor node and copy it to the node where the Ceph File System will be mounted.

Procedure

1. On a Monitor host, create a client user.

   ```
   ceph auth get-or-create client.<id> <capabilities>
   ```

   Specify the client ID and desired capabilities.

   - To restrict the client to only mount and work within a certain directory:

     ```
     ceph auth get-or-create client.1 mon 'allow r' mds 'allow r,
     allow rw path=<directory>' osd 'allow rw'
     ```

     For example, to restrict the client to only mount and work within the **/home/cephfs/** directory:

     ```
     [root@monitor ~]# ceph auth get-or-create client.1 mon 'allow r'
     ```

```
mds 'allow r, allow rw path=/home/cephfs' osd 'allow rw'
[client.1]
    key = AQACNoZXhrzqIRAABPKHTach4x03JeNadeQ9Uw==
```

- To restrict the client to only write to and read from a particular pool in the cluster:

```
ceph auth get-or-create client.1 mon 'allow r' mds 'allow rw' osd
'allow rw pool=<pool>'
```

For example, to restrict the client to only write to and read from the **data** pool:

```
[root@monitor ~]# ceph auth get-or-create client.1 mon 'allow r'
mds 'allow rw' osd 'allow rw pool=data'
```

- To prevent the client from modifying the data pool that is used for files and directories:

```
[root@monitor ~]# ceph auth get-or-create client.1 mon 'allow r'
mds 'allow rwp' osd 'allow rw'
```

2. Verify the created key:

```
ceph auth get client.<id>
```

For example:

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

3. Copy the client keyring from the Monitor host to the **/etc/ceph/** directory on the client host.

```
scp root@<monitor>:/etc/ceph/ceph.client.1.keyring
/etc/ceph/ceph.client.1.keyring
```

Replace **<monitor>** with the Monitor host name or IP, for example:

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

4. Set the appropriate permissions for the keyring file.

```
chmod 644 <keyring>
```

Specify the path to the keyring, for example:

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

Additional Resources

- The *User Management* chapter in the *Administration Guide* for Red Hat Ceph Storage 3

3.4. MOUNTING THE CEPH FILE SYSTEM AS A KERNEL CLIENT

You can mount the Ceph File System as a kernel client:

- Manually by using the **mount** command-line utility
- Automatically by adding an entry to the **/etc/fstab** file

IMPORTANT

Clients on Linux distributions aside from Red Hat Enterprise Linux (RHEL) are permitted but not supported. If there are issues found in the cluster (e.g. the MDS) when using these clients, Red Hat will address them, but if the cause is found to be on the client side, the issue will have to be addressed by the kernel vendor.

3.4.1. Prerequisites

- On the client node, enable the Red Hat Ceph Storage 3 Tools repository:
  - On Red Hat Enterprise Linux, use:

```
[root@client ~]# subscription-manager repos enable rhel-7-server-
rhceph-3-tools-rpms
```

  - On Ubuntu, use:

```
[user@client ~]$ sudo bash -c 'umask 0077; echo deb
https://customername:customerpasswd@rhcs.download.redhat.com/3-
updates/Tools $(lsb_release -sc) main | tee
/etc/apt/sources.list.d/Tools.list'
[user@client ~]$ sudo bash -c 'wget -O -
https://www.redhat.com/security/fd431d51.txt | apt-key add -'
[user@client ~]$ sudo apt-get update
```

- If you use the **cephx** authentication, create and copy the client keyring to the node client node. See Section 3.3, “Creating Ceph File System Client Users” for details.
- Copy the Ceph configuration file from a Monitor node to the client node.

```
scp root@<monitor>:/etc/ceph/ceph.conf /etc/ceph/ceph.conf
```

  Replace **<monitor>** with the Monitor host name or IP, for example:

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

- Set the appropriate permissions for the configuration file.

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

3.4.2. Manually Mounting the Ceph File System as a kernel Client

To manually mount the Ceph File System as a kernel client, use the **mount** utility.

Procedure

1. Create a mount directory.

```
mkdir -p <mount-point>
```

For example:

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

2. Mount the Ceph File System. To specify multiple monitor addresses, either separate them with commas in the **mount** command, or configure a DNS server so that a single host name resolves to multiple IP addresses and pass that host name to the **mount** command. If you use the **cephx** authentication, specify user and path to the secret file.

```
mount -t ceph <monitor1-host-name>:6789,<monitor2-host-name>:6789,
<monitor3-host-name>:6789:/ <mount-point> -o name=<user-
name>,secretfile=<path>
```

```
[user@client ~]$ mount -t ceph mon1:6789,mon2:6789,mon3:6789:/
/mnt/cephfs -o name=user,secretfile=/etc/ceph/user.secret
```

3. Verify that the file system is successfully mounted:

```
stat -f <mount-point>
```

For example:

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

Additional Resources

- The **mount(8)** manual page
- The DNS Servers chapter in the Networking Guide for Red Hat Enterprise Linux 7
- The User Management chapter in the Administration Guide for Red Hat Ceph Storage 3

3.4.3. Automatically Mounting the Ceph File System as a kernel Client

To automatically mount a Ceph File System on start, edit the **/etc/fstab** file.

Prerequisites

- Consider to mount the file system manually first. See Section 3.4.2, “Manually Mounting the Ceph File System as a kernel Client” for details.
- If you want to use the **secretefile=** mounting option, install the **ceph-common** package.

Procedure

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

```
mkdir -p <mount-point>
```

For example:

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

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

```
#DEVICE                 PATH           TYPE     OPTIONS
<host-name>:<port>:/,   <mount-point>  ceph     [name=<user-name>,
<host-name>:<port>:/,                         secret=<key>|
<host-name>:<port>:/                        secretfile=<file>,
                                              [<mount-options>]
```

Specify the Monitors host names, the ports they use, and the mount point. If you use the **cephx** authentication, specify the user name and the path to its secret or secret file. Note that the **secretfile** option requires the **ceph-common** package to be installed. In addition, specify required mount options. Consider to use the **_netdev** option that ensures that the file system is mounted after the networking subsystem to prevent networking issues. For example:

```
#DEVICE         PATH                    TYPE    OPTIONS
mon1:6789:/,    /mnt/cephfs             ceph    name=admin,
mon2:6789:/,                                 secretfile=
mon3:6789:/                                     /home/secret.key,
                                                _netdev,noatime 00
```

The file system will be mounted on the next boot.

3.5. MOUNTING THE CEPH FILE SYSTEM AS A FUSE CLIENT

You can mount the Ceph File System as a File System in User Space (FUSE) client:

- Manually by using the **ceph-fuse** command-line utility
- Automatically by adding an entry to the **/etc/fstab** file

3.5.1. Prerequisites

- On the client node, enable the Red Hat Ceph Storage 3 Tools repository:
  - On Red Hat Enterprise Linux, use:

```
[root@client ~]# subscription-manager repos enable rhel-7-server-
rhceph-3-tools-rpms
```

  - On Ubuntu, use:

```
[user@client ~]$ sudo bash -c 'umask 0077; echo deb
https://customername:customerpasswd@rhcs.download.redhat.com/3-
updates/Tools $(lsb_release -sc) main | tee
/etc/apt/sources.list.d/Tools.list'
[user@client ~]$ sudo bash -c 'wget -O -
https://www.redhat.com/security/fd431d51.txt | apt-key add -'
[user@client ~]$ sudo apt-get update
```

- If you use the **cephx** authentication, create and copy the client keyring to the node client node. See Section 3.3, “Creating Ceph File System Client Users” for details.

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

```
scp root@<monitor>:/etc/ceph/ceph.conf /etc/ceph/ceph.conf
```

Replace **<monitor>** with the Monitor host name or IP, for example:

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

- Set the appropriate permissions for the configuration file.

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

3.5.2. Manually Mounting the Ceph File System as a FUSE Client

To mount a Ceph File System as a File System in User Space (FUSE) client, use the **ceph-fuse** utility.

Prerequisites

- On the node where the Ceph File System will be mounted, install the **ceph-fuse** package.
  - On Red Hat Enterprise Linux, use:

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

  - On Ubuntu, use:

```
[user@client ~]$ sudo apt-get install ceph-fuse
```

Procedure

1. Create a directory to serve as a mount point. Note that if you used the **path** option with MDS capabilities, the mount point must be within what is specified by **path**.

```
mkdir <mount-point>
```

For example:

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

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

```
ceph-fuse -n client.<client-name> <mount-point>
```

For example:

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

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

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

- If you restricted the client to a only mount and work within a certain directory, use the **-r** option to instruct the client to treat that path as its root:

  ```
  ceph-fuse -n client.<client-name/id> <mount-point> -r <path>
  ```

  For example, to instruct the client with ID **1** to treat the **/home/cephfs/** directory as its root:

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

3. Verify that the file system is successfully mounted:

   ```
   stat -f <mount-point>
   ```

   For example:

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

Additional Resources

- The **ceph-fuse(8)** manual page *
- The *User Management* chapter in the *Administration Guide* for Red Hat Ceph Storage 3

3.5.3. Automatically Mounting the Ceph File System as a FUSE Client

To automatically mount a Ceph File System on start, edit the **/etc/fstab** file.

Prerequisites

- Consider to mount the file system manually first. See Section 3.4.2, “Manually Mounting the Ceph File System as a kernel Client” for details.

Procedure

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

   ```
   mkdir -p <mount-point>
   ```

   For example:

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

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

   ```
   #DEVICE     PATH            TYPE        OPTIONS
   none        <mount-point>   fuse.ceph   ceph.id=<user-id>
                                           [,ceph.conf=<path>],
                                           _netdev,defaults  0 0
   ```

Specify the use ID, for example **admin**, not **client-admin**, and the mount point. Use the **conf** option if you store the Ceph configuration file somewhere else than in the default location. In addition, specify required mount options. Consider to use the **_netdev** option that ensures that the file system is mounted after the networking subsystem to prevent networking issues. For example:

```
#DEVICE     PATH         TYPE         OPTIONS
none        /mnt/ceph    fuse.ceph    ceph.id=admin,

ceph.conf=/etc/ceph/cluster.conf,
                                      _netdev,defaults  0 0
```

The file system will be mounted on the next boot.

CHAPTER 4. ADMINISTERING CEPH FILE SYSTEMS

This chapter describes common Ceph File System administrative tasks.

- To map a directory to a particular MDS rank, see Section 4.2, “Mapping Directory Trees to MDS Ranks”.
- To disassociate a directory from a MDS rank, see Section 4.3, “Disassociating Directory Trees from MDS Ranks”.
- To work with files and directory layouts, see Section 4.4, “Working with File and Directory Layouts”.
- To add a new data pool, see Section 4.5, “Adding Data Pools”.

4.1. PREREQUISITES

- Deploy a Ceph Storage Cluster if you do not have one. For details, see the *Installation Guide for Red Hat Enterprise Linux* or *Ubuntu*.
- Install and configure Ceph Metadata Server daemons (**ceph-mds**). For details, see the *Installation Guide for Red Hat Enterprise Linux* or *Ubuntu* Chapter 2, *Configuring Metadata Server Daemons*.
- Create and mount the Ceph File System. For details, see Chapter 3, *Deploying Ceph File Systems*.

4.2. MAPPING DIRECTORY TREES TO MDS RANKS

This section describes how 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 cluster.

IMPORTANT

Note that an internal balancer already dynamically spreads the application load. Therefore, map directory trees to ranks only 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

- Configure multiple active MDS daemons. See Section 2.6, “Configuring Multiple Active Metadata Server Daemons” for details.
- Ensure that the **attr** package is installed on the client node with mounted Ceph File System.

Procedure

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

```
setfattr -n ceph.dir.pin -v <rank> <directory>
```

For example, to assign the **/home/ceph-user/** directory all of its subdirectories to rank 2:

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

Additional Resources

- Section 4.3, “Disassociating Directory Trees from MDS Ranks”

4.3. DISASSOCIATING DIRECTORY TREES FROM MDS RANKS

This section describes how to disassociate a directory from a particular active Metadata Server (MDS) rank.

Prerequisites

- Ensure that the **attr** package is installed on the client node with mounted Ceph File System.

Procedure

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

```
setfattr -n ceph.dir.pin -v -1 <directory>
```

For example, to disassociate the **/home/ceph-user/** directory from a MDS rank:

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

Note that any separately mapped subdirectories of **/home/ceph-user/** are not affected.

Additional Resources

- Section 4.2, “Mapping Directory Trees to MDS Ranks”

4.4. WORKING WITH FILE AND DIRECTORY LAYOUTS

This section describes how to:

- Understand file and directory layouts
- Set the layouts
- View the layouts
- Remove the directory layouts

4.4.1. Prerequisites

- Make sure that the **attr** package is installed.

4.4.2. Understanding 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. See Layouts Inheritance for more details.

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`**

The File and Directory Layout Fields table lists available layout fields that you can set on files and directories.

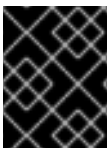
Table 4.1. File and Directory Layout Fields

Field	Description	Type
`pool`	ID or name of the pool to store file's data objects. Note that the pool must part of the set of data pools of the Ceph file system. See Section 4.5, "Adding Data Pools" for details.	string
`pool_namespace`	Namespace to write objects to. Empty by default, that means the default namespace is used.	string
`stripe_unit`	The size in bytes of a block of data used in the RAID 0 distribution of a file. All stripe units for a file have equal size. The last stripe unit is typically incomplete. That means it represents the data at the end of the file as well as unused space beyond it up to the end of the fixed stripe unit size.	integer
`stripe_count`	The number of consecutive stripe units that constitute a RAID 0 "stripe" of file data.	integer
`object_size`	Size of RADOS objects in bytes in which file data are chunked.	integer

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.

4.4.3. Setting File and Directory Layouts

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.

Procedure

- To modify layout fields on a file or directory:

```
setfattr -n ceph.<type>.layout.<field> -v <value> <path>
```

Replace:

- **<type>** with **file** or **dir**
- **<field>** with the name of the field, see the File and Directory Layouts Fields table for details.
- **<value>** with the new value of the field
- **<path>** with the path to the file or directory

For example, to set the **stripe_unit** field to **1048576** on the **test** file:

```
$ setfattr -n ceph.file.layout.stripe_unit -v 1048576 test
```

Additional Resources

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

4.4.4. Viewing File and Directory Layouts

This section describes how to use the **getfattr** command to view layout fields on a file or directory.

Procedure

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

```
getfattr -n ceph.<type>.layout <path>
```

Replace * **<path>** with the path to the file or directory * **<type>** with **file** or **dir**

For example, to view file layouts on the **/home/test/** file:

```
$ getfattr -n ceph.dir.layout /home/test
ceph.dir.layout="stripe_unit=4194304 stripe_count=2
object_size=4194304 pool=cephfs_data"
```

NOTE

Directories do not have an explicit layout until you set it (see Section 4.4.3, “Setting File and Directory Layouts”). Consequently, an attempt to view the layout fails if you never modified the layout.

- To view individual layout fields on a file or directory:

```
getfattr -n ceph.<type>.layout.<field> <path>
```

Replace:

- **<type>** with **file** or **dir**
- **<field>** with the name of the field, see the File and Directory Layouts Fields table for details.
- **<path>** with the path to the file or directory

For example, to view the **pool** field of the **test** file:

```
$ getfattr -n ceph.file.layout.pool test
ceph.file.layout.pool="cephfs_data"
```

NOTE

When viewing the **pool** field, the pool is usually indicated by its name. However, when you just created the pool, it can be indicated by its ID.

Additional Resources

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

4.4.5. Removing Directory Layouts

This section describes how to use the **setfattr** command to remove layouts from a directory.

NOTE

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

Procedure

- To remove a layout from a directory:

```
setfattr -x ceph.dir.layout <path>
```

Replace:

- **<path>** with the path to the directory

For example:

```
$ setfattr -x ceph.dir.layout /home/cephfs
```

- To remove the **pool_namespace** field:

```
$ setfattr -x ceph.dir.layout.pool_namespace <directory>
```

Replace:

- **<path>** with the path to the directory

For example:

```
$ 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

4.5. 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 (and optionally, within a namespace of that pool) using file and directory layouts. See Section 4.4, “Working with File and Directory Layouts” for details.

Procedure

Use the following commands from a Monitor host and as the **root** user.

1. Create a new data pool.

   ```
   ceph osd pool create <name> <pg_num>
   ```

   Replace:

   - **<name>** with the name of the pool
   - **<pg_num>** with the number of placement groups (PGs)

   For example:

   ```
   [root@monitor]# ceph osd pool create cephfs_data_ssd 64
   pool 'cephfs_data_ssd' created
   ```

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

   ```
   ceph mds add_data_pool <name>
   ```

   Replace:

- **<name>** with the name of the pool

For example:

```
[root@monitor]# ceph mds add_data_pool cephfs_data_ssd
added data pool 6 to fsmap
```

3. Verify that the pool was successfully added:

```
[root@monitor]# 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. See Section 3.3, “Creating Ceph File System Client Users” for details.

CHAPTER 5. UNMOUNTING CEPH FILE SYSTEMS

This chapter describes how to unmount Ceph File System mounted as kernel or File System in User Space (FUSE) clients.

5.1. UNMOUNTING CEPH FILE SYSTEMS MOUNTED AS KERNEL CLIENTS

This section shows how to unmount a Ceph File System that is mounted as a kernel client.

Procedure

- To unmount a Ceph File System mounted as a kernel client:

  ```
  umount <mount-point>
  ```

  Specify the mount point where the file system is mounted:

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

Additional Resources

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

5.2. UNMOUNTING CEPH FILE SYSTEMS MOUNTED AS FUSE CLIENTS

This section shows how to unmount a Ceph File System that is mounted as a File System in User Space (FUSE) client.

Procedure

- To unmount a Ceph File System mounted in FUSE:

  ```
  fusermount -u <mount-point>
  ```

  Specify the mount point where the file system is mounted

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

Additional Resources

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

APPENDIX A. TROUBLESHOOTING

A.1. CEPHFS HEALTH MESSAGES

Cluster health checks

The Ceph monitor daemons generate health messages in response to certain states of the MDS cluster. Below is the list of the cluster 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 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 additionally 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 (default is 4 seconds). If an MDS daemon fails to send a message within the time specified by the **`mds_beacon_grace`** option (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. This conditions have human readable messages, and additionally 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 Section 2.8, “Understanding MDS Cache Size Limits” 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 have 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 longer 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. This message appears if the MDS cache size is 50% greater than its limit (by default). See Section 2.8, “Understanding MDS Cache Size Limits” for details.

APPENDIX B. CONFIGURATION REFERENCE

B.1. MDS CONFIGURATION REFERENCE

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 max file size

Description

The maximum allowed file size to set when creating a new file system.

Type

64-bit Integer Unsigned

Default

`1ULL << 40`

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 (disabled 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 (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 (for 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 will fragment directories.

Type

Boolean

Default

`false`

mds bal split size

Description

The maximum directory size before the MDS will split a directory fragment into smaller bits.

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. Used 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. Used 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 will migrate any subtree that is larger than this fraction of the target subtree size.

Type

Float

Default

`0.3`

mds bal minchunk

Description

Ceph will ignore 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 (`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 will randomly export subtrees between nodes (testing only).

Type

32-bit Integer

Default

`0`

mds thrash fragments

Description

Ceph will randomly fragment or merge directories.

Type

32-bit Integer

Default

`0`

mds dump cache on map

Description

Ceph will dump the MDS cache contents to a file on each MDS map.

Type

Boolean

Default

`false`

mds dump cache after rejoin

Description

Ceph will dump MDS cache contents to a file after rejoining the cache during recovery.

Type

Boolean

Default

`false`

mds verify scatter

Description

Ceph will assert that various scatter/gather invariants are `true` (for developers only).

Type

Boolean

Default

`false`

mds debug scatterstat

Description

Ceph will assert that various recursive statistics invariants are `true` (for developers only).

Type

Boolean

Default

`false`

mds debug frag

Description

Ceph will verify directory fragmentation invariants when convenient (for developers only).

Type

Boolean

Default

`false`

mds debug auth pins

Description

The debug authentication pin invariants (for developers only).

Type

Boolean

Default

`false`

mds debug subtrees

Description

The debug subtree invariants (for developers only).

Type

Boolean

Default

`false`

mds kill mdstable at

Description

Ceph will inject MDS failure in MDS Table code (for developers only).

Type

32-bit Integer

Default

`0`

mds kill export at

Description

Ceph will inject MDS failure in the subtree export code (for developers only).

Type

32-bit Integer

Default

`0`

mds kill import at

Description

Ceph will inject MDS failure in the subtree import code (for developers only).

Type

32-bit Integer

Default

`0`

mds kill link at

Description

Ceph will inject MDS failure in hard link code (for developers only).

Type

32-bit Integer

Default

`0`

mds kill rename at

Description

Ceph will inject MDS failure in the rename code (for developers only).

Type

32-bit Integer

Default

`0`

mds wipe sessions

Description

Ceph will delete all client sessions on startup (for testing only).

Type

Boolean

Default

`0`

mds wipe ino prealloc

Description

Ceph will delete inode preallocation metadata on startup (for testing only).

Type

Boolean

Default

`0`

mds skip ino

Description

The number of inode numbers to skip on startup (for testing only).

Type

32-bit Integer

Default

`0`

mds standby for name

Description

The MDS daemon will 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 will be 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 (**`hot standby`**).

Type

Boolean

Default

`false`

B.2. JOURNALER CONFIGURATION REFERENCE

journaler allow split entries

Description

Allow an entry to span a stripe boundary.

Type

Boolean

Required

No

Default

`true`

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

B.3. FUSE CLIENT CONFIGURATION REFERENCE

This section lists configuration options for 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. See

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

IMPORTANT

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)