Red Hat Ceph Storage 2

Ceph File System Guide (Technology Preview)

Configuring and mounting Ceph file systems.

Last Updated: 2018-02-15

Red Hat Ceph Storage 2 Ceph File System Guide (Technology Preview)

Configuring and mounting Ceph file systems.

Legal Notice

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

Abstract

This guide describes how to create and configure the Ceph Metadata Server (MDS) and how to create and mount the Ceph File System (CephFS).

Table of Contents

CHAPTER 1. WHAT IS THE CEPH FILE SYSTEM (CEPHFS)? **3**
1.1. FEATURES 3
1.2. LIMITATIONS 4
1.3. DIFFERENCES FROM POSIX COMPLIANCE 5

CHAPTER 2. INSTALLING AND CONFIGURING CEPH METADATA SERVERS (MDS) **7**
2.1. PREREQUISITES 7
2.2. INSTALLING A CEPH METADATA SERVER 7
2.3. CONFIGURING A CEPH METADATA SERVER 7
2.3.1. Terminology 8
2.3.2. Configuring Standby Daemons 9

CHAPTER 3. CREATING CEPH FILE SYSTEMS **11**
3.1. PREREQUISITES 11
3.2. CREATING CEPH FILE SYSTEMS 11

CHAPTER 4. MOUNTING AND UNMOUNTING CEPH FILE SYSTEMS **13**
4.1. CLIENT AUTHENTICATION 13
4.2. MOUNTING CEPH FILE SYSTEMS AS KERNEL CLIENTS 15
4.3. MOUNTING CEPH FILE SYSTEMS IN USER SPACE (FUSE) 16
4.4. MOUNTING CEPH FILE SYSTEMS PERMANENTLY IN /ETC/FSTAB 18
4.5. UNMOUNTING CEPH FILE SYSTEMS 19

CHAPTER 5. TROUBLESHOOTING **20**
5.1. CEPHFS HEALTH MESSAGES 20

APPENDIX A. CONFIGURATION REFERENCE **22**
A.1. MDS CONFIGURATION REFERENCE 22
A.2. JOURNALER CONFIGURATION REFERENCE 36
A.3. FUSE CLIENT CONFIGURATION REFERENCE 37
Developer Options 43

CHAPTER 1. WHAT IS THE CEPH FILE SYSTEM (CEPHFS)?

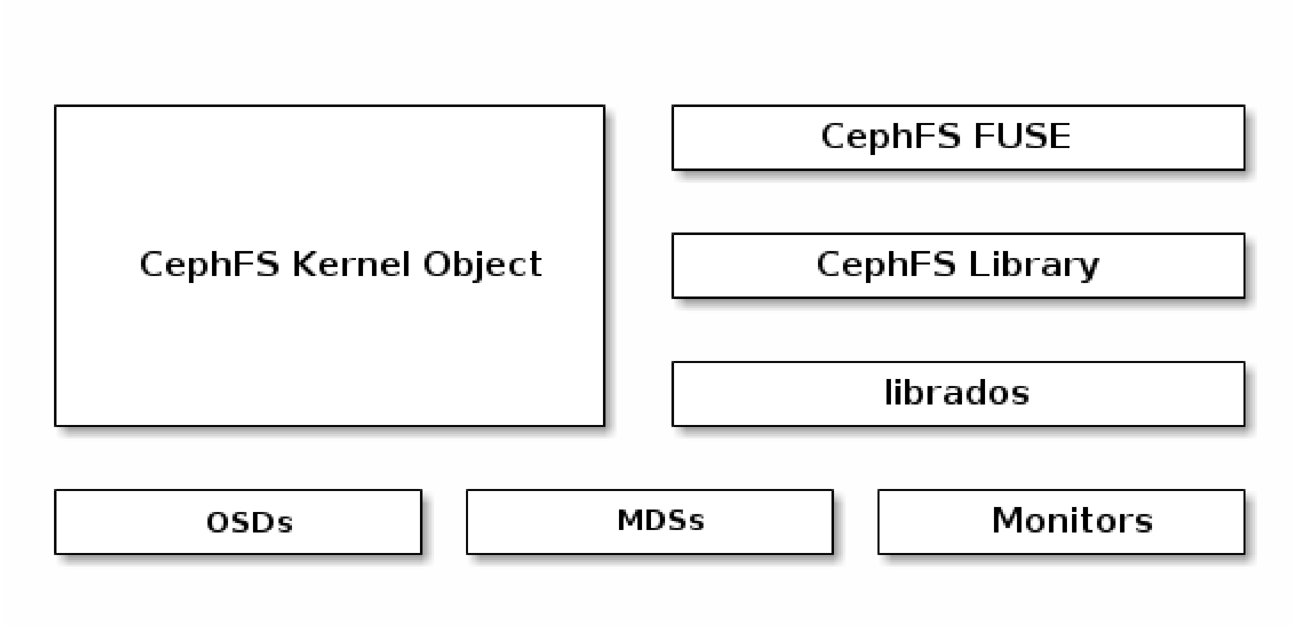
The Ceph File System (CephFS) is a file system compatible with POSIX standards that uses a Ceph Storage Cluster to store its data. The Ceph File System uses the same Ceph Storage Cluster system as the Ceph Block Device, Ceph Object Gateway, or `librados` API.

IMPORTANT

The Ceph File System is a Technology Preview only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information on Red Hat Technology Preview features support scope, see https://access.redhat.com/support/offerings/techpreview/.

In addition, see Section 1.2, "Limitations" for details on current CephFS limitations and experimental features.



To run the Ceph File System, you must have a running Ceph Storage Cluster with at least one Ceph Metadata Server (MDS) running. For details on installing the Ceph Storage Cluster, see the Installation Guide for Red Hat Enterprise Linux or Installation Guide for Ubuntu . See Chapter 2, *Installing and Configuring Ceph Metadata Servers (MDS)* for details on installing the Ceph Metadata Server.

1.1. FEATURES

The Ceph File System introduces the following features and enhancements:

Scalability

The Ceph File System is highly scalable because clients read directly from and write to all 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.

File and Directory Layouts

The Ceph File System allows users to configure file and directory layouts to use multiple pools.

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 1.2, “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.

To enable the client quotas, set the `client quota` option to `true` in the Ceph configuration file:

```
[client]

client quota = true
```

1.2. LIMITATIONS

The Ceph File System is provided as a Technical Preview and as such, there are several 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]

fuse_default_permission=0
client_acl_type=posix_acl
```

Then restart the Ceph services.

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 active MDS

By default, only configurations with one active MDS are supported. Having more active MDS can cause the Ceph File System to fail.
If you understand the risks and still wish to use multiple active MDS, increase the value of the **max_mds** option and set the **allow_multimds** option to **true** in the Ceph configuration file.

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
```

FUSE clients cannot be mounted permanently on Red Hat Enterprise Linux 7.2

The **util-linux** package shipped with Red Hat Enterprise Linux 7.2 does not support mounting CephFS FUSE clients in **/etc/fstab**. Red Hat Enterprise Linux 7.3 includes a new version of **util-linux** that supports mounting CephFS FUSE clients permanently.

The kernel clients in Red Hat Enterprise Linux 7.3 do not support the pool_namespace layout setting

As a consequence, files written from FUSE clients with a namespace set might not be accessible from Red Hat Enterprise Linux 7.3 kernel clients. Attempts to read or set the **ceph.file.layout.pool_namespace** extended attribute fail with the "No such attribute" error.

1.3. DIFFERENCES FROM POSIX COMPLIANCE

The Ceph File System aims to adhere to 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, there are a few places where CephFS diverges from strict POSIX semantics for various reasons:

- 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 the 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. INSTALLING AND CONFIGURING CEPH METADATA SERVERS (MDS)

The Ceph Metadata Server (MDS) node runs the MDS daemon (**ceph-mds**), which manages metadata related to files stored on the Ceph File System. The MDS daemon also coordinates access to the shared Ceph Storage Cluster.

2.1. PREREQUISITES

The following procedure assumes that:

- You have a working Ceph Storage Cluster (see the *Storage Cluster Installation* chapter in the Installation Guide for Red Hat Enterprise Linux or Installation Guide for Ubuntu).

- You have an administration node with Ansible installed (see the *Installing Ceph Ansible* section in the Installation Guide for Red Hat Enterprise Linux or Installation Guide for Ubuntu).

- On the MDS node, you have performed the tasks in listed the *Prerequisites* chapter of the Installation Guide for Red Hat Enterprise Linux or Installation Guide for Ubuntu . Especially, ensure to enable the Red Hat Ceph Storage 2 Tools repository. See Enabling Ceph Repositories on Red Hat Enterprise Linux or Enabling Ceph Repositories on Ubuntu for details.

2.2. INSTALLING A CEPH METADATA SERVER

Use the Ansible automation application to install a Ceph Metadata Server. Perform the following steps on the Ansible administration server:

1. Add a new section **[mdss]** to the **/etc/ansible/hosts** file:

   ```
   [mdss]
   <mdss-hostname>
   ```

   Replace **<mdss-hostname>** with the host name of the node where you want to install the Ceph Metadata Server.

2. Navigate to the Ansible configuration directory, **/usr/share/ceph-ansible/**:

   ```
   $ cd /usr/share/ceph-ansible
   ```

3. Run the Ansible playbook:

   ```
   $ ansible-playbook site.yml
   ```

2.3. CONFIGURING A CEPH METADATA SERVER

The Ceph Metadata Servers (MDS) have two modes:

- active

- standby

The first MDS that you started becomes **active**. The rest of the MDS are in **standby** mode.

When the active MDS becomes unresponsive, the monitor will wait the number of seconds specified by the **mds_beacon_grace** option. Then the monitor marks the MDS as **laggy**. When this happens, one of the standby servers becomes active depending on your configuration. See Section 2.3.2, "Configuring Standby Daemons" for details.

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

2.3.1. Terminology

FSCID

A Ceph cluster can have zero or more Ceph File Systems. Ceph File Systems have a human readable name (set by the **fs new** command) and an integer ID. The ID is called the File System Cluster ID, or **FSCID**.

Ranks

Each Ceph File System has a number of ranks, one by default, which start at zero.

Ranks are how the metadata workload is shared between multiple MDS (**ceph-mds**) daemons. The number of ranks is the maximum number of MDS daemons that may be active at one time. Each MDS handles the subset of the file system metadata that is assigned to that rank.

Each MDS daemon initially starts without a rank. The monitor cluster 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.

Ranks can be:

- **Up** - A rank is **up** once it is assigned to a daemon.
- **Failed** - A rank is **failed** if it is not associated with an instance of the MDS daemon.
- **Damaged** - A rank is damaged when its metadata is corrupted or missing. Damaged ranks will not be assigned to any MDS daemon until you fix the problem and use the **ceph mds repaired** command on the damaged rank.

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

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

Daemon 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.

GID

Each time a daemon starts, it is also assigned a **GID**, which is unique to the process lifetime of the daemon.

Referring to MDS daemons

Most of the administrative commands that refer to MDS daemons accept a flexible argument format.

A rank can be optionally qualified with a leading file system name or ID. If a daemon is in standby mode (meaning that it does not currently have a rank assigned), it can only be referred to by GID or name.

For example, an MDS daemon is called **myhost** and has GID 5446. It was assigned rank 0 in the file system **myfs**, which has FSCID 3. The following examples show possible forms of the **fail** command:

```
ceph mds fail 5446     # GID
ceph mds fail myhost   # Daemon name
ceph mds fail 0        # Unqualified rank
ceph mds fail 3:0      # FSCID and rank
ceph mds fail myfs:0   # File system name and rank
```

2.3.2. Configuring Standby Daemons

There are four configuration settings that control how daemons behave in standby mode:

- **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)

These settings can be set in the Ceph configuration file (**ceph.conf** by default) 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.

By default, if none of these settings are used, all MDS daemons that do not hold a rank will be used as standby daemons for any rank.

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**, then 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

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

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.

For more information about MSD configuration options, see Configuration Reference.

Configuration Examples

The following examples of parts of the Ceph configuration file can be:

- in the main Ceph configuration file present on all servers
- in different configuration files on each server that contain just configuration related to that server

Simple pair

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
```

Two MDS clusters

There are two file systems and four MDS daemons, each file 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
```

CHAPTER 3. CREATING CEPH FILE SYSTEMS

3.1. PREREQUISITES

To use the Ceph File System, you must have:

a working Ceph Storage Cluster

See the Installation Guide for Red Hat Enterprise Linux and Installation Guide for Ubuntu for details.

at least one Ceph Metadata Server

See Installing and Configuring Ceph Metadata Server (MDS) for details.

at least two pools; one for data and one for metadata

When configuring these pools, consider:

- Using a higher replication level for the metadata pool, as any data loss in this pool can render the whole file system inaccessible.
- Using storage with lower latency such as Solid-state Drive (SSD) disks for the metadata pool, because this directly affects the observed latency of file system operations on clients.

See the Pools chapter in the Storage Strategies guide for details on pools.

3.2. CREATING CEPH FILE SYSTEMS

Before creating the Ceph File System, ensure that you have the **ceph-common** package installed and if not, install it.

- On Red Hat Enterprise Linux:

  ```
  # yum install ceph-common
  ```

- On Ubuntu:

  ```
  $ sudo apt-get install ceph-common
  ```

To create a Ceph File System:

```
ceph fs new <file_system_name> <metadata> <pool>
```

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

```
$ ceph fs new cephfs cephfs-metadata cephfs_data
```

Once the file system is created, the Ceph Metadata Server (MDS) enters to the **active** state:

```
$ ceph mds stat
e5: 1/1/1 up {0=a=up:active}
```

After creating the Ceph File System, mount it. See Mounting Ceph File Systems for details.

NOTE

By default, only one Ceph File System can be created in a cluster. See Section 1.2, “Limitations” for details.

CHAPTER 4. MOUNTING AND UNMOUNTING CEPH FILE SYSTEMS

There are two ways to **temporarily** mount a Ceph File System:

- as a kernel client (Section 4.2, “Mounting Ceph File Systems as Kernel Clients”)
- using the FUSE client (Section 4.3, “Mounting Ceph File Systems in User Space (FUSE)”)

On details on mounting Ceph File Systems **permanently**, see Section 4.4, “Mounting Ceph File Systems Permanently in `/etc/fstab`”.

Before mounting a CephFS client, create a client keyring with capabilities that specifies client access rights and permissions. See Section 4.1, “Client Authentication” for details.

4.1. CLIENT AUTHENTICATION

To restrict the Ceph File System clients to the lowest possible level of authority needed, use Ceph authentication capabilities.

CephFS supports the following restrictions:

- path restriction
- OSD restriction
- layout modification restriction

Path Restriction

By default, clients are not restricted in what paths they are allowed to mount. Further, when clients mount a subdirectory, for example, `/home/<user>`, the MDS does not by default verify that subsequent operations are locked within that directory.

To restrict clients to only mount and work within a certain directory, use path-based MDS authentication capabilities. For example, to restrict the MDS daemon to write metadata only to a particular directory, specify that directory while creating the client capabilities:

```
ceph auth get-or-create client.<client-name/id> mon 'allow r' mds 'allow
r, allow rw path=<directory>' osd 'allow rw pool=data'
```

Example

The following example command restricts the MDS to write metadata only to the `/home/cephfs/` directory. Also, it restricts the CephFS client to perform read and write operations only within the `data` pool:

```
$ ceph auth get-or-create client.1 mon 'allow r' mds 'allow r, allow rw
path=/home/cephfs' osd 'allow rw pool=data'
[client.1]
    key = AQACNoZXhrzqIRAABPKHTach4x03JeNadeQ9Uw==
```

To view the created key:

```
$ ceph auth get client.1
```

```
exported keyring for client.1
[client.1]
    key = AQACNoZXhrzqIRAABPKHTach4x03JeNadeQ9Uw==
    caps mds = "allow r, allow rw path=/home/cephfs"
    caps mon = "allow r"
    caps osd = "allow rw pool=data"
```

Path restriction using the authentication capabilities is the most common way to restrict clients. See the User Management chapter in the Administration Guide for details on authentication capabilities.

When a client has capabilities that restrict it to a path, use the **-r** option with the **ceph-fuse** command so that the client will treat that path as its root:

```
ceph-fuse -n client.<client-name/id> --keyring=<path_to_keyring> <mount-
point> -r <directory>
```

Example

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

```
# ceph-fuse -n client.1 --keyring=/etc/ceph/client.1.keyring /mnt/cephfs -
r /home/cephf
```

NOTE

If you use the default location and name of the client keyring, that is **/etc/ceph/ceph.client.<client-ID>.keyring**, you do not have to use the **--keyring** option.

OSD restriction

To prevent clients from writing to or reading data from pools other than those in use for the Ceph File System, set an OSD authentication capability that restricts access to the CephFS data pools:

```
client.<client-name/id>
    key: <key>
    caps: [mds] allow rw
    caps: [mon] allow r
    caps: [osd] allow rw pool=<pool-name>
```

To restrict clients from writing data, use **r** instead of **rw**:

```
client.<client-name/id>
    key: <key>
    caps: [mds] allow rw
    caps: [mon] allow r
    caps: [osd] allow r pool=<pool-name>
```

This does not affect the ability of the clients to update file system metadata for files it has read access to, but it prevents them from persistently writing data in a way that would be visible to other clients.

Example:

To restrict client with id **1** to have read and write access to pool **data** and read access to pool **stack**:

```
client.1
    key: AQAz7EVWygILFRAAdIcuJ12opU/JKyfFmxhuaw==
    caps: [mds] allow rw
    caps: [mon] allow r
    caps: [osd] allow rw pool=data, allow r pool=stack
```

See the User Management chapter in the Administration Guide for details.

Layout Modification Restriction

To prevent clients from modifying the data pool used for files or directories, use the **p** modifier in MDS authentication capabilities.

Example

In the following snippet **`client.0`** can modify the pool used for files, but **`client.1`** cannot:

```
client.0
    key: AQAz7EVWygILFRAAdIcuJ12opU/JKyfFmxhuaw==
    caps: [mds] allow rwp
    caps: [mon] allow r
    caps: [osd] allow rw pool=data

client.1
    key: AQAz7EVWygILFRAAdIcuJ12opU/JKyfFmxhuaw==
    caps: [mds] allow rw
    caps: [mon] allow r
    caps: [osd] allow rw pool=data
```

4.2. MOUNTING CEPH FILE SYSTEMS AS KERNEL CLIENTS

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

1. On the client node, enable the Red Hat Ceph Storage 2 Tools repository. For details, see the Enabling Ceph Repositories section in the Installation Guide for Red Hat Enterprise Linux or the Enabling Ceph Repositories section in the Installation Guide for Ubuntu .

2. Ensure that the **`ceph-common`** package is installed on the client and if not, install it:

   - On Red Hat Enterprise Linux:

     ```
     # yum install ceph-common
     ```

   - On Ubuntu:

     ```
     $ sudo apt-get install ceph-common
     ```

3. 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. For details on setting DNS servers see the DNS Servers chapter in the Networking Guide for Red Hat Enterprise Linux 7.

```
mount -t ceph <monitor1-host-name>:6789,<monitor2-host-name>:6789,
<monitor3-host-name>:6789:/ <mount-point>
```

Example:

```
# mount -t ceph mon1:6789,mon2:6789,mon3:6789:/ /mnt/cephfs
```

To mount a Ceph File System with the **cephx** authentication enabled, specify a user name and a secret file:

```
mount -t ceph <monitor-hostname>:6789:/ <mount-point> -o name=
<username>, secretfile=<secret-file>
```

Example

```
# mount -t ceph mon1:6789:/ /mnt/cephfs -o
name=user,secretfile=/etc/ceph/user.secret
```

For details on **cephx**, see the User Management chapter in the Administration Guide.

For more information about **mount**, see the **mount(8)** manual page.

4.3. MOUNTING CEPH FILE SYSTEMS IN USER SPACE (FUSE)

To mount a Ceph File System as a FUSE client:

1. On the client node, enable the Red Hat Ceph Storage 2 Tools repository. For details, see the Enabling Ceph Repositories section in the Installation Guide for Red Hat Enterprise Linux or the Enabling Ceph Repositories section in the Installation Guide for Ubuntu.

2. Ensure that the **ceph-common** and **ceph-fuse** packages are installed on the client and if not, install them.

   - On Red Hat Enterprise Linux:

     ```
     # yum install ceph-common ceph-fuse
     ```

   - On Ubuntu:

     ```
     $ sudo apt-get ceph-common ceph-fuse
     ```

3. Copy the Ceph configuration file from the monitor host to the **/etc/ceph/** directory on the client host:

   ```
   scp root@<mon-host>:/etc/ceph/ceph.conf /etc/ceph/ceph.conf
   ```

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

   ```
   # scp root@192.168.0.1:/etc/ceph/ceph.conf /etc/ceph/ceph.conf
   ```

4. On the administration or monitor host, create the client user with correct authentication capabilities and output the user keyring to a file:

```
ceph auth get-or-create client.<client-name/id> mon 'allow r' mds
'allow r, allow rw path=<directory>' osd 'allow rw pool=<pool>' -o
<file_name>
```

Specify the client name or ID, the CephFS working directory, pool and the output file. For example:

```
$ ceph auth get-or-create client.1 mon 'allow r' mds 'allow r, allow
rw path=/' osd 'allow rw pool=data' -o ceph.client.1.keyring
[client.1]
 key = AQACNoZXhrzqIRAABPKHTach4x03JeNadeQ9Uw==
```

5. Copy the client keyring from the monitor host to the **/etc/ceph/** directory on the client host:

   ```
   scp root@<mon-host>:/ceph.client.1.keyring
   /etc/ceph/ceph.client.1.keyring
   ```

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

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

6. Ensure that the Ceph configuration file and the keyring have correct permissions:

   ```
   # chmod 644 /etc/ceph/ceph.conf
   # chmod 644 /etc/ceph/ceph.client.1.keyring
   ```

7. Create a directory to serve as a mount point. Note that the mount point must be within what is permitted by the client capabilities by the **path** option:

   ```
   $ mkdir <mountpoint>
   ```

   For example:

   ```
   $ mkdir /mnt/mycephfs
   ```

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

   ```
   ceph-fuse -n client.<client-name> -m <monitor1-host-name>:6789,
   <monitor2-host-name>:6789, <monitor3-host-name>:6789 <mountpoint>
   ```

   For example:

   ```
   # ceph-fuse -n client.1 -m mon1:6789, mon2:6789, mon3:6789
   /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:

   ```
   # ceph-fuse -n client.1 -m mon1:6789, mon2:6789, mon3:6789 --
   keyring=/etc/ceph/client1.keyring /mnt/mycephfs
   ```

-

For more information about **ceph-fuse** see the **ceph-fuse(8)** manual page.

4.4. MOUNTING CEPH FILE SYSTEMS PERMANENTLY IN /ETC/FSTAB

To automatically mount Ceph File Systems on startup, add them to the **/etc/fstab** file. The form of the entry depends on how the Ceph File System is mounted.

In all cases, use the **_netdev** option. This option ensures that the file system is mounted after the networking subsystem to prevent networking issues.

Ceph File System mounted as a kernel client

```
#DEVICE                        PATH            TYPE     OPTIONS
<mon1-hostanme>:<port>:/,  <mountpoint>   ceph     [name=username
<mon1-hostanme>:<port>:/,                         ,secret=secretkey|
<mon1-hostanme>:<port>:/             secretfile=
         path_to_secretfile],
         [<mount.options>]
```

Example

```
mon1:6789:/,      /mnt/cephfs   ceph     name=admin,
mon2:6789:/,     secretfile=
mon3:6789:/     /etc/ceph/secret.key,
          _netdev,
         noatime 0 0
```

IMPORTANT

The **name** and **secret** or **secretfile** options are mandatory when Ceph authentication is enabled.

Ceph File System mounted as a FUSE client

```
#DEVICE                                  PATH          TYPE
OPTIONS
id=<user-ID>[,conf=<configuration_file>] <mount-point> fuse.ceph _netdev,

defaults
         0 0
```

Examples

```
id=client1      /mnt/ceph  fuse.ceph   _netdev,
        defaults
        0 0
```

```
id=myuser,conf=/etc/ceph/ceph.conf   /mnt/ceph2  fuse.ceph  _netdev,
        defaults
        0 0
```

The **DEVICE** field is a comma-delimited list of options to pass to the command line. Ensure to use the ID (for example, **admin**, not **client.admin**). You can pass any valid **ceph-fuse** option to the command line this way.

IMPORTANT

The **util-linux** package shipped with Red Hat Enterprise Linux 7.2 does not support mounting CephFS FUSE clients in **/etc/fstab**. Red Hat Enterprise Linux 7.3 includes a new version of **util-linux** that supports mounting CephFS FUSE clients permanently.

4.5. UNMOUNTING CEPH FILE SYSTEMS

Unmounting Ceph File Systems mounted as kernel clients

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

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

Example

```
# umount /mnt/cephfs
```

See the **umount(8)** manual page for details.

Unmounting Ceph File Systems mounted as FUSE

To unmount a Ceph File System mounted in FUSE:

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

Example

```
# fusermount -u /mnt/cephfs
```

See the **ceph-fuse(8)** manual page for details.

CHAPTER 5. TROUBLESHOOTING

5.1. 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 the size specified by the `mds_cache_size` option, 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).

"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 actual cache size in inodes is at least 50% greater than the value specified by the `mds_cache_size` option (default is 100000).

APPENDIX A. CONFIGURATION REFERENCE

A.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. See Standby Replay for details.

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 size

Description

The number of inodes to cache.

Type

32-bit Integer

Default

`100000`

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`

A.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`

A.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 Path Restriction for details.

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

Description

Enable client quotas if set to `true`.

Type

`Boolean`

Default

`false`

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)