Red Hat Ceph Storage File Systems

As a storage administrator, you can configure the Ceph Metadata Server (MDS) and how to create, mount, and work the Ceph File System (CephFS). Get to know the features, system components, and limitations to manage a CephFS environment.

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

The Ceph File System has the following features:

Parallel data streams

CephFS is a parallel file system. Clients and storage servers can exchange data through multiple-streams in parallel, which helps increase throughput and scalability.

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 simultaneously.

Multiple file systems

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

High availability

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

Configurable file and directory layouts

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

POSIX Access Control Lists (ACL)

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

Client quotas

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

- For more information about installing Ceph Metadata servers, see Managing the MDS service.
- For more information about creating Ceph File Systems, see “Deploying the CephFS” on page 33.

Using CephFS

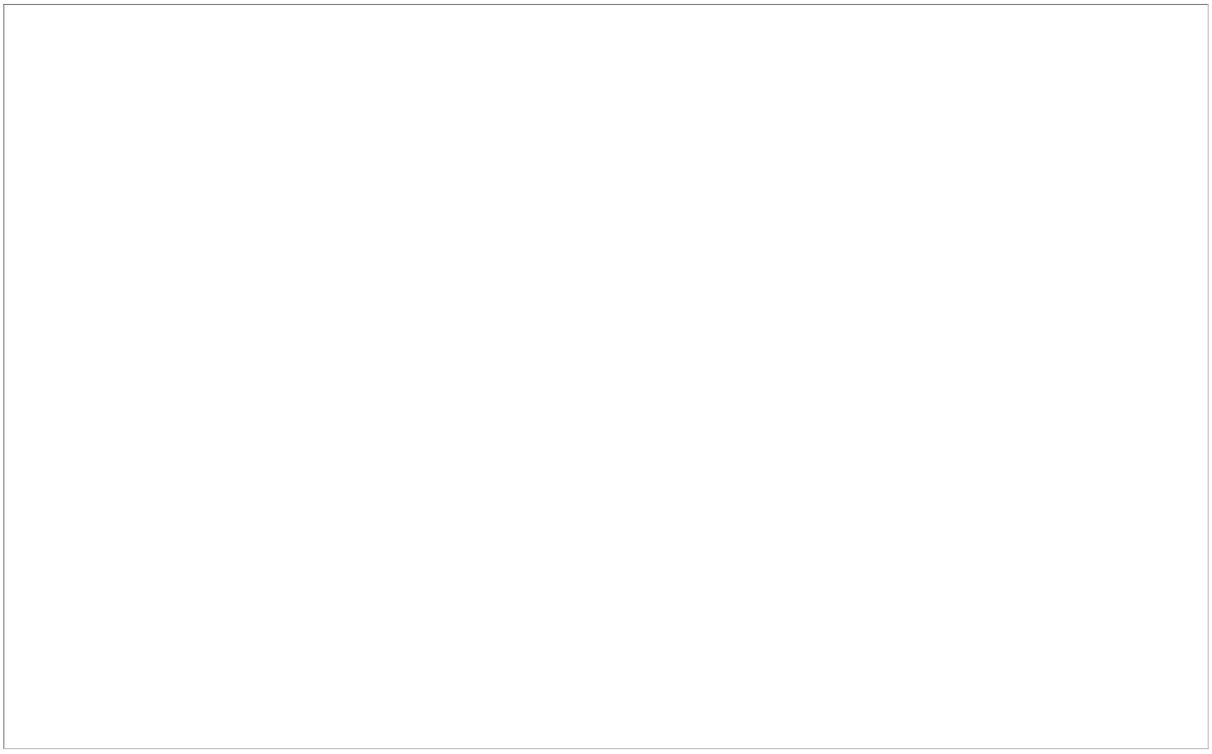
Understand how Ceph File Systems (CephFS) integrates within the Ceph cluster, its data flow, the role of the Metadata Server (MDS) and aspects of volume management.

Understanding the CephFS data flow

Understand the data flow in CephFS to recognize the interactions between the CephFS client, Metadata Server (MDS), and the Ceph Storage Cluster.

“Figure: CephFS Data Flow” on page 2 illustrates the role of Metadata Server in CephFS.

Figure: CephFS Data Flow



The following are the primary functions of the CephFS Client and Metadata Server:

CephFS Client

Mounting

Connects to CephFS by using either `kcephfs` or `ceph-fuse`.

Metadata requests

Requests metadata from the MDS for file operations.

Metadata Server

Metadata Operations

Manages file system metadata including file creation, directory management, and access control.

Caching

Improves performance by caching frequently accessed metadata. Caching is distributed and coordinated with clients and other multi-MDS

Ceph Storage Cluster

RADOS Storage

Handles distributed object storage.

OSDs

Manage data storage, retrieval, and replication across the cluster.

Data Access and Storage

Accessing Data

After retrieving metadata from the MDS, the client accesses data directly from OSDs. Data is stored as objects within the Ceph Storage Cluster, with the CRUSH algorithm ensuring efficient distribution and redundancy.

Comparison with other FSaaS Solutions

Following are some of the key comparisons between CephFS and other SaaS solutions:

Scalability

CephFS inherits all the benefits of the core Ceph internal storage engine RADOS. File systems can be increased in capacity and performance by adding more storage nodes. The data is automatically distributed and protetected.

Performance

CephFS is a parallel filesystem. Datastreams can run in parallel from multiple CephFS clients to multiple Ceph storage servers and OSDs. Metadata operations are running independently of the data operations.

Integration

CephFS is a POSIX-compatible file system. The integration of applications and infrastructure software such as Backup and Recovery solutions can rely on the POSIX nature of CephFS.

Feature Set

CephFS provides parallel data access and throughput and performance can be scaled up and down by the number of storage nodes and storage devices. CephFS like the rest of Ceph is designed to have no single point of failure and data redundancy is a built-in feature of the Ceph RADOS architecture. Available physical storage capacities are managed in thinly provisioned pools which all share the same infrastructure.

Data Protection in CephFS

Because CephFS builds on top of RADOS, it inherits all the data durability available from that system. File data and metadata are regularly scrubbed and node failures are automatically handled. CephFS also provides metadata scrubbing that you can start.

For geographically resilient scale storage, you can use CephFS snapshots and your favorite file system backup solution to generate consistent versions. If a data center disaster occurs, CephFS provides fsck like tools to rebuild the file system from data objects. For help if your system reports data issues, contact Red Hat Support.

Configuring CephFS

Learn how to configure Ceph File Systems and understand health messages.

Health messages

Understand the different health messages for the Ceph File System.

For more information, see “Configuring Metadata Server daemons” on page 5.

Cluster health checks

The Ceph Monitor daemons generate the following health messages in response to certain states of the Metadata Server (MDS):

mds rank(s) <ranks> have failed

One or more MDS ranks are not currently assigned to any MDS daemon. The storage cluster will not recover until a suitable replacement daemon starts.

mds rank(s) <ranks> are damaged

One or more MDS ranks has encountered severe damage to its stored metadata, and cannot start again until the metadata is repaired.

mds cluster is degraded

One or more MDS ranks are not currently up and running, clients might pause metadata I/O until this situation is resolved. This includes ranks being failed or damaged, and includes ranks which are running on an MDS but are not in the `active` state yet — for example, ranks in the `replay` state.

mds <names> are laggy

The MDS daemons are supposed to send beacon messages to the monitor in an interval specified by the `mds_beacon_interval` option, the default is 4 seconds. If an MDS daemon fails to send a message within the time specified by the `mds_beacon_grace` option, the default is 15 seconds. The Ceph Monitor marks the MDS daemon as `laggy` and automatically replaces it with a standby daemon if any is available.

Daemon-reported health checks

The MDS daemons can identify a variety of unwanted conditions, and return them in the output of the `ceph status` command. These conditions have human readable messages, and also have a unique code starting `MDS_HEALTH`, which appears in JSON output.

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`.

> **Note:** Increasing `mds_log_max_segments` is recommended if the trim warning is encountered. However, ensure that this configuration is reset back to its default when the cluster health recovers and the trim warning is seen no more. It is recommended to set `mds_log_max_segments` to 256 to allow the MDS to catch up with trimming.

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). For more information, see “Metadata Server cache size limits” on page 19.

Client *NAME* failing to advance its oldest client/flush tid

Code: `MDS_HEALTH_CLIENT_OLDEST_TID, MDS_HEALTH_CLIENT_OLDEST_TID_MANY`

The CephFS protocol for communicating between clients and MDS servers uses a field called **oldest tid** to inform the MDS of which client requests are fully complete so that the MDS can forget about them. If an unresponsive client is failing to advance this field, the MDS might be prevented from properly cleaning up resources used by client requests. This message appears if a client has more requests than the number specified by the `max_completed_requests` option (default is 100000) that are complete on the MDS side but have not yet been accounted for in the client's `oldest tid` value.

Metadata damage detected

Code: `MDS_HEALTH_DAMAGE`

Corrupt or missing metadata was encountered when reading from the metadata pool. This message indicates that the damage was sufficiently isolated for the MDS to continue operating, although client accesses to the damaged subtree return I/O errors. Use the `damage ls` administration socket command to view details on the damage. This message appears as soon as any damage is encountered.

MDS in read-only mode

Code: `MDS_HEALTH_READ_ONLY`

The MDS has entered into read-only mode and will return the EROFS error codes to client operations that attempt to modify any metadata. The MDS enters into read-only mode:

- If it encounters a write error while writing to the metadata pool.

– If the administrator forces the MDS to enter into read-only mode by using the **`force_readonly`** administration socket command.

***N* slow requests are blocked**

Code: `MDS_HEALTH_SLOW_REQUEST`

One or more client requests have not been completed promptly, indicating that the MDS is either running very slowly, or encountering a bug. Use the `ops` administration socket command to list outstanding metadata operations. This message appears if any client requests have taken more time than the value specified by the `mds_op_complaint_time` option (default is 30 seconds).

Too many inodes in cache

Code: `MDS_HEALTH_CACHE_OVERSIZED`

The MDS has failed to trim its cache to comply with the limit set by the administrator. If the MDS cache becomes too large, the daemon might exhaust available memory and terminate unexpectedly. By default, this message appears if the MDS cache size is 50% greater than its limit.

Configuring Metadata Server daemons

Use this list of commands for Metadata Server (MDS) daemon configuration.

Metadata Server (MDS) daemon configuration commands

Command	Description	Type	Default
`mon_force_standby_active`	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.	Boolean	`true`
`max_mds`	The number of active MDS daemons during cluster creation. Set under the `[mon]` or `[global]` section in the Ceph configuration file.	32-bit Integer	`1`
`mds_cache_memory_limit`	The memory limit that the MDS enforces for its cache. Use this parameter instead of the MDS cache size parameter.	64-bit Integer Unsigned	`1073741824`
`mds_cache_reservation`	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.	Float	`0.05`

Command	Description	Type	Default
mds_cache_size	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.	32-bit Integer	`0`
mds_cache_mid	The insertion point for new items in the cache LRU, from the top.	Float	`0.7`
mds_dir_commit_ratio	The fraction of directory that contains erroneous information before Ceph commits using a full update instead of partial update.	Float	`0.5`
mds_dir_max_commit_size	The maximum size of a directory update in MB before Ceph breaks the directory into smaller transactions.	32-bit Integer	`10`
mds_decay_halflife	The half-life of the MDS cache temperature.	Float	`5`
mds_beacon_interval	The frequency, in seconds, of beacon messages sent to the monitor.	Float	`4`
mds_beacon_grace	The interval without beacons before Ceph declares a MDS `laggy` and possibly replaces it.	Float	`15`
mds_blacklist_interval	The blacklist duration for failed MDS daemons in the OSD map.	Float	`24.0*60.0`
mds_session_timeout	The interval, in seconds, of client inactivity before Ceph times out capabilities and leases.	Float	`60`
mds_session_autoclose	The interval, in seconds, before Ceph closes a `laggy` client's session.	Float	`300`
mds_reconnect_timeout	The interval, in seconds, to wait for clients to reconnect during a MDS restart.	Float	`45`
mds_tick_interval	How frequently the MDS performs internal periodic tasks.	Float	`5`
mds_dirstat_min_interval	The minimum interval, in seconds, to try to avoid propagating recursive statistics up the tree.	Float	`1`

Command	Description	Type	Default
mds_scatter_nudge_interval	How quickly changes in directory statistics propagate up.	Float	5
mds_client_prealloc_inos	The number of inode numbers to preallocate per client session.	32-bit Integer	1000
mds_early_reply	Determines whether the MDS allows clients to see request results before they commit to the journal.	Boolean	true
mds_use_tmap	Use trivialmap for directory updates.	Boolean	true
mds_default_dir_hash	The function to use for hashing files across directory fragments.	32-bit Integer	2, that is, rjenkins
mds_log	Set to true if the MDS should journal metadata updates. Disable for benchmarking only.	Boolean	true
mds_log_skip_corrupt_events	Determines whether the MDS tries to skip corrupt journal events during journal replay.	Boolean	false
mds_log_max_events	The maximum events in the journal before Ceph initiates trimming. Set to -1 to disable limits.	32-bit Integer	-1
mds_log_max_segments	The maximum number of segments or objects in the journal before Ceph initiates trimming. Set to -1 to disable limits.	32-bit Integer	30
mds_log_max_expiring	The maximum number of segments to expire in parallels.	32-bit Integer	20
mds_log_eopen_size	The maximum number of inodes in an EOpen event.	32-bit Integer	100
mds_bal_sample_interval	Determines how frequently to sample directory temperature when making fragmentation decisions.	Float	3
mds_bal_replicate_threshold	The maximum temperature before Ceph attempts to replicate metadata to other nodes.	Float	8000
mds_bal_unreplicate_threshold	The minimum temperature before Ceph stops replicating metadata to other nodes.	Float	0
mds_bal_frag	Determines whether or not the MDS fragments directories.	Boolean	false

Command	Description	Type	Default
mds_bal_split_size	The maximum directory size before the MDS splits a directory fragment into smaller bits. The root directory has a default fragment size limit of 10000.	32-bit Integer	`10000`
mds_bal_split_rd	The maximum directory read temperature before Ceph splits a directory fragment.	Float	`25000`
mds_bal_split_wr	The maximum directory write temperature before Ceph splits a directory fragment.	Float	`10000`
mds_bal_split_bits	The number of bits by which to split a directory fragment.	32-bit Integer	`3`
mds_bal_merge_size	The minimum directory size before Ceph tries to merge adjacent directory fragments.	32-bit Integer	`50`
mds_bal_merge_rd	The minimum read temperature before Ceph merges adjacent directory fragments.	Float	`1000`
mds_bal_merge_wr	The minimum write temperature before Ceph merges adjacent directory fragments.	Float	`1000`
mds_bal_interval	The frequency, in seconds, of workload exchanges between MDS nodes.	32-bit Integer	`10`
mds_bal_fragment_interval	The frequency, in seconds, of adjusting directory fragmentation.	32-bit Integer	`5`
mds_bal_idle_threshold	The minimum temperature before Ceph migrates a subtree back to its parent.	Float	`0`
mds_bal_max	The number of iterations to run balancer before Ceph stops. For testing purposes only.	32-bit Integer	`-1`
mds_bal_max_until	The number of seconds to run balancer before Ceph stops. For testing purposes only.	32-bit Integer	`-1`
mds_bal_mode	The method for calculating MDS load:	32-bit Integer	`0`
mds_bal_min_rebalance	The minimum subtree temperature before Ceph migrates.	Float	`0.1`

Command	Description	Type	Default
mds_bal_min_start	The minimum subtree temperature before Ceph searches a subtree.	Float	`0.2`
mds_bal_need_min	The minimum fraction of target subtree size to accept.	Float	`0.8`
mds_bal_need_max	The maximum fraction of target subtree size to accept.	Float	`1.2`
mds_bal_midchunk	Ceph migrates any subtree that is larger than this fraction of the target subtree size.	Float	`0.3`
mds_bal_minchunk	Ceph ignores any subtree that is smaller than this fraction of the target subtree size.	Float	`0.001`
mds_bal_target_removal_min	The minimum number of balancer iterations before Ceph removes an old MDS target from the MDS map.	32-bit Integer	`5`
mds_bal_target_removal_max	The maximum number of balancer iterations before Ceph removes an old MDS target from the MDS map.	32-bit Integer	`10`
mds_replay_interval	The journal poll interval when in `standby-replay` mode for a `hot standby`.	Float	`1`
mds_shutdown_check	The interval for polling the cache during MDS shutdown.	32-bit Integer	`0`
mds_thrash_exports	Ceph randomly exports subtrees between nodes. For testing purposes only.	32-bit Integer	`0`
mds_thrash_fragments	Ceph randomly fragments or merges directories.	32-bit Integer	`0`
mds_dump_cache_on_map	Ceph dumps the MDS cache contents to a file on each MDS map.	Boolean	`false`
mds_dump_cache_after_rejoin	Ceph dumps MDS cache contents to a file after rejoining the cache during recovery.	Boolean	`false`
mds_verify_scatter	Ceph asserts that various scatter/gather invariants are `true`. For developer use only.	Boolean	`false`
mds_debug_scatterstat	Ceph asserts that various recursive statistics invariants are `true`. For developer use only.	Boolean	`false`

Command	Description	Type	Default
mds_debug_frag	Ceph verifies directory fragmentation invariants when convenient. For developer use only.	Boolean	`false`
mds_debug_auth_pins	The debug authentication pin invariants. For developer use only.	Boolean	`false`
mds_debug_subtrees	Debugging subtree invariants. For developer use only.	Boolean	`false`
mds_kill_mdstable_at	Ceph injects a MDS failure in a MDS Table code. For developer use only.	32-bit Integer	`0`
mds_kill_export_at	Ceph injects a MDS failure in the subtree export code. For developer use only.	32-bit Integer	`0`
mds_kill_import_at	Ceph injects a MDS failure in the subtree import code. For developer use only.	32-bit Integer	`0`
mds_kill_link_at	Ceph injects a MDS failure in a hard link code. For developer use only.	32-bit Integer	`0`
mds_kill_rename_at	Ceph injects a MDS failure in the rename code. For developer use only.	32-bit Integer	`0`
mds_wipe_sessions	Ceph deletes all client sessions on startup. For testing purposes only.	Boolean	`0`
mds_wipe_ino_prealloc	Ceph deletes inode preallocation metadata on startup. For testing purposes only.	Boolean	`0`
mds_skip_ino	The number of inode numbers to skip on startup. For testing purposes only.	32-bit Integer	`0`
mds_standby_for_name	The MDS daemon is a standby for another MDS daemon of the name specified in this setting.	String	N/A
mds_standby_for_rank	An instance of the MDS daemon is a standby for another MDS daemon instance of this rank.	32-bit Integer	`-1`
mds_standby_replay	Determines whether the MDS daemon polls and replays the log of an active MDS when used as a `hot standby`.	Boolean	`false`

Configuring journaler

Use this list of commands for journaler configuration.

journaler_write_head_interval

Description

How frequently to update the journal head object.

Type

Integer

Required

No

Default

`15`

journaler_prefetch_periods

Description

How many stripe periods to read ahead on journal replay.

Type

Integer

Required

No

Default

`10`

journaler_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`

Configuring Ceph File System mirrors

Use this list of commands for Ceph File System (CephFS) mirror configuration.

`cephfs_mirror_max_concurrent_directory_syncs`

Description

Maximum number of directory snapshots that can be synchronized concurrently by `cephfs-mirror` daemon. Controls the number of synchronization threads.

Type

Integer

Default

`3`

Min

`1`

`cephfs_mirror_action_update_interval`

Description

Interval in seconds to process pending mirror update actions.

Type

`secs`

Default

`2`

Min

`1`

`cephfs_mirror_restart_mirror_on_blocklist_interval`

Description

Interval in seconds to restart blocklisted mirror instances. Setting to zero (`0`) disables restarting blocklisted instances.

Type

`secs`

Default

`30`

Min

`0`

`cephfs_mirror_max_snapshot_sync_per_cycle`

Description

Maximum number of snapshots to mirror when a directory is picked up for mirroring by worker threads.

Type

Integer

Default

`3`

Min

`1`

`cephfs_mirror_directory_scan_interval`

Description

Interval in seconds to scan configured directories for snapshot mirroring.

Type

Integer

Default

`10`

Min

`1`

`cephfs_mirror_max_consecutive_failures_per_directory`

Description

Number of consecutive snapshot synchronization failures to mark a directory as “failed”. Failed directories are retried for synchronization less frequently.

Type

Integer

Default

`10`

Min

`0`

`cephfs_mirror_retry_failed_directories_interval`

Description

Interval in seconds to retry synchronization for failed directories.

Type

Integer

Default

`60`

Min

`1`

`cephfs_mirror_restart_mirror_on_failure_interval`

Description

Interval in seconds to restart failed mirror instances. Setting to zero (`0`) disables restarting failed mirror instances.

Type

`secs`

Default

`20`

Min

`0`

`cephfs_mirror_mount_timeout`

Description

Timeout in seconds for mounting primary or secondary CephFS by the `cephfs-mirror` daemon. Setting this to a higher value could result in the mirror daemon getting stalled when mounting a file system if the cluster is not reachable. This option is used to override the usual `client_mount_timeout`.

Type

`secs`

Default

`10`

Min

`0`

CephFS and the Metadata Server

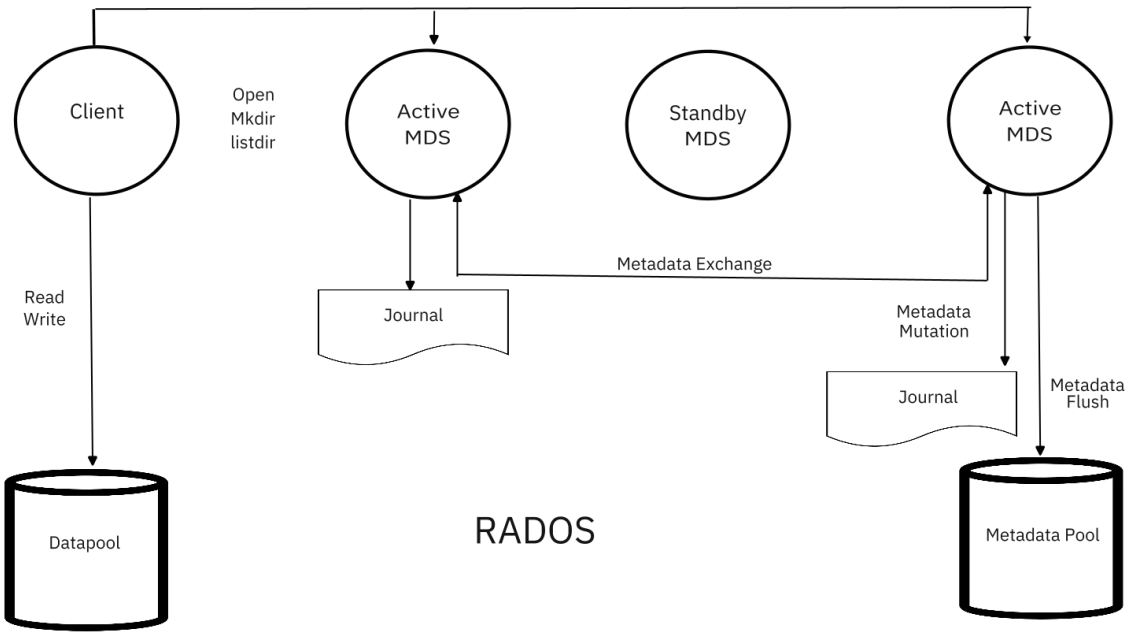
Ceph File System (CephFS) is a scalable distributed file system that relies on the Metadata Server (MDS) to efficiently manage metadata and coordinate file operations.

Role of Metadata Server in CephFS

The Metadata Server (MDS) is critical to CephFS, managing file system metadata and coordinating file operations across clients.

Figure 1 illustrates the role of Metadata Server in CephFS.

Figure: Role of Metadata Server in CephFS



The following outlines the key functions of the MDS:

Active MDS

Handles metadata-related client requests, such as file creation and directory management, while maintaining a cache to optimize performance.

Standby MDS

Provides high availability by taking over in the event of an active MDS failure. A standby MDS configured as standby-replay continuously applies journal changes to ensure a swift failover.

Optimizing MDS performance

Optimizing MDS performance is essential for maintaining a high-performing and reliable CephFS deployment. Key strategies include configuring single active MDS, multiple active MDS instances and Standby MDS. In CephFS, different MDS configurations can be used based on the scale and demands of the environment. The following outlines the various MDS deployment options:

Single active MDS

For smaller deployments or environments with limited metadata workloads, a single active MDS is typically sufficient. This configuration simplifies management and reduces overhead.

Multiple active MDS daemons

In larger deployments with substantial metadata workloads, multiple active MDS daemons can enhance performance by distributing the metadata load. Adjust the **`max_mds`** parameter to dynamically balance the load across multiple MDS instances, improving system responsiveness and reducing latency.

Standby MDS

Ensures high availability by automatically taking over if an active MDS fails. Configuring standby MDS as standby-replay can improve failover speed by continuously applying journal changes.

MDS limits and configuration

To ensure optimal performance in CephFS, you must configure various limits related to memory and metadata management. The following outlines key parameters to consider:

Memory limits

Configure the cache size with **`mds_cache_memory_limit`** and set a cache reservation with **`mds_cache_reservation`** to allocate sufficient memory for metadata operations, optimizing performance.

File and directory limits

CephFS supports horizontal scaling by adding more MDS daemons to manage increased workloads and maintain performance as metadata demands grow.

CephFS volumes, subvolumes, and subvolume groups

Ceph File System (CephFS) volumes, subvolumes, and subvolume groups are managed through a Ceph Manager (MGR) module.

In CephFS, several constructs are essential for efficiently managing data across various deployment scenarios and integrations. Each component serves a unique purpose, allowing for better organization, control, and policy application.

The following is a breakdown of these component constructs:

Volumes

Volumes represent the entire CephFS file system, serving as the foundational layer for file-based storage. Usually there is only one CephFS volume per Ceph cluster.

Subvolumes

Subvolumes act as per application or tenant directory trees within volumes, enabling you to organize and manage data more effectively within the larger file system. Subvolumes are usually the basis for user or application file-shares.

Subvolume groups

The subvolume groups act according to application or tenant directory trees within volumes, enabling you to organize and manage data more effectively within the larger file system. Subvolumes are usually the basis for user or application file shares.

Differences and Usage

To effectively manage data within CephFS, it's important to understand the distinct roles played by volumes, subvolumes, and subvolume groups. Each of these constructs offers unique capabilities that contribute to the overall efficiency and organization of the file system.

The following lists the differences and usage of each component:

Volumes

Volumes provide a high-level overview and control of the entire file system, essential for overall management. Generally, there is one volume per Ceph cluster.

Subvolumes

Subvolumes allow for detailed data organization and quota management within volumes, helping to fine-tune resource allocation.
Subvolumes are ideal for creating shares for users, applications, and persistent shared file systems for Kubernetes.

Subvolume groups

Subvolume groups enable the efficient application of policies across groups of subvolumes, making it easier to manage large datasets and enforce administrative rules.
Subvolume groups can be used to simplify the management of multiple subvolumes that all belong to the same application or service.

Volume Management

Effective volume management is crucial for optimizing CephFS performance and security.

Single versus multiple volumes

Multiple file system support is stable and ready to use. This functionality allows configuring separate file systems with full data separation on separate pools. However, it is recommended to consolidate file system workloads onto a single CephFS file system, when possible. Consolidate the workloads to avoid over-allocating resources to MDS servers that can be underutilized.

Security considerations

Implement robust access controls and security policies for CephFS volumes, subvolumes, and subvolume groups to help ensure data protection and compliance for users, applications, and tenants.

File system components

The CephFS is comprised of clients and Metadata Servers (MDS). Understand the component layers of the Ceph File system and the underlying core storage cluster components.

The CephFS has two primary components:

Clients

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

Metadata Servers (MDS)

The MDS does the following:

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

Ceph File System components can be divided into four layers, as illustrated in “Figure: Ceph File System component layers” on page 17.

The first, bottom layer represents the underlying core storage cluster components:

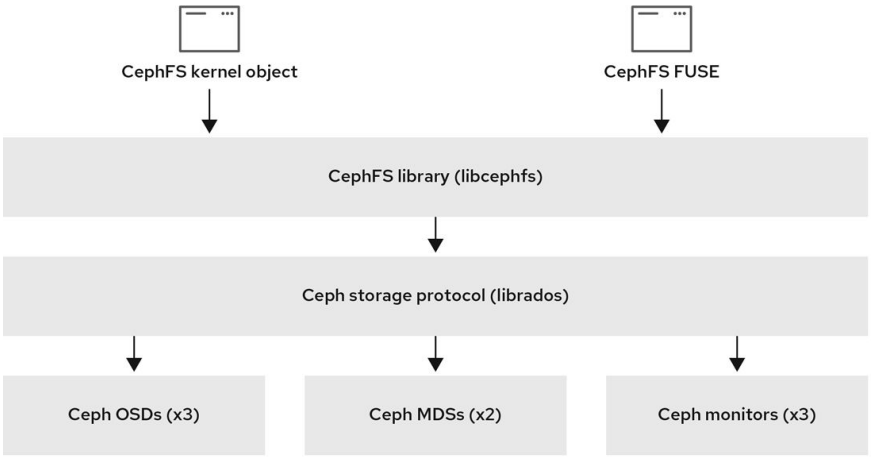
- Ceph OSDs (`ceph-osd`) where the Ceph File System data and metadata are stored.
- Ceph Metadata Servers (`ceph-mds`) that manages Ceph File System metadata.
- Ceph Monitors (`ceph-mon`) that manages the master copy of the cluster map.

The second level is the Ceph storage protocol layer. This represents the Ceph native `librados` library for interacting with the core storage cluster.

The third layer, on top of the Ceph storage protocol, is the CephFS library layer includes the CephFS `libcephfs` library that works on top of `librados` and represents the Ceph File System.

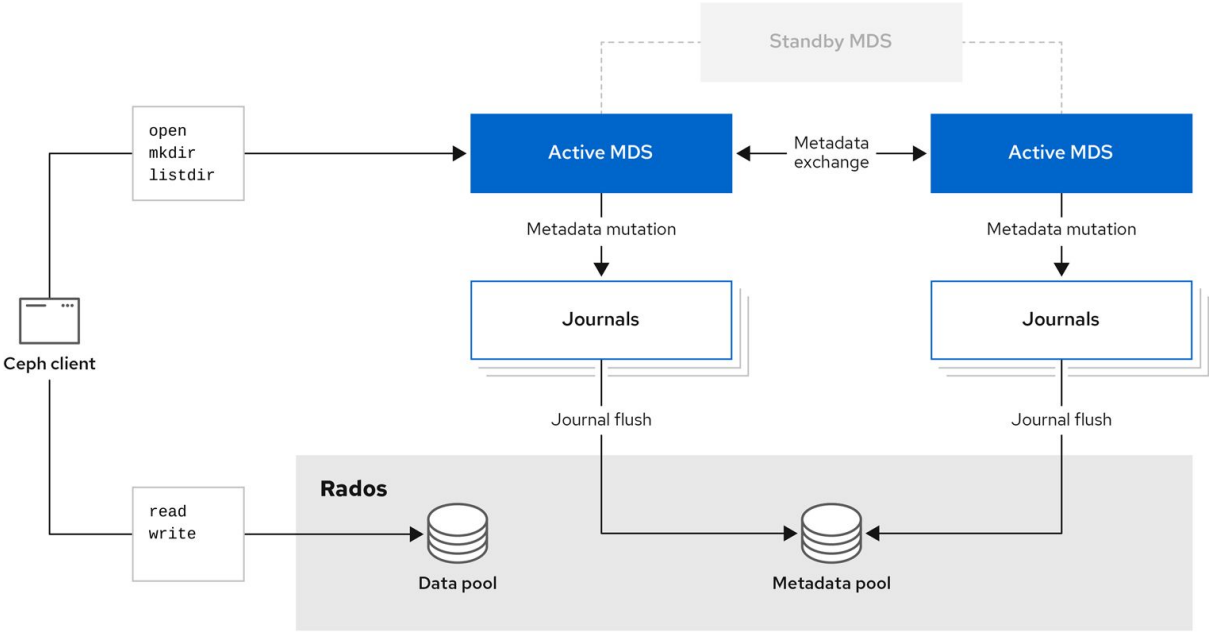
The top layer represents the two types of Ceph clients that can access the Ceph File Systems: CephFS kernel object and CephFS FUSE.

Figure: Ceph File System component layers



“Figure: Interaction between the Ceph File System components” on page 17 shows more details on how the Ceph File System components interact with each other.

Figure: Interaction between the Ceph File System components



For more information, see:

- “Managing the MDS service using the Ceph Orchestrator” on page 20
- “Deploying the CephFS” on page 33

CephFS and SELinux

Using Security-Enhanced Linux (SELinux) is supported on Ceph File Systems (CephFS) environments.

Set any SELinux file type with CephFS, along with assigning a particular SELinux type on individual files. This support applies to the CephFS Metadata Server (MDS), the CephFS File System in User Space (FUSE) clients, and the CephFS kernel clients.

Related information

Using SELinux

CephFS limitations and the POSIX standards

Learn about the CephFS limitations and the POSIX standards.

The CephFS diverges from the strict POSIX semantics in the following ways:

- If a client’s attempt to write a file fails, the write operations are not necessarily atomic. For example, the client might call the `write()` system call on a file that is opened with the `O_SYNC` flag with an 8 MB buffer and then terminates unexpectedly. In this case, 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 nonzero offset might often work but is not guaranteed to do so. Calling `seekdir()` to offset 0 will always work. This is equivalent to the `rewinddir()` system call.
- Sparse files propagate incorrectly to the `st_blocks` field of the `stat()` system call. CephFS does not explicitly track parts of a file that are allocated or written to because the `st_blocks` field is always populated by the quotient of file size divided by block size. This behavior causes utilities, such as `du`, to overestimate used space.
- When the `mmap()` system call maps a file into memory on multiple hosts, write operations are not coherently propagated to caches of other hosts. That is, if a page is cached on host A, and then updated on host B, host A page is not coherently invalidated.
- CephFS clients present a hidden `.snap` directory that is used to access, create, delete, and rename snapshots. Although this directory is excluded from the `readdir()` system call, any process that tries to create a file or directory with the same name returns an error. The name of this hidden directory can be changed at mount time with the `-o snapdirname=.<new_name>` option or by using the `client_snapdir` configuration option.

For more information, see:

- “Managing the MDS service using the Ceph Orchestrator” on page 20
- “Deploying the CephFS” on page 33

CephFS Metadata Server

Learn about the different states of the Ceph File System (CephFS) Metadata Server (MDS), along with learning about CephFS MDS ranking mechanics, configuring the MDS standby daemon, and cache size limits. Knowing these concepts help you configure the MDS daemons for a storage environment.

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

- A running, and operational Red Hat Ceph Storage cluster.
  For more information, see "Installing Red Hat Ceph Storage" on page 0.
- Ceph Metadata Server daemons (`ceph-mds`) installed.

Metadata Server daemon states

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

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

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

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

What happens when the active MDS daemon fails

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

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

Metadata Server ranks

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

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

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

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

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

Rank states

Ranks can be as follows:

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

Metadata Server cache size limits

Learn how to limit the size of the Ceph File System (CephFS) Metadata Server (MDS) cache.

You can limit the size of the Ceph File System (CephFS) Metadata Server (MDS) cache by using either the memory limit or the inode count.

Memory limit

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

Important: Use memory limits instead of inode count limits.

Note: The default value for `mds_cache_memory_limit` is 4 GB. Since the default value is outside the recommended range, users are recommended to set the value within the mentioned range.

Inode count

Use the `mds_cache_size` option. By default, limiting the MDS cache by inode count is disabled.

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

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

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

For more information, see “Configuring Metadata Server daemons” on page 5.

File system affinity

You can configure a Ceph File System (CephFS) to prefer a particular Ceph Metadata Server (MDS) over another Ceph MDS.

In some cases you may have two Ceph Metadata Servers, a newer and faster hardware and an older and slower one. You can specify the preference of the newer hardware, by setting the `mds_join_fs` option, which enforces this file system affinity. Ceph Monitors give preference to MDS standby daemons with `mds_join_fs` equal to the file system name with the failed rank. The standby-replay daemons are selected before choosing another standby daemon. If no standby daemon exists with the `mds_join_fs` option, then the Ceph Monitors choose an ordinary standby for replacement or any other available standby as a last resort. The Ceph Monitors periodically examine the Ceph File Systems to see whether a standby with a stronger affinity is available to replace the Ceph MDS that has a lower affinity.

Managing the MDS service using the Ceph Orchestrator

Use the Ceph Orchestrator with `cephadm` in the backend to deploy the MDS service.

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

When managing the MDS service by using the Ceph Orchestrator, be sure to have:

- A running Red Hat Ceph Storage cluster.
- Root-level access to all the nodes.
- Hosts added to the cluster.
- All manager, monitor, and OSD daemons deployed.

Deploying the MDS service with the command-line interface

Using the Ceph Orchestrator, you can deploy the Metadata Server (MDS) service by using the `placement` specification in the command-line interface.

PREREQUISITE

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

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

Ceph File System (CephFS) requires one or more MDS.

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

For more information about creating Ceph File Systems, see Creating Ceph File Systems.

To deploy the MDS service with the command-line interface, log in to the `cephadm` shell.

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

Use one of the following options to deploy MDS daemons using placement specification.

Creating the MDS daemons through the command-line

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

  ```
  ceph fs volume create FILESYSTEM_NAME --
  placement="NUMBER_OF_DAEMONS HOST_NAME_1 HOST_NAME_2 HOST_NAME_3"
  ```

  > **Note:** By default, replicated pools are created for this command.

  For example,

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

Creating the pools, CephFS, and then deploying MDS service using placement specification

1. Create the pools for CephFS.

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

   For example,

   ```
   [ceph: root@host01 /]# ceph osd pool create cephfs_data 64
   [ceph: root@host01 /]# ceph osd pool create cephfs_metadata 64
   ```

2. Create the file system for the data pools and metadata pools.

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

   For example,

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

3. Deploy MDS service using the **`ceph orch apply`** command.

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

For example,

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

Verifying the MDS service deployment

1. List the service.
   For example,

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

2. Check the CephFS status.
   For example,

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

3. List the hosts, daemons, and processes.

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

   For example,

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

Deploying the MDS service using the service specification

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

PREREQUISITE

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

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

> **Note:** Ensure that you have at least two pools, one for the Ceph File System (CephFS) data and one for the CephFS metadata.

1. Create the `mds.yaml` file.

   ```
   touch mds.yaml
   ```

2. Edit the `mds.yaml` file.

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

   For example,

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

3. Mount the YAML file under a directory in the container.

For example,

```
[root@host01 ~]# cephadm shell --mount mds.yaml:/var/lib/ceph/mds/mds.yaml
```

4. Navigate to the ceph/mds/ directory.
For example,

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

5. Log into the cephadm shell.

```
cephadm shell
```

6. Navigate to the /ceph/mds/ directory.
For example,

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

7. Deploy MDS service using service specification.

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

For example,

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

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

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

For example,

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

AFTER COMPLETING THE TASK

Verify the MDS service deployment.

1. List the service.

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

2. List the hosts, daemons, and processes.

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

For example,

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

Removing the MDS service using the Ceph Orchestrator

Remove the service either by using the **ceph orch rm** command or by removing the file system and the associated pools.

PREREQUISITE

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

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

To remove the MDS service with the command-line interface, log in to the `cephadm` shell.

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

Use one of the following options to remove MDS daemons.

Removing the MDS service from the entire cluster through the command line

Use the **`ceph orch rm`** command to remove the MDS service from the entire cluster.

1. List the service.

   ```
   ceph orch ls
   ```

   For example,

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

2. Remove the service.

   ```
   ceph orch rm SERVICE_NAME
   ```

   For example,

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

Removing the CephFS volume, associated pools, and the services

1. Set the configuration parameter **`mon_allow_pool_delete`** to `true`.

   ```
   ceph config set mon mon_allow_pool_delete true
   ```

   For example,

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

2. Remove the file system, its data, and metadata pools.

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

   This command also tries to remove the MDS using the enabled `ceph-mgr` Orchestrator module.

   ```
   ceph fs volume rm cephfs-new --yes-i-really-mean-it
   ```

   For example,

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

Verifying the MDS service removal

- List the hosts, daemons, and processes, by using the **`ceph orch ps`** command.
  For example,

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

Configuring file system affinity

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

PREREQUISITE

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

- A healthy, and running Red Hat Ceph Storage.

- A Ceph File System created.
- Root-level access to a Ceph Monitor node.

For more information about file system affinity, see "File system affinity" on page 20.

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

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

Standby daemons:

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

2. Set the file system affinity.

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

For example,

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

After a Ceph MDS failover event, the file system favors the standby daemon for which the affinity is set, as shown in the following example.

Note the `mds.b` daemon now has the `join_fscid=27` in the file system dump output.

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

Standby daemons:

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

> **Note:** If a file system is in a degraded or undersized state, then no failover occurs to enforce the file system affinity.

Configuring multiple active Metadata Server daemons

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

PREREQUISITE

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

- Ceph administration capabilities on the MDS node.
- Root-level access to a Ceph Monitor node.

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

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

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

   For example, to increase the number of active MDS daemons to two in the CephFS called `cephfs`:

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

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

2. Verify the number of active MDS daemons.

   ```
   ceph fs status NAME
   ```

   For example:

   ```
   [root@mon ~]# ceph fs status cephfs
   cephfs - 0 clients
   ======
   +------+--------+-------+---------------+-------+-------+--------+--------+
   | RANK | STATE  |  MDS  |    ACTIVITY   |  DNS  |  INOS |  DIRS  |  CAPS  |
   +------+--------+-------+---------------+-------+-------+--------+--------+
   |  0   | active | node1 | Reqs:    0 /s |   10  |   12  |   12   |   0    |
   |  1   | active | node2 | Reqs:    0 /s |   10  |   12  |   12   |   0    |
   +------+--------+-------+---------------+-------+-------+--------+--------+
   +-----------------+----------+-------+-------+
   |       POOL      |   TYPE   |  USED | AVAIL |
   +-----------------+----------+-------+-------+
   | cephfs_metadata | metadata | 4638  | 26.7G |
   |   cephfs_data   |   data   |    0  | 26.7G |
   +-----------------+----------+-------+-------+

   +-------------+
   | STANDBY MDS |
   +-------------+
   |    node3    |
   +-------------+
   ```

Related information

Managing Ceph users

Configuring the number of standby daemons

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

PREREQUISITE

Be sure that you have root-level access to a Ceph Monitor node.

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

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

> **Note:** Setting the *NUMBER* to zero disables the daemon health check.

For example, to set the expected standby daemon count to two:

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

Configuring the standby-replay Metadata Server

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

PREREQUISITE

Be sure that you have root-level access to a Ceph Monitor node.
This specific standby-replay daemon follows the active MDS's metadata journal. The standby-replay daemon is only used by the active MDS of the same rank, and is not available to other ranks.

> **Important:** If using standby-replay, then every active MDS must have a standby-replay daemon.

– Set the standby-replay for a particular CephFS.

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

> **Note:** Setting the *NUMBER* to zero disables the daemon health check.

For example, to enable the standby-replay daemons to be assigned to the active Ceph MDS daemons, set the Boolean value to1

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

For more information, see "Using the ceph mds fail command" on page 75

Ephemeral pinning policies

An ephemeral pin is a static partition of subtrees, and can be set with a policy that uses extended attributes. A policy can automatically set ephemeral pins to directories. When setting an ephemeral pin to a directory, it is automatically assigned to a particular rank, as to be uniformly distributed across all Ceph MDS ranks.

Determining which rank gets assigned is done by a consistent hash and the directory's inode number.
Ephemeral pins do not persist when the directory's inode is dropped from file system cache. When failing over a Ceph Metadata Server (MDS), the ephemeral pin is recorded in its journal so the Ceph MDS standby server does not lose this information.

> **Note:** Installation of the `attr` and `jq` package must be installed as a prerequisite for the ephemeral pinning policies.

There are two types of policies for using ephemeral pins:

Distributed

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

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

Example

```
[root@host01 mount]# setfattr -n ceph.dir.pin.distributed -v 1 dir1/
```

Random

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

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

Example

```
[root@host01 mount]# setfattr -n ceph.dir.pin.random -v  0.01 dir1/
```

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

After enabling pinning, you can **verify** by running either of the following commands.

Syntax

```
getfattr -n ceph.dir.pin.random DIRECTORY_PATH
getfattr -n ceph.dir.pin.distributed DIRECTORY_PATH
```

Example

```
[root@host01 mount]# getfattr -n ceph.dir.pin.distributed dir1/
# file: dir1/
ceph.dir.pin.distributed="1"

[root@host01 mount]# getfattr -n ceph.dir.pin.random dir1/
# file: dir1/
ceph.dir.pin.random="0.01"
```

Example

```
ceph tell mds.a get subtrees | jq '.[] | [.dir.path, .auth_first, .export_pin]'
```

If the directory is pinned, the value of `export_pin` is `0` if it is pinned to rank `0`, `1` if it is pinned to rank `1`, and so on. If the directory is not pinned, the value is `-1`.

To **remove** a partitioning policy, remove the extended attributes or set the value to `0`.

Syntax

```
 setfattr -n ceph.dir.pin.distributed -v 0 DIRECTORY_PATH
```

Example

```
[root@host01 mount]# setfattr -n ceph.dir.pin.distributed -v 0 dir1/
```

You can **verify** by running either of the following commands:

Syntax

```
getfattr -n ceph.dir.pin.distributed DIRECTORY_PATH
```

Example

```
[root@host01 mount]# getfattr -n ceph.dir.pin.distributed dir1/
```

For export pins, remove the extended attribute or set the extended attribute to `-1`.

Syntax

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

Example

```
[root@host01 mount]# setfattr -n ceph.dir.pin -v -1 dir1/
```

For more information about manually setting pins, see “Manually pinning directory trees to a particular rank” on page 29.

Manually pinning directory trees to a particular rank

You can manually override the dynamic balancer with explicit mappings of metadata to a particular Ceph Metadata Server (MDS) rank. This can be done to evenly spread the load of an application or to limit the impact of users metadata requests on the CephFS cluster.

PREREQUISITE

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

- A running Red Hat Ceph Storage cluster.
- A running Ceph File System.
- Root-level access to the CephFS client.
- The `attr` package installed.

Manually pinning directories is also known as an export pin by setting the `ceph.dir.pin` extended attribute.

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

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

(1) Directories `a/` and `a/b` both start without an export pin set.
(2) Directories `a/` and `a/b` are now pinned to rank `1`.
(3) Directory `a/b` is now pinned to rank `0` and directory `a/` and the rest of its subdirectories are still pinned to rank `1`.

For more information about automatically setting pins, see “Ephemeral pinning policies” on page 27.

- Set the export pin on a directory.

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

  For example:

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

Decreasing the number of active Metadata Server daemons

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

PREREQUISITE

A rank that is to be removed, must first be active. This means that you must have the same number of MDS daemons as specified by the **max_mds** parameter.
Before you begin be sure that you have root-level access to a Ceph Monitor node.

For more information about Metadata Server daemons, see “Metadata Server daemon states” on page 19 and “Configuring multiple active Metadata Server daemons” on page 25.

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

   ```
   ceph fs status NAME
   ```

   For example:

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

+------+--------+-------+---------------+-------+-------+--------+--------+
| RANK | STATE  |  MDS  |    ACTIVITY   |  DNS  |  INOS |  DIRS  |  CAPS  |
+------+--------+-------+---------------+-------+-------+--------+--------+
|  0   | active | node1 | Reqs:    0 /s |   10  |   12  |   12   |   0    |
|  1   | active | node2 | Reqs:    0 /s |   10  |   12  |   12   |   0    |
+------+--------+-------+---------------+-------+-------+--------+--------+
+-----------------+----------+-------+-------+
|       POOL      |   TYPE   |  USED | AVAIL |
+-----------------+----------+-------+-------+
| cephfs_metadata | metadata | 4638  | 26.7G |
|   cephfs_data   |   data   |    0  | 26.7G |
+-----------------+----------+-------+-------+

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

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

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

For example:

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

3. Wait for the storage cluster to stabilize to the new **max_mds** value by watching the Ceph File System status.
4. Verify the number of active MDS daemons.

```
ceph fs status NAME
```

For example:

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

+------+--------+-------+---------------+-------+-------+--------+--------+
| RANK | STATE  |  MDS  |    ACTIVITY   |  DNS  |  INOS |  DIRS  |  CAPS  |
+------+--------+-------+---------------+-------+-------+--------+--------+
|  0   | active | node1 | Reqs:    0 /s |   10  |   12  |   12   |   0    |
+------+--------+-------+---------------+-------+-------+--------|--------+
+-----------------+----------+-------+-------+
|       POOl      |   TYPE   |  USED | AVAIL |
+-----------------+----------+-------+-------+
| cephfs_metadata | metadata | 4638  | 26.7G |
|   cephfs_data   |   data   |    0  | 26.7G |
+-----------------+----------+-------+-------+

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

Viewing metrics for Ceph metadata server clients

You can use the command-line interface to view the metrics for the Ceph metadata server (MDS).

PREREQUISITE

- A running Red Hat Ceph Storage cluster.

CephFS uses `Perf Counters` to track metrics. You can view the metrics using the **`counter dump`** command.

1. Get the name of the *mds* service:

Syntax:

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

2. Check the MDS per client metrics:
   Syntax:

```
[ceph: root@mds-host01 /]# ceph tell MDS_SERVICE_NAME counter dump
```

   Example:

```
[root@ceph2-hk-n-0mfqao-node1-installer ~]# ceph tell mds.cephfs.ceph2-hk-n-0mfqao-
node4.isztbk counter dump

[
  {
    "key": "mds_client_metrics",
    "value": [
      {
        "labels": {
          "fs_name": "cephfs",
          "id": "24379"
        },
        "counters": {
          "num_clients": 4
        }
      }
    ]
  },
  {
    "key": "mds_client_metrics-cephfs",
    "value": [
      {
        "labels": {
          "client": "client.24413",
          "rank": "0"
        },
        "counters": {
          "cap_hits": 56,
          "cap_miss": 9,
          "avg_read_latency": 0E-9,
          "avg_write_latency": 0E-9,
          "avg_metadata_latency": 0E-9,
          "dentry_lease_hits": 2,
          "dentry_lease_miss": 12,
          "opened_files": 0,
          "opened_inodes": 9,
          "pinned_icaps": 4,
          "total_inodes": 9,
          "total_read_ops": 0,
          "total_read_size": 0,
          "total_write_ops": 0,
          "total_write_size": 0
        }
      },
      {
        "labels": {
          "client": "client.24502",
          "rank": "0"
        },
        "counters": {
          "cap_hits": 921403,
          "cap_miss": 102382,
          "avg_read_latency": 0E-9,
          "avg_write_latency": 0E-9,
          "avg_metadata_latency": 0E-9,
          "dentry_lease_hits": 17117,
          "dentry_lease_miss": 204710,
          "opened_files": 0,
          "opened_inodes": 9,
          "pinned_icaps": 7,
          "total_inodes": 9,
          "total_read_ops": 0,
          "total_read_size": 0,
          "total_write_ops": 1,
          "total_write_size": 132
        }
      },
```

```
      {
        "labels": {
          "client": "client.24508",
          "rank": "0"
        },
        "counters": {
          "cap_hits": 928694,
          "cap_miss": 103183,
          "avg_read_latency": 0E-9,
          "avg_write_latency": 0E-9,
          "avg_metadata_latency": 0E-9,
          "dentry_lease_hits": 17217,
          "dentry_lease_miss": 206348,
          "opened_files": 0,
          "opened_inodes": 9,
          "pinned_icaps": 7,
          "total_inodes": 9,
          "total_read_ops": 0,
          "total_read_size": 0,
          "total_write_ops": 1,
          "total_write_size": 132
        }
      },
      {
        "labels": {
          "client": "client.24520",
          "rank": "0"
        },
        "counters": {
          "cap_hits": 56,
          "cap_miss": 9,
          "avg_read_latency": 0E-9,
          "avg_write_latency": 0E-9,
          "avg_metadata_latency": 0E-9,
          "dentry_lease_hits": 2,
          "dentry_lease_miss": 12,
          "opened_files": 0,
          "opened_inodes": 9,
          "pinned_icaps": 4,
          "total_inodes": 9,
          "total_read_ops": 0,
          "total_read_size": 0,
          "total_write_ops": 0,
          "total_write_size": 0
        }
      }
    ]
  }
]
```


Client metrics description:

CephFS exports client metrics as `Labeled Perf Counters`, which you can use to monitor the client performance. CephFS exports the below client metric.

Client metrics description		
Name	**Type**	**Description**
cap_hits	Gauge	Percentage of file capability hits over total number of caps.
cap_miss	Gauge	Percentage of file capability misses over total number of caps.
avg_read_latency	Gauge	Mean value of the read latencies.
avg_write_latency	Gauge	Mean value of the write latencies.
avg_metadata_latency	Gauge	Mean value of the metadata latencies
dentry_lease_hits	Gauge	Percentage of dentry lease hits handed out over the total dentry lease request.
dentry_lease_miss	Gauge	Percentage of dentry lease misses handed out over the total dentry lease requests.
opened_files	Gauge	Number of opened files.
opened_inodes	Gauge	Number of opened inode.

Name	Type	Description
pinned_icaps	Gauge	Number of pinned Inode Caps.
total_inodes	Gauge	Total number of Nodes.
total_read_ops	Gauge	Total number of read operations generated by all process.
total_read_size	Gauge	Number of bytes read in input/output operations generated by all process.
total_write_ops	Gauge	Total number of write operations generated by all process.
total_write_size	Gauge	Number of bytes written in input/output operations generated by all processes.

Deploying the CephFS

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

The deployment workflow is three steps:

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

Before deploying CephFS , be sure that you have a running and healthy Red Hat Ceph Storage cluster and the Ceph Metadata Server daemon (`ceph-mds`) installed and configured.

For more information, see:

- “Managing the MDS service using the Ceph Orchestrator” on page 20
- “Creating Ceph File Systems” on page 34
- “Creating client users for a Ceph File System” on page 38
- “Mounting the Ceph File System as a kernel client” on page 39
- “Mounting the Ceph File System as a FUSE client” on page 42
- “CephFS Metadata Server” on page 18

Layout, quota, snapshot, and network restrictions

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

> **Important:** All user capability flags, except `rw`, must be specified in alphabetical order.

For more information about setting the Ceph user capabilities, see “Creating client users for a Ceph File System” on page 38.

Layouts and quotas

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

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

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

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

Snapshots

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

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

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

Network

Restrict clients connecting from a particular network.

The optional network and prefix length is in CIDR notation.

In the following example, the optional network and prefix length is `10.0.0.0/8`.

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

Creating Ceph File Systems

Create multiple Ceph File Systems (CephFS) on a Ceph Monitor node.

PREREQUISITE

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

- A running and healthy Red Hat Ceph Storage cluster.
- Root-level access to a Ceph Monitor node.
- Root-level access to a Ceph client node.

For more information, see the following:

- “Creating client users for a Ceph File System” on page 38
- “Mounting the Ceph File System as a kernel client” on page 39
- “Mounting the Ceph File System as a FUSE client” on page 42
- “CephFS limitations and the POSIX standards” on page 18
- Pools overview

1. Configure the client node to use the Ceph storage cluster.
   a. Enable the Red Hat Ceph Storage 9 tools repository.

      ```
      subscription-manager repos --enable=rhceph-9-tools-for-rhel-9-x86_64-rpms
      ```

   b. Install the `ceph-fuse` package.

```
dnf install ceph-fuse
```

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

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

Replace *MONITOR_NODE_NAME* with the Ceph Monitor hostname or IP address.
For example,

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

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

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

Replace *MONITOR_NODE_NAME* with the Ceph Monitor hostname or IP address.
For example,

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

e. Set the appropriate permissions for the configuration file.

```
chmod 644 /etc/ceph/ceph.conf
```

2. Create a Ceph File System on a Ceph Monitor node.

```
ceph fs volume create FILE_SYSTEM_NAME
```

For example,

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

Repeat this step to create more file systems.

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

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

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

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

> **Important:** The Ceph client can only see the CephFS it is authorized for.

For example,

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

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

Note: Optionally, you can add a safety measure by specifying the `root_squash` option. This prevents accidental deletion scenarios by disallowing clients with a `uid=0` or `gid=0` to do write operations, but still allows read operations.
In the following example, `root_squash` is enabled for the file system `cephfs01`, except within the `/volumes` directory tree.

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

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

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

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

For example,

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

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

```
mkdir PATH_TO_NEW_DIRECTORY_NAME
```

For example,

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

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

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

For example,

```
[root@client ~]# ceph-fuse /mnt/mycephfs/ -n client.1 --client-fs=cephfs01
ceph-fuse[555001]: starting ceph client
2022-05-09T07:33:27.158+0000 7f11feb81200 -1 init, newargv = 0x55fc4269d5d0
newargc=15
ceph-fuse[555001]: starting fuse
```

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

Adding an erasure-coded pool to a Ceph File System

By default, Ceph uses replicated pools for data pools. However, another erasure-coded data pool to the Ceph File System can be added, as needed.

PREREQUISITE

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

- A running Red Hat Ceph Storage cluster.
- An existing Ceph File System.
- Pools that use BlueStore OSDs.
- Root-level access to a Ceph Monitor node.
- The `attr` package installed.

Ceph File Systems (CephFS) backed by erasure-coded pools use less overall storage compared to Ceph File Systems that are backed by replicated pools. While erasure-coded pools use less overall storage, they also use more memory and processor resources than replicated pools.

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

For more information, see:

- “CephFS Metadata Server” on page 18
- “Creating Ceph File Systems” on page 34
- “Erasure code pools overview” on page 0

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

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

   For example:

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

2. Verify that the pool was added.

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

3. Enable overwrites on the erasure-coded pool.

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

   For example:

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

4. Verify the status of the Ceph File System.

   ```
   ceph fs status FILE_SYSTEM_NAME
   ```

   For example:

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

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

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

```
ceph fs add_data_pool FILE_SYSTEM_NAME DATA_POOL_NAME
```

In the following example, the new data pool, `cephfs-data-ec01`, is added to the existing erasure-coded file system, `cephfs-ec`.

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

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

```
ceph fs status FILE_SYSTEM_NAME
```

For example:

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

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

7. Set the file layout on a new directory.

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

In the following example, all new files that are created in the `/mnt/cephfs/newdir` directory inherit the directory layout and places the data in the newly added erasure-coded pool.

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

Creating client users for a Ceph File System

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

PREREQUISITE

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

- A running Red Hat Ceph Storage cluster.
- The Ceph Metadata Server daemon (`ceph-mds`) installed and configured.
- Root-level access to a Ceph Monitor node.
- Root-level access to a Ceph client node.

For more information, see Ceph user management.

1. Log in to the `cephadm` shell on the monitor node.

```
cephadm shell
```

2. Create a client user on a Ceph Monitor node.

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

Example for restricting the client to only writing in the `temp` directory of filesystem `cephfs_a`

```
[ceph: root@host01 ~]# ceph fs authorize cephfs_a client.1 / r /temp rw

client.1
  key = AQBSdFhcGZFUDRAAcKhG9Cl2HPiDMMRv4DC43A==
```

Example for completely restricting the client to the `temp` directory, by removing the root (/) directory

```
[ceph: root@host01 ~]# ceph fs authorize cephfs_a client.1 /temp rw
```

> **Note:** Supplying `all` or asterisk (`*`) as the file system name grants access to every file system. Typically, it is necessary to put the asterisk in quotations to protect it from the shell.

3. Verify the created key.

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

For example,

```
[ceph: root@host01 ~]# ceph auth get client.1

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

4. Copy the keyring to the client.
   a. On the Ceph Monitor node, export the keyring to a file.

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

For example,

```
[ceph: root@host01 ~]# ceph auth get client.1 -o ceph.client.1.keyring
exported keyring for client.1
```

   b. Copy the client keyring from the Ceph Monitor node to the `/etc/ceph/` directory on the client node.
   Replace *CLIENT_NODE_NAME* with the Ceph client node name or IP.
   For example,

```
[ceph: root@host01 ~]# scp /ceph.client.1.keyring root@client01:/etc/ceph/
ceph.client.1.keyring
```

5. From the client node, set the appropriate permissions for the keyring file.

```
chmod 644 ceph.client.ID.keyring
```

For example,

```
[root@client01 ~]# chmod 644 /etc/ceph/ceph.client.1.keyring
```

Mounting the Ceph File System as a kernel client

The Ceph File System (CephFS) can be mounted as a kernel client, either manually or automatically on system start.

PREREQUISITE

Before mounting a Ceph File System as a kernel client, be sure that you have:

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

– An existing Ceph File System.

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

– See the `mount(8)` manual page.

– For more information about creating a Ceph user, see "Ceph user management" on page 0.

– For more information about creating Ceph File Systems, see "Creating Ceph File Systems" on page 34.

Configure the client node to use the Ceph storage cluster.

1. Enable the Red Hat Ceph Storage 9 Tools repository.

```
subscription-manager repos --enable=rhceph-9-tools-for-rhel-9-x86_64-rpms
```

Repeat on all client nodes of the Red Hat Ceph Storage.

2. Install the `ceph-common` package.

```
dnf install ceph-common
```

3. Copy the Ceph client keyring from the Ceph Monitor node to the client node.

Note: The client user should already be created for the keyring file to exist. For information about creating client users for Ceph File Systems, see "Creating client users for a Ceph File System" on page 38.

```
scp /etc/ceph/ceph.client.admin.keyring root@CLIENT_NODE_NAME:/etc/ceph/
ceph.client.admin.keyring
```

Replace *CLIENT_NODE_NAME* with the Ceph client host name or IP address.
For example,

```
[ceph: root@host01 ~]# scp /etc/ceph/ceph.client.admin.keyring
root@ceph.client.admin.keyring:/etc/ceph/ceph.client.admin.keyring
```

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

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

Replace *CLIENT_NODE_NAME* with the Ceph client host name or IP address.
For example,

```
[ceph: root@host01 ~]# scp /etc/ceph/ceph.conf root@client01:/etc/ceph/ceph.conf
```

5. Configure the client node to use the Ceph storage cluster.

```
chmod 644 /etc/ceph/ceph.conf
```

6. Choose either `automatically` or `manually`.
   – For automatic mounting, continue to "Mounting automatically" on page 40.
   – For manual mounting, continue to "Mounting manually" on page 41 .

Mounting automatically

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

```
mkdir -p MOUNT_POINT
```

For example:

```
[root@client01 ~]# mkdir -p /mnt/cephfs
```

2. Edit the `/etc/fstab` file.

```
#DEVICE                 PATH           TYPE        OPTIONS
MON_0_HOST:PORT,              MOUNT_POINT        ceph      name=CLIENT_ID,
MON_1_HOST:PORT,                                           ceph.client_mountpoint=/
VOL/SUB_VOL_GROUP/SUB_VOL/UID_SUB_VOL, fs=FILE_SYSTEM_NAME, [ADDITIONAL_OPTIONS]
MON_2_HOST:PORT:/VOL/SUB_VOL/UID_SUB_VOL
```

In the `/etc/fstab` file,

- The first column (`#DEVICE`) sets the Ceph Monitor host names and the port number.
- The second column (`PATH`) set the mount point.
- The third column (`TYPE`) sets the file system type. In this case, `ceph`, for CephFS.
- The fourth column (`OPTIONS`) sets the various options. These include the user name and the secret file that use the `name` and `secretfile` options. You can also set specific volumes, subvolume groups, and subvolumes that use the `ceph.client_mountpoint` option.
  Set the `_netdev` option to ensure that the file system is mounted after the networking subsystem starts to prevent hanging and networking issues. If you do not need access time information, then setting the `noatime` option can increase performance.
- Set the fifth and sixth columns (`DUMP` and `FSCK`) to `0`.

For example,

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

The Ceph File System is mounted on the next system start.

Note:

- `mount.ceph` can read keyring files directly. As such, a secret file is no longer necessary. Specify the client ID with `name=`*`CLIENT_ID`*, and `mount.ceph` will find the correct keyring file.
- You can replace the Monitor host names with the string `:/` and `mount.ceph` will read the Ceph configuration file to determine which Monitors to connect to.

Mounting manually

1. Create a mount directory on the client node.

```
mkdir -p MOUNT_POINT
```

For example,

```
[root@client01 ~]# mkdir -p /mnt/cephfs
```

2. Mount the Ceph File System.
   To specify multiple Ceph Monitor addresses, in the **mount** command:
   a. Separate them with commas.
   b. Specify the mount point.

c. Set the client name.

> **Note:** `mount.ceph` can read keyring files directly, therefore a secret file is not necessary. It is enough to specify the client ID with `name=CLIENT_ID`, and `mount.ceph` finds the correct keyring file.

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

For example:

```
[root@client01 ~]# mount -t ceph mon1:6789,mon2:6789,mon3:6789:/ /mnt/cephfs -o
name=1,fs=cephfs01
```

> **Note:**
>
> - Configure a DNS server so that a single host name resolves to multiple IP addresses. That single host name can then be used with the **mount** command, instead of supplying a comma-separated list.
> - You can also replace the Monitor host names with the string `:/` and `mount.ceph` reads the Ceph configuration file to determine which Monitors to connect to.
> - Set the `nowsync` option to asynchronously run file creation and removal on the Red Hat Ceph Storage clusters. This improves the performance of some workloads by avoiding round-trip latency for these system calls without impacting consistency. The `nowsync` option requires kernel clients with Red Hat Enterprise Linux 9.2 or later.
>   For example,
>
>   ```
>   [root@client01 ~]# mount -t ceph mon1:6789,mon2:6789,mon3:6789:/ /mnt/
>   cephfs -o nowsync,name=1,fs=cephfs01
>   ```

3. Verify that the file system is successfully mounted.

   ```
   stat -f MOUNT_POINT
   ```

   For example,

   ```
   [root@client01 ~]# stat -f /mnt/cephfs
   ```

Mounting the Ceph File System as a FUSE client

The Ceph File System (CephFS) can be mounted as a FUSE client, either manually or automatically on system start.

PREREQUISITE

Before mounting a Ceph File System as a FUSE client, be sure that you have:

- Root-level access to a Linux-based client node.
- Root-level access to a Ceph Monitor node.
- An existing Ceph File System.
- For more information, see the `ceph-fuse(8)` manual page.
- For more information about creating a Ceph user, see “Ceph user management” on page 0.
- For more information about creating Ceph File Systems, see “Creating Ceph File Systems” on page 34.

Configure the client node to use the Ceph storage cluster.

1. Enable the Red Hat Ceph Storage 9 Tools repository.

```
subscription-manager repos --enable=rhceph-9-tools-for-rhel-9-x86_64-rpms
```

Repeat on client nodes of the Red Hat Ceph Storage storage cluster.

2. Install the `ceph-fuse` package.

```
dnf install ceph-fuse
```

3. Copy the Ceph client keyring from the Ceph Monitor node to the client node.

> **Note:** The client user must already be created.

```
scp /etc/ceph/ceph.client.admin.keyring root@CLIENT_NODE_NAME:/etc/ceph/
ceph.client.admin.keyring
```

Replace *CLIENT_NODE_NAME* with the Ceph client host name or IP address.

For example,

```
[ceph: root@host01 ~]# scp /etc/ceph/ceph.client.admin.keyring root@client01:/etc/
ceph/ceph.client.admin.keyring
```

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

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

Replace *CLIENT_NODE_NAME* with the Ceph client host name or IP address.

For example,

```
[ceph: root@host01 ~]# scp /etc/ceph/ceph.conf root@client01:/etc/ceph/ceph.conf
```

5. From the client node, set the appropriate permissions for the configuration file.

```
chmod 644 /etc/ceph/ceph.conf
```

6. Choose either `automatically` or `manually`.
   - For automatic mounting, continue to “Mounting automatically” on page 43.
   - For manual mounting, continue to “Mounting manually” on page 44.

Mounting automatically

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

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

```
mkdir PATH_TO_MOUNT_POINT
```

For example,

```
[root@client01 ~]# mkdir /mnt/mycephfs
```

2. Edit the `/etc/fstab` file.

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

```
HOST_NAME:PORT:/
ceph.client_fs=FILE_SYSTEM_NAME,ceph.name=USERNAME,ceph.keyring=/etc/ceph/
KEYRING_FILE,
                                                    [ADDITIONAL_OPTIONS]
```

In the `/etc/fstab` file,

- The first column (`#DEVICE`) sets the Ceph Monitor host names and the port number.
- The second column (`PATH`) set the mount point.
- The third column (`TYPE`) sets the file system type. In this case, `fuse.ceph`, for CephFS.
- The fourth column (`OPTIONS`) sets the various options. These include the user name and the keyring that use the `ceph.name` and `ceph.keyring` options. You can also set specific volumes, subvolume groups, and subvolumes that use the `ceph.client_mountpoint` option.
  To specify which Ceph File System to access, use the `ceph.client_fs` option. Set the `_netdev` option to ensure that the file system is mounted after the networking subsystem starts to prevent hanging and networking issues. If you do not need access time information, then setting the `noatime` option can increase performance. If you want to automatically reconnect after an eviction, then set the `client_reconnect_stale=true` option.
- Set the fifth and sixth columns (`DUMP` and `FSCK`) to `0`.

For example,

```
# DEVICE         PATH               TYPE         OPTIONS         DUMP  FSCK
 mon1:6789,      /mnt/mycephfs       fuse.ceph    ceph.id=1,         0     0
 mon2:6789,                                      ceph.client_mountpoint=/my_vol/
my_sub_vol_group/my_sub_vol/0,
 mon3:6789:/
 ceph.client_fs=cephfs01,ceph.name=client.1,ceph.keyring=/etc/ceph/client1.keyring,
                                              _netdev,defaults
```

The Ceph File System is mounted on the next system start.

Mounting manually

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

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

   ```
   mkdir PATH_TO_MOUNT_POINT
   ```

   For example,

   ```
   [root@client01 ~]# mkdir /mnt/mycephfs
   ```

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

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

   For example,

   ```
   [root@client01 ~]# ceph-fuse -n client.1 --client_fs cephfs01 /mnt/mycephfs
   ```

   Note:

   - If you do not use the default name and location of the user keyring (`/etc/ceph/ceph.client.CLIENT_ID.keyring`) then use the **`--keyring`** option to specify the path to the user keyring. For example,

```
[root@client01 ~]# ceph-fuse -n client.1 --keyring=/etc/ceph/
client.1.keyring /mnt/mycephfs
```

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

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

For example,

```
[root@client01 ~]# ceph-fuse -n client.1 /mnt/cephfs -r /home/cephfs
```

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

```
[root@client01 ~]# ceph-fuse -n client.1 /mnt/cephfs --
client_reconnect_stale=true
```

3. Verify that the file system is successfully mounted.

```
stat -f MOUNT_POINT
```

For example,

```
[root@client01 ~]# stat -f /mnt/cephfs
```

Managing CephFS volumes

The `volumes` module for the Ceph Manager daemon (`ceph-mgr`) implements the ability to export Ceph File Systems (CephFS).

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

- CephFS volumes
- CephFS subvolumes
- CephFS subvolume groups

For more information, see “Managing Ceph users” on page 0.

CephFS volumes

You can create, list, and remove Ceph File System (CephFS) volumes. CephFS volumes are an abstraction for Ceph File Systems.

Before working with CephFS volume be sure that you have:

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

Creating a CephFS volume

Ceph Orchestrator is a module for Ceph Manager that creates a Metadata Server (MDS) for the Ceph File System (CephFS). This section describes how to create a CephFS volume.

Creating a CephFS volume creates the Ceph File system along with the data and metadata pools.

- Create a CephFS volume on the monitor node.

```
ceph fs volume create VOLUME_NAME
```

For example,

```
[ceph: root@host01 /]# ceph fs volume create cephfs
```

Listing CephFS volumes

Learn how to list Ceph File System volumes.

PREREQUISITE

Be sure that you have a CephFS volume.

– List the CephFS volume.

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

Viewing information about a CephFS volume

List basic details about a Ceph File System (CephFS) volume, such as attributes of data and metadata pools of the CephFS volume, pending subvolumes deletion count, and the like.

PREREQUISITE

Be sure that you have a CephFS volume.

– View information about a CephFS volume.

```
ceph fs volume info VOLUME_NAME
```

For example:

```
[ceph: root@host01 /]# ceph fs volume info cephfs
{
    "mon_addrs": [
        "192.168.1.7:40977",
    ],
    "pending_subvolume_deletions": 0,
    "pools": {
        "data": [
            {
                "avail": 106288709632,
                "name": "cephfs.cephfs.data",
                "used": 4096
            }
        ],
        "metadata": [
            {
                "avail": 106288709632,
                "name": "cephfs.cephfs.meta",
                "used": 155648
            }
        ]
    },
    "used_size": 0
}
```

The output of the **`ceph fs volume info`** command includes:

`mon_addrs`

List of monitor addresses.

`pending_subvolume_deletions`

Number of subvolumes pending deletion.

`pools`

Attributes of data and metadata pools.

`avail`

The amount of free space available in bytes.

name

Name of the pool.

used

The amount of storage consumed in bytes.

used_size

Current used size of the CephFS volume in bytes.

Removing a CephFS volume

Ceph Orchestrator is a module for Ceph Manager that removes the Metadata Server (MDS) for the Ceph File System (CephFS). Learn how to remove the Ceph File System (CephFS) volume.

PREREQUISITE

Be sure that you have a CephFS volume.

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

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

2. Remove the CephFS volume.

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

For example:

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

CephFS subvolumes

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

Before working with CephFS subvolumes be sure that you have:

- A working Red Hat Ceph Storage cluster with CephFS deployed.
- A minimum read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.

Creating a file system subvolume

Learn how to create a Ceph File System (CephFS) subvolume.

PREREQUISITE

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

- A volume.
  For more information, see Creating a CephFS volume.
- A subvolume group.
  For more information, see Creating a file system subvolume group.

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

Note:

- you can specify its data pool layout, UID, GID, and file mode in octal numerals. By default, the subvolume group is created with an octal file mode '755', uid '0', gid '0', and data pool layout of its parent directory.
- To set quotas while creating a subvolume group, see Setting and managing quotas on a file system subvolume group.

You can also specify an unicode normalization form by using the `normalization` option. This will be used to internally mangle file names so that unicode characters that can be represented by different unicode code point sequences are all mapped to the representation, which means that they will all access the same file. However, users will continue to see the same name that they used when the file was created. The valid values for the unicode normalization form are:

nfd

canonical decomposition (default)

nfc

canonical decomposition, followed by canonical composition

nfkd

compatibility decomposition

nfkc

compatibility decomposition, followed by canonical composition

To learn more about unicode normalization forms see https://unicode.org/reports/tr15

- Create a CephFS subvolume.

  Note: The `--group_name` parameter is optional. If the subvolume needs to be created in a subvolume group, then `--group_name` needs to be passed in the command.

  Note:

  - Subvolume names containing whitespace must be enclosed in quotes when using the CLI. If whitespace is not quoted, the command-line parser interprets the name as multiple arguments and may lead to unintended behavior or incorrect subvolume creation.
  - To correctly specify a subvolume name with spaces, use quotes. For example,

    ```
    ceph fs subvolume create cephfs "sv 1" --group_name smb
    ```

  ```
  ceph fs subvolume create VOLUME_NAME SUBVOLUME_NAME[--size SIZE_IN_BYTES--
  group_name SUBVOLUME_GROUP_NAME--pool_layout DATA_POOL_NAME --uid UID --gid GID--
  mode OCTAL_MODE] [--namespace-isolated][--normalization <form>] [--casesensitive bool]
  ```

  For example:

  ```
  [root@mon ~]# ceph fs subvolume create cephfs01 sub0 --group_name subgroup0 --
  namespace-isolated
  ```

  The command succeeds even if the subvolume exists.

Listing a file system subvolume

Learn how to list a Ceph File System (CephFS) subvolume.

PREREQUISITE

Be sure that you have a CephFS subvolume.

- List the CephFS subvolume.

  > **Note:** The `--group_name` parameter is optional. If the subvolume needs to be listed from within a subvolume group, then `--group_name` needs to be passed in the command.

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

  For example,

  ```
  [root@mon ~]# ceph fs subvolume ls cephfs01 --group_name subgroup0
  [

  {

  "name": "sub0"

  }

  ]
  ```

Resizing a file system subvolume

Learn how to resize a Ceph File System (CephFS) subvolume.

PREREQUISITE

Be sure that you have a CephFS subvolume.

> **Note:** The **`ceph fs subvolume resize`** command resizes the subvolume quota by using the size that is specified by `new_size`. The `--no_shrink` flag prevents the subvolume from shrinking below the currently used size of the subvolume. The subvolume can be resized to an infinite size by passing `inf` or `infinite` as the `new_size`.

- Resize a CephFS subvolume.

  > **Note:** The `--group_name` parameter is optional. If the subvolume needs to be resized in a subvolume group, then `--group_name` needs to be passed in the command.

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

  For example:

  ```
  [root@mon ~]# ceph fs subvolume resize cephfs01 sub0 1024000000 --group_name subgroup0
  --no_shrink
  ```

Fetching absolute path of a file system subvolume

Learn how to fetch the absolute path of a Ceph File System (CephFS) subvolume.

PREREQUISITE

Be sure that you have a CephFS subvolume.

- Fetch the absolute path of the CephFS subvolume.

Note: The `--group_name` parameter is optional. If the subvolume needs to be fetched from within a subvolume group, then `--group_name` needs to be passed in the command.

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

For example,

```
[root@mon ~]# ceph fs subvolume getpath cephfs01 sub0 --group_name subgroup0

/volumes/subgroup0/sub0/81992fd4-9765-420e-85f9-2545d31ff136
```

Fetching metadata of a file system subvolume

Learn how to fetch metadata of a Ceph File System (CephFS) subvolume.

PREREQUISITE

Be sure that you have a CephFS subvolume.

– Fetch the metadata of a CephFS subvolume.

Note: The `--group_name` parameter is optional. If the subvolume needs to be fetched from within a subvolume group, then `--group_name` needs to be passed in the command.

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

For example:

```
[root@mon ~]# ceph fs subvolume info cephfs01 sub0 --group_name subgroup0
```

Example output:

```
# ceph fs subvolume info cephfs01 sub0
{
    "atime": "2023-07-14 08:52:46",
    "bytes_pcent": "0.00",
    "bytes_quota": 1024000000,
    "bytes_used": 0,
    "created_at": "2023-07-14 08:52:46",
    "ctime": "2023-07-14 08:53:54",
    "data_pool": "cephfs.cephfs01.data",
    "features": [
        "snapshot-clone",
        "snapshot-autoprotect",
        "snapshot-retention"
    ],
    "flavor": "2",
    "gid": 0,
    "mode": 16877,
    "mon_addrs": [
        "10.0.208.172:6789",
        "10.0.211.197:6789",
        "10.0.209.212:6789"
    ],
    "mtime": "2023-07-14 08:52:46",
    "path": "/volumes/_nogroup/sub0/834c5cbc-f5db-4481-80a3-aca92ff0e7f3",
    "pool_namespace": "",
    "state": "complete",
    "type": "subvolume",
    "uid": 0
}
```

The output format is JSON and contains the following fields:

- `atime`: access time of subvolume path in the format "YYYY-MM-DD HH:MM:SS".

- `bytes_pcent`: quota used in percentage if quota is set, else displays "undefined".
- `bytes_quota`: quota size in bytes if quota is set, else displays "infinite".
- `bytes_used`: current used size of the subvolume in bytes.
- `created_at`: time of creation of subvolume in the format "YYYY-MM-DD HH:MM:SS".
- `ctime`: change time of subvolume path in the format "YYYY-MM-DD HH:MM:SS".
- `data_pool`: data pool the subvolume belongs to.
- `features`: features supported by the subvolume, such as, "snapshot-clone", "snapshot-autoprotect", or "snapshot-retention".
- `flavor`: subvolume version, either `1` for version one or `2` for version two.
- `gid`: group ID of subvolume path.
- `mode`: mode of subvolume path.
- `mon_addrs`: list of monitor addresses.
- `mtime`: modification time of subvolume path in the format "YYYY-MM-DD HH:MM:SS".
- `path`: absolute path of a subvolume.
- `pool_namespace`: RADOS namespace of the subvolume.
- `state`: current state of the subvolume, such as, "complete" or "snapshot-retained".
- `type`: subvolume type indicating whether it is clone or subvolume.
- `uid`: user ID of subvolume path.

Creating snapshot of a file system subvolume

Learn how to create snapshots of a Ceph File System (CephFS) subvolume.

PREREQUISITE

Be sure that you have a CephFS subvolume.
In addition to read (`r`) and write (`w`) capabilities, clients also require `s` flag on a directory path within the file system.

Verify that the `s` flag is set on the directory.

```
ceph auth get CLIENT_NAME
```

Output example:

```
CLIENT_NAME
    key = AQAz7EVWygILFRAAdIcuJ12opU/JKyfFmxhuaw==
    caps mds = allow rw, allow rws path=DIRECTORY_PATH
    caps mon = allow r
    caps osd = allow rw tag cephfs01 data=DIRECTORY_NAME
```

In the following example, `client.0` can create or delete snapshots in the `bar` directory of file system `cephfs_a`.

```
[root@mon ~]# ceph auth get client.0
[client.0]
    key = AQAz7EVWygILFRAAdIcuJ12opU/JKyfFmxhuaw==
    caps mds = "allow rw, allow rws path=/bar"
    caps mon = "allow r"
    caps osd = "allow rw tag cephfs data=cephfs_a"
```

– Create a snapshot of the Ceph File System subvolume.

> **Note:** The `--group_name` parameter is optional. If the subvolume is present within a subvolume group, then `--group_name` needs to be passed in the command.

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

For example:

```
[root@mon ~]# ceph fs subvolume snapshot create cephfs01 sub0 snap0 --group_name
subgroup0
```

Cloning subvolumes from snapshots

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

PREREQUISITE

Be sure that you have a CephFS subvolume.
In addition to read (`r`) and write (`w`) capabilities, clients also require `s` flag on a directory path within the file system.

Verify that the `s` flag is set on the directory.

```
ceph auth get CLIENT_NAME
```

Output example:

```
CLIENT_NAME
    key = AQAz7EVWygILFRAAdIcuJ12opU/JKyfFmxhuaw==
    caps mds = allow rw, allow rws path=DIRECTORY_PATH
    caps mon = allow r
    caps osd = allow rw tag cephfs01 data=DIRECTORY_NAME
```

In the following example, `client.0` can create or delete snapshots in the `bar` directory of file system `cephfs_a`.

```
[root@mon ~]# ceph auth get client.0
[client.0]
    key = AQAz7EVWygILFRAAdIcuJ12opU/JKyfFmxhuaw==
    caps mds = "allow rw, allow rws path=/bar"
    caps mon = "allow r"
    caps osd = "allow rw tag cephfs data=cephfs_a"
```

For more information, see “Managing Ceph users” on page 0.

> **Note:** Cloning is inefficient for very large data sets.

1. Create a Ceph File System (CephFS) volume.
   This creates the CephFS file system, its data, and metadata pools.

   > **Note:** The `--group_name` parameter is optional. If the subvolume needs to be created in a subvolume group, then `--group_name` needs to be passed in the command.

   ```
   ceph fs volume create VOLUME_NAME
   ```

   For example,

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

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

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

For example,

```
[root@mon ~]# ceph fs subvolumegroup create cephfs01 subgroup0
```

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

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

For example,

```
[root@mon ~]# ceph fs subvolume create cephfs01 sub0 subgroup0
```

4. Create a snapshot of a subvolume.

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

For example,

```
[root@mon ~]# ceph fs subvolume snapshot create cephfs01 sub0 snap0  --group_name
subgroup0
```

5. Initiate a clone operation.

> **Note:** By default, cloned subvolumes are created in the default group.

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

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

   For example,

```
[root@mon ~]# ceph fs subvolume snapshot clone cephfs01 sub0 snap0 clone0
```

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

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

   For example,

```
[root@mon ~]# ceph fs subvolume snapshot clone cephfs01 sub0 snap0 clone0 --
group_name subgroup0
```

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

```
ceph fs subvolume snapshot
clone VOLUME_NAME SUBVOLUME_NAME SNAP_NAME TARGET_CLONE_NAME --
target_group_name SUBVOLUME_GROUP_NAME
```

   For example,

```
[root@mon ~]# ceph fs subvolume snapshot clone cephfs01 sub0 snap0 clone0 --
target_group_name subgroup1
```

6. Check the status of the clone operation.

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

For example,

```
[root@mon ~]# ceph fs clone status cephfs01 clone0 --group_name subgroup1

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

Listing snapshots of a file system subvolume

Learn how to list the snapshots of a Ceph File system (CephFS) subvolume.

PREREQUISITE

Be sure that you have a CephFS subvolume and snapshots of the subvolume.

– List the snapshots of a CephFS subvolume.

> **Note:** The `--group_name` parameter is optional. If the subvolume is present within the subvolume group, then `--group_name` needs to be passed in the command.

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

For example:

```
[root@mon ~]# ceph fs subvolume snapshot ls cephfs01 sub0 --group_name subgroup0

[
{
"name": "snap0"
}
]
```

Fetching metadata of the snapshots of a file system subvolume

Learn how to list the snapshots of a Ceph File system (CephFS) subvolume.

PREREQUISITE

Be sure that you have a CephFS subvolume and snapshots of the subvolume.

– Fetch the metadata of the snapshots of a CephFS subvolume.

> **Note:** The `--group_name` parameter is optional. If the subvolume is present within a subvolume group, then `--group_name` needs to be passed in the command.

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

For example:

```
[root@mon ~]# ceph fs subvolume snapshot info cephfs01 sub0 snap0 --group_name
subgroup0
```

Example output:

```
{
    "created_at": "2022-05-09 06:18:47.330682",
    "data_pool": "cephfs_data",
    "has_pending_clones": "no"
}
```

The output format is JSON and contains the following fields:

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

Removing a file system subvolume

Learn how to remove the Ceph File System (CephFS) subvolume.

PREREQUISITE

Be sure that you have a CephFS subvolume.

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

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

– Remove a CephFS subvolume.

> **Note:** The `--group_name` parameter is optional. If the subvolume needs to be removed from within in a subvolume group, then `--group_name` needs to be passed in the command.

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

For example:

```
[root@mon ~]# ceph fs subvolume rm cephfs01 sub0 --group_name subgroup0 --retain-
snapshots
```

Recreating a subvolume from a retained snapshot

To recreate a subvolume from a retained snapshot, use

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

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

For example:

```
[root@mon ~]# ceph fs subvolume snapshot clone cephfs01 sub0 snap0 sub1 --group_name
subgroup0 --target_group_name subgroup0
```

Verifying the state of a file subvolume

Verify the state of a file subvolume, by using the **`subvolume info`** command.
For example,

```
[root@mon ~]# ceph fs subvolume info cephfs01 sub1 subgroup0

{
"atime": "2025-01-08 04:21:37",
"bytes_pcent": "undefined",
"bytes_quota": "infinite",
"bytes_used": 0,
"created_at": "2025-01-08 04:42:46",
"ctime": "2025-01-08 04:42:47",
"data_pool": "cephfs.cephfs01.data",
"earmark": "",
"features": [
"snapshot-clone",
"snapshot-autoprotect",
"snapshot-retention"
],
"flavor": 2,
"gid": 0,
"mode": 16877,
"mon_addrs": [
"10.0.64.12:6789",
"10.0.67.102:6789",
"10.0.66.135:6789"
],
"mtime": "2025-01-08 04:21:37",
"path": "/volumes/subgroup0/sub1/6bd615e9-8985-4c25-8dd9-c80d9ebb8033",
"pool_namespace": "fsvolumens_sub0",
"state": "complete",
"type": "clone",
"uid": 0
}
```

Removing snapshot of a file system subvolume

Learn how to remove snapshots of a Ceph File System (CephFS) subvolume group.

PREREQUISITE

Be sure that you have a CephFS subvolume and a snapshot of the subvolume group.
Using the `--force` flag allows the command to succeed that would otherwise fail if the snapshot did not exist.

- Remove the snapshot of the CephFS subvolume.

> **Note:** The `--group_name` parameter is optional. If the subvolume is present within a subvolume group, then `--group_name` needs to be passed in the command.

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

For example:

```
[root@mon ~]# ceph fs subvolume snapshot rm cephfs01 sub0 snap0 --group_name subgroup0
--force

[]
```

Managing Character Mapping for subvolume

Learn how to manage character mapping in CephFS.

The CephFS subvolume and subvolumegroup interfaces provide a porcelain API to manage character mapping `ceph.dir.charmap` settings. These settings support name normalization and case sensitivity options.

Configuring Character Mapping

Configure character mapping for subvolume.

To configure character mapping for subvolume:

```
ceph fs subvolume charmap set vol_name subvol group_name=name  setting value
```

Reading Character Mapping

Read character mapping for subvolume.

To read the character mapping for subvolume:

```
ceph fs subvolume charmap get vol_name subvol group_name=name setting
```

For example,

```
ceph fs subvolume charmap get vol subvol group_name=csi casesensitive
```

To read the full character mapping for subvolume:

```
ceph fs subvolume charmap get vol_name subvol group_name=name
```

For example,

```
ceph fs subvolumegroup charmap get vol csi
```

Outputs:

```
{"casesensitive":false,"normalization":"nfd","encoding":"utf8"}
```

Removing Character Mapping

Remove character mapping for subvolume.

> **Note:** A charmap can only be removed when a subvolume is empty.

To remove character mapping for subvolume:

```
ceph fs subvolume charmap rm vol_name subvol group_name=name
```

CephFS subvolume groups

You can create, list, fetch absolute path, and remove Ceph File System (CephFS) subvolume groups. subvolume groups are abstractions at a directory level that effects policies. For example, file layouts, across a set of subvolumes.

> **Note:** The subvolume group snapshot feature is not supported. You can only list and remove the existing snapshots of these subvolume groups.

Before working with Ceph File System subvolume groups be sure that you have:

- A working Red Hat Ceph Storage cluster with CephFS deployed.
- A minimum read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.

Creating a file system subvolume group

Learn how to create a Ceph File System (CephFS) subvolume group.

Note:

- When creating a subvolume group, you can specify its data pool layout, UID, GID, and file mode in octal numerals. By default, the subvolume group is created with an octal file mode '755', uid '0', gid '0', and data pool layout of its parent directory.
- To set quotas while creating a subvolume group, see "Setting and managing quotas on a file system subvolume group" on page 58.

- Create a CephFS subvolume group.

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

For example,

```
[ceph: root@host01 /]# ceph fs subvolumegroup create cephfs01 subgroup0
```

The command succeeds even if the subvolume group already exists.

Setting and managing quotas on a file system subvolume group

Learn how to set and manage quotas on a Ceph File System (CephFS) subvolume group.

1. Set quotas while creating a subvolume group by providing size in bytes.

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

For example:

```
[ceph: root@host01 /]# ceph fs subvolumegroup create cephfs01 subvolgroup_2
10737418240
```

2. Resize a subvolume group.

```
ceph fs subvolumegroup resize VOLUME_NAME GROUP_NAME new_size [--no_shrink]
```

For example:

```
[ceph: root@host01 /]# ceph fs subvolumegroup resize cephfs01 subvolgroup_2
20737418240
[
    {
        "bytes_used": 10768679044
    },
    {
        "bytes_quota": 20737418240
    },
    {
        "bytes_pcent": "51.93"
    }
]
```

3. Fetch the metadata of a subvolume group.

```
ceph fs subvolumegroup info VOLUME_NAME GROUP_NAME
```

For example:

```
[ceph: root@host01 /]# ceph fs subvolumegroup info cephfs01 subvolgroup_2
{
    "atime": "2022-10-05 18:00:39",
    "bytes_pcent": "51.85",
    "bytes_quota": 20768679043,
    "bytes_used": 10768679044,
    "created_at": "2022-10-05 18:00:39",
    "ctime": "2022-10-05 18:21:26",
    "data_pool": "cephfs.cephfs01.data",
    "gid": 0,
    "mode": 16877,
    "mon_addrs": [
        "60.221.178.236:1221",
        "205.64.75.112:1221",
        "20.209.241.242:1221"
    ],
    "mtime": "2022-10-05 18:01:25",
    "uid": 0
}
```

Listing file system subvolume groups

Learn how to list the Ceph File System (CephFS) subvolume groups.

PREREQUISITE

Be sure that you have a CephFS subvolume group.

- List the CephFS subvolume groups.

  ```
  ceph fs subvolumegroup ls VOLUME_NAME
  ```

  For example:

  ```
  [ceph: root@host01 /]# ceph fs subvolumegroup ls cephfs01
  [
  {
  "name": "subgroup0"
  }
  ]
  ```

Fetching the absolute path of a file system subvolume group

Learn how to fetch the absolute path of a Ceph File System (CephFS) subvolume group.

PREREQUISITE

Be sure that you have a CephFS subvolume group.

- Fetch the absolute path of the CephFS subvolume group.

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

  For example:

  ```
  [ceph: root@host01 /]# ceph fs subvolumegroup getpath cephfs01 subgroup0
  /volumes/subgroup0
  ```

Listing snapshots of a file system subvolume group

Learn how to list the snapshots of a Ceph File System (CephFS) subvolume group.

PREREQUISITE

Be sure that you have a CephFS subvolume group and snapshots of a subvolume group.

- List the snapshots of a CephFS subvolume group.

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

For example:

```
[ceph: root@host01 /]# ceph fs subvolumegroup snapshot ls cephfs01 subgroup0

[]
```

Removing snapshot of a file system subvolume group

Learn how to remove snapshots of a Ceph File System (CephFS) subvolume group.

PREREQUISITE

Be sure that you have a CephFS subvolume group and snapshots of a subvolume group.

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

- Remove the snapshot of the CephFS subvolume group.

```
ceph fs subvolumegroup snapshot rm VOLUME_NAME GROUP_NAMESNAP_NAME  [--force]
```

For example:

```
[ceph: root@host01 /]# ceph fs subvolumegroup snapshot rm cephfs01 subgroup0 snap0 --
force
```

Removing a file system subvolume group

Learn how to remove the Ceph File System (CephFS) subvolume group.

PREREQUISITE

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

- No subvolumes under the targeted subvolume group.
  For more information, see “Removing a file system subvolume” on page 55.
- No retained snaphshots under the targeted subvolume group.
  For more information, see “Removing snapshot of a file system subvolume” on page 56.

> **Note:** The removal of a subvolume group fails if it is not empty or nonexistent. The `--force` flag allows the nonexistent subvolume group to be removed.

- Remove the CephFS subvolume group.

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

For example:

```
[ceph: root@host01 /]# ceph fs subvolumegroup rm cephfs01 subgroup0 --force
```

Managing Character Mapping for subvolume group

Learn how to manage character mapping in CephFS.

The CephFS subvolume and subvolumegroup interfaces provide a porcelain API to manage character mapping `ceph.dir.charmap` settings. These settings support name normalization and case sensitivity options.

Configuring Character Mapping

Configure character mapping for a subvolumegroup.

To configure the character mapping for subvolumegroup:

```
ceph fs subvolumegroup charmap set vol_name group_name setting value
```

For example,

```
ceph fs subvolumegroup charmap set vol csi normalization nfd
```

Outputs:

```
{"casesensitive":true,"normalization":"nfd","encoding":"utf8"}
```

Reading Character Mapping

Read character mapping for a subvolumegroup.

To read the character mapping for subvolumegroup:

```
ceph fs subvolumegroup charmap get vol_name group_name setting
```

To read the full character mapping for subvolumegroup:

```
ceph fs subvolumegroup charmap get vol_name group_name
```

For example,

```
ceph fs subvolumegroup charmap get vol csi
```

Outputs:

```
{"casesensitive":false,"normalization":"nfd","encoding":"utf8"}
```

Removing Character Mapping

Remove character mapping for a subvolumegroup.

> **Note:** A charmap can only be removed when a subvolumegroup is empty.

To remove the character mapping for subvolumegroup:

```
ceph fs subvolumegroup charmap rm <vol_name> <group_name>
```

For example,

```
ceph fs subvolumegroup charmap rm vol csi
```

Outputs:

```
{}
```

Metadata information on CephFS subvolumes

You can set, get, list, and remove metadata information of Ceph File System (CephFS) subvolumes.

The custom metadata is for users to store their metadata in subvolumes. Users can store the key-value pairs similar to `xattr` in a Ceph File System.

Before working with the metadata on CephFS subvolumes be sure that you have:

- A running Red Hat Ceph Storage cluster.
- A CephFS volume, subvolume group, and subvolume created.

Viewing subvolume metrics for CephFS metadata server clients

To access subvolume-level metrics, a client can mount a Ceph File System (CephFS) or a specific subvolume.

To access client-specific metrics, a user mounts the complete CephFS. This exposes client-specific metrics of all the operations at the CephFS volume level.

> **Note:** When a user mounts an entire CephFS, it leads to performance degradation of the specific file system. To avoid this, users should perform a subvolume level mount.

Using a subvolume level mount has the following benefits:

- It improves the file system performance.
- It enables clients to identify and isolate their operations to specific subvolume paths.

Metrics for throughput, IOPS, latency (read and write), byte consumption, and quota limits are implemented at the subvolume level. These subvolume metrics are collected from all Metadata Servers (MDSs), shared with the metrics aggregator, and aggregated within a sliding window.

Sliding window

A sliding window is a fixed-size time interval that moves forward incrementally. In CephFS, the sliding window monitors continuous streams of file system activity and performance data. A subvolume is considered active if operations occur within the sliding window. The default window size is 30 seconds.

> **Note:** If no activity is detected during this interval, the subvolume is considered inactive.

The subvolume state (active or inactive) impacts on metrics generation.

Active

The subvolume generates metrics immediately.

Inactive

The subvolume does not generate any metrics and no value is reported.

Sliding window interval for subvolume metrics

The sliding window algorithm includes the `subv_metrics_window_interval` output value. Both the default and minimum values are defined at 30 seconds.

> **Note:**
> - The sliding windows is only available for subvolume metrics, not for client metrics.
> - An administrator can configure different default sliding window values for different file systems.

```
name: subv_metrics_window_interval
  type: secs
  level: dev
  desc: subvolume metrics sliding window interval, seconds
  long_desc: interval in seconds to hold values in sliding window for subvolume metrics
  default: 30
  min: 30
  services:
  - mds
```

- Get the `subv_metrics_window_interval` default value by using the **`ceph config get mds`** command.

```
ceph config get mds subv_metrics_window_interval
```

For example,

```
[root@mon ~]# ceph config get mds subv_metrics_window_interval
30
```

– Change the `subv_metrics_window_interval` default value by using the **`ceph config set mds`** command.

```
ceph config set mds subv_metrics_window_interval 60
```

For example,

```
[root@mon ~]# ceph config get mds subv_metrics_window_interval
30
[root@mon ~]# ceph config set mds subv_metrics_window_interval 60
[root@mon ~]# ceph config get mds subv_metrics_window_interval
60
```

Getting subvolume metrics

CephFS exports aggregated metrics per subvolume using the client `asok` command **`dump_subvolume_metrics`** . To view all metrics, including the **`mds_subvolume_metrics`** section, run the following command:

```
ceph tell mds.cephfs.ceph-gene-amk-diyngp-node4.paipgc counter dump
```

CephFS exports gauge subvolume metrics, as defined in “Subvolume performance metrics description” on page 63.

Subvolume performance metrics description

Name	Type	Description
`mds_subvolume_metrics`	Array	List of subvolume metric objects .
`labels`	Object	Metadata identifying the file system and subvolume path.
`fs_name`	String	Name of the file system.
`subvolume_path`	String	Path of the subvolume within the file system.
`counters`	Object	Performance metrics for the subvolume.
`avg_read_iops`	Gauge	Average read IOPS (input/output operations per second) over the sliding window.
`avg_read_tp_Bps`	Gauge	Average read throughput in bytes per second.
`avg_read_lat_msec`	Gauge	Average read latency in milliseconds.
`avg_write_iops`	Gauge	Average write IOPS over the sliding window.
`avg_write_tp_Bps`	Gauge	Average write throughput in bytes per second.
`avg_write_lat_msec`	Gauge	Average write latency in milliseconds.
`quota_bytes`	Gauge	The quota limit in bytes configured for the subvolume. A value of 0 indicates no quota is set (unlimited).
`used_bytes`	Gauge	The number of bytes currently consumed by the subvolume.

The subvolume metrics are dumped as a part of the same command. The `mds_subvolume_metrics` section in the output of **`counter dump`** command displays the metrics for each client.

The following is an example with an unlimited quota:

```
"mds_subvolume_metrics": [
    {
        "labels": {
            "fs_name": "cephfs",
            "subvolume_path": "/volumes/_nogroup/test_subvolume"
        },
        "counters": {
            "avg_read_iops": 4,
            "avg_read_tp_Bps": 399,
            "avg_read_lat_msec": 0,
            "avg_write_iops": 2780,
            "avg_write_tp_Bps": 11374658,
            "avg_write_lat_msec": 409,
            "quota_bytes": 0,
            "used_bytes": 2097285
        }
    }
]
```

The following example shows a subvolume with `quota_bytes` set to 10 GB and a current usage (`used_bytes`) of approximately 8.75 MB.

```
"mds_subvolume_metrics": [
    {
        "labels": {
            "fs_name": "cephfs",
            "subvolume_path": "/volumes/_nogroup/test_subvolume"
        },
        "counters": {
            "avg_read_iops": 4,
            "avg_read_tp_Bps": 399,
            "avg_read_lat_msec": 0,
            "avg_write_iops": 2780,
            "avg_write_tp_Bps": 11374658,
            "avg_write_lat_msec": 409,
            "quota_bytes": 10737418240,
            "used_bytes": 8754412
        }
    }
]
```

Setting custom metadata on the file system subvolume

Learn how to set custom metadata on the file system subvolume as a key-value pair.

Important: Custom metadata on a subvolume is not preserved when creating a snapshot of the subvolume, and hence, is also not preserved when cloning the subvolume snapshot.

Note:

- If the `key_name` exists, then the old value is replaced by the new value.
- Be sure that the *KEY_NAME* and *VALUE* is a string of ASCII characters, as specified in python's `string.printable`. The *KEY_NAME* is case-insensitive and is always stored in lowercase.

1. Set the metadata on the CephFS subvolume.

   ```
   ceph fs subvolume metadata set VOLUME_NAME SUBVOLUME_NAME KEY_NAME VALUE [--
   group_name SUBVOLUME_GROUP_NAME]
   ```

   For example:

   ```
   [ceph: root@host01 /]# ceph fs subvolume metadata set cephfs01 sub0 test_meta cluster
   --group_name subgroup0
   ```

2. Optional: Set the custom metadata with a space in the *KEY_NAME*.
   This creates another metadata with *KEY_NAME* as `test meta` for the *VALUE* `cluster`.

For example:

```
[ceph: root@host01 /]# ceph fs subvolume metadata set cephfs01 sub0 "test meta"
cluster --group_name subgroup0
```

3. Optional: Set the same metadata with a different value.
   For example:

```
[ceph: root@host01 /]# ceph fs subvolume metadata set cephfs01 sub0 "test_meta"
cluster2 --group_name subgroup0
```

Getting custom metadata on the file system subvolume

Learn how to get the custom metadata, the key-value pairs, of a Ceph File System (CephFS) in a volume, and optionally, in a specific subvolume group.

AFTER COMPLETING THE TASK

Get the metadata on the CephFS subvolume:

```
ceph fs subvolume metadata get VOLUME_NAME SUBVOLUME_NAME KEY_NAME [--
group_name SUBVOLUME_GROUP_NAME]
```

For example:

```
[ceph: root@host01 /]# ceph fs subvolume metadata get cephfs01 sub0 test_meta --group_name
subgroup0

cluster
```

Listing custom metadata on the file system subvolume

Learn how to list the custom metadata associated with the key of a Ceph File System (CephFS) in a volume, and optionally, in a specific subvolume group.

List the metadata on the CephFS subvolume.

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

For example:

```
[ceph: root@host01 /]# ceph fs subvolume metadata ls cephfs01 sub0
{
    "test_meta": "cluster"
}
```

Removing custom metadata from the file system subvolume

Learn how to remove the custom metadata, the key-value pairs, of a Ceph File System (CephFS) in a volume, and optionally, in a specific subvolume group.

1. Remove the custom metadata on the CephFS subvolume.

   ```
   ceph fs subvolume metadata rm VOLUME_NAME SUBVOLUME_NAME KEY_NAME [--
   group_name SUBVOLUME_GROUP_NAME]
   ```

   For example:

   ```
   [ceph: root@host01 /]# ceph fs subvolume metadata rm cephfs01 sub0 test_meta --
   group_name subgroup0
   ```

2. List the metadata.
   For example:

```
[ceph: root@host01 /]# ceph fs subvolume metadata ls cephfs01 sub0 --group_name
subgroup0

{}
```

Administering CephFS

Learn how to perform common Ceph File System (CephFS) administrative tasks.

Common CephFS administrative tasks are:

- Monitoring CephFS metrics in real time
- Mapping a directory to a particular MDS rank
- Disassociating a directory from an MDS rank
- Adding a new data pool
- Working with files and directory layouts
- Removing a Ceph File System
- Client features
- Using the **`ceph mds fail`** command
- Manually evict a CephFS client

Before performing administrative tasks on Ceph File Systems, be sure that you have the following:

- A running, and healthy Red Hat Ceph Storage cluster.
- Ceph Metadata Server daemons (`ceph-mds`) installed and configured.
- A Ceph File System that is created and mounted.

For more information, see:

- "Deploying the CephFS" on page 33
- Installing

Using the `cephfs-top` utility

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

PREREQUISITE

Before using the `cephfs-top` utility, be sure that you have the following:

- A healthy and running Red Hat Ceph Storage cluster.
- Deployment of a Ceph File System.
- Root-level access to a Ceph client node.
- `cephfs-top` package installed.

Currently, the `cephfs-top` utility supports nearly 10,000 clients.

> **Note:** Not all of the performance stats are available in the Red Hat Enterprise Linux 9 kernel.

1. Enable the Red Hat Ceph Storage 9 Tools repository:

```
subscription-manager repos --enable=rhceph-9-tools-for-rhel-9-x86_64-rpms
```

2. Install the `cephfs-top` package.

```
dnf install cephfs-top
```

For example,

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

3. Enable the Ceph Manager `stats` plug-in.

```
ceph mgr module enable stats
```

For example,

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

4. Create the `client.fstop` Ceph user.

```
ceph auth get-or-create client.fstop mon 'allow r' mds 'allow r' osd 'allow r' mgr
'allow r' > /etc/ceph/ceph.client.fstop.keyring
```

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

5. Start the `cephfs-top` utility, by using the **`cephfs-top`** command.
For example:

```
[root@client ~]# cephfs-top
cephfs-top - Thu Oct 20 07:29:35 2022
Total Client(s): 3 - 2 FUSE, 1 kclient, 0 libcephfs
HELP: All filesystem info [Press 'm' to select a filesystem and 'q' to quit]

Filesystem: cephfs01 - 2 client(s)

  client_id mount_root chit(%) dlease(%) ofiles oicaps oinodes rtio(MB) raio(MB)
rsp(MB/s) wtio(MB) waio(MB) wsp(MB/s) rlatavg(ms) rlatsd(ms) wlatavg(ms) wlatsd(ms)
mlatavg(ms) mlatsd(ms) mount_point@host/addr
  4880     /          100.00   0.0          0      1    0       0.0      0.0      0.0
       0.0     0.00     0.0       0.0         0.0        0.0          0.0        0.27
        0.0       /mnt/cephfs@example/127.0.0.1
  4885     /          100.00   0.0          0      1    0       0.0      0.0      0.0
       0.0     0.00     0.0       0.0         0.0        0.0          0.0        0.35
        0.0        N/A@cephfs@example/127.0.0.1

Filesystem: cephfs02 - 2 client(s)

  client_id mount_root chit(%) dlease(%) ofiles oicaps oinodes rtio(MB) raio(MB)
rsp(MB/s) wtio(MB) waio(MB) wsp(MB/s) rlatavg(ms) rlatsd(ms) wlatavg(ms) wlatsd(ms)
mlatavg(ms) mlatsd(ms) mount_point@host/addr
  4889     /          100.0   0.0           0      1      0       0.0      0.0
 0.0           0.0      0.0      0.0       0.0          0.0        0.0          0.0
   0.45        0.0        /mnt/cephfs@example/127.0.0.1
```

`cephfs-top` utility interactive commands

Select a particular file system and view the metrics related to that file system with the `cephfs-top` utility interactive commands.

`m`

Description

Displays a menu of file systems for selection.

`q`

Description

Exits the utility if you are at the home screen with all file system information. If you are not at the home screen, it redirects you back to the home screen.

s

Description

Designates the sort field. **cap_hit** is the default.

l

Description

Sets the limit on the number of clients to be displayed.

r

Description

Resets the sort field and limit value to the default.

The metrics display can be scrolled using the Arrow Keys, PgUp/PgDn, Home/End and mouse.

Example of entering and exiting the file system selection menu

```
[root@client ~]# m

                        Filesystems
Press "q" to go back to home (all filesystem info) screen



                         cephfs01
                         cephfs02
[root@client ~]# q

cephfs-top - Thu Oct 20 07:29:35 2022
Total Client(s): 3 - 2 FUSE, 1 kclient, 0 libcephfs
```

`cephfs-top` utility options

Use the `cephfs-top` utility command with various options.

Example

```
[root@client ~]# cephfs-top --selftest
selftest ok
```

This list describes the different options and their descriptions.

`--cluster` *`NAME_OF_THE_CLUSTER`*

With this option, you can connect to the non-default cluster name. The default name is `ceph`.

`--id` *`USER`*

This is a client which connects to the Ceph cluster and is `fstop` by default.

`--selftest`

With this option, you can perform a selftest. This mode performs a sanity check of `stats` module.

`--conffile` *`PATH_TO_THE_CONFIGURATION_FILE`*

With this option, you can provide a path to the Ceph cluster configuration file.

`-d/--delay` *`INTERVAL_IN_SECONDS`*

The `cephfs-top` utility refreshes statistics every second by default. With this option, you can change a refresh interval.

> **Note:** Interval should be greater than or equal to 1 seconds. Fractional seconds are honored.

`--dump`

With this option, you can dump the metrics to stdout without creating a curses display use.

`--dumpfs` *`FILE_SYSTEM_NAME`*

With this option, you can dump the metrics of the given filesystem to stdout without creating a curses display use.

Using the MDS autoscaler module

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

PREREQUISITE

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

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

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

- `max_mds` file system setting
- `standby_count_wanted` file system setting

The Ceph monitor daemons are still responsible for promoting or stopping MDS according to these settings. The `mds_autoscaler` adjusts the number of MDS that are created by the orchestrator.

- Enable the MDS autoscaler module.
  For example:

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

Unmounting Ceph File Systems mounted as kernel clients

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

For more information, see the `umount(8)` manual page.

PREREQUISITE

Before you begin, be sure that you have root-level access to the node that is mounting.

- Unmount a Ceph File System mounted as a kernel client.

```
umount MOUNT_POINT
```

  For example:

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

Unmounting Ceph File Systems mounted as FUSE clients

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

PREREQUISITE

Before beginning this task, be sure that you have root-level access to the FUSE client node.
For more information, see the `ceph-fuse(8)` manual page.

- Unmount a Ceph File System mounted in FUSE.

```
fusermount -u MOUNT_POINT
```

  For example,

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

Mapping directory trees to Metadata Server daemon ranks

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

PREREQUISITE

Before you begin, verify that the `attr` package is installed on the CephFS client node with a mounted Ceph File System. Also, be sure that you have the following:

- At least two active MDS daemons.
- Installation of `attr` package.
- User access to the CephFS client node.

> **Important:** An internal balancer already dynamically spreads the application load. Therefore, only map directory trees to ranks for certain carefully chosen applications.
> In addition, when a directory is mapped to a rank, the balancer cannot split it. Therefore, a large number of operations within the mapped directory can overload the rank and the MDS daemon that manages it.

For more information, see:

- “Layout, quota, snapshot, and network restrictions” on page 33
- “Manually pinning directory trees to a particular rank” on page 29
- “Configuring multiple active Metadata Server daemons” on page 25

1. Add the `p` flag to the Ceph user’s capabilities.

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

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

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

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

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

   In the following example, the `/temp` directory is assigned and all of its subdirectories to rank 2.

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

Disassociating directory trees from Metadata Server daemon ranks

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

PREREQUISITE

Before disassociating directory trees, be sure that you have user access to the Ceph File System (CephFS) client node.
Also, ensure that the `attr` package is installed on the client node with a mounted CephFS.

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

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

  For example:

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

Note: Any separately mapped subdirectories of `/home/ceph-user/` are not affected.

AFTER COMPLETING THE TASK

- Syntax

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

Example

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

Related information

Mapping directory trees to Metadata Server daemon ranks

Multiple data pools use-cases

Ceph File System (CephFS) supports creating multiple data pools in a CephFS volume. Each data pool can be assigned to a dedicated subvolume or directory in the same file system. This helps administrators optimize their Ceph cluster's performance and storage behavior.

CephFS is a distributed, POSIX-compliant file system. All CephFS deployments require a metadata RADOS pool. By default, a single, additional RADOS pool is provisioned for file data. Deploying additional data pools can be beneficial for supporting multiple workloads, storage media types, and data protection strategies.

Note: When creating a new CephFS, it is recommended to configure the initial data pool as a replicated SSD pool, rather than using Erasure Coding (EC) or deploying on HDDs. The CephFS metadata and default data pools may be colocated on the same SSDs.

Note: If you do not configure the first data pool as a replicated SSD pool, the configurations cannot be retrofitted unless you:

- Recreate the entire CephFS filesystem
- Migrate all existing data to the new setup

The following lists the use-cases for multiple data pools:

File System as a Service (FSaaS)

Service providers can offer differentiated storage tiers by mapping subvolumes to pools with distinct performance and durability characteristics.

Workload segmentation

Assign different workloads to subvolumes or directories backed by pools optimized for their specific needs. For optimized performance and cost efficiency:

- Use fast media (SSDs) and replicated pools for throughput-intensive workloads requiring low latency and high IOPS.
- Store archival or infrequently accessed data in subvolumes or directories backed by erasure-coded (EC) HDD or Quad-Level Cell (QLC) SSD pools to reduce total cost of ownership (TCO) while maintaining durability.

Benefits of using multiple pools

Administrators can optimize their Ceph clusters by aligning subvolume configurations with workload priorities by selecting appropriate attributes.

Attribute selection considerations for CephFS optimization		
Attributes	**Types**	
Data Protection Scheme	Replicated pools	EC pools
	Recommended for performance-sensitive workloads.	Suitable for cost-efficient, durable storage.
Storage Media	SSD	HDD or QLC SSD
	High throughput and low latency.	Cost-effective capacity for less demanding workloads.
Performance vs. Cost Trade-offs	Throughput-optimized	TCO-optimized
	SSDs with replication for high-performance workloads.	HDDs or QLC SSDs with erasure coding for archival or backup workloads.

By leveraging these attributes and following recommendations, administrators can optimize CephFS deployments to meet diverse operational requirements.

Adding data pools

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

PREREQUISITE

Before you begin, be sure that you have root-level access to the Ceph Monitor node.
Adding data pools can be useful for:

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

Before using another data pool in the Ceph File System, you must add a data pool.

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

1. Create a data pool.

   ```
   ceph osd pool create POOL_NAME
   ```

   Replace *POOL_NAME* with the name of the pool.
   For example:

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

   pool 'cephfs_data_ssd' created
   ```

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

   ```
   ceph fs add_data_pool FS_NAME POOL_NAME
   ```

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

   For example:

```
[ceph: root@host01 /]# ceph fs add_data_pool cephfs cephfs_data_ssd

added data pool 6 to fsmap
```

3. Verify that the pool was successfully added.
   For example:

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

name: cephfs, metadata pool: cephfs_metadata, data pools: [cephfs_data
cephfs_data_ssd]
```

4. Optional: Remove a data pool from the file system.

```
ceph fs rm_data_pool FS_NAME POOL_NAME
```

   For example:

```
[ceph: root@host01 /]# ceph fs rm_data_pool cephfs cephfs_data_ssd

removed data pool 6 from fsmap
```

   After removing a data pool, verify that the pool was successfully removed.
   For example:

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

name: cephfs, metadata pool: cephfs_metadata, data pools: [cephfs_data]
```

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

Related information

Overview of file and directory layouts
Creating client users for a Ceph File System

Taking down a CephFS cluster

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

PREREQUISITE

Be sure that you have root-level access to a Ceph Monitor node.
You can take the CephFS cluster down quickly to test the deletion of a file system and bring the Metadata Server (MDS) daemons down. For example, when practicing a disaster recovery scenario. Doing this sets the `jointable` flag to prevent the MDS standby daemons from activating the file system.

Use one of the following procedures to take down a CephFS cluster. To bring the CephFS cluster back up, see “Bringing the CephFS cluster back up” on page 74.

– Mark the CephFS cluster down.

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

  For example:

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

– Quickly take down a CephFS cluster.

```
ceph fs fail FS_NAME
```

  For example:

```
[ceph: root@host01 /]# ceph fs fail cephfs
```

Bringing the CephFS cluster back up

Use one of the following procedures to bring a CephFS cluster back up after taking it down.

– Bring the CephFS cluster back up.

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

For example:

```
[ceph: root@host01 /]# ceph fs set cephfs down false
```

– Get the CephFS cluster back up, by setting `cephfs` to `joinable`.

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

For example:

```
[ceph: root@host01 /]# ceph fs set cephfs joinable true
cephfs marked joinable; MDS may join as newly active.
```

Removing a CephFS

Learn how to remove a Ceph File System.

PREREQUISITE

Before removing a CephFS, be sure that you have root-level access to a Ceph Monitor node and to back up your data.
You can remove a Ceph File System (CephFS). Before doing so, consider backing up all the data and verifying that all clients have unmounted the file system locally.

> **Warning:** This operation is destructive and will make the data that is stored on the Ceph File System permanently inaccessible.

1. Mark the storage cluster as down:

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

   Replace *FS_NAME* with the name of the Ceph File System that you want to remove.
   For example:

   ```
   [ceph: root@host01 /]# ceph fs set cephfs down true
   cephfs marked down.
   ```

2. Display the status of the Ceph File System.

   ```
   ceph fs status
   ```

   For example:

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

cephfs - 0 clients
======
+------------------+----------+-------+-------+
|       POOL       |   TYPE   |  USED | AVAIL |
+------------------+----------+-------+-------+
|cephfs.cephfs.meta | metadata | 31.5M |  52.6G|
|cephfs.cephfs.data |   data   |    0  |  52.6G|
+------------------+----------+-------+--------+
               STANDBY MDS
cephfs.ceph-host01
cephfs.ceph-host02
cephfs.ceph-host03
```

3. Remove the Ceph File System.

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

Replace *FS_NAME* with the name of the Ceph File System that you want to remove.
For example:

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

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

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

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

Related information

Deleting a pool

Using the `ceph mds fail` command

Use the **`ceph mds fail`** command to mark an MDS daemon as failed or restart a running MDS daemon.

PREREQUISITE

Before using the **`ceph mds fail`** command, make sure that the Ceph MDS daemons are installed and configured.

Marking an MDS daemon as failed

If the daemon was active and a suitable standby daemon was available, and if the standby daemon was active after disabling the `standby-replay` configuration, using this command forces a failover to the standby daemon. By disabling the `standby-replay` daemon, this prevents new `standby-replay` daemons from being assigned.

Restarting a running MDS daemon

If the daemon was active and a suitable standby daemon was available, the "failed" daemon becomes a standby daemon.

For more information about MDS daemons, see:

- “Decreasing the number of active Metadata Server daemons” on page 29
- “Configuring the number of standby daemons” on page 26
- “Metadata Server ranks” on page 19
- Fail an MDS daemon.

```
ceph mds fail MDS_NAME
```

Replace the *MDS_NAME* with the name of the `standby-replay` MDS node.

Note: Use the **`ceph fs status`** command to find the Ceph MDS name.

For example:

```
[ceph: root@host01 /]# ceph mds fail example01
```

Client features

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

Important: CephFS clients missing newly added features are evicted automatically. You can list all the CephFS features by using the **`fs features ls`** command. You can add or remove requirements by using the **`fs required_client_features`** command.

Use the following syntax:

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

Client features

Client feature	Description
`reply_encoding`	The Ceph Metadata Server (MDS) encodes reply requests in extensible format, if the client supports this feature.
`reclaim_client`	The Ceph MDS allows a new client to reclaim another, perhaps a dead, client's state.
`lazy_caps_wanted`	When a stale client resumes, the Ceph MDS only needs to re-issue the capabilities that are explicitly wanted, if the client supports this feature.
`multi_reconnect`	After a Ceph MDS failover event, the client sends a reconnect message to the MDS to reestablish cache states. A client can split large reconnect messages into multiple messages.
`deleg_ino`	A Ceph MDS delegates inode numbers to a client, if the client supports this feature. Delegating inode numbers is a prerequisite for a client to do async file creation.
`metric_collect`	CephFS clients can send performance metrics to a Ceph MDS.
`alternate_name`	CephFS clients can set and understand alternative names for directory entries. This feature allows for encrypted file names.

CephFS client evictions

When a Ceph File System (CephFS) client is unresponsive or misbehaving, it might be necessary to forcibly stop, or evict it from accessing the CephFS.

Evicting a CephFS client prevents it from communicating further with Metadata Server (MDS) daemons and Ceph OSD daemons. If a CephFS client is buffering I/O to the CephFS at the time of eviction, then any un-

flushed data will be lost. The CephFS client eviction process applies to all client types: FUSE mounts, kernel mounts, and any process that uses `libcephfs` API library.

You can evict CephFS clients automatically, if they fail to communicate promptly with the MDS daemon, or manually. For more information, see “Manually evicting a CephFS client” on page 77.

Automatic client eviction

These scenarios cause an automatic CephFS client eviction:

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

Blocklist CephFS clients

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

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

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

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

Manually evicting a CephFS client

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

PREREQUISITE

Before evicting a Ceph File System client, be sure that you have root-level access to the Ceph Monitor node.

1. Review the client list.

```
ceph tell DAEMON_NAME client ls
```

   For example:

```
[ceph: root@host01 /]# ceph tell mds.0 client ls
[
    {
        "id": 47385,
        "entity": {
            "name": {
                "type": "client",
                "num": 47385
            },
            "addr": {
                "type": "any",
                "addr": "10.8.128.23:0",
                "nonce": 428447873
            }
        },
        "state": "open",
        "num_leases": 0,
        "num_caps": 0,
        "request_load_avg": 0,
        "uptime": 829961.12384965899,
        "requests_in_flight": 0,
        "num_completed_requests": 0,
        "num_completed_flushes": 0,
        "reconnecting": false,
        "recall_caps": {
            "value": 0,
            "halflife": 60
        },
        "release_caps": {
            "value": 0,
            "halflife": 60
        },
        "recall_caps_throttle": {
            "value": 0,
            "halflife": 1.5
        },
        "recall_caps_throttle2o": {
            "value": 0,
            "halflife": 0.5
        },
        "session_cache_liveness": {
            "value": 0,
            "halflife": 300
        },
        "cap_acquisition": {
            "value": 0,
            "halflife": 10
        },
        "delegated_inos": [],
        "inst": "client.47385 10.8.128.23:0/428447873",
        "completed_requests": [],
        "prealloc_inos": [],
        "client_metadata": {
            "client_features": {
                "feature_bits": "0x00000000000fffff"
            },
            "metric_spec": {
                "metric_flags": {
                    "feature_bits": "0x000000000000ffff"
                }
            },
            "ceph_sha1": "3c9b67d46bf428c8eb52f31dfd4c722a2e896cf7",
            "ceph_version": "ceph version 17.2.6-96.el9cp
(3c9b67d46bf428c8eb52f31dfd4c722a2e896cf7) quincy (stable)",
            "entity_id": "admin",
            "hostname": "node03",
            "mount_point": "/mnt/cephfs_io_tbmutle114_1",
            "pid": "16699",
            "root": "/"
        }
]
```

2. Evict the specified CephFS client.

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

For example:

```
[ceph: root@host01 /]# ceph tell mds.0 client evict id=4305
```

Removing a CephFS client from the blocklist

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

PREREQUISITE

Before removing a Ceph File System client, be sure that you have root-level access to the Ceph Monitor node.

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

1. Review the blocklist.
   For example:

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

   listed 1 entries
   127.0.0.1:0/3710147553 2022-05-09 11:32:24.716146
   ```

2. Remove the CephFS client from the blocklist.

   ```
   ceph osd blocklist rm CLIENT_NAME_OR_IP_ADDR
   ```

   For example:

   ```
   [ceph: root@host01 /]# ceph osd blocklist rm 127.0.0.1:0/3710147553

   un-blocklisting 127.0.0.1:0/3710147553
   ```

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

   ```
   recover_session=clean
   ```

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

   ```
   client_reconnect_stale=true
   ```

Related information

Mounting the Ceph File System as a FUSE client

Disabling volumes plug-in

Learn how to disable volume plug-ins.

By default the volumes plug-in is enabled and set to always on. However, in certain cases, it might be appropriate to disable it. For example, when a CephFS is in a degraded state, the volumes plug-in commands can accumulate in MGR instead of getting served. This eventually causes policy throttles to take effect and the MGR becomes unresponsive.

In this event, you can disable the volumes plug-in even though it is an always on module in MGR.

Run the following command:

```
ceph mgr module disable volumes --yes-i-really-mean-i
```

> **Warning:** This command disables operations and removes commands of the volumes plug-in as it disables all CephFS services on the Ceph cluster that is accessed through this plug-in.

Another way for disabling volume plug-ins is to disable asynchronous threads that are started by volumes plug-ins for cloning and purging trash.

CephFS subvolume quiesce

The subvolume quiesce provides enterprise-level consistency assurance for multi-client applications with one or more subvolumes. With this feature, you can pause the input/output (IO) to a set of subvolumes of a volume (file system). The ability to pause across all clients helps ensure that any persistent checkpoints the application reaches before the pause are recoverable from the snapshots that are made during the pause.

In cases where many write operations are run, CephFS snapshots do not provide strong consistent backup and disaster recovery for distributed applications. The subvolume quiesce helps provide the consistency that is needed at an enterprise level.

The `volumes` plug-in provides a command-line interface (CLI) to initiate and await the pause for a set of subvolumes. This pause is called a quiesce, which is also the command name.

The `fs quiesce` function is based on a lower-level `quiesce db` service that is provided by the Metadata Server (MDS) daemons, which operates at a file system path for granularity. The `volumes` plug-in maps the subvolume names to their corresponding paths on the file system and then issues the corresponding `quiesce db` command to the MDS.

```
ceph fs quiesce VOLUME_NAME --set-id ID-NAME [GROUP_NAME/]SUB_NAME... --await

# Now, you can perform actions while the IO pause is active, like taking snapshots.

ceph fs quiesce VOLUME_NAME --set-id ID-NAME --release --await
```

After the release command is successful, all members of the set are confirmed as still paused and are released.

Operations

You can request the quiesce for a set of one or more subvolumes (that is paths in a file system). This set is referred to as the quiesce set. A unique set ID identifies every quiesce set. You can manipulate a quiesce set in the following ways:

include

Include one or more subvolumes - quiesce set members.

exclude

Exclude one or more members.

cancel

Cancel the set asynchronously aborting the pause on all its current members.

release

Release the set requesting the end of the pause from all members and expecting an acknowledgment from all clients.

query

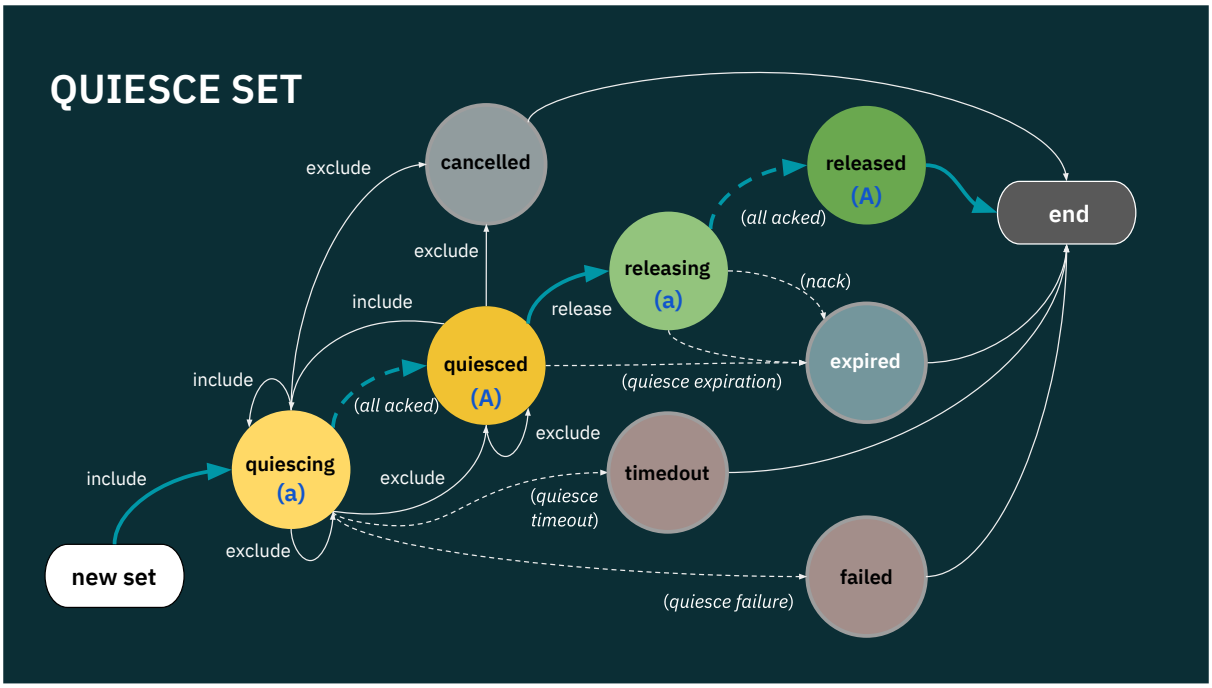
Query the current state of a set by ID or all active or known sets.

cancel all

Cancel all active sets in case an immediate resume of IO is required.

The operations that are listed before are non-blocking that is they attempt the intended modification and return with an up-to-date version of the target set, whether the operation was successful or not. The set can change states due to the modification, and the version that is returned in the response is guaranteed to be consistent with other potentially successful operations from the same control loop batch.

Some states are set `awaitable`. Any of the commands that are mentioned before can be amended with an await modifier, which causes the commands to block on the set after applying their intended modification, as long as the resulting set state is `awaitable`. The command remains blocked until the set reaches the awaited state, gets modified by another command, or transitions into another state. The return code unambiguously identifies the exit condition, and the contents of the response always carry the latest known set state.



`Awaitable` states on the diagram are marked with (a) or (A). Blocking versions of the operations pends while the set is in an (a) state and completes with success if it reaches an (A) state. If the set is already at an (A) state, the operation completes immediately with success.

Most of the operations require a set-ID. The exceptions are:

- Creation of a new set without specifying a set ID.
- Query of active or all known sets.
- The cancel all.

You can create a new set by including members through the **`include`** or **`reset`** commands. You can specify a set ID, and if it is a new ID then the set is created with the specified members in the `QUIESCING` state.

When no set ID is specified while including or resetting the members, then a new set with a unique set ID is created. The set ID is known to the caller by inspecting the output through the following command:

```
ceph fs quiesce fs1 sub1 --set-id=unique-id
{
    "epoch": 3,
    "set_version": 1,
    "sets": {
        "unique-id": {
            "version": 1,
            "age_ref": 0.0,
            "state": {
                "name": "TIMEDOUT",
                "age": 0.0
            },
            "timeout": 0.0,
            "expiration": 0.0,
            "members": {
                "file:/volumes/_nogroup/sub1/b1fcce76-3418-42dd-aa76-f9076d047dd3": {
                    "excluded": false,
                    "state": {
                        "name": "QUIESCING",
                        "age": 0.0
                    }
                }
            }
        }
    }
}
```

However, in the output of this example, the set that is created successfully is already `TIMEDOUT`. This is expected since the timeout parameter is not specified for this quiesce, and in the output it is initialized to 0 by default, along with the expiration.

CephFS subvolume quiesce parameters

Learn about the different parameters that can be used while quiescing the CephFS subvolumes.

Use the following parameters that can be used while quiescing:

- “Timeout” on page 82
- “Awaiting” on page 84
- “Quiesce-Await and Expiration” on page 85
- “If Version” on page 85

Timeout

The two timeout parameters, **`timeout`** and **`expiration`**, are the main guards against accidentally causing a DOS condition for the application. You can add the **`--timeout`** or **`--expiration`** arguments to any commands to update the values for an active set. If these arguments are present, the values are applied before the action requested by the command.

In the following example, `unique-id` set is in a terminal state. No changes are allowed to sets that are in their terminal states that is which are inactive.

```
$ ceph fs quiesce fs1 --set-id=unique-id --timeout=10 > /dev/null
Error EPERM:
```

The following example shows a set-ID that is not specified so the system created a new one. In the output, the set ID is `8988b419`. The command is successful and the set is `QUIESCING`.

```
ceph fs quiesce fs1 sub1 --timeout 60
{
    "epoch": 3,
    "set_version": 2,
    "sets": {
        "8988b419": {
            "version": 2,
            "age_ref": 0.0,
            "state": {
                "name": "QUIESCING",
                "age": 0.0
            },
            "timeout": 60.0,
            "expiration": 0.0,
            "members": {
                "file:/volumes/_nogroup/sub1/b1fcce76-3418-42dd-aa76-f9076d047dd3": {
                    "excluded": false,
                    "state": {
                        "name": "QUIESCING",
                        "age": 0.0
                    }
                }
            }
        }
    }
}
```

The following example is adding more members to the set while quiescing.

> **Note:** The **`--include`** parameter is optional. Include parameter is assumed if no operation is given while members are provided.

```
ceph fs quiesce fs1 --set-id 8988b419 --include sub2 sub3
{
    "epoch": 3,
    "set_version": 3,
    "sets": {
        "8988b419": {
            "version": 3,
            "age_ref": 0.0,
            "state": {
                "name": "QUIESCING",
                "age": 30.7
            },
            "timeout": 60.0,
            "expiration": 0.0,
            "members": {
                "file:/volumes/_nogroup/sub1/b1fcce76-3418-42dd-aa76-f9076d047dd3": {
                    "excluded": false,
                    "state": {
                        "name": "QUIESCING",
                        "age": 30.7
                    }
                },
                "file:/volumes/_nogroup/sub2/bc8f770e-7a43-48f3-aa26-d6d76ef98d3e": {
                    "excluded": false,
                    "state": {
                        "name": "QUIESCING",
                        "age": 0.0
                    }
                },
                "file:/volumes/_nogroup/sub3/24c4b57b-e249-4b89-b4fa-7a810edcd35b": {
                    "excluded": false,
                    "state": {
                        "name": "QUIESCING",
                        "age": 0.0
                    }
                }
            }
        }
    }
}
```

The timeout argument specifies how much time is to be given to the system to reach the QUIESCED state on the set. However, since new members can be added to an active set at any time, the timeout is not measured from the set creation time. Hence, the timeout is tracked per member and every member has timeout seconds to quiesce. If any member takes longer than the timeout seconds, the whole set is marked as `timedout` and the pause is released.

After the set is in the `quiesced` state, it begins the expiration timer. This timer is tracked per set as a whole, not per member. After the expiration seconds elapse, the set changes into an `expired` state, unless a dedicated operation actively releases it or cancels it.

It is possible to add new members to a QUIESCED set. In this case, the set transitions back to `quiescing`, and the new members have their timeout to quiesce. If they succeed, then the set is again `quiesced` and the expiration timer restarts.

> **Warning:**
> The expiration timer does not apply when a set is `quiescing`. It is reset to the value of the expiration property when the set becomes `quiesced`.
>
> Timeout does not apply to members that are `quiesced`.

Awaiting

To wait for the quiesce set to reach the `quiesced` state, you must await it at some point. You can give the **`--await`** argument along with other arguments to let the system know that it must wait for the set to reach the quiesced state.

There are two types of `await`: **`quiesce await`** and **`release await`**. **`Quiesce await`** is the default, and **`release await`** requires the **`--release`** parameter to be present in the argument list.

> **Note:** Issuing a **`quiesce await`** when the set is not `quiescing` is not allowed.

Trying to **`--release`** a set that is not `quiesced` gives an EPERM error regardless of whether await is requested alongside. However, it is not an error to **`release await`** an already released set or to **`quiesce await`** a `quiesced` set.

Since a set is awaited after the application of the **`--await-augmented`** command, the awaited operation can mask a successful result with its error.

The following example shows to **`cancel-await`** a set:

```
ceph fs quiesce fs1 --set-id set1 --cancel --await
{
    // ...
    "sets": {
        "set1": {
            // ...
            "state": {
                "name": "CANCELED",
                "age": 0
            },
            // ...
        }
    }
}
Error EPERM:
```

Awaiting a canceled set is not allowed and results in an EINVAL error. The `--cancel` operation succeeds synchronously for a set in an active state.

When awaiting, you can specify a maximum duration, you want this await request to block. If the target awaited state is not reached within the specified duration, then an error EINPROGRESS is returned.

To resolve the EINPROGRESS error, use the argument **`--await-for`**=*SECONDS*. **`--await`** argument is equivalent to **`--await-for=infinity`**. If both **`--await`** and **`--await-for`** are present, then the time limit from **`--await-for`** is considered.

For example,

```
time ceph fs quiesce fs1 sub1 --timeout=10 --await-for=2
{
    "epoch": 6,
    "set_version": 3,
    "sets": {
        "c3c1d8de": {
            "version": 3,
            "age_ref": 0.0,
            "state": {
                "name": "QUIESCING",
                "age": 2.0
            },
            "timeout": 10.0,
            "expiration": 0.0,
            "members": {
                "file:/volumes/_nogroup/sub1/b1fcce76-3418-42dd-aa76-f9076d047dd3": {
                    "excluded": false,
                    "state": {
                        "name": "QUIESCING",
                        "age": 2.0
                    }
                }
            }
        }
    }
}
Error EINPROGRESS:
ceph fs quiesce fs1 sub1 --timeout=10 --await-for=2  0.41s user 0.04s system 17% cpu 2.563
total
```

Quiesce-Await and Expiration

`Quiesce await` resets the internal expiration timer. This parameter allows for a watchdog approach to a multistep process under the IO pause by repeatedly **`--awaiting`** an already `quiesced` set. Consider the following example script:

```
set -e   # (1)
ceph fs quiesce fs1 sub1 sub2 sub3 --timeout=30 --expiration=10 --set-id="snapshots" --
await # (2)
ceph fs subvolume snapshot create a sub1 snap1-sub1  # (3)
ceph fs quiesce fs1 --set-id="snapshots" --await  # (4)
ceph fs subvolume snapshot create a sub2 snap1-sub2  # (3)
ceph fs quiesce fs1 --set-id="snapshots" --await  # (4)
ceph fs subvolume snapshot create a sub3 snap1-sub3  # (3)
ceph fs quiesce fs1 --set-id="snapshots" --release --await  # (5)
```

> **Warning:** This example uses arbitrary timeouts to convey the concept. In real life, the values must be carefully chosen following the actual system requirements and specifications

The goal of the script is to take consistent snapshots of 3 subvolumes. In the example, the `bash -e` option (1) is set to exit the script if any or the following commands return with a nonzero status.

An input/output (IO) pause is requested for the three subvolumes (2). The timeouts are set to allow the system to spend up to 30 seconds to reach the quiesced state across all members. The members stay quiesced for up to 10 seconds before the quiesce expires and the IO is resumed. The **`--await`** is also specified to only proceed after the quiesce is reached.

The script proceeds with the set of command pairs that take the snapshot and call **`--await`** on set to extend the expiration timeout for 10 more seconds (3,4). This approach gives up to 10 seconds for every snapshot but also allows taking as many snapshots as needed without losing the IO pause. This approach also helps get consistent snapshots. If you want, you can update the expiration every time you call for `await`.

If any of the snapshots get stuck and takes longer than 10 seconds to complete, then the next call to **`--await`** returns an error since the set is in the `expired` state, which is not an `awaitable` state. This helps to limit the impact on the applications in such scenarios.

If Version

It is not enough to only observe the successful quiesce or release. Another client can make a concurrent change to the set.

Consider this example:

```
ceph fs quiesce fs1 sub1 sub2 sub3 --timeout=30 --expiration=60 --set-id="snapshots" --
await  #(1)
ceph fs subvolume snapshot create a sub1 snap1-sub1  # (2)
ceph fs subvolume snapshot create a sub2 snap1-sub2  # (3)
ceph fs subvolume snapshot create a sub3 snap1-sub3  # (4)
ceph fs quiesce fs1 --set-id="snapshots" --release --await  # (5)
```

The sequence looks good, and the release (5) completes successfully. However, it can be that before snap for sub3 (4) is taken, another session excludes sub3 from the set, resuming its input/output (IO):

```
ceph fs quiesce fs1 --set-id="snapshots" --exclude sub3
```

As removing a member from a set does not affect its `quiesced` state, the release command (5) has no reason to fail. It acknowledges the two unexcluded members sub1 and sub2 and reports success.

To address this or similar problems, the quiesce command supports an optimistic concurrency mode. To activate it, pass an **`--if-version=<version>`** that compares the database version of the set and the operation only proceeds if the values match. Otherwise, the command is not run and the return status is `ESTALE`.

Every command that modifies a set returns this set on the stdout, regardless of the exit status. In the example, notice that every set carries a version property, which gets updated whenever this set is modified.

In this example, the initial quiesce command (1) would return the newly created set with ID snapshots and some version, for example, consider 13. Since no other changes are expected to the set while taking the snapshots with the commands (2,3,4), the release command (5) might look like this:

```
ceph fs quiesce fs1 --set-id="snapshots" --release --await --if-version=13 # (5)
```

This way, the release command returns ESTALE instead of 0, indicating that something is not right with the quiesce set and the snapshots are not consistent.

> **Note:** When **`--if-version`** and the command returns `ESTALE`, the requested action is not run. It means that the script wants to run some unconditional command on the set to adjust the state according to the requirements.

The other use of the **`--if-version`** argument is for automation software. It is possible to create a new quiesce set with a given set ID. Drivers like the CSI for Kubernetes can use their internal request ID to eliminate the need to keep an extra mapping to the quiesce set ID. However, to guarantee uniqueness, the driver verifies that the set is new. For that, **`if-version=0`** can be used, and it only creates the new set if no other set with this ID is present in the database:

```
ceph fs quiesce fs1 sub1 sub2 sub3 --set-id="external-id" --if-version=0
```

CephFS quotas

Use Ceph File System (CephFS) quotas to restrict the number of bytes or the number of files that are stored in the directory structure. Ceph File System quotas are fully supported by using a FUSE client or by using Kernel clients, version 4.17 or newer. Learn how to view, set, and remove quotas on any directory in the file system.

When configuring quotas, be sure that you have the following:

- A running, and healthy Red Hat Ceph Storage cluster.
- Deployment of a Ceph File System.
- Root-level access to the Ceph client node.
- The `attr` package installed.

For more information, see:

- “Deploying the CephFS” on page 33.
- The `getfattr(1)` manual page.
- The `setfattr(1)` manual page.

Limitations

- CephFS quotas rely on the cooperation of the client mounting the file system to stop writing data when it reaches the configured limit. However, quotas alone cannot prevent an adversarial, untrusted client from filling the file system.
- Once processes that write data to the file system reach the configured limit, a short period of time elapses between when the amount of data reaches the quota limit, and when the processes stop writing data. The time period generally measures in the tenths of seconds. However, processes continue to write data during that time. The amount of additional data that the processes write depends on the amount of time elapsed before they stop.
- When using path-based access restrictions, be sure to configure the quota on the directory to which the client is restricted, or to a directory nested beneath it. If the client has restricted access to a specific path based on the MDS capability, and the quota is configured on an ancestor directory that the client cannot access, the client will not enforce the quota. For example, if the client cannot access the `/home/` directory and the quota is configured on `/home/`, the client cannot enforce that quota on the directory `/home/user/`.
- Snapshot file data that has been deleted or changed does not count towards the quota.

Viewing quotas

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

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

For more information, see the `getfattr(1)` manual page.

View CephFS quotes in one of the following ways.

- Using a byte-limit quota.

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

For example:

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

In this example, `100000000` equals 100 MB.

- Using a file-limit quota.

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

For example:

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

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

Setting quotas

Learn how to use the **`setfattr`** command and the `ceph.quota` extended attributes to set the quota for a directory.

For more information, see the `setfattr(1)` manual page.

Set CephFS quotes in one of the following ways.

- Using a byte-limit quota.

  Note: The following values are supported for byte-limit quota: K, Ki, M, Mi, G, Gi, T, Ti.

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

  For example:

  ```
  [root@client ~]# setfattr -n ceph.quota.max_bytes -v 2T /mnt/cephfs/
  ```

- Using a file-limit quota.

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

  For example:

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

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

  Note: Only numerical values are supported for the file *LIMIT_VALUE*.

Removing quotas

Learn how to use the **`setfattr`** command and the `ceph.quota` extended attributes to remove a quota from a directory.

For more information, see the `setfattr(1)` manual page.

Remove CephFS quotes in one of the following ways.

- Using a byte-limit quota.

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

  For example:

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

  In this example, `100000000` equals 100 MB.

- Using a file-limit quota.

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

  For example:

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

File and directory layouts

You can control how file or directory data is mapped to objects.

Be sure that you have the following when working with file and directory layouts:

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

- The `attr` package installed.

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

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

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

For more information, see:

- “Deploying the CephFS” on page 33
- The `getfattr(1)` manual page.
- The `setfattr(1)` manual page.

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 to the layout in the directory structure.

Setting file and directory layout fields

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

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

For more information, see the `setfattr(1)` manual page.

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

For example:

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

Viewing file and directory layout fields

Use the **`getfattr`** command to view layout fields on a file or directory.

For more information, see the `getfattr(1)` manual page.

- View layout fields on a file or directory as a single string.

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

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

Replace:

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

For example:

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

Viewing individual layout fields

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

For more information, see the `getfattr(1)` manual page.

– 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.
- *PATH* with the path to the file or directory.

For example:

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

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

Removing directory layouts

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

> **Note:** When you set a file layout, you cannot change or remove it.

For more information, see the `setfattr(1)` manual page.

1. Remove a layout from a directory.

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

   For example:

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

2. Remove the `pool_namespace` field.

   > **Note:** The `pool_namespace` field is the only field that you can remove separately.

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

   For example:

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

CephFS snapshots

Use CephFS snapshots to create an immutable, point-in-time view of the file system. CephFS directory. CephFS snapshots are asynchronous, and you can choose which directory snapshots are created in.

Create snapshots in one of the following ways:

- With the Ceph Dashboard.
  For more information, see "Managing CephFS snapshots" on page 0.
- By invoking the **mkdir** command within the `.snap` hidden directory
- By using CLI commands.

Snapshots can also be automatically created at specific intervals through snapshot schedules and policies.

Before creating CephFS snapshots, be sure to have CephFS deployed and root-level access to the Ceph MetaData Server (MDS) node.

The Ceph File System (CephFS) snapshotting feature is enabled by default on new Ceph File Systems, but must be manually enabled on existing Ceph File Systems. CephFS snapshots are asynchronous and are kept in a special hidden directory in the CephFS directory named `.snap`. You can specify snapshot creation for any directory within a Ceph File System. When specifying a directory, the snapshot also includes all the subdirectories within it.

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

For more information, see "Deploying the CephFS" on page 33 and "CephFS snapshot schedules" on page 94.

CephFS snapshots usage

Use CephFS snapshots for data protection and disaster recovery.

Data protection

Snapshots allow users to recover files from specific points in time. It is useful for creating backups without disrupting ongoing operations.

Disaster recovery

Snapshots are used for CephFS mirroring which enables to replicate CephFS data asynchronously to another Ceph cluster in another location.

CephFS limitations

Before using CephFS, get to know the CephFS limitations.

Snapshot names

Snapshot names cannot start with an underscore (_) and cannot exceed 240 characters.

Performance impact

While creating snapshots is fast, the performance impact can be noticeable during heavy I/O operations due to the asynchronous nature of data flushing.

Creating a snapshot for a Ceph File System

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

> **Note:** For a new Ceph File System, snapshots are enabled by default.

1. Log into the `cephadm` shell.

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

2. For existing Ceph File Systems, enable the snapshot feature.

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

For example:

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

3. Create a snapshot subdirectory under the `.snap` directory.

```
mkdir NEW_DIRECTORY_PATH
```

The following example creates the `new-snaps` subdirectory, informing the Ceph Metadata Server (MDS) to start making snapshots.

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

Deleting snapshots

> **Important:** Attempting to delete root-level snapshots, which might contain underlying snapshots, will fail.

– Delete snapshots.

```
rmdir NEW_DIRECTORY_PATH
```

For example:

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

Cloning subvolumes from snapshots

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

PREREQUISITE

Be sure that you have a CephFS subvolume.
In addition to read (`r`) and write (`w`) capabilities, clients also require `s` flag on a directory path within the file system.

Verify that the `s` flag is set on the directory.

```
ceph auth get CLIENT_NAME
```

Output example:

```
CLIENT_NAME
    key = AQAz7EVWygILFRAAdIcuJ12opU/JKyfFmxhuaw==
    caps mds = allow rw, allow rws path=DIRECTORY_PATH
    caps mon = allow r
    caps osd = allow rw tag cephfs01 data=DIRECTORY_NAME
```

In the following example, `client.0` can create or delete snapshots in the `bar` directory of file system `cephfs_a`.

```
[root@mon ~]# ceph auth get client.0
[client.0]
    key = AQAz7EVWygILFRAAdIcuJ12opU/JKyfFmxhuaw==
    caps mds = "allow rw, allow rws path=/bar"
    caps mon = "allow r"
    caps osd = "allow rw tag cephfs data=cephfs_a"
```

For more information, see “Managing Ceph users” on page 0.

Note: Cloning is inefficient for very large data sets.

1. Create a Ceph File System (CephFS) volume.
   This creates the CephFS file system, its data, and metadata pools.

   Note: The `--group_name` parameter is optional. If the subvolume needs to be created in a subvolume group, then `--group_name` needs to be passed in the command.

   ```
   ceph fs volume create VOLUME_NAME
   ```

   For example,

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

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

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

   For example,

   ```
   [root@mon ~]# ceph fs subvolumegroup create cephfs01 subgroup0
   ```

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

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

   For example,

   ```
   [root@mon ~]# ceph fs subvolume create cephfs01 sub0 subgroup0
   ```

4. Create a snapshot of a subvolume.

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

   For example,

   ```
   [root@mon ~]# ceph fs subvolume snapshot create cephfs01 sub0 snap0  --group_name
   subgroup0
   ```

5. Initiate a clone operation.

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

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

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

For example,

```
[root@mon ~]# ceph fs subvolume snapshot clone cephfs01 sub0 snap0 clone0
```

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

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

For example,

```
[root@mon ~]# ceph fs subvolume snapshot clone cephfs01 sub0 snap0 clone0 --
group_name subgroup0
```

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

```
ceph fs subvolume snapshot
clone VOLUME_NAME SUBVOLUME_NAME SNAP_NAME TARGET_CLONE_NAME --
target_group_name SUBVOLUME_GROUP_NAME
```

For example,

```
[root@mon ~]# ceph fs subvolume snapshot clone cephfs01 sub0 snap0 clone0 --
target_group_name subgroup1
```

6. Check the status of the clone operation.

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

For example,

```
[root@mon ~]# ceph fs clone status cephfs01 clone0 --group_name subgroup1

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

CephFS snapshot schedules

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

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

Scheduling snapshots for a Ceph File System (CephFS) is managed by the `snap_schedule` Ceph Manager module. This module provides an interface to add, query, and delete snapshot schedules, and to manage the retention policies. This module also implements the **`ceph fs snap-schedule`** command, with several subcommands to manage schedules, and retention policies. All subcommands take the CephFS volume path and subvolume path arguments to specify the file system path when using multiple Ceph File Systems. Not

specifying the CephFS volume path, the argument defaults to the first file system listed in the `fs_map`, and not specifying the subvolume path argument defaults to nothing.

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

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

For more information about snapshots, see “CephFS snapshots” on page 91.

Adding a snapshot schedule for a CephFS

Add a snapshot schedule for a CephFS path that does not exist yet. You can create one or more schedules for a single path. Schedules are considered different, if their repeat interval and start times are different.

PREREQUISITE

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

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

A CephFS path can have only one retention policy, but a retention policy can have multiple count-time period pairs.

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

1. Log in to the `cephadm` shell on a Ceph Manager node:
   For example,

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

2. Enable the `snap_schedule` module.
   For example,

   ```
   [ceph: root@host01 ~]# ceph mgr module enable snap_schedule
   ```

3. Log in to the `cephadm` shell on a Ceph Metadata Server node.
   For example,

   ```
   [root@host02 ~]# cephadm shell
   ```

4. Add a schedule for a Ceph File System.

   > **Note:** *`START_TIME`* is represented in ISO 8601 format.

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

In the following example, a snapshot schedule is created for the CephFS `/cephfs` volume, snapshotting every hour, and starts on 27 June 2022 9:50 PM.

```
[ceph: root@host02 ~]# ceph fs snap-schedule add /cephfs_kernelf739cwtus2/pmo9axbwsi
1h 2022-06-27T21:50:00 --fs cephfs01
```

5. Add a retention policy for snapshots of a CephFS volume path.

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

For example:

```
[ceph: root@host02 ~]# ceph fs snap-schedule retention add /cephfs h 14 (1)
[ceph: root@host02 ~]# ceph fs snap-schedule retention add /cephfs d 4 (2)
[ceph: root@host02 ~]# ceph fs snap-schedule retention add /cephfs 14h4w (3)
```

(1) This example keeps 14 snapshots at least one hour apart.
(2) This example keeps 4 snapshots at least one day apart.
(3) This example keeps 14 hourly, and 4 weekly snapshots.

6. List the snapshot schedules to verify that the new schedule is created.

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

The following example lists all schedules in the directory tree.

```
[ceph: root@host02 ~]# ceph fs snap-schedule list /cephfs --recursive=true
```

7. Check the status of a snapshot schedule.

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

The following example displays the status of the snapshot schedule for the CephFS `/cephfs` path in JSON format. If not specified, the default format is plain text.

```
[ceph: root@host02 ~]# ceph fs snap-schedule status /cephfs --format=json
```

Adding a snapshot schedule for CephFS subvolume

Create snapshot schedules and retention policies for directories in a subvolume, subvolume within a default group, and for subvolumes within a non-default group.

PREREQUISITE

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

- A running and healthy Red Hat Ceph Storage cluster with Ceph File System (CephFS) deployed.
- A minimum of read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- A CephFS subvolume and subvolume group created.

Schedules are considered different, if their repeat interval and start times are different.

Add a snapshot schedule for a CephFS file path that does not exist yet. A CephFS path can have only one retention policy, but a retention policy can have multiple count-time period pairs.

> **Note:**

- After the scheduler module is enabled, running the **ceph fs snap-schedule** command displays the available subcommands and their usage format.
- Subvolumes belonging to the default subvolume group and non-default subvolume group can be scheduled for creating snapshots. However, the commands are different.

You can add snapshot schedules to the directories within a subvolume, subvolume in the default group and to the subvolume in a non-default group.

- Add a snapshot schedule to a directory in a subvolume.
  - Get the absolute path of the subvolume where directory exists:
    Syntax:

    ```
    ceph fs subvolume getpath VOLUME_NAME SUBVOLUME_NAME --group-
    name SUBVOLUME_GROUP_NAME
    ```

    Example:

    ```
    [ceph: root@host02 /]# ceph fs subvolume getpath cephfs01 sub0 --group_name
    subgroup0

    /volumes/subgroup0/sub0/db992be0-aacd-414e-8abd-e1bfeb37a4a2
    ```

  - Add a snapshot schedule to a directory in the subvolume:

    Note: *START_TIME* is represented in ISO 8601 format.

    Note: Path in snap-schedule command is <absolute_path_of_ subvolume>/<relative_path_of_test_dir>. Get the absolute_path of the subvolume, from step "1" on page 97.

    Syntax:

    ```
    ceph fs snap-schedule add SUBVOLUME_DIR_PATH SNAP_SCHEDULE [START_TIME] --
    fs CEPH_FILE_SYSTEM_NAME
    ```

    In this example, test_dir is the directory for which the schedule is added.

    ```
    [ceph: root@host02 /]# ceph fs snap-schedule add /volumes/subgroup0/
    sub0/4dee0788-32a0-47e6-ac4b-bd8568a6989b/test_dir 1h --fs cephfs01

    Schedule set for path /volumes/subgroup0/sub0/4dee0788-32a0-47e6-ac4b-
    bd8568a6989b/test_dir
    ```

- Add a snapshot schedule for a CephFS subvolume in the default group:
  Syntax:

  ```
  ceph fs snap-schedule add /.. SNAP_SCHEDULE [START_TIME] --fs CEPH_FILE_SYSTEM_NAME --
  subvol SUBVOLUME_NAME
  ```

  Important: The path (/) must be defined and cannot be left empty. There is no dependency on path string value, you can define it with one of the following characters:
  / - /..

  In this example, a snapshot schedule for the CephFS subvolume sub0 volume is created, taking snapshots every hour, and starts on 27 June 2022 9:50 PM.

```
[ceph: root@host02 /]# ceph fs snap-schedule add / 1h 2022-06-27T21:50:00 --fs cephfs01
--subvol sub0
```

- Add a snapshot schedule for a CephFS subvolume in the non-default group:
  Syntax:

```
ceph fs snap-schedule add /.. SNAP_SCHEDULE [START_TIME] --fs CEPH_FILE_SYSTEM_NAME  --
subvol SUBVOLUME_NAME --group NON_DEFAULT_SUBVOLGROUP_NAME
```

> **Important:** The `path ( / )` must be defined and cannot be left empty. There is no dependency on `path` string value, you can define it with one of the following characters:
> / - /..

In this example, a snapshot schedule is added for the Ceph subvolume `sv_non_def_1` belonging to the non default group `subgroup0`.

```
[ceph: root@host02 /]# ceph fs snap-schedule add / 2M --subvol sub0 --group subgroup0
```

Adding retention policy for snapshot schedules of a CephFS volume path

PREREQUISITE

Before adding a retention policy, you must have a snapshot schedule in place.
To define how many snapshots to retain in the volume path at any time, you must add a retention policy after you have created a snapshot schedule.

You can create retention policies for directories within a subvolume, subvolume in a default group and to the subvolume in a non-default group.

- Add a retention policy to a snapshot schedule created for a directory in a subvolume.

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

For example,

```
[ceph: root@host02 /]# ceph fs snap-schedule retention add /volumes/subvolgroup_1/
subvol_1 h 14 (1)
[ceph: root@host02 /]# ceph fs snap-schedule retention add /volumes/subvolgroup_1/
subvol_1 d 4 (2)
[ceph: root@host02 /]# ceph fs snap-schedule retention add /volumes/subvolgroup_1/
subvol_1 14h4w (3)
```

(1) This example keeps 14 snapshots at least one hour apart.
(2) This example keeps 4 snapshots at least one day apart.
(3) This example keeps 14 hourly, and 4 weekly snapshots.

- Add a retention policy to a snapshot schedule created for a subvolume in the default group.

```
ceph fs snap-schedule retention add / [COUNT_TIME_PERIOD_PAIR] TIME_PERIOD COUNT --
fs CEPH_FILE_SYSTEM_NAME --subvol SUBVOLUME_NAME
```

> **Important:** The `path ( / )` must be defined and cannot be left empty. There is no dependency on `path` string value, you can define it with one of the following characters:
> / - /..

For example,

```
[ceph: root@host02 /]# ceph fs snap-schedule retention add / 5h --fs cephfs --subvol
sv_sched
Retention added to path /volumes/sv_sched/e704342a-ff07-4763-bb0b-a46d9dda6f27/..
```

- Add a retention policy to the snapshot schedule created for a subvolume group in the non-default group.

```
ceph fs snap-schedule retention add / [COUNT_TIME_PERIOD_PAIR] TIME_PERIOD COUNT --
fs CEPH_FILE_SYSTEM_NAME --subvol SUBVOLUME_NAME --group NON_DEFAULT_SUBVOLGROUP_NAME
```

> **Important:** The `path ( / )` must be defined and cannot be left empty. There is no dependency on `path` string value, you can define it with one of the following characters:
> / - /..

For example,

```
[ceph: root@host02 /]# ceph fs snap-schedule retention add / 5h --fs cephfs --subvol
sv_sched --group subvolgroup_cg
Retention added to path /volumes/subvolgroup_cg/sv_sched/e704342a-ff07-4763-bb0b-
a54j0dda7f16/..
```

Listing CephFS snapshot schedules

PREREQUISITE

Before listing CephFS snapshots schedules, be sure that you have both read and write capability on the Ceph Manager nodes.
By listing and adhering to snapshot schedules, you can ensure robust data protection and efficient management.

PREREQUISITE

Before listing CephFS snapshots schedules, be sure that you have both read and write capability on the Ceph Manager nodes.

- List the snapshot schedules.

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

The following example lists all schedules in the directory tree.

```
[ceph: root@host02 /]# ceph fs snap-schedule list /volumes/subgroup0/
sub0/4dee0788-32a0-47e6-ac4b-bd8568a6989b/test_dir --recursive=true

/volumes/subgroup0/sub0/4dee0788-32a0-47e6-ac4b-bd8568a6989b/test_dir 1h
```

Checking status of CephFS snapshot schedules

PREREQUISITE

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

- Read and write capability on the Ceph Manager nodes.
- A snapshot schedule.

You can check the status of snapshot schedules using the command in the procedure for snapshots created in a directory of a subvolume, for a subvolume in the default subvolume group and for a subvolume created in a non-default group.

PREREQUISITE

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

- Read and write capability on the Ceph Manager nodes.
- A snapshot schedule.
- Check the status of a snapshot schedule created for a directory in a subvolume:

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

The following example displays the status of the snapshot schedule for the `/volumes/subvolgroup_1/subvol_1` path in JSON format. If not specified, the default format is plain text.

```
/volumes/subgroup0/sub0/4dee0788-32a0-47e6-ac4b-bd8568a6989b/test_dir

{"fs": "cephfs01", "subvol": null, "group": null, "path": "/volumes/subgroup0/
sub0/4dee0788-32a0-47e6-ac4b-bd8568a6989b/test_dir", "rel_path": "/volumes/subgroup0/
sub0/4dee0788-32a0-47e6-ac4b-bd8568a6989b/test_dir", "schedule": "1h", "retention": {},
"start": "2025-01-09T00:00:00", "created": "2025-01-09T10:00:29", "first": null,
"last": null, "last_pruned": null, "created_count": 0, "pruned_count": 0, "active":
true}
```

- Check the status of the snapshot schedule created for a subvolume in the default group:

```
ceph fs snap-schedule status --fs CEPH_FILE_SYSTEM_NAME --subvol SUBVOLUME_NAME
```

> **Important:** The `path ( / )` must be defined and cannot be left empty. There is no dependency on `path` string value, you can define it with one of the following characters:
> / - /..

For example,

```
[ceph: root@host02 /]# ceph fs snap-schedule status --fs cephfs --subvol sv_sched
{"fs": "cephfs", "subvol": "sv_sched", "group": "subvolgroup_cg", "path": "/volumes/
subvolgroup_cg/sv_sched/e704342a-ff07-4763-bb0b-a46d9dda6f27/..", "rel_path": "/
volumes/subvolgroup_cg/sv_sched/e704342a-ff07-4763-bb0b-a46d9dda6f27/..", "schedule":
"1h", "retention": {"h": 5}, "start": "2024-05-21T00:00:00", "created":
"2024-05-21T09:18:58", "first": null, "last": null, "last_pruned": null,
"created_count": 0, "pruned_count": 0, "active": true}proc_cephfs_adding-snapshot-
schedule-of-a-file-system-subvolumeproc_cephfs_adding-snapshot-schedule-of-a-file-
system-subvolumeproc_cephfs_adding-snapshot-schedule-of-a-file-system-subvolume
```

- Check the status of the snapshot schedule created for a subvolume in the non-default group:

```
ceph fs snap-schedule status --fs CEPH_FILE_SYSTEM_NAME --subvol SUBVOLUME_NAME --
group NON_DEFAULT_SUBVOLGROUP_NAME
```

> **Important:** The `path ( / )` must be defined and cannot be left empty. There is no dependency on `path` string value, you can define it with one of the following characters:
> / - /..

For example,

```
[ceph: root@host02 /]# ceph fs snap-schedule status --fs cephfs --subvol sv_sched --
group subvolgroup_cg
{"fs": "cephfs", "subvol": "sv_sched", "group": "subvolgroup_cg", "path": "/volumes/
subvolgroup_cg/sv_sched/e564329a-kj87-4763-gh0y-b56c8sev7t23/..", "rel_path": "/
volumes/subvolgroup_cg/sv_sched/e704342a-ff07-4763-bb0b-a46d9dda6f27/..", "schedule":
"1h", "retention": {"h": 5}, "start": "2024-05-21T00:00:00", "created":
"2024-05-21T09:18:58", "first": null, "last": null, "last_pruned": null,
"created_count": 0, "pruned_count": 0, "active": true}
```

Activating snapshot schedule for a CephFS

Learn how to manually set the snapshot schedule to active for a Ceph File System (CephFS).

PREREQUISITE

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

- A running, and healthy Red Hat Ceph Storage cluster with Ceph File System (CephFS) deployed.
- A minimum of read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.

- Activate the snapshot schedule.

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

  In the following example, all schedules for the CephFS `/cephfs` path are activated.

```
[ceph: root@host01 ~]# ceph fs snap-schedule activate /cephfs
```

AFTER COMPLETING THE TASK

Verify that the snapshot schedule is activated, by using the **`snap-schedule status`** command. For more information, see "Checking status of CephFS snapshot schedules" on page 99.

Activating snapshot schedule for a CephFS subvolume

Learn how to manually set the snapshot schedule to active for a Ceph File System (CephFS) sub volume.

PREREQUISITE

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

- A working Red Hat Ceph Storage cluster with a Ceph File System (CephFS) deployed.
- A minimum of read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- Activate the snapshot schedule created for a directory in a subvolume group.

```
ceph fs snap-schedule activate SUB_VOLUME_PATH REPEAT_INTERVAL
```

  In the following example, all schedules for the CephFS `/volumes/_nogroup/subvol_1/85a615da-e8fa-46c1-afc3-0eb8ae64a954/..` path are activated.

```
[ceph: root@host01 ~]# ceph fs snap-schedule activate /volumes/_nogroup/
subvol_1/85a615da-e8fa-46c1-afc3-0eb8ae64a954/..
```

- Activate the snapshot schedule created for a subvolume in the default group.

```
ceph fs snap-schedule activate /.. REPEAT_INTERVAL --fs CEPH_FILE_SYSTEM_NAME --
subvol SUBVOLUME_NAME
```

> **Important:** The `path ( / )` must be defined and cannot be left empty. There is no dependency on `path` string value, you can define it with one of the following characters:
> / - /..

  For example,

```
[ceph: root@host02 /]# ceph fs snap-schedule activate / --fs cephfs --subvol sv_sched
Schedule activated for path /volumes/subvolgroup_cg/sv_sched/e704342a-ff07-4763-bb0b-
a46d9dda6f27/..
```

- Activate the snapshot schedule created for a subvolume in the non-default group:

```
ceph fs snap-schedule activate /.. [_REPEAT_INTERVAL_] --fs CEPH_FILE_SYSTEM_NAME --
subvol SUBVOLUME_NAME --group NON_DEFAULT_GROUP_NAME
```

> **Important:** The `path ( / )` must be defined and cannot be left empty. There is no dependency on `path` string value, you can define it with one of the following characters:
> / - /..

For example,

```
[ceph: root@host02 /]# ceph fs snap-schedule activate / --fs cephfs --subvol sv_sched
--group subvolgroup_cg
Schedule activated for path /volumes/subvolgroup_cg/sv_sched/e704342a-ff07-4763-bb0b-
a46d9dda6f27/..
```

AFTER COMPLETING THE TASK

Verify that the snapshot schedule is activated, by using the **snap-schedule status** command. For more information, see "Checking status of CephFS snapshot schedules" on page 99.

Deactivating snapshot schedule for a CephFS

Learn how to manually set the snapshot schedule to inactive for a Ceph File System (CephFS). This action excludes the snapshot from scheduling until it is activated again.

PREREQUISITE

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

- A working Red Hat Ceph Storage cluster with a Ceph File System (CephFS) deployed.
- Minimum of read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- A snapshot schedule that is created and in active state.
- Deactivate a snapshot schedule for a CephFS path.

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

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

```
[ceph: root@host02 ~]# ceph fs snap-schedule deactivate /cephfs 1d
```

AFTER COMPLETING THE TASK

Verify that the snapshot schedule is deactivated, by using the **snap-schedule status** command. For more information, see "Checking status of CephFS snapshot schedules" on page 99.

Deactivating snapshot schedule for a CephFS subvolume

Learn how to manually set the snapshot schedule to inactive for a Ceph File System (CephFS) subvolume. This action will exclude the snapshot from scheduling until it is activated again.

PREREQUISITE

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

- A working Red Hat Ceph Storage cluster with a Ceph File System (CephFS) deployed.
- Minimum of read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- A snapshot schedule that is created and in active state.
- Deactivate a snapshot schedule for a CephFS directory in the subvolume.

```
ceph fs snap-schedule deactivate SUBVOLUME_DIR_PATH REPEAT_INTERVAL
```

> **Important:** The `path ( / )` must be defined and cannot be left empty. There is no dependency on `path` string value, you can define it with one of the following characters:
> / - /..

The following example deactivates the daily snapshots for the `/volumes/_nogroup/subvol_1/85a615da-e8fa-46c1-afc3-0eb8ae64a954/..` path, pausing any further snapshot creation.

```
[ceph: root@host02 ~]# ceph fs snap-schedule deactivate /volumes/_nogroup/
subvol_1/85a615da-e8fa-46c1-afc3-0eb8ae64a954/.. 1d
```

- Deactivate the snapshot schedule created for a subvolume in a default group.

```
ceph fs snap-schedule deactivate / REPEAT_INTERVAL --fs CEPH_FILE_SYSTEM_NAME --
subvol SUBVOLUME_NAME
```

> **Important:** The `path ( / )` must be defined and cannot be left empty. There is no dependency on `path` string value, you can define it with one of the following characters:
> / - /..

For example,

```
[ceph: root@host02 ~]# ceph fs snap-schedule deactivate / --fs cephfs --subvol
sv_sched
Schedule deactivated for path /volumes/subvolgroup_cg/sv_sched/e704342a-ff07-4763-bb0b-
a46d9dda6f27/..
```

- Deactivate the snapshot schedule created for a subvolume in the non-default group.

```
ceph fs snap-schedule deactivate / REPEAT_INTERVAL --fs CEPH_FILE_SYSTEM_NAME --
subvol SUBVOLUME_NAME --group NON-DEFAULT_GROUP_NAME
```

> **Important:** The `path ( / )` must be defined and cannot be left empty. There is no dependency on `path` string value, you can define it with one of the following characters:
> / - /..

For example,

```
[ceph: root@host02 ~]# ceph fs snap-schedule deactivate / --fs cephfs --subvol sv_sched
--group subvolgroup_cg
Schedule deactivated for path /volumes/subvolgroup_cg/sv_sched/e704342a-ff07-4763-bb0b-
a46d9dda6f27/..
```

AFTER COMPLETING THE TASK

Verify that the snapshot schedule is deactivated, by using the **`snap-schedule status`** command. For more information, see “Checking status of CephFS snapshot schedules” on page 99.

Removing a snapshot schedule for a CephFS

Learn how to remove snapshot schedule of a Ceph File System (CephFS).

PREREQUISITE

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

- A working Red Hat Ceph Storage cluster with a Ceph File System (CephFS) deployed.
- Minimum of read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- A snapshot schedule that is created.

1. Remove a specific snapshot schedule.

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

   This example removes the specific snapshot schedule for the /cephfs volume that is snapshotting every four hours and that started on 16 May 2023 2:00 PM.

   ```
   [ceph: root@host02 ~]# ceph fs snap-schedule remove /cephfs 4h 2023-05-16T14:00:00
   ```

2. Remove all snapshot schedules for a specific CephFS volume.

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

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

   ```
   [ceph: root@host02 ~]# ceph fs snap-schedule remove /cephfs
   ```

AFTER COMPLETING THE TASK

Validate the snapshot schedule, by using the **snap-schedule list** command. For more information, see "Listing CephFS snapshot schedules" on page 99.

Removing a snapshot schedule for a CephFS subvolume

Learn how to remove snapshot schedule of a Ceph File System (CephFS) subvolume.

PREREQUISITE

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

- A working Red Hat Ceph Storage cluster with a Ceph File System (CephFS) deployed.
- Minimum of read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- A snapshot schedule.

  Remove specific snapshots schedules in any of the following ways.

- Remove a specific snapshot schedule created for a directory in the subvolume.

  ```
  ceph fs snap-schedule remove SUBVOLUME_DIR_PATH REPEAT_INTERVAL START_TIME
  ```

  The following example removes the specific snapshot schedule for the /volumes/_nogroup/subvol_1/85a615da-e8fa-46c1-afc3-0eb8ae64a954/.. volume that is taking snapshots every four hours and started on 16 May 2022 2:00 PM.

  ```
  [ceph: root@host02 ~]# ceph fs snap-schedule remove /volumes/_nogroup/
  subvol_1/85a615da-e8fa-46c1-afc3-0eb8ae64a954/.. 4h 2022-05-16T14:00:00
  ```

- Remove a specific snapshot schedule created for a subvolume in a default group.

  ```
  ceph fs snap-schedule remove / --fs CEPH_FILESYSTEM_NAME --subvol SUBVOLUME_NAME
  ```

  > **Important:** The path (/) must be defined and cannot be left empty. There is no dependency on path string value, you can define it with one of the following characters:
  > / - /..

  For example,

  ```
  [ceph: root@host02 ~]# ceph fs snap-schedule remove / --fs cephfs --subvol sv_sched
  Schedule removed for path /volumes/subvolgroup_cg/sv_sched/e704342a-ff07-4763-bb0b-
  a46d9dda6f27/..
  ```

- Remove a specific snapshot schedule created for a subvolume in a non-default group.

```
ceph fs snap-schedule remove / --fs CEPH_FILESYSTEM_NAME --subvol SUBVOLUME_NAME --
group NON-DEFAULT_GROUP_NAME
```

> **Important:** The `path ( / )` must be defined and cannot be left empty. There is no dependency on `path` string value, you can define it with one of the following characters:
> / - /..

For example,

```
[ceph: root@host02 ~]# ceph fs snap-schedule remove / --fs cephfs --subvol sv_sched --
group subvolgroup_cg
Schedule removed for path /volumes/subvolgroup_cg/sv_sched/e704342a-ff07-4763-bb0b-
a46d9dda6f27/..
```

AFTER COMPLETING THE TASK

Validate the snapshot schedule, by using the **`snap-schedule list`** command. For more information, see “Listing CephFS snapshot schedules” on page 99.

Removing snapshot schedule retention policy for a CephFS

Learn how to remove the retention policy of the scheduled snapshots for a Ceph File System (CephFS).

PREREQUISITE

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

- A working Red Hat Ceph Storage cluster with a Ceph File System (CephFS) deployed.
- Minimum of read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- A snapshot schedule that is created for a CephFS volume path.
- Remove a retention policy on a CephFS path.

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

For example,

```
[ceph: root@host02 ~]# ceph fs snap-schedule retention remove /cephfs h 4 (1)
[ceph: root@host02 ~]# ceph fs snap-schedule retention remove /cephfs 14d4w (2)
```

(1) This example removes 4 hourly snapshots.
(2) This example removes 14 daily and 4 weekly snapshots.

AFTER COMPLETING THE TASK

Verify the retention policy, by using the **`snap-schedule status`** command. For more information, see “Checking status of CephFS snapshot schedules” on page 99.

Removing snapshot schedule retention policy for a CephFS subvolume

Learn how to remove the retention policy of the scheduled snapshots for a Ceph File System (CephFS) subvolume.

PREREQUISITE

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

- A working Red Hat Ceph Storage cluster with a Ceph File System (CephFS) deployed.

- Minimum of read access on the Ceph Monitor.
- Read and write capability on the Ceph Manager nodes.
- A snapshot schedule created either for a CephFS subvolume directory or a subvolume in default group or a subvolume in non-default group.

Remove retention policy in any of the following ways.

- Remove a retention policy created on snapshot schedule for a directory in the subvolume.

```
ceph fs snap-schedule retention
remove SUBVOLUME_DIR_PATH TIME_PERIOD_PAIR TIME_PERIOD COUNT
```

For example,

```
[ceph: root@host02 ~]# ceph fs snap-schedule retention remove /volumes/_nogroup/
subvol_1/85a615da-e8fa-46c1-afc3-0eb8ae64a954/.. h 4 (1)
[ceph: root@host02 ~]# ceph fs snap-schedule retention remove /volumes/_nogroup/
subvol_1/85a615da-e8fa-46c1-afc3-0eb8ae64a954/.. 14d4w (2)
```

(1) This example removes 4 hourly snapshots.
(2) This example removes 14 daily and 4 weekly snapshots.

- Remove a retention policy created on snapshot schedule for a subvolume in a default group.

```
ceph fs snap-schedule retention remove / TIME_PERIOD_PAIR TIME_PERIOD COUNT --
fs CEPH_FILESYSTEM_NAME --subvol SUBVOLUME_NAME
```

> **Important:** The `path ( / )` must be defined and cannot be left empty. There is no dependency on `path` string value, you can define it with one of the following characters:
> / - /..

For example,

```
[ceph: root@host02 ~]# ceph fs snap-schedule retention remove / 5h --fs cephfs --subvol
sv_sched --group subvolgroup_cg
Retention removed from path /volumes/subvolgroup_cg/sv_sched/e704342a-ff07-4763-bb0b-
a46d9dda6f27/..
```

- Remove a retention policy created on snapshot schedule for a subvolume in a non-default group.

```
ceph fs snap-schedule retention remove / TIME_PERIOD_PAIR TIME_PERIOD COUNT --
fs CEPH_FILESYSTEM_NAME --subvol SUBVOLUME_NAME --group NON-DEFAULT_GROUP_NAME
```

> **Important:** The `path ( / )` must be defined and cannot be left empty. There is no dependency on `path` string value, you can define it with one of the following characters:
> / - /..

For example,

```
[ceph: root@host02 ~]# ceph fs snap-schedule retention remove / 5h --fs cephfs --subvol
sv_sched --group subvolgroup_cg
Retention removed from path /volumes/subvolgroup_cg/sv_sched/e704342a-ff07-4763-bb0b-
a46d9dda6f27/..
```

AFTER COMPLETING THE TASK

Verify the retention policy, by using the **`snap-schedule status`** command. For more information, see “Checking status of CephFS snapshot schedules” on page 99.

CephFS snapshot mirroring (Technology Preview)

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

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

Note: CephFS mirroring is currently available as a Technology Preview feature. In production environments, use other synchronization methods, such as `rsync`. You can combine these methods with periodic CephFS snapshots to help ensure data consistency. For more information, see Recovering a Ceph Monitor when using BlueStore.

To use CephFS snapshot mirroring, both the source and the target storage clusters must be running Red Hat Ceph Storage 7.0 or later.

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

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

Important: Synchronizing hard links is not supported. Hard linked files get synchronized as regular files.

The CephFS snapshot mirroring includes features, for example snapshot incarnation or high availability. These can be managed through Ceph Manager `mirroring` module, which is the recommended control interface.

Ceph Manager Module and interfaces

The Ceph Manager `mirroring` module is disabled by default. It provides interfaces for managing mirroring of directory snapshots. Ceph Manager interfaces are mostly wrappers around monitor commands for managing CephFS mirroring. They are the recommended control interface.

The Ceph Manager `mirroring` module is implemented as a Ceph Manager plugin. It is responsible for assigning directories to the `cephfs-mirror` daemons for synchronization.

The Ceph Manager `mirroring` module also provides a family of commands to control mirroring of directory snapshots. The `mirroring` module does not manage the `cephfs-mirror` daemons. The stopping, starting, restarting, and enabling of the `cephfs-mirror` daemons is controlled by `systemctl`, but managed by `cephadm`.

Note: Mirroring module commands use the `fs snapshot mirror` prefix as compared to the monitor commands with the `fs mirror` prefix. Assure that you are using the module command prefix to control the mirroring of directory snapshots.

Snapshot incarnation

A snapshot might be deleted and created again with the same name and different content. The user could synchronize an "old" snapshot earlier and create the snapshot again when the mirroring was disabled. Using snapshot names to infer the point-of-continuation results in the “new” snapshot, an incarnation, never getting picked up for synchronization.

Snapshots on the secondary file system store the `snap-id` of the snapshot it was synchronized from. This metadata is stored in the `SnapInfo` structure on the Ceph Metadata Server.

High availability

You can deploy multiple `cephfs-mirror` daemons on two or more nodes to achieve concurrency in synchronization of directory snapshots. When `cephfs-mirror` daemons are deployed or terminated, the Ceph Manager `mirroring` module discovers the modified set of `cephfs-mirror` daemons and rebalances the directory assignment amongst the new set thus providing high availability.

`cephfs-mirror` daemons share the synchronization load by using a simple M/N policy, where M is the number of directories and N is the number of `cephfs-mirror` daemons.

Readdition of Ceph File System mirror peers

When readding or reassigning a peer to a CephFS in another cluster, make sure that all mirror daemons stopped synchronization to the peer. You can verify readdition of CephFS mirror peers with the **`fs mirror status`** command. The Peer UUID does not show up in the command output.

Purge synchronized directories from the peer before readding it to another CephFS, especially those directories that might exist in the new primary file system. Purging synchronized directories is not required if you are readding a peer to the same primary file system it was earlier synchronized from.

For more information about the **`fs mirror status`** command, see "Viewing the mirror status for a Ceph File System" on page 111.

Configuring a snapshot mirror for a Ceph File System

Learn how to configure a Ceph File System (CephFS) for mirroring to replicate snapshots to another CephFS on a remote Red Hat Ceph Storage cluster.

PREREQUISITE

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

- The source and the target storage clusters must be healthy and running the latest Red Hat Ceph Storage version.
- Root-level access to a Ceph Monitor node in the source and the target storage clusters.
- At least one CephFS deployed on your storage cluster.

The time taken for synchronizing to a remote storage cluster depends on the file size and the total number of files in the mirroring path.

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

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

2. On the source storage cluster, deploy the CephFS mirroring daemon.
   This command creates a Ceph user called `cephfs-mirror` and deploys the `cephfs-mirror` daemon on the given node.

   ```
   ceph orch apply cephfs-mirror ["NODE_NAME"]
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph orch apply cephfs-mirror "node1.example.com"
   Scheduled cephfs-mirror update...
   ```

   a. Deploy multiple CephFS mirroring daemons and achieve high availability.

      ```
      ceph orch apply cephfs-mirror --placement="PLACEMENT_SPECIFICATION"
      ```

      The following example deploys three `cephfs-mirror` daemons on different hosts.

      > **Warning:** Do not separate the hosts with commas. Using commas results in the following error:
      >
      > `Error EINVAL: name component must include only a-z, 0-9, and -`

```
[ceph: root@host01 /]# ceph orch apply cephfs-mirror --placement="3 host1 host2
host3"
Scheduled cephfs-mirror update...
```

3. On the target storage cluster, create a user for each CephFS peer.

```
ceph fs authorize FILE_SYSTEM_NAME CLIENT_NAME / rwps
```

For example,

```
[ceph: root@host01 /]# ceph fs authorize cephfs client.mirror_remote / rwps
[client.mirror_remote]
        key = AQCjZ5Jg739AAxAAxduIKoTZbiFJ0lgose8luQ==
```

4. On the source storage cluster, enable the CephFS mirroring module.

```
ceph mgr module enable mirroring
```

For example,

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

5. On the source storage cluster, enable mirroring on a Ceph File System.

```
ceph fs snapshot mirror enable FILE_SYSTEM_NAME
```

For example,

```
[ceph: root@host01 /]# ceph fs snapshot mirror enable cephfs
```

a. Disable snapshot mirroring.

> **Warning:** Disabling snapshot mirroring on a file system removes the configured peers. You have to import the peers again by bootstrapping them.

```
ceph fs snapshot mirror disable FILE_SYSTEM_NAME
```

For example,

```
[ceph: root@host01 /]# ceph fs snapshot mirror disable cephfs
```

6. Prepare the target peer storage cluster.

a. On a target node, enable the `mirroring` Ceph Manager module.

```
ceph mgr module enable mirroring
```

For example,

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

b. On the same target node, create the peer bootstrap.

```
ceph fs snapshot mirror peer_bootstrap
create FILE_SYSTEM_NAME CLIENT_NAME SITE_NAME
```

The *SITE_NAME* is a user-defined string to identify the target storage cluster.

For example,

```
[ceph: root@host01 /]# ceph fs snapshot mirror peer_bootstrap create cephfs
client.mirror_remote remote-site
{"token":
"eyJmc2lkIjogIjBkZjE3MjE3LWRmY2QtNDAzMC05MDc5LTM2Nzk4NTVkNDJlZiIsICJmaWxlc3lzdGVt
IjogImJhY2t1cF9mcyIsICJ1c2VyIjogImNsaWVudC5taXJyb3JfcGVlcl9ib290c3RyYXAiLCAic2l0Z
V9uYW1lIjogInNpdGUtcmVtb3RlIiwgImtleSI6ICJBUUFhcDBCZ0xtRmpOeEFBVnNyZXozai9YYUV0T2
UrbUJEZlJDZz09IiwgIm1vbl9ob3N0IjogIlt2MjoxOTIuMTY4LjAuNTo0MDkxOCx2MToxOTIuMTY4LjA
uNTo0MDkxOV0ifQ==
"}
```

c. Copy the token string between the double quotes, from step “6.b” on page 109 for use in step “7” on page 110.

7. On the source storage cluster, import the bootstrap token from the target storage cluster.

```
ceph fs snapshot mirror peer_bootstrap import FILE_SYSTEM_NAME TOKEN
```

For example,

```
[ceph: root@host01 /]# ceph fs snapshot mirror peer_bootstrap import cephfs
eyJmc2lkIjogIjBkZjE3MjE3LWRmY2QtNDAzMC05MDc5LTM2Nzk4NTVkNDJlZiIsICJmaWxlc3lzdGVtIjogIm
JhY2t1cF9mcyIsICJ1c2VyIjogImNsaWVudC5taXJyb3JfcGVlcl9ib290c3RyYXAiLCAic2l0ZV9uYW1lIjog
InNpdGUtcmVtb3RlIiwgImtleSI6ICJBUUFhcDBCZ0xtRmpOeEFBVnNyZXozai9YYUV0T2UrbUJEZlJDZz09Ii
wgIm1vbl9ob3N0IjogIlt2MjoxOTIuMTY4LjAuNTo0MDkxOCx2MToxOTIuMTY4LjAuNTo0MDkxOV0ifQ==
```

8. On the source storage cluster, list the CephFS mirror peers.

```
ceph fs snapshot mirror peer_list FILE_SYSTEM_NAME
```

For example,

```
[ceph: root@host01 /]# ceph fs snapshot mirror peer_list cephfs
{"e5ecb883-097d-492d-b026-a585d1d7da79": {"client_name": "client.mirror_remote",
"site_name": "remote-site", "fs_name": "cephfs", "mon_host":
"[v2:10.0.211.54:3300/0,v1:10.0.211.54:6789/0]
[v2:10.0.210.56:3300/0,v1:10.0.210.56:6789/0]
[v2:10.0.210.65:3300/0,v1:10.0.210.65:6789/0]"}}
```

a. Remove a snapshot peer.

```
ceph fs snapshot mirror peer_remove FILE_SYSTEM_NAME PEER_UUID
```

> **Note:** For more information about how to find the peer UUID value, see step “4” on page 112 of “Viewing the mirror status for a Ceph File System” on page 111.

For example,

```
[ceph: root@host01 /]# ceph fs snapshot mirror peer_remove cephfs
e5ecb883-097d-492d-b026-a585d1d7da79
```

9. On the source storage cluster, configure a directory for snapshot mirroring.

```
ceph fs snapshot mirror add FILE_SYSTEM_NAME PATH
```

> **Important:** Only absolute paths inside the Ceph File System are valid.

> **Note:** The Ceph Manager `mirroring` module normalizes the path. For example, the `/d1/d2/../dN` directories are equivalent to `/d1/d2`. Once a directory has been added for mirroring, its ancestor directories and subdirectories are prevented from being added for mirroring.

For example,

```
[ceph: root@host01 /]# ceph fs snapshot mirror add cephfs /volumes/_nogroup/subvol_1
```

a. Stop snapshot mirroring for a directory.

```
ceph fs snapshot mirror remove FILE_SYSTEM_NAME PATH
```

For example,

```
[ceph: root@host01 /]# ceph fs snapshot mirror remove cephfs /home/user1
```

Viewing the mirror status for a Ceph File System

The Ceph File System (CephFS) mirror daemon (`cephfs-mirror`) gets asynchronous notifications about changes in the CephFS mirroring status, along with peer updates. The CephFS mirroring module provides a mirror daemon status interface to check mirror daemon status.

PREREQUISITE

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

- A running Red Hat Ceph Storage cluster.
- A minimum of one deployment of a Ceph File System with mirroring enabled.
- Root-level access to the node running the CephFS mirroring daemon.

For more detailed information, you can query the `cephfs-mirror` admin socket with commands to retrieve the mirror status and peer status.

1. Log in to the `cephadm` shell.
   For example,

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

2. Check the `cephfs-mirror` daemon status.

   Note: For more detailed information, use the admin socket interface as detailed in step “5” on page 112.

   ```
   ceph fs snapshot mirror daemon status
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph fs snapshot mirror daemon status
   [
     {
       "daemon_id": 15594,
       "filesystems": [
         {
           "filesystem_id": 1,
           "name": "cephfs",
           "directory_count": 1,
           "peers": [
             {
               "uuid": "e5ecb883-097d-492d-b026-a585d1d7da79",
               "remote": {
                 "client_name": "client.mirror_remote",
                 "cluster_name": "remote-site",
                 "fs_name": "cephfs"
               },
               "stats": {
                 "failure_count": 1,
                 "recovery_count": 0
               }
             }
           ]
         }
       ]
     }
   ]
   ```

3. Find the Ceph File System ID on the node running the CephFS mirroring daemon.

```
ceph --admin-daemon PATH_TO_THE_ASOK_FILE help
```

> **Note:** When mirroring is disabled, the respective **`fs mirror status`** command for the file system does not show up in the **`help`** command.

In the following example, the Ceph File System ID is `cephfs@11`.

```
[ceph: root@host01 /]# ceph --admin-daemon /var/run/ceph/1011435c-9e30-4db6-
b720-5bf482006e0e/ceph-client.cephfs-mirror.node1.bndvox.asok help
{
    ...
    "fs mirror peer status cephfs@11 1011435c-9e30-4db6-b720-5bf482006e0e": "get peer
mirror status",
    "fs mirror status cephfs@11": "get filesystem mirror status",
    ...
}
```

4. View the mirror status.

```
ceph --admin-daemon PATH_TO_THE_ASOK_FILE fs mirror
status FILE_SYSTEM_NAME @_FILE_SYSTEM_ID
```

For example,

```
[ceph: root@host01 /]# ceph --admin-daemon /var/run/ceph/1011435c-9e30-4db6-
b720-5bf482006e0e/ceph-client.cephfs-mirror.node1.bndvox.asok fs mirror status
cephfs@11
{
  "rados_inst": "192.168.0.5:0/1476644347",
  "peers": {
      "1011435c-9e30-4db6-b720-5bf482006e0e": { (1)
          "remote": {
              "client_name": "client.mirror_remote",
              "cluster_name": "remote-site",
              "fs_name": "cephfs"
          }
      }
  },
  "snap_dirs": {
      "dir_count": 1
  }
}
```

(1) This is the unique peer UUID.

5. View the peer status.

```
ceph --admin-daemon PATH_TO_ADMIN_SOCKET fs mirror
status FILE_SYSTEM_NAME@FILE_SYSTEM_ID PEER_UUID
```

For example,

```
[ceph: root@host01 /]# ceph --admin-daemon /var/run/ceph/cephfs-mirror.asok fs mirror
peer status cephfs@11 1011435c-9e30-4db6-b720-5bf482006e0e
{
  "/home/user1": {
      "state": "idle", (1)
      "last_synced_snap": {
          "id": 120,
          "name": "snap1",
          "sync_duration": 0.079997898999999997,
          "sync_time_stamp": "274900.558797s"
      },
      "snaps_synced": 2, (2)
      "snaps_deleted": 0, (3)
      "snaps_renamed": 0
  }
}
```

The `state` can be one of these three values:

(1) `idle` means that the directory is currently not being synchronized.

(2) `syncing` means that the directory is currently being synchronized.

(3) `failed` means that the directory has reached the upper limit of consecutive failures.
The default number of consecutive failures is 10, and the default retry interval is 60 seconds.

Displaying the directory to which `cephfs-mirror` daemon is mapped

To display the directory to which `cephfs-mirror` daemon is mapped:

```
ceph fs snapshot mirror dirmap FILE_SYSTEM_NAME PATH
```

Example 1 (mirror daemons are running)

```
[ceph: root@host01 /]# ceph fs snapshot mirror dirmap cephfs /volumes/_nogroup/subvol_1
{
  "instance_id": "25184", (1)
  "last_shuffled": 1661162007.012663,
  "state": "mapped"
}
```

(1) `instance_id` is the RADOS instance-ID that is associated with a `cephfs-mirror` daemon.

Example 2 (no mirror daemons are running)

```
[ceph: root@host01 /]# ceph fs snapshot mirror dirmap cephfs /volumes/_nogroup/subvol_1
{
  "reason": "no mirror daemons running",
  "state": "stalled" (1)
}
```

(1) `stalled` state means the CephFS mirroring is stalled.

Viewing metrics for Ceph File System snapshot mirroring

Viewing these metrics helps in monitoring the performance and the sync progress.

PREREQUISITE

- A running Red Hat Ceph Storage cluster.
- A minimum of one deployment of a Ceph File System snapshot mirroring enabled.
- Root-level access to the node running the Ceph File System mirroring daemon.

Check Ceph File System snapshot mirror health and volume metrics by using the **`counter dump`**.

1. Get the name of the `asok` file. The `asok` file is available where the mirroring daemon is running and is located at `/var/run/ceph/` within the `cephadm` shell.
2. Check the mirroring metrics and synchronization status by running the following command on the node running the CephFS mirroring daemon.
   Syntax:

   ```
   [ceph: root@mirror-host01 /]# ceph --admin-daemon ASOK_FILE_NAME counter dump
   ```

   Example:

```
[ceph: root@mirror-host01 /]# ceph --admin-daemon ceph-client.cephfs-mirror.ceph1-hk-
n-0mfqao-node7.pnbrlu.2.93909288073464.asok counter dump

[
  {
    "key": "cephfs_mirror",
    "value": [
      {
        "labels": {},
        "counters": {
          "mirrored_filesystems": 1,
          "mirror_enable_failures": 0
        }
      }
    ]
  },
  {
    "key": "cephfs_mirror_mirrored_filesystems",
    "value": [
      {
        "labels": {
          "filesystem": "cephfs"
        },
        "counters": {
          "mirroring_peers": 1,
          "directory_count": 1
        }
      }
    ]
  },
  {
    "key": "cephfs_mirror_peers",
    "value": [
      {
        "labels": {
          "peer_cluster_filesystem": "cephfs",
          "peer_cluster_name": "remote_site",
          "source_filesystem": "cephfs",
          "source_fscid": "1"
        },
        "counters": {
          "snaps_synced": 1,
          "snaps_deleted": 0,
          "snaps_renamed": 0,
          "sync_failures": 0,
          "avg_sync_time": {
            "avgcount": 1,
            "sum": 4.216959457,
            "avgtime": 4.216959457
          },
          "sync_bytes": 132
        }
      }
    ]
  }
]
```

Metrics description:

Labeled `Perf Counters` generate metrics which can be consumed by the OCP/ODF dashboard to provide monitoring of geo replication in the OCP and ACM dashboard and elsewhere.

This would generate the progress of `cephfs_mirror` syncing and provide monitoring capability. The exported metrics enable monitoring based on the following alerts.

mirroring_peers

The number of peers involved in mirroring.

directory_count

The total number of directories being synchronized.

mirrored_filesystems

The total number of file systems which are mirrored.

mirror_enable_failures

Enable mirroring failures.

snaps_synced

The total number of snapshots successfully synchronized.

sync_bytes

The total bytes being synchronized

sync_failures

The total number of failed snapshot synchronizations.

snaps_deleted

The total number of snapshots deleted.

snaps_renamed

The total number of snapshots renamed.

avg_synced_time

The average time taken by all snapshot synchronizations.

last_synced_start

The sync start time of the last synced snapshot.

last_synced_end

The sync end time of the last synced snapshot.

last_synced_duration

The time duration of the last synchronization.

last_synced_bytes

The total bytes being synchronized for the last synced snapshot.