Configuring Red Hat Ceph Storage

Use this information to help configure Red Hat Ceph Storage at start and run time.

As a storage administrator, you need to have a basic understanding of how to view the Ceph configuration, and how to set the Ceph configuration options for the Red Hat Ceph Storage cluster. You can view and set the Ceph configuration options at runtime.

As part of the Ceph authentication configuration, consider key rotation for your Ceph and gateway daemons for increased security. Key rotation can be done either through the command-line or the Ceph dashboard. For more information see Creating user capabilities and “Enabling key rotation” on page 0.

Be sure that the Red Hat Ceph Storage software is installed before configuring.

Ceph configuration

Red Hat Ceph Storage clusters typically have a configuration file that is created and defined through a deployment tool. Configuration files can also be created manually.

Deployment tools, such as `cephadm`, typically create an initial Ceph configuration file for you. However, you can manually create one, if you prefer to bootstrap a Red Hat Ceph Storage cluster without using a deployment tool.

Red Hat Ceph Storage clusters have a configuration, which defines:

- Cluster Identity
- Authentication settings
- Ceph daemons
- Network configuration
- Node names and addresses
- Paths to keyrings
- Paths to OSD log files
- Other runtime options

For more information about `cephadm` and the Ceph orchestrator, see Operations.

Configuration database

The Ceph Monitor manages a configuration database of Ceph options that centralize configuration management by storing configuration options for the entire storage cluster. By centralizing the Ceph configuration in a database, this simplifies storage cluster administration.

The priority order that Ceph uses to set options is:

1. Compiled-in default values
2. Ceph cluster configuration database
3. Local `ceph.conf` file
4. Runtime override, using the **`ceph daemon DAEMON-NAME config set`** or **`ceph tell DAEMON-NAME injectargs`** commands

There are still a few Ceph options that can be defined in the local Ceph configuration file, which is `/etc/ceph/ceph.conf` by default.

`cephadm` uses a basic `ceph.conf` file that only contains a minimal set of options for connecting to Ceph Monitors, authenticating, and fetching configuration information. In most cases, `cephadm` uses only the `mon_host` option. To avoid using `ceph.conf` only for the `mon_host` option, use DNS SRV records to perform operations with Monitors.

Important: Use the **`assimilate-conf`** administrative command to move valid options into the configuration database from the `ceph.conf` file. For more information about **`assimilate-conf`**, see "Administrative commands" on page 2.

Ceph allows you to make changes to the configuration of a daemon at runtime. This capability can be useful for increasing or decreasing the logging output, by enabling or disabling debug settings, and can even be used for runtime optimization.

Note: When the same option exists in the configuration database and the Ceph configuration file, the configuration database option has a lower priority than what is set in the Ceph configuration file.

Sections and masks

Just as you can configure Ceph options globally, per daemon type, or by a specific daemon in the Ceph configuration file, you can also configure the Ceph options in the configuration database according to these sections. These sections are described in "Configuration database sections" on page 2.

Configuration database sections

Section	Description
`global`	Affects all daemons and clients.
`mon`	Affects all Ceph Monitors.
`mgr`	Affects all Ceph Managers.
`osd`	Affects all Ceph OSDs.
`mds`	Affects all Ceph Metadata Servers.
`client`	Affects all Ceph Clients, including mounted file systems, block devices, and RADOS Gateways.

Ceph configuration options can have a mask associated with them. These masks can further restrict which daemons or clients the options apply to.

Masks have two forms:

`type:location`

The `type` is a CRUSH property, for example, `rack` or `host`. The `location` is a value for the property type. For example, `host:foo` limits the option only to daemons or clients running on the `foo` host.

For example:

```
ceph config set osd/host:magna045 debug_osd 20
```

`class:device-class`

The `device-class` is the name of the CRUSH device class, such as `hdd` or `ssd`. For example, `class:ssd` limits the option only to Ceph OSDs backed by solid state drives (SSD). This mask has no effect on non-OSD daemons of clients.

For example:

```
ceph config set osd/class:hdd osd_max_backfills 8
```

Administrative commands

The Ceph configuration database can be administered with the subcommand **`ceph config ACTION`**. These are the actions you can do:

`ls`

Lists the available configuration options.

dump

Dumps the entire configuration database of options for the storage cluster.

get WHO

Dumps the configuration for a specific daemon or client. For example, *WHO* can be a daemon, like `mds.a`.

set WHO OPTION VALUE

Sets a configuration option in the Ceph configuration database, where *WHO* is the target daemon, *OPTION* is the option to set, and *VALUE* is the desired value.

show WHO

Shows the reported running configuration for a running daemon. These options might be different from those stored by the Ceph Monitors if there is a local configuration file in use or options have been overridden on the command line or at run time. Also, the source of the option values is reported as part of the output.

assimilate-conf -i INPUT_FILE -o OUTPUT_FILE

Assimilate a configuration file from the *INPUT_FILE* and move any valid options into the Ceph Monitors' configuration database. Any options that are unrecognized, invalid, or cannot be controlled by the Ceph Monitor return in an abbreviated configuration file stored in the *OUTPUT_FILE*. This command can be useful for transitioning from legacy configuration files to a centralized configuration database. Note that when you assimilate a configuration and the Monitors or other daemons have different configuration values set for the same set of options, the end result depends on the order in which the files are assimilated.

help OPTION -f json-pretty

Displays help for a particular *OPTION* using a JSON-formatted output.

For more information, see "Setting a specific configuration at runtime" on page 4.

Using the Ceph metavariables

Metavariables simplify Ceph storage cluster configuration dramatically. When a metavariable is set in a configuration value, Ceph expands the metavariable into a concrete value.

Metavariables are powerful when used within the `[global]`, `[osd]`, `[mon]`, or `[client]` sections of the Ceph configuration file. However, you can also use them with the administration socket. Ceph metavariables are similar to Bash shell expansion.

Ceph supports the following metavariables, as listed in "Supported Ceph metavariables" on page 3:

Supported Ceph metavariables

Metavariable	Description	Example	Default
`$cluster`	Expands to the Ceph storage cluster name. Useful when running multiple Ceph storage clusters on the same hardware.	`/etc/ceph/$cluster.keyring`	`ceph`
`$type`	Expands to one of `osd` or `mon`, depending on the type of instant daemon.	`/var/lib/ceph/$type`	None.
`$id`	Expands to the daemon identifier. For `osd.0`, this would be `0`.	`/var/lib/ceph/$type/$cluster-$id`	None.
`$host`	Expands to the host name of the instant daemon.	N/A	None.
`$name`	Expands to `$type.$id`.	`/var/run/ceph/$cluster-$name.asok`	None.

Viewing the Ceph configuration at runtime

The Ceph configuration files can be viewed at boot time and run time.

Prerequisites

- Root-level access to the Ceph node.
- Access to admin keyring.

Procedure

1. To view a runtime configuration, log in to a Ceph node running the daemon and run the following command.

   Syntax

   ```
   ceph daemon DAEMON_TYPE.ID config show
   ```

   To see the configuration for `osd.0`, you can log into the node containing `osd.0` and run this command:

   Example

   ```
   [root@osd ~]# ceph daemon osd.0 config show
   ```

2. For additional options, specify a daemon and `help`.

   Example

   ```
   [root@osd ~]# ceph daemon osd.0 help
   ```

Viewing a specific configuration at runtime

Configuration settings for Red Hat Ceph Storage can be viewed at runtime from the Ceph Monitor node.

Prerequisites

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

Procedure

1. Log into a Ceph node and and run the following command.

   Syntax

   ```
   ceph daemon DAEMON_TYPE.ID config get PARAMETER
   ```

   Example

   ```
   [root@mon ~]# ceph daemon osd.0 config get public_addr
   ```

Setting a specific configuration at runtime

To set a specific Ceph configuration at runtime, use the **`ceph config set`** command.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to the Ceph Monitor or OSD nodes.

Procedure

1. Set the configuration on all Monitor or OSD daemons :

   Syntax

```
ceph config set DAEMON CONFIG-OPTION VALUE
```

Example

```
[root@mon ~]# ceph config set osd debug_osd 10
```

2. Validate that the option and value are set:

   Example

   ```
   [root@mon ~]# ceph config dump
   osd      advanced debug_osd  10/10
   ```

   - To remove the configuration option from all daemons:

     Syntax

     ```
     ceph config rm DAEMON CONFIG-OPTION VALUE
     ```

     Example

     ```
     [root@mon ~]# ceph config rm osd debug_osd
     ```

   - To set the configuration for a specific daemon:

     Syntax

     ```
     ceph config set DAEMON.DAEMON-NUMBER CONFIG-OPTION VALUE
     ```

     Example

     ```
     [root@mon ~]# ceph config set osd.0 debug_osd 10
     ```

   - To validate that the configuration is set for the specified daemon:

     Example

     ```
     [root@mon ~]# ceph config dump
     osd.0      advanced debug_osd     10/10
     ```

   - To remove the configuration for a specific daemon:

     Syntax

     ```
     ceph config rm DAEMON.DAEMON-NUMBER CONFIG-OPTION
     ```

     Example

     ```
     [root@mon ~]# ceph config rm osd.0 debug_osd
     ```

Note: If you use a client that does not support reading options from the configuration database, or if you still need to use `ceph.conf` to change your cluster configuration for other reasons, run the following command:

```
ceph config set mgr mgr/cephadm/manage_etc_ceph_ceph_conf false
```

Be sure to maintain and distribute the `ceph.conf` file across the storage cluster.

OSD Memory Target

BlueStore keeps OSD heap memory usage under a designated target size with the `osd_memory_target` configuration option.

The option `osd_memory_target` sets OSD memory based upon the available RAM in the system. Use this option when TCMalloc is configured as the memory allocator, and when the `bluestore_cache_autotune` option in BlueStore is set to `true`.

Ceph OSD memory caching is more important when the block device is slow; for example, traditional hard drives, because the benefit of a cache hit is much higher than it would be with a solid state drive. However, this must be weighed into a decision to collocate OSDs with other services, such as in a hyper-converged infrastructure (HCI) or other applications.

Setting the OSD memory target

Use the `osd_memory_target` option to set the maximum memory threshold for all OSDs in the storage cluster, or for specific OSDs.

An OSD with an `osd_memory_target` option set to 16 GB might use up to 16 GB of memory.

> **Note:** Configuration options for individual OSDs take precedence over the settings for all OSDs.

Prerequisites

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

Procedure

1. To set `osd_memory_target` for all OSDs in the storage cluster:

   Syntax

   ```
   ceph config set osd osd_memory_target VALUE
   ```

   VALUE is the number of GBytes of memory to be allocated to each OSD in the storage cluster.

2. To set `osd_memory_target` for a specific OSD in the storage cluster:

   Syntax

   ```
   ceph config set osd.id osd_memory_target VALUE
   ```

   `.id` is the ID of the OSD and *VALUE* is the number of GB of memory to be allocated to the specified OSD. For example, to configure the OSD with ID 8 to use up to 16 GBytes of memory:

   Example

   ```
   [ceph: root@host01 /]# ceph config set osd.8 osd_memory_target 16G
   ```

3. To set an individual OSD to use one maximum amount of memory and configure the rest of the OSDs to use another amount, specify the individual OSD first:

   Example

   ```
   [ceph: root@host01 /]# ceph config set osd osd_memory_target 16G
   [ceph: root@host01 /]# ceph config set osd.8 osd_memory_target 8G
   ```

Reference

- To configure Red Hat Ceph Storage to autotune OSD memory usage, see Automatically tuning OSD memory.

Automatically tuning OSD memory

The OSD daemons adjust the memory consumption based on the `osd_memory_target` configuration option. The option `osd_memory_target` sets OSD memory based upon the available RAM in the system.

If Red Hat Ceph Storage is deployed on dedicated nodes that do not share memory with other services, `cephadm` automatically adjusts the per-OSD consumption based on the total amount of RAM and the number of deployed OSDs.

> **Important:** By default, the **`osd_memory_target_autotune`** parameter is set to **`true`** in the Red Hat Ceph Storage cluster.

```
ceph config set osd osd_memory_target_autotune true
```

`cephadm` starts with a fraction `mgr/cephadm/autotune_memory_target_ratio`, which defaults to `0.7` of the total RAM in the system, subtract off any memory consumed by non-autotuned daemons such as non-OSDS and for OSDs for which `osd_memory_target_autotune` is false, and then divide by the remaining OSDs.

The `osd_memory_target` parameter is calculated as follows:

```
osd_memory_target = TOTAL_RAM_OF_THE_OSD * (1048576) *
(autotune_memory_target_ratio) / NUMBER_OF_OSDS_IN_THE_OSD_NODE -
(SPACE_ALLOCATED_FOR_OTHER_DAEMONS)
```

SPACE_ALLOCATED_FOR_OTHER_DAEMONS can optionally include the following daemon space allocations:

- Alertmanager: 1 GB
- Grafana: 1 GB
- Ceph Manager: 4 GB
- Ceph Monitor: 2 GB
- Node-exporter: 1 GB
- Prometheus: 1 GB

For example, if a node has 24 OSDs and has 251 GB RAM space, then `osd_memory_target` is `7860684936`.

The final targets are reflected in the configuration database with options. You can view the limits and the current memory consumed by each daemon from the `ceph orch ps` output under `MEM LIMIT` column.

> **Note:**
> The default setting of `osd_memory_target_autotune` `true` is unsuitable for hyperconverged infrastructures where compute and Ceph storage services are colocated. In a hyperconverged infrastructure, the `autotune_memory_target_ratio` can be set to `0.2` to reduce the memory consumption of Ceph.
>
> ```
> [ceph: root@host01 /]# ceph config set mgr mgr/cephadm/autotune_memory_target_ratio
> 0.2
> ```

You can manually set a specific memory target for an OSD in the storage cluster.

```
[ceph: root@host01 /]# ceph config set osd.123 osd_memory_target 7860684936
```

You can manually set a specific memory target for an OSD host in the storage cluster.

```
ceph config set osd/host:HOSTNAME osd_memory_target TARGET_BYTES
```

```
[ceph: root@host01 /]# ceph config set osd/host:host01 osd_memory_target 1000000000
```

Note:
Enabling `osd_memory_target_autotune` overwrites existing manual OSD memory target settings. To prevent daemon memory from being tuned even when the `osd_memory_target_autotune` option or other similar options are enabled, set the `_no_autotune_memory` label on the host.

```
ceph orch host label add HOSTNAME _no_autotune_memory
```

You can exclude an OSD from memory autotuning by disabling the autotune option and setting a specific memory target.

```
[ceph: root@host01 /]# ceph config set osd.123 osd_memory_target_autotune false
[ceph: root@host01 /]# ceph config set osd.123 osd_memory_target 16G
```

MDS Memory Cache Limit

MDS servers keep their metadata in a separate storage pool, named `cephfs_metadata` and are the users of Ceph OSDs.

For Ceph File Systems, MDS servers support an entire Red Hat Ceph Storage cluster, not just a single storage device within the storage cluster. As a result, their memory requirements can be significant. This is the case particularly if the workload consists of small to medium-size files, where the ratio of metadata to data is higher.

For example, to set the `mds_cache_memory_limit` to 2000000000 bytes, run:

```
ceph_conf_overrides:
  mds:
    mds_cache_memory_limit=2000000000
```

Note: For a large Red Hat Ceph Storage cluster with a metadata-intensive workload, do not put an MDS server on the same node as other memory-intensive services. Doing so gives you the option to allocate more memory to MDS, for example, sizes greater than 100 GB.

For more information, see Metadata Server cache size limits.

General configuration options

Understand the general configuration options for Ceph.

Note: Typically, these configuration options are set automatically by deployment tools, such as `cephadm`.

`fsid`

Description: The file system ID. One per cluster.

Type: >UUID

Required: No.

Default: N/A. Usually generated by deployment tools.

`admin_socket`

Description: The socket for executing administrative commands on a daemon, irrespective of whether Ceph monitors have established a quorum.

Type: >String

Required: No

Default: `/var/run/ceph/$cluster-$name.asok`

pid_file

Description: The file in which the monitor or OSD will write its PID. For instance, `/var/run/$cluster/$type.$id.pid` creates `/var/run/ceph/mon.a.pid` for the `mon` with id `a` running in the `ceph` cluster. The `pid file` is removed when the daemon stops gracefully. If the process is not daemonized (meaning it runs with the `-f` or `-d` option), the `pid file` is not created.

Type: >String

Required: No

Default: No

chdir

Description: The directory Ceph daemons change to once they are up and running. Default `/` directory recommended.

Type: >String

Required: No

Default: `/`

max_open_files

Description: If set, when the Red Hat Ceph Storage cluster starts, Ceph sets the `max_open_fds` at the OS level (that is, the max # of file descriptors). It helps prevent Ceph OSDs from running out of file descriptors.

Type: >64-bit Integer

Required: No

Default: `0`

fatal_signal_handlers

Description: If set, we will install signal handlers for SEGV, ABRT, BUS, ILL, FPE, XCPU, XFSZ, SYS signals to generate a useful log message.

Type: >Boolean

Default: `true`

Ceph network configuration

It is important to understand the network environment that the Red Hat Ceph Storage cluster will operate in, in order to configure the Red Hat Ceph Storage.

Understanding and configuring the Ceph network options ensures optimal performance and reliability of the overall storage cluster.

Be sure to have active network connectivity and a Red Hat Ceph Storage cluster installed before configuring your Ceph network.

For more information, see “Network configuration options” on page 18 and Ceph on-wire encryption.

Network configuration for Ceph

The Ceph storage cluster does not perform request routing or dispatching on behalf of the Ceph client. Instead, Ceph clients make requests directly to Ceph OSD daemons. Ceph OSDs perform data replication on behalf of Ceph clients, which means replication and other factors impose additional loads on the networks of Ceph storage clusters.

Ceph has one network configuration requirement that applies to all daemons. The Ceph configuration file must specify the `host` for each daemon.

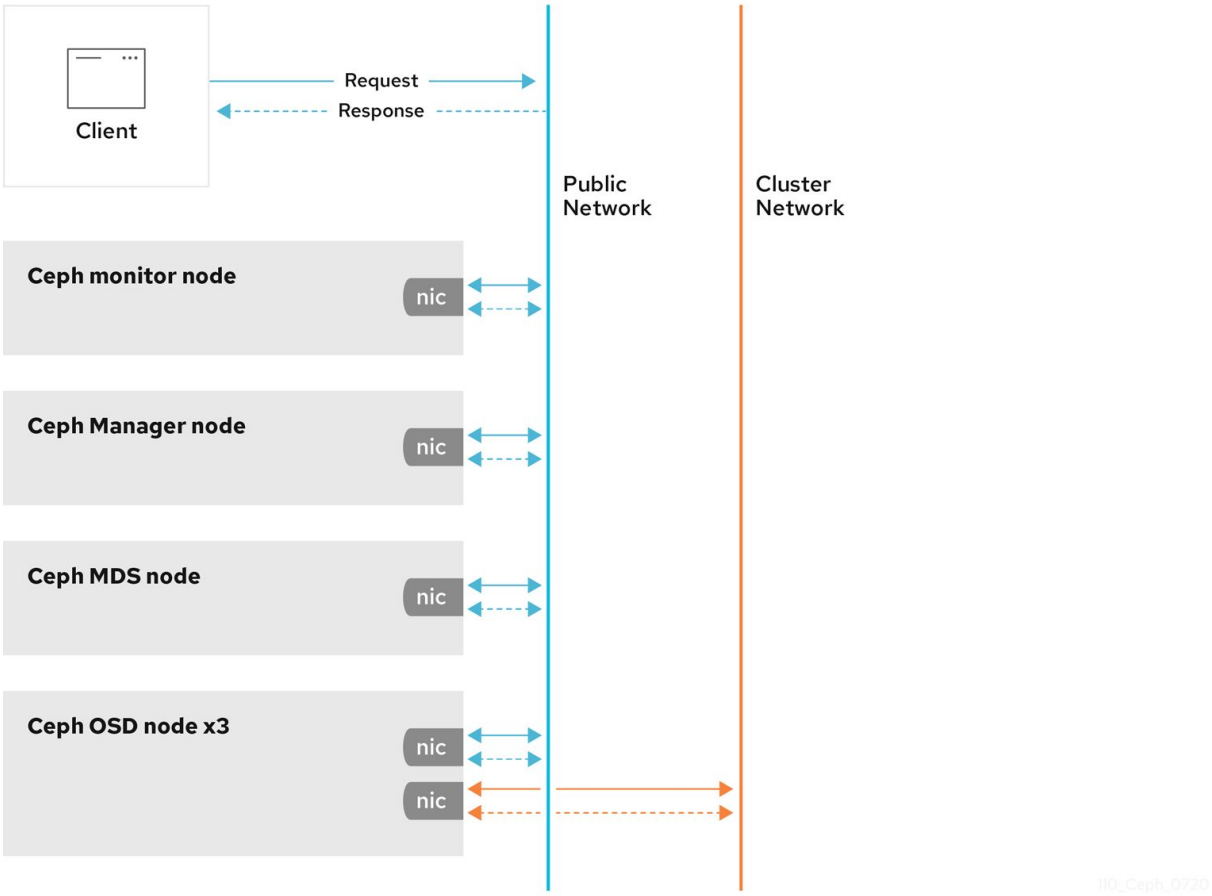
Some deployment utilities, such as `cephadm` creates a configuration file for you. Do not set these values if the deployment utility does it for you.

> **Important:**

- The `host` option is the short name of the node, not its FQDN. It is not an IP address.
- All Ceph clusters must use a public network. However, unless you specify an internal cluster network, Ceph assumes a single public network. Ceph can function with a public network only, but for large storage clusters, you will see significant performance improvement with a second private network for carrying only cluster-related traffic.
- It is recommended to run a Ceph storage cluster with two networks. One public network and one private network.

To support two networks, each Ceph Node needs to have more than one network interface card (NIC).

Figure: Network architecture



Consider operating two separate networks for better performance and security.

- **Performance:** Ceph OSDs handle data replication for the Ceph clients. When Ceph OSDs replicate data more than once, the network load between Ceph OSDs easily dwarfs the network load between Ceph clients and the Ceph storage cluster. This can introduce latency and create a performance problem. Recovery and rebalancing can also introduce significant latency on the public network.
- **Security**: While most people are generally civil, some actors will engage in what is known as a Denial of Service (DoS) attack. When traffic between Ceph OSDs gets disrupted, peering may fail and placement groups may no longer reflect an `active + clean` state, which may prevent users from reading and writing data. A great way to defeat this type of attack is to maintain a completely separate cluster network that does not connect directly to the internet.

Network configuration settings are not required. Ceph can function with a public network only, assuming a public network is configured on all hosts running a Ceph daemon. However, Ceph allows you to establish much more specific criteria, including multiple IP networks and subnet masks for your public network. You can also establish a separate cluster network to handle OSD heartbeat, object replication, and recovery traffic.

Do not confuse the IP addresses you set in the configuration with the public-facing IP addresses network clients might use to access your service. Typical internal IP networks are often `192.168.0.0` or `10.0.0.0`.

Important: If you specify more than one IP address and subnet mask for either the public or the private network, the subnets within the network must be capable of routing to each other. Additionally, make sure you include each IP address and subnet in your IP tables and open ports for them as necessary.

Note: Ceph uses CIDR notation for subnets, for example, `10.0.0.0/24`.

When you configured the networks, you can restart the cluster or restart each daemon. Ceph daemons bind dynamically, so you do not have to restart the entire cluster at once if you change the network configuration.

For common option descriptions and usage information, see "Network configuration options" on page 18.

Network messenger

Messenger is the Ceph network layer implementation. Both `simple` and `async` messenger types are supported.

The default messenger type is `async`. To change the messenger type, specify the `ms_type` configuration setting in the `[global]` section of the Ceph configuration file.

Note: For the `async` messenger, Red Hat supports the `posix` transport type, but does not currently support `rdma` or `dpdk`. By default, the `ms_type` setting in Red Hat Ceph Storage reflects `async+posix`, where `async` is the messenger type and `posix` is the transport type.

`SimpleMessenger`

The `SimpleMessenger` implementation uses TCP sockets with two threads per socket. Ceph associates each logical session with a connection. A pipe handles the connection, including the input and output of each message. While `SimpleMessenger` is effective for the `posix` transport type, it is not effective for other transport types such as `rdma` or `dpdk`.

`AsyncMessenger`

`AsyncMessenger` is the default messenger type. `AsyncMessenger` implementation uses TCP sockets with a fixed-size thread pool for connections, which should be equal to the highest number of replicas or erasure-code chunks. The thread count can be set to a lower value if performance degrades due to a low CPU count or a high number of OSDs per server.

Note: Other transport types, such as `rdma` and `dpdk` are not currently supported.

For more information about using on-wire encryption with the Ceph messenger version 2 protocol, see Ceph on-wire encryption.

For more information about asynchronous messenger options, see "Network configuration options" on page 18.

Configuring a public network

While Ceph functions well with only a public network, you can establish more specific criteria, including multiple IP networks, for your public network.

PREREQUISITE

Before you begin, make sure that you have the Red Hat Ceph Storage software installed.
To configure Ceph networks, use the **`config set`** command within the `cephadm` shell.

Note: The network addresses that you set in your network configuration are different from the public-facing IP addresses that network clients might use to access your service.

Ceph can work with only a public network. You can also establish more specific criteria, including multiple IP networks for your public network.

You can also establish a separate, private cluster network to handle OSD heartbeat, object replication, and recovery traffic. For more information about the private network, see “Configuring a private network” on page 13.

Note:

- Ceph uses CIDR notation for subnets, for example, 10.0.0.0/24. Typical internal IP networks are often 192.168.0.0/24 or 10.0.0.0/24.
- If you specify more than one network address for either the public or the cluster network, the subnets within the network must be capable of routing to each other. Make sure that you include each IP address in your firewall rules and open ports for them as necessary.

Using the public network configuration allows you to specifically define network addresses and subnets for the public network.

Note: The `public_network` option cannot be updated at runtime. After updating the Ceph configuration database, reconfigure and restart each Monitor daemon to apply the new value.

For common option descriptions and usage information, see “Network configuration options” on page 18.

1. Log in to the `cephadm` shell.
   For example,

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

2. Configure the public network with the network address in CIDR notation.

   ```
   ceph config set global public_network NETWORK_ADDRESS_CIDR
   ```

   For example,

   ```
   [ceph: root@host01 /]# ceph config set global public_network
   "10.58.145.0/24,10.58.146.0/24,10.58.148.0/24"
   ```

3. Get the list of Monitor daemons in the storage cluster.
   For example,

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

4. Verify the current value in the Monitor bootstrap configuration file.

   ```
   cat /var/lib/ceph/FSID/mon.MON_ID/config
   ```

   For example,

   ```
   [root@host01 ~]# cat /var/lib/ceph/a4b15188-c43e-11ee-9388-5cba2c8ed070/mon.host01/
   config

   [mon.host01]
   public network = 10.58.145.0/24,10.58.146.0/24
   ```

5. Reconfigure the Monitor daemon to update the Monitor bootstrap configuration file.

   ```
   ceph orch daemon reconfig mon.MON_ID
   ```

For example,

```
[ceph: root@host01 /]# ceph orch daemon reconfig mon.host01
```

6. Verify that the Monitor bootstrap configuration file contains the updated value.

```
cat /var/lib/ceph/FSID/mon.MON_ID/config
```

For example,

```
[root@host01 ~]# cat /var/lib/ceph/a4b15188-c43e-11ee-9388-5cba2c8ed070/mon.host01/
config

[mon.host01]
public network = 10.58.145.0/24,10.58.146.0/24,10.58.148.0/24
```

7. Restart the Monitor daemon.

```
ceph orch daemon restart mon.MON_ID
```

> **Note:** Wait for the Monitor daemon to regain quorum before reconfiguring and restarting the next Monitor daemon. Repeat this procedure for each Monitor daemon in the cluster.

For example,

```
[ceph: root@host01 /]# ceph orch daemon restart mon.host01
```

8. Optional: To restart the cluster, from the admin node as a root user run the **`systemctl`** command.

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

For example,

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

Configuring a private network

Network configuration settings are not required. Ceph assumes a public network with all hosts operating on it, unless you specifically configure a cluster network, also known as a *private network*.

If you create a cluster network, OSDs routes heartbeat, object replication, and recovery traffic over the cluster network. This can improve performance, compared to using a single network.

> **Important:** For added security, the cluster network should not be reachable from the public network or the Internet.

To assign a cluster network, use the **`--cluster-network`** option with the `cephadm bootstrap` command. The cluster network that you specify must define a subnet in CIDR notation (for example, 10.90.90.0/24 or fe80::/64).

You can also configure the `cluster_network` after bootstrap.

For more information about invoking `cephadm bootstrap`, see Bootstrapping a new storage cluster.

Before you begin

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

- Access to the Ceph software repository.
- Root-level access to all nodes in the storage cluster.

Procedure

1. Run the `cephadm bootstrap command` from the initial node that you want to use as the Monitor node in the storage cluster. Include the `--cluster-network` option in the command.

```
cephadm bootstrap --mon-ip IP-ADDRESS --registry-url registry.redhat.io --registry-
username USER_NAME --registry-password PASSWORD --cluster-network NETWORK-IP-ADDRESS
```

   For example,

```
[root@host01 ~]# cephadm bootstrap --mon-ip 10.10.128.68 --registry-url
registry.redhat.io --registry-username myuser1 --registry-password mypassword1 --
cluster-network 10.10.0.0/24
```

2. To configure the `cluster_network` after bootstrap, run the `config set` command and redeploy the daemons

   a. Log in to the `cephadm` shell.
   For example,

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

   b. Configure the cluster network with the subnet.

```
ceph config set global cluster_network IP_ADDRESS_WITH_SUBNET
```

   For example,

```
[ceph: root@host01 /]# ceph config set global cluster_network 10.10.0.0/24
```

   c. Get the list of services in the storage cluster, by running the **`ceph orch ls`** command.
   For example,f

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

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

```
[ceph: root@host01 /]# ceph orch restart mon
```

   e. If you want to restart the cluster, on the admin node as a root user, run **`systemctl restart`** command.

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

   For example,

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

Configuring multiple public networks to the cluster

When the user wants to place the Ceph Monitor daemons on hosts belonging to multiple network subnets, configuring multiple public networks to the cluster is necessary.

An example of usage is a stretch cluster mode used for Advanced Cluster Management (ACM) in Metro DR for OpenShift Data Foundation.

You can configure multiple public networks to the cluster during bootstrap and once bootstrap is complete.

Before you begin

Before adding a host be sure that you have a running Red Hat Ceph Storage cluster.

Procedure

1. Bootstrap a Ceph cluster configured with multiple public networks.

a. Prepare a `ceph.conf` file containing a `mon` public network section.

> **Important:** At least one of the provided public networks must be configured on the current host used for bootstrap.

```
[mon]
 public_network = PUBLIC_NETWORK1, PUBLIC_NETWORK2
```

The following is an example with three public networks to be provided for bootstrap.

```
[mon]
 public_network = 10.40.0.0/24, 10.41.0.0/24, 10.42.0.0/24
```

b. Bootstrap the cluster by providing the `ceph.conf` file as input.

> **Note:** During the bootstrap you can include any other arguments that you want to provide.

```
cephadm --image IMAGE_URL bootstrap --mon-ip MONITOR_IP -c PATH_TO_CEPH_CONF
```

> **Note:** Alternatively, an *IMAGE_ID* (such as, `13ea90216d0be03003d12d7869f72ad9de5cec9e54a27fd308e01e467c0d4a0a`) can be used instead of *IMAGE_URL*.

For example,

```
cephadm --image admin.lab.redhat.com:5000/rhceph-9-rhel9:latest bootstrap –mon-ip
10.40.0.0/24 -c /etc/ceph/ceph.conf
```

2. Add new hosts to the subnets.

> **Note:** The host being added must be reachable from the host that the active manager is running on.

a. Install the cluster's public SSH key in the new host's root user's `authorized_keys` file.

```
# ssh-copy-id -f -i /etc/ceph/ceph.pub root@NEW-HOST
```

For example,

```
[root@host01 ~]# ssh-copy-id -f -i /etc/ceph/ceph.pub root@host02
[root@host01 ~]# ssh-copy-id -f -i /etc/ceph/ceph.pub root@host03
```

b. Add the new node to the Ceph cluster.

```
ceph orch host add NEW_HOST IP [LABEL1 ...]
```

For example,

```
ceph orch host add host02 10.41.0.102 label1
ceph orch host add host03 10.42.0.103 label2
```

> **Note:**

- It is best to explicitly provide the host IP address. If an IP is not provided, then the host name will be immediately resolved via DNS and that IP will be used.
- One or more labels can also be included to immediately label the new host. For example, by default the `_admin` label will make `cephadm` maintain a copy of the `ceph.conf` file and a `client.admin` keyring file in `/etc/ceph`.

3. Add the networks configurations for the public network parameters to a running cluster. Be sure that the subnets are separated by commas and that the subnets are listed in subnet/mask format.

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

For example,

```
[root@host01 ~]# ceph config set global public_network
"10.40.0.0/24,10.41.0.0/24,10.42.0.0/24"
```

Note: For an existing cluster, configure `public_network` in the global section of the Ceph configuration database. After updating the configuration database, reconfigure and restart the Monitor daemons to apply the new value. For more information, see "Configuring a public network" on page 11.

If necessary, update the `mon` specifications to place the `mon` daemons on hosts within the specified subnets.

Reference

- For more information about adding hosts, see Adding hosts.
- For more information about stretch clusters, see Stretch clusters for Ceph storage.

Verifying firewall rules are configured for default Ceph ports

Verify that the host's firewall allows connection on the default TCP ports.

By default, Red Hat Ceph Storage daemons use TCP ports 6800—7100 to communicate with other hosts in the cluster.

Note: If your network has a dedicated firewall, you might need to verify its configuration in addition to following this procedure. See the firewall's documentation for more information.

See the firewall's documentation for more information.

Prerequisites

- Root-level access to the host.

Procedure

1. Verify the host's `iptables` configuration.
   a. List active rules.

```
[root@host1 ~]# iptables -L
```

   b. Verify the absence of rules that restrict connectivity on TCP ports 6800—7100.
   Example

```
REJECT all -- anywhere anywhere reject-with icmp-host-prohibited
```

2. Verify the host's `firewalld` configuration.
    a. List ports open on the host.
    Syntax

    ```
    firewall-cmd --zone ZONE --list-ports
    ```

    Example

    ```
    [root@host1 ~]# firewall-cmd --zone default --list-ports
    ```

    b. Verify the range is inclusive of TCP ports 6800—7100.

Firewall settings for Ceph Monitor node

You can enable encryption for all Ceph traffic over the network with the introduction of the messenger version 2 protocol. The `secure` mode setting for messenger v2 encrypts communication between Ceph daemons and Ceph clients, giving you end-to-end encryption.

Messenger v2 Protocol

The second version of Ceph's on-wire protocol, `msgr2`, includes several new features:

- A secure mode encrypts all data moving through the network.
- Encapsulation improvement of authentication payloads.
- Improvements to feature advertisement and negotiation.

The Ceph daemons bind to multiple ports allowing both the legacy, v1-compatible, and the new, v2-compatible, Ceph clients to connect to the same storage cluster. Ceph clients or other Ceph daemons connecting to the Ceph Monitor daemon will try to use the `v2` protocol first, if possible, but if not, then the legacy `v1` protocol will be used. By default, both messenger protocols, `v1` and `v2`, are enabled. The new v2 port is 3300, and the legacy v1 port is 6789, by default.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Access to the Ceph software repository.
- Root-level access to the Ceph Monitor node.

Procedure

1. Add rules using the following example.

    ```
    [root@mon ]# sudo iptables -A INPUT -i IFACE -p tcp -s IP-ADDRESS/NETMASK --dport 6789
    -j ACCEPT
    [root@mon ]# sudo iptables -A INPUT -i IFACE -p tcp -s IP-ADDRESS/NETMASK --dport 3300
    -j ACCEPT
    ```

    a. Replace `IFACE` with the public network interface (for example, `eth0`, `eth1`, and so on).

    b. Replace `IP-ADDRESS` with the IP address of the public network and `NETMASK` with the netmask for the public network.

2. For the `firewalld` daemon, run the following commands.

    ```
    [root@mon ~]# firewall-cmd --zone=public --add-port=6789/tcp
    [root@mon ~]# firewall-cmd --zone=public --add-port=6789/tcp --permanent
    [root@mon ~]# firewall-cmd --zone=public --add-port=3300/tcp
    [root@mon ~]# firewall-cmd --zone=public --add-port=3300/tcp --permanent
    ```

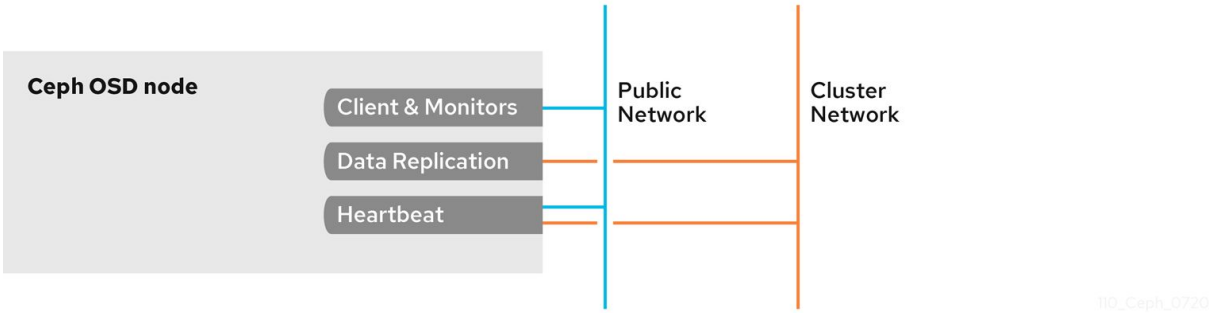
Firewall settings for Ceph OSDs

By default, Ceph OSDs bind to the first available ports on a Ceph node beginning at port 6800. Open at least four ports beginning at port 6800 for each OSD that runs on the node.

The four ports are used as follows:

- One for talking to clients and monitors on the public network.
- One for sending data to other OSDs on the cluster network.
- Two for sending heartbeat packets on the cluster network.

Figure: OSD firewall



Ports are node-specific. However, you might need to open more ports than the number of ports needed by Ceph daemons running on that Ceph node in the event that processes get restarted and the bound ports do not get released. Consider opening a few additional ports in case a daemon fails and restarts without releasing the port such that the restarted daemon binds to a new port. Also, consider opening the port range of 6800—7300 on each OSD node.

If you set separate public and cluster networks, you must add rules for both the public network and the cluster network, because clients will connect using the public network and other Ceph OSD Daemons will connect using the cluster network.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Access to the Ceph software repository.
- Root-level access to the Ceph OSD nodes.

Procedure

1. Add rules using the following example:

```
[root@mon ~]# sudo iptables -A INPUT -i IFACE  -m multiport -p tcp -s IP-ADDRESS/
NETMASK --dports 6800:7300 -j ACCEPT
```

   a. Replace `IFACE` with the public network interface (for example, `eth0`, `eth1`, and so on).

   b. Replace `IP-ADDRESS` with the IP address of the public network and `NETMASK` with the netmask for the public network.

2. For the `firewalld` daemon, run the following:

```
[root@mon ~] # firewall-cmd --zone=public --add-port=6800-7300/tcp
[root@mon ~] # firewall-cmd --zone=public --add-port=6800-7300/tcp --permanent
```

If you put the cluster network into another zone, open the ports within that zone as appropriate.

Network configuration options

Understand the various network configuration options for Ceph.

- “Common options” on page 19
- “Host options” on page 20
- “TCP options” on page 20
- “Bind options” on page 21

– “Asynchronous messenger options” on page 21

Common options

These are the common network configuration options for Ceph.

`public_network`

Description: The network address and CIDR bit count of the public (front-side) network (for example, `192.168.0.0/24`). Set in `[global]`. You can specify comma-delimited subnets.

> **Note:** The `public_network` option cannot be updated at runtime. After updating the Ceph configuration database, reconfigure and restart the Monitor daemons to apply the new value. For more information, see “Configuring a public network” on page 11.

Type: `<network-address>/<CIDR-bit-count> [, <network-address>/<CIDR-bit-count>]`

Required: No

Default: N/A

`public_addr`

Description: The IP address for the public (front-side) network. Set for each daemon.

Type: IP Address

Required: No

Default: N/A

`cluster_network`

Description: The network address and CIDR bit count of the cluster network (for example, `10.0.0.0/24`). Set in `[global]`. You can specify comma-delimited subnets.

Type: `<network-address>/<CIDR-bit-count> [, <network-address>/<CIDR-bit-count>]`

Required: No

Default: N/A

`cluster_addr`

Description: The IP address for the cluster network. Set for each daemon.

Type: Address

Required: No

Default: N/A

`ms_type`

Description: The messenger type for the network transport layer. Red Hat supports the `simple` and the `async` messenger type using `posix` semantics.

Type: String.

Required: No.

Default: `async+posix`

`ms_public_type`

Description: The messenger type for the network transport layer of the public network. It operates identically to `ms_type`, but is applicable only to the public or front-side network. This setting enables Ceph to use a different messenger type for the public or front-side and cluster or back-side networks.

Type: String.

Required: No.

Default: None.

`ms_cluster_type`

Description: The messenger type for the network transport layer of the cluster network. It operates identically to `ms_type`, but is applicable only to the cluster or back-side network. This setting enables Ceph to use a different messenger type for the public or front-side and cluster or back-side networks.

Type: String.

Required: No.

Default: None.

Host options

You must declare at least one Ceph Monitor in the Ceph configuration file, with a `mon addr` setting under each declared monitor. Ceph expects a `host` setting under each declared monitor, metadata server and OSD in the Ceph configuration file.

> **Important:** Do not use `localhost`. Use the short name of the node, not the fully-qualified domain name (FQDN). Do not specify any value for `host` when using a third party deployment system that retrieves the node name for you.

`mon_addr`

Description: A list of `<hostname>:<port>` entries that clients can use to connect to a Ceph monitor. If not set, Ceph searches `[mon.*]` sections.

Type: String

Required: No

Default: N/A

`host`

Description: The host name. Use this setting for specific daemon instances (for example, `[osd.0]`).

Type: String

Required: Yes, for daemon instances.

Default: `localhost`

TCP options

Ceph disables TCP buffering by default.

`ms_tcp_nodelay`

Description: Ceph enables `ms_tcp_nodelay` so that each request is sent immediately (no buffering). Disabling Nagle's algorithm increases network traffic, which can introduce congestion. If you experience large numbers of small packets, you may try disabling `ms_tcp_nodelay`, but be aware that disabling it will generally increase latency.

Type: Boolean

Required: No

Default: `true`

`ms_tcp_rcvbuf`

Description: The size of the socket buffer on the receiving end of a network connection. Disabled by default.

Type: 32-bit Integer

Required: No

Default: 0

Bind options

The bind options configure the default port ranges for the Ceph OSD daemons. The default range is `6800:7100`. You can also enable Ceph daemons to bind to IPv6 addresses.

> **Important:** Verify that the firewall configuration allows you to use the configured port range.

`ms_bind_port_min`

Description: The minimum port number to which an OSD daemon will bind.

Type: 32-bit Integer

Default: `6800`

Required: No

`ms_bind_ipv6`

Description: Enables Ceph daemons to bind to IPv6 addresses.

Type: Boolean

Default: `false`

Required: No

Asynchronous messenger options

These Ceph messenger options configure the behavior of `AsyncMessenger`.

`ms_async_op_threads`

Description: Initial number of worker threads used by each `AsyncMessenger` instance. This configuration setting **SHOULD** equal the number of replicas or erasure code chunks, but it may be set lower if the CPU core count is low or the number of OSDs on a single server is high.

Type: 64-bit Unsigned Integer

Required: No

Default: `3`

Ceph Monitor configuration

Use the default configuration values for the Ceph Monitor or customize them according to the intended workload.

All storage clusters have at least one monitor. A Ceph Monitor configuration usually remains fairly consistent, but you can add, remove or replace a Ceph Monitor in a storage cluster.

Ceph monitors maintain a primary copy of the cluster map. That means a Ceph client can determine the location of all Ceph monitors and Ceph OSDs just by connecting to one Ceph monitor and retrieving a current cluster map.

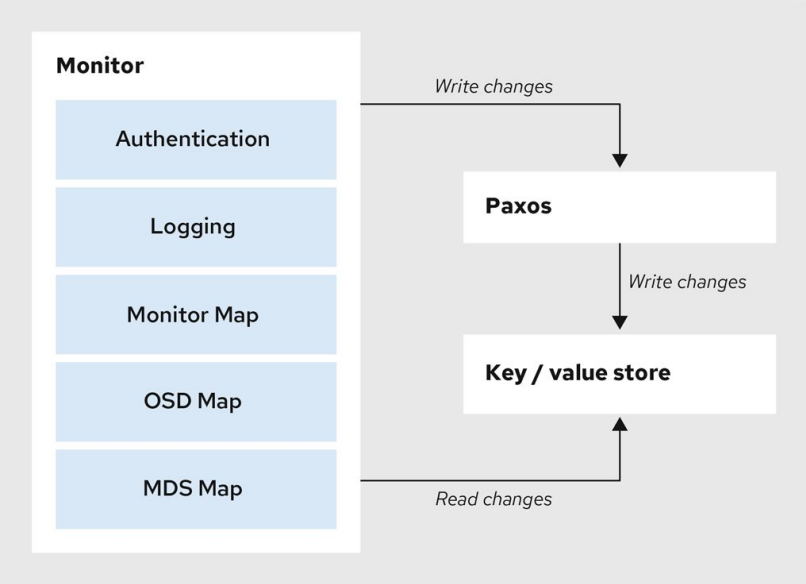
> **Important:** Be sure to have a Red Hat Ceph Storage cluster that is installed before customizing values for Ceph Monitor.

Before Ceph clients can read from or write to Ceph OSDs, they must connect to a Ceph Monitor first. With a current copy of the cluster map and the CRUSH algorithm, a Ceph client can compute the location for any object. The ability to compute object locations allows a Ceph client to talk directly to Ceph OSDs, which is an important aspect of Ceph's high scalability and performance.

The primary role of the Ceph Monitor is to maintain a primary copy of the cluster map. Ceph Monitors also provide authentication and logging services. Ceph Monitors write all changes in the monitor services to a single Paxos instance, and Paxos writes the changes to a key-value store for strong consistency. Ceph Monitors can query the most recent version of the cluster map during synchronization operations. Ceph Monitors use the key-

value store's snapshots and iterators, by using the `rocksdb` database to complete store-wide synchronization. "Figure: Paxos" on page 22 shows the flow of the Ceph Monitors toward a single Paxos instance.

Figure: Paxos



Important: In director-deployed Red Hat Ceph Storage environments, replacing Controller nodes can affect the Ceph Monitor service and lead to storage outages if Monitor IP addresses change. For more information, see "Managing the Ceph Monitor service with Red Hat OpenStack Platform" on page 0.

For more information, see "Ceph Monitor configuration options" on page 28.

Viewing the Ceph Monitor configuration database

You can view Ceph Monitor configuration in the configuration database.

Prerequisites

- A running Red Hat Ceph Storage cluster.
- Root-level access to a Ceph Monitor host.

Procedure

1. Log into the `cephadm` shell.

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

2. Use the `ceph config` command to view the configuration database.

   Example

```
[ceph: root@host01 /]# ceph config get mon
```

For more information about the options available for the `ceph config` command, use `ceph config -h`.

Cluster maps

The cluster map is a composite of maps, including the monitor map, the OSD map, and the placement group map.

The cluster map tracks a number of important events:

- Which processes are `in` the Red Hat Ceph Storage cluster.
- Which processes that are `in` the Red Hat Ceph Storage cluster are `up` and running or `down`.
- Whether, the placement groups are `active` or `inactive`, and `clean` or in some other state.
- Other details that reflect the current state of the cluster, such as:
  - the total amount of storage space or
  - the amount of storage used.

When there is a significant change in the state of the cluster, for example, a Ceph OSD goes down, a placement group falls into a degraded state, and so on. The cluster map gets updated to reflect the current state of the cluster. Additionally, the Ceph monitor also maintains a history of the prior states of the cluster. The monitor map, OSD map, and placement group map each maintain a history of their map versions. Each version is called an **epoch**.

When operating the Red Hat Ceph Storage cluster, keeping track of these states is an important part of the cluster administration.

Ceph Monitor quorum

Run a production Red Hat Ceph Storage cluster with at least three Ceph Monitors to ensure high availability. When you run multiple monitors, you can specify the initial monitors that must be members of the storage cluster to establish a quorum. This may reduce the time it takes for the storage cluster to come online.

A cluster will run sufficiently with a single monitor. However, a single monitor is a single-point-of-failure. To ensure high availability in a production Ceph storage cluster, run Ceph with multiple monitors so that the failure of a single monitor will not cause a failure of the entire storage cluster.

When a Ceph storage cluster runs multiple Ceph Monitors for high availability, Ceph Monitors use the Paxos algorithm to establish consensus about the master cluster map. A consensus requires a majority of monitors running to establish a quorum for consensus about the cluster map. For example, 1; 2 out of 3; 3 out of 5; 4 out of 6; and so on.

```
[mon]
mon_initial_members = a,b,c
```

> **Note:** Majority of the monitors in the storage cluster must be able to reach each other in to establish a quorum. You can decrease the initial number of monitors to establish a quorum with the `mon_initial_members` option.

Ceph Monitor consistency

When you add monitor settings to the Ceph configuration file, you need to be aware of some of the architectural aspects of Ceph Monitors. Ceph imposes strict consistency requirements for a Ceph Monitor when discovering another Ceph Monitor within the cluster. Whereas Ceph clients and other Ceph daemons use the Ceph configuration file to discover monitors, monitors discover each other using the monitor map (`monmap`), not the Ceph configuration file.

A Ceph Monitor always refers to the local copy of the monitor map when discovering other Ceph Monitors in the Red Hat Ceph Storage cluster. Using the monitor map instead of the Ceph configuration file avoids errors that could break the cluster. For example, typos in the Ceph configuration file when specifying a monitor address or port. Since monitors use monitor maps for discovery and they share monitor maps with clients and other Ceph daemons, the monitor map provides monitors with a strict guarantee that their consensus is valid.

Strict consistency when applying updates to the monitor maps

As with any other updates on the Ceph Monitor, changes to the monitor map always run through a distributed consensus algorithm called Paxos. The Ceph Monitors must agree on each update to the monitor map, such as adding or removing a Ceph Monitor, to ensure that each monitor in the quorum has the same version of the monitor map. Updates to the monitor map are incremental so that Ceph Monitors have the latest agreed-upon version and a set of previous versions.

Maintaining history

Maintaining a history enables a Ceph Monitor that has an older version of the monitor map to catch up with the current state of the Red Hat Ceph Storage cluster.

If Ceph Monitors discovered each other through the Ceph configuration file instead of through the monitor map, it would introduce additional risks because the Ceph configuration files are not updated and distributed automatically. Ceph Monitors might inadvertently use an older Ceph configuration file, fail to recognize a Ceph Monitor, fall out of a quorum, or develop a situation where Paxos is not able to determine the current state of the system accurately.

Bootstrap the Ceph Monitor

In most configuration and deployment cases, tools that deploy Ceph, such as `cephadm`, might help bootstrap the Ceph monitors by generating a monitor map for you.

A Ceph monitor requires a few explicit settings:

- **File System ID**: The `fsid` is the unique identifier for your object store. Since you can run multiple storage clusters on the same hardware, you must specify the unique ID of the object store when bootstrapping a monitor. Using deployment tools, such as `cephadm`, will generate a file system identifier, but you can also specify the `fsid` manually.
- **Monitor ID**: A monitor ID is a unique ID assigned to each monitor within the cluster. By convention, the ID is set to the monitor's hostname. This option can be set using a deployment tool, using the `ceph` command, or in the Ceph configuration file. In the Ceph configuration file, sections are formed as follows:

  Example

  ```
  [mon.host1]
  [mon.host2]
  ```

- **Keys**: The monitor must have secret keys.

Reference

- For more information about `cephadm` and the Ceph orchestrator, see Operations.

Minimum configuration for a Ceph Monitor

The bare minimum monitor settings for a Ceph Monitor in the Ceph configuration file includes a host name for each monitor if it is not configured for DNS and the monitor address. The Ceph Monitors run on port 6789 and 3300 by default.

> **Important:** Do not edit the Ceph configuration file.

> **Note:** This minimum configuration for monitors assumes that a deployment tool generates the `fsid` and the `mon.` key for you.

You can use the following commands to set or read the storage cluster configuration options.

`ceph config dump`

Dumps the entire configuration database for the whole storage cluster.

`ceph config generate-minimal-conf`

Generates a minimal `ceph.conf` file.

`ceph config get WHO`

Dumps the configuration for a specific daemon or a client, as stored in the Ceph Monitor's configuration database.

`ceph config set WHO OPTION VALUE`

Sets the configuration option in the Ceph Monitor's configuration database.

`ceph config show WHO`

Shows the reported running configuration for a running daemon.

`ceph config assimilate-conf -i INPUT_FILE -o OUTPUT_FILE`

Ingests a configuration file from the input file and moves any valid options into the Ceph Monitors' configuration database.

Here, the *WHO* parameter might be name of the section or a Ceph daemon, *OPTION* is a configuration file, and *VALUE* can be either `true` or `false`.

Important: When a Ceph daemon needs a config option prior to getting the option from the config store, you can set the configuration by running the following command:

```
ceph cephadm set-extra-ceph-conf
```

This command adds text to all the daemon's `ceph.conf` files. It is a workaround and is NOT a recommended operation.

Unique identifier for Ceph

Each Red Hat Ceph Storage cluster has a unique identifier (`fsid`).

If specified, the unique identifier usually appears under the `[global]` section of the configuration file. Deployment tools usually generate the `fsid` and store it in the monitor map, so the value may not appear in a configuration file. The `fsid` makes it possible to run daemons for multiple clusters on the same hardware.

Important: Do not set the `fsid` value if you use a deployment tool that does this automatically.

Ceph Monitor data store

Ceph provides a default path where Ceph monitors store data.

Important: For optimal performance in a production Red Hat Ceph Storage cluster, run Ceph monitors on separate drives from Ceph OSDs.

Note: A dedicated `/var/lib/ceph` partition should be used for the MON database with a size between 50 and 100 GB.

Ceph monitors call the `fsync()` function often, which can interfere with Ceph OSD workloads.

Ceph monitors store their data as key-value pairs. Using a data store prevents recovering Ceph monitors from running corrupted versions through Paxos, and it enables multiple modification operations in one single atomic batch, among other advantages.

Important: Only change the default data location if necessary. If you modify the default location, make it uniform across Ceph monitors by setting it in the `[mon]` section of the configuration file.

Storage capacity

Monitor your storage capacity to protect against data loss.

When a Red Hat Ceph Storage cluster gets close to its maximum capacity (specified by the `mon_osd_full_ratio` parameter), Ceph prevents you from writing to or reading from Ceph OSDs as a safety measure to prevent data loss. Therefore, letting a production Red Hat Ceph Storage cluster approach its full ratio is not a good practice, because it sacrifices high availability. The default full ratio is `.95`, or 95% of capacity. This a very aggressive setting for a test cluster with a small number of OSDs.

> **Tip:** When monitoring a cluster, be alert to warnings related to the `nearfull` ratio. This means that a failure of some OSDs could result in a temporary service disruption if one or more OSDs fails. Consider adding more OSDs to increase storage capacity.

A common scenario for test clusters involves a system administrator removing a Ceph OSD from the Red Hat Ceph Storage cluster to watch the cluster re-balance. Then, removing another Ceph OSD, and so on until the Red Hat Ceph Storage cluster eventually reaches the full ratio and locks up.

> **Important:** Red Hat recommends a bit of capacity planning even with a test cluster. Planning enables you to gauge how much spare capacity you will need in to maintain high availability.

Ideally, you want to plan for a series of Ceph OSD failures where the cluster can recover to an `active + clean` state without replacing those Ceph OSDs immediately. You can run a cluster in an `active + degraded` state, but this is not ideal for normal operating conditions.

The following diagram depicts a simplistic Red Hat Ceph Storage cluster containing 33 Ceph Nodes with one Ceph OSD per host, each Ceph OSD Daemon reading from and writing to a 3TB drive. So this exemplary Red Hat Ceph Storage cluster has a maximum actual capacity of 99TB. With a `mon osd full ratio` of `0.95`, if the Red Hat Ceph Storage cluster falls to 5 TB of remaining capacity, the cluster will not allow Ceph clients to read and write data. So, the Red Hat Ceph Storage cluster's operating capacity is 95 TB, not 99 TB.

Figure: Storage capacity

It is normal in such a cluster for one or two OSDs to fail. A less frequent but reasonable scenario involves a rack's router or power supply failing, which brings down multiple OSDs simultaneously, for example, OSDs 7-12. In such a scenario, you should still strive for a cluster that can remain operational and achieve an `active + clean` state, even if that means adding a few hosts with additional OSDs in short order. If your capacity utilization is too high, you might not lose data, but you could still sacrifice data availability while resolving an outage within a failure domain if capacity utilization of the cluster exceeds the full ratio. For this reason, Red Hat recommends at least some rough capacity planning.

Identify two numbers for your cluster:

- the number of OSDs
- the total capacity of the cluster

To determine the mean average capacity of an OSD within a cluster, divide the total capacity of the cluster by the number of OSDs in the cluster. Consider multiplying that number by the number of OSDs you expect to fail simultaneously during normal operations (a relatively small number). Finally, multiply the capacity of the cluster by the full ratio to arrive at a maximum operating capacity. Then, subtract the amount of data from the OSDs you expect to fail to arrive at a reasonable full ratio. Repeat the foregoing process with a higher number of OSD failures (for example, a rack of OSDs) to arrive at a reasonable number for a near full ratio.

Ceph heartbeat

Ceph monitors know about the cluster by requiring reports from each OSD, and by receiving reports from OSDs about the status of their neighboring OSDs.

Ceph provides reasonable default settings for interaction between monitor and OSD, however, you can modify them as needed.

Ceph Monitor synchronization role

When you run a production cluster with multiple monitors which is recommended, each monitor checks to see if a neighboring monitor has a more recent version of the cluster map. For example, a map in a neighboring monitor with one or more epoch numbers higher than the most current epoch in the map of the instant monitor. Periodically, one monitor in the cluster might fall behind the other monitors to the point where it must leave the quorum, synchronize to retrieve the most current information about the cluster, and then rejoin the quorum.

Synchronization roles

For the purposes of synchronization, monitors can assume one of three roles:

Leader

The Leader is the first monitor to achieve the most recent Paxos version of the cluster map.

Provider

The Provider is a monitor that has the most recent version of the cluster map, but was not the first to achieve the most recent version.

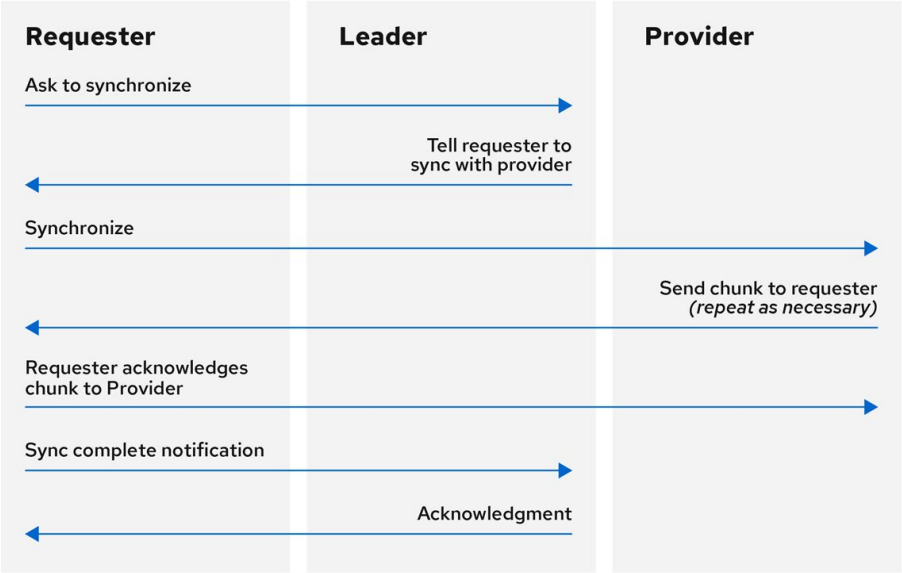
Requester

The Requester is a monitor that has fallen behind the leader and must synchronize to retrieve the most recent information about the cluster before it can rejoin the quorum.

These roles enable a leader to delegate synchronization duties to a provider, which prevents synchronization requests from overloading the leader and improving performance.

In “Figure: Ceph Monitor synchronization roles” on page 27, the requester has learned that it has fallen behind the other monitors. The requester asks the leader to synchronize, and the leader tells the requester to synchronize with a provider.

Figure: Ceph Monitor synchronization roles



Monitor synchronization

Synchronization always occurs when a new monitor joins the cluster. During runtime operations, monitors can receive updates to the cluster map at different times. This means the leader and provider roles may migrate from one monitor to another. If this happens while synchronizing, for example, a provider falls behind the leader, the provider can terminate synchronization with a requester.

Once synchronization is complete, Ceph requires trimming across the cluster. Trimming requires that the placement groups are `active + clean`.

Time synchronization

Ceph daemons pass critical messages to each other, which must be processed before daemons reach a timeout threshold. If the clocks in Ceph monitors are not synchronized, it can lead to a number of anomalies.

For example:

- Daemons ignoring received messages such as outdated timestamps.
- Timeouts triggered too soon or late when a message was not received in time.

> **Tip:** Install NTP on the Ceph monitor hosts to ensure that the monitor cluster operates with synchronized clocks.

Clock drift may still be noticeable with NTP even though the discrepancy is not yet harmful. Ceph clock drift and clock skew warnings can get triggered even though NTP maintains a reasonable level of synchronization. Increasing your clock drift may be tolerable under such circumstances. However, a number of factors such as workload, network latency, configuring overrides to default timeouts, and other synchronization options can influence the level of acceptable clock drift without compromising Paxos guarantees.

Ceph Monitor configuration options

Understand the various Ceph monitor configuration options that can be set up during deployment.

You can set these configuration options with the **`ceph config set mon CONFIGURATION_OPTION VALUE`** command.

Table title, optional			
Configuration option	**Description**	**Type**	**Default**
`mon_force_quorum_join`	Force monitor to join quorum even if it has been previously removed from the map	Boolean	`False`
`mon_dns_srv_name`	The service name used for querying the DNS for the monitor hosts/ addresses.	String	`ceph-mon`
`fsid`	The cluster ID. One per cluster.	UUID	N/A. May be generated by a deployment tool if not specified.
`mon_data_size_warn`	Ceph issues a `HEALTH_WARN` status in the cluster log when the monitor's data store reaches this threshold. The default value is 15GB.	Integer	`15*1024*1024*1024*`
`mon_data_avail_warn`	Ceph issues a `HEALTH_WARN` status in the cluster log when the available disk space of the monitor's data store is lower than or equal to this percentage.	Integer	`30`
`mon_data_avail_crit`	Ceph issues a `HEALTH_ERR` status in the cluster log when the available disk space of the monitor's data store is lower or equal to this percentage.	Integer	`5`
`mon_warn_on_cache_pools_without_hit_sets`	Ceph issues a `HEALTH_WARN` status in the cluster log if a cache pool does not have the **`hit_set_type`** parameter set.	Boolean	True
`mon_warn_on_crush_straw_calc_version_zero`	Ceph issues a `HEALTH_WARN` status in the cluster log if the CRUSH's `straw_calc_version` is zero. See CRUSH tunables overview for details.	Boolean	True
`mon_warn_on_legacy_crush_tunables`	Ceph issues a `HEALTH_WARN` status in the cluster log if CRUSH tunables are too old (older than `mon_min_crush_required_version`).	Boolean	True

Configuration option	Description	Type	Default
mon_crush_min_required_version	This setting defines the minimum tunable profile version required by the cluster.	String	hammer
mon_warn_on_osd_down_out_interval_zero	Ceph issues a HEALTH_WARN status in the cluster log if the mon_osd_down_out_interval setting is zero, because the Leader behaves in a similar manner when the **noout** flag is set. Administrators find it easier to troubleshoot a cluster by setting the **noout** flag. Ceph issues the warning to ensure administrators know that the setting is zero.	Boolean	True
mon_health_to_clog	This setting enables Ceph to send a health summary to the cluster log periodically.	Boolean	True
mon_health_detail_to_clog	This setting enable Ceph to send a health details to the cluster log periodically.	Boolean	True
mon_op_complaint_time	Number of seconds after which the Ceph Monitor operation is considered blocked after no updates.	Integer	30
mon_health_to_clog_tick_interval	How often (in seconds) the monitor sends a health summary to the cluster log. A non-positive number disables it. If the current health summary is empty or identical to the last time, the monitor will not send the status to the cluster log.	Float	60.000000
mon_health_to_clog_interval	How often (in seconds) the monitor sends a health summary to the cluster log. A non-positive number disables it. The monitor will always send the summary to the cluster log.	Integer	600
mon_sync_timeout	The number of seconds the monitor will wait for the next update message from its sync provider before it gives up and bootstraps again.	Double	60.000000

Configuration option	Description	Type	Default
mon_sync_max_payload_size	The maximum size for a sync payload (in bytes).	32-bit Integer	1045676
paxos_max_join_drift	The maximum Paxos iterations before we must first sync the monitor data stores. When a monitor finds that its peer is too far ahead of it, it will first sync with data stores before moving on.	Integer	10
paxos_stash_full_interval	How often (in commits) to stash a full copy of the PaxosService state. Currently this setting only affects mds, mon, auth, and mgr PaxosServices.	Integer	25
paxos_propose_interval	Gather updates for this time interval before proposing a map update.	Double	1.0
paxos_min	The minimum number of paxos states to keep around	Integer	500
paxos_min_wait	The minimum amount of time to gather updates after a period of inactivity.	Double	0.05
paxos_trim_min	Number of extra proposals tolerated before trimming	Integer	250
paxos_trim_max	The maximum number of extra proposals to trim at a time	Integer	500
paxos_service_trim_min	The minimum amount of versions to trigger a trim (0 disables it)	Integer	250
paxos_service_trim_max	The maximum amount of versions to trim during a single proposal (0 disables it)	Integer	500
mon_mds_force_trim_to	Force monitor to trim mdsmaps to this point (0 disables it. Dangerous, use with care)	Integer	0
mon_osd_force_trim_to	Force monitor to trim osdmaps to this point, even if there is PGs not clean at the specified epoch (0 disables it. dangerous, use with care)	Integer	0
mon_osd_cache_size	The size of osdmaps cache, not to rely on underlying store's cache	Integer	500

Configuration option	Description	Type	Default
`mon_election_timeout`	On election proposer, maximum waiting time for all ACKs in seconds.	`Float`	`5`
`mon_lease_renew_interval_factor`	`mon lease * mon lease renew interval factor` will be the interval for the Leader to renew the other monitor's leases. The factor should be less than `1.0`.	`Float`	`0.6`
`mon_lease_ack_timeout_factor`	The Leader will wait `mon lease * mon lease ack timeout factor` for the Providers to acknowledge the lease extension.	`Float`	`2.0`
`mon_min_osdmap_epochs`	Minimum number of OSD map epochs to keep at all times.	32-bit Integer	`500`
`mon_max_pgmap_epochs`	Maximum number of PG map epochs the monitor should keep.	32-bit Integer	`500`
`mon_max_log_epochs`	Maximum number of Log epochs the monitor should keep.	32-bit Integer	`500`
`mon_tick_interval`	A monitor's tick interval in seconds.	32-bit Integer	`5`
`mon_clock_drift_allowed`	The clock drift in seconds allowed between monitors.	Float	`.050`
`mon_clock_drift_warn_backoff`	Exponential backoff for clock drift warnings.	Float	`5`
`mon_timecheck_interval`	The time check interval (clock drift check) in seconds for the leader.	Float	`300.0`
`mon_timecheck_skew_interval`	The time check interval (clock drift check) in seconds when in the presence of a skew in seconds for the Leader.	Float	`30.0`
`mon_max_osd`	The maximum number of OSDs allowed in the cluster.	32-bit Integer	`10000`
`mon_globalid_prealloc`	The number of global IDs to pre-allocate for clients and daemons in the cluster.	32-bit Integer	`10000`
`mon_subscribe_interval`	The refresh interval, in seconds, for subscriptions. The subscription mechanism enables obtaining the cluster maps and log information.	Double	`86400.000000`

Configuration option	Description	Type	Default
mon_stat_smooth_intervals	Ceph will smooth statistics over the last N PG maps.	Integer	6
mon_probe_timeout	Number of seconds the monitor will wait to find peers before bootstrapping.	Double	2.0
mon_daemon_bytes	The message memory cap for metadata server and OSD messages (in bytes).	64-bit Integer Unsigned	400ul << 20
mon_max_log_entries_per_event	The maximum number of log entries per event.	Integer	4096
mon_osd_prime_pg_temp	Enables or disable priming the PGMap with the previous OSDs when an out OSD comes back into the cluster. With the true setting, the clients will continue to use the previous OSDs until the newly in OSDs as that PG peered.	Boolean	true
mon_osd_prime_pg_temp_max_time	How much time in seconds the monitor should spend trying to prime the PGMap when an out OSD comes back into the cluster.	Float	0.5
mon_osd_pool_ec_fast_read	Whether turn on fast read on the pool or not. It will be used as the default setting of newly created erasure pools if fast_read is not specified at create time.	Boolean	False
mon_mds_skip_sanity	Skip safety assertions on FSMap, in case of bugs where we want to continue anyway. Monitor terminates if the FSMap sanity check fails, but we can disable it by enabling this option.	Boolean	False
mon_max_mdsmap_epochs	The maximum amount of mdsmap epochs to trim during a single proposal.	Integer	500
mon_config_key_max_entry_size	The maximum size of config-key entry (in bytes).	Integer	65536
mon_warn_pg_not_scrubbed_ratio	The percentage of the scrub max interval past the scrub max interval to warn.	float	0.5

Configuration option	Description	Type	Default
mon_warn_pg_not_deep_scrubbed_ratio	The percentage of the deep scrub interval past the deep scrub interval to warn	float	0.75
mon_scrub_interval	How often, in seconds, the monitor scrub its store by comparing the stored checksums with the computed ones of all the stored keys.	Integer	3600*24
mon_scrub_timeout	The timeout to restart scrub of mon quorum participant does not respond for the latest chunk.	Integer	5 min
mon_scrub_max_keys	The maximum number of keys to scrub each time.	Integer	100
mon_scrub_inject_missing_keys	The probability of injecting missing keys into mon scrub.	Float	0
mon_compact_on_start	Compact the database used as Ceph Monitor store on ceph-mon start. A manual compaction helps to shrink the monitor database and improve its performance if the regular compaction fails to work.	Boolean	False
mon_compact_on_bootstrap	Compact the database used as Ceph Monitor store on bootstrap. The monitor starts probing each other for creating a quorum after bootstrap. If it times out before joining the quorum, it will start over and bootstrap itself again.	Boolean	False
mon_compact_on_trim	Compact a certain prefix (including paxos) when we trim its old states.	Boolean	True
mon_osd_mapping_pgs_per_chunk	We calculate the mapping from the placement group to OSDs in chunks. This option specifies the number of placement groups per chunk.	Integer	4096
rados_mon_op_timeout	Number of seconds to wait for a response from the monitor before returning an error from a RADOS operation. 0 means at limit, or no wait time.	Double	0

Ceph authentication configuration

Authenticating users and services is important to the security of the Red Hat Ceph Storage cluster. Red Hat Ceph Storage includes the Cephx protocol, as the default, for cryptographic authentication, and the tools to manage authentication in the storage cluster.

Be sure to have a Red Hat Ceph Storage cluster installed before configuring user authentication.

For more information, see “Cephx configuration options” on page 38.

Cephx authentication

The `cephx` protocol is enabled by default.

Cryptographic authentication has some computational costs, though they are generally low. If the network environment connecting clients and hosts is considered safe and you cannot afford authentication computational costs, you can disable it. When deploying a Ceph storage cluster, the deployment tool creates the `client.admin` user and keyring.

> **Important:** Use authentication. If authentication is disabled you are at risk of a man-in-the-middle attack altering client and server messages, which can lead to significant security issues.

Enabling and disabling Cephx

Enabling Cephx requires that you have deployed keys for the Ceph Monitors and OSDs. When toggling Cephx authentication on or off, you do not have to repeat the deployment procedures.

Enabling Cephx

When `cephx` is enabled, Ceph will look for the keyring in the default search path, which includes `/etc/ceph/$cluster.$name.keyring`. You can override this location by adding a `keyring` option in the `[global]` section of the Ceph configuration file, but this is not recommended.

Run the following procedures to enable `cephx` on a cluster with authentication disabled. If you or your deployment utility have already generated the keys, you may skip the steps related to generating keys.

Prerequisites

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

Procedure

1. Create a `client.admin` key, and save a copy of the key for your client host:

```
[root@mon ~]# ceph auth get-or-create client.admin mon 'allow *' osd 'allow *' -o /
etc/ceph/ceph.client.admin.keyring
```

> **Warning:** This will erase the contents of any existing `/etc/ceph/client.admin.keyring` file. Do not perform this step if a deployment tool has already done it for you.

2. Create a keyring for the monitor cluster and generate a monitor secret key:

```
 [root@mon ~]# ceph-authtool --create-keyring /tmp/ceph.mon.keyring --gen-key -n mon.
--cap mon 'allow *'
```

3. Copy the monitor keyring into a `ceph.mon.keyring` file in every monitor `mon data` directory. For example, to copy it to `mon.a` in cluster `ceph`, use the following:

```
 [root@mon ~]# cp /tmp/ceph.mon.keyring /var/lib/ceph/mon/ceph-a/keyring
```

4. Generate a secret key for every OSD, where _ID_ is the OSD number:

```
 ceph auth get-or-create osd.ID mon allow rwx osd allow * -o /var/lib/ceph/osd/ceph-
ID/keyring
```

5. By default the cephx authentication protocol is enabled.

> **Note:** If the cephx authentication protocol was disabled previously by setting the authentication options to none, then by removing the following lines under the [global] section in the Ceph configuration file (/etc/ceph/ceph.conf) will reenable the cephx authentication protocol:
>
> ```
> auth_cluster_required = none
>  auth_service_required = none
>  auth_client_required = none
> ```

6. Start or restart the Ceph storage cluster.

> **Important:**
> Enabling cephx requires downtime because the cluster needs to be completely restarted, or it needs to be shut down and then started while client I/O is disabled. These flags need to be set before restarting or shutting down the storage cluster:
>
> ```
> [root@mon ~]# ceph osd set noout
>  [root@mon ~]# ceph osd set norecover
>  [root@mon ~]# ceph osd set norebalance
>  [root@mon ~]# ceph osd set nobackfill
>  [root@mon ~]# ceph osd set nodown
>  [root@mon ~]# ceph osd set pause
> ```

Once cephx is enabled and all PGs are active and clean, unset the flags:

```
[root@mon ~]# ceph osd unset noout
 [root@mon ~]# ceph osd unset norecover
 [root@mon ~]# ceph osd unset norebalance
 [root@mon ~]# ceph osd unset nobackfill
 [root@mon ~]# ceph osd unset nodown
 [root@mon ~]# ceph osd unset pause
```

Disabling Cephx

If your cluster environment is relatively safe, you can offset the computation expense of the running authentication, by disabling Cephx.

The following procedure describes how to disable Cephx.

> **Important:** Red Hat recommends enabling authentication.

However, it may be easier during setup or troubleshooting to temporarily disable authentication.

Prerequisites

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

Procedure

1. Disable cephx authentication by setting the following options in the [global] section of the Ceph configuration file:

Example

```
auth_cluster_required = none
auth_service_required = none
auth_client_required = none
```

2. Start or restart the Ceph storage cluster.

Cephx user keyrings

When you run Ceph with authentication enabled, the `ceph` administrative commands and Ceph clients require authentication keys to access the Ceph storage cluster.

The most common way to provide these keys to the `ceph` administrative commands and clients is to include a Ceph keyring under the `/etc/ceph/` directory. The file name is usually `ceph.client.admin.keyring` or `$cluster.client.admin.keyring`. If you include the keyring under the `/etc/ceph/` directory, you do not need to specify a `keyring` entry in the Ceph configuration file.

> **Important:** Copy the Red Hat Ceph Storage cluster keyring file to nodes where you will run administrative commands, because it contains the `client.admin` key.

To do so, run the following command:

```
# scp USER@HOSTNAME:/etc/ceph/ceph.client.admin.keyring /etc/ceph/ceph.client.admin.keyring
```

Replace *USER* with the user name used on the host with the `client.admin` key and *HOSTNAME* with the host name of that host.

> **Note:** Ensure the `ceph.keyring` file has appropriate permissions set on the client machine.

You can specify the key itself in the Ceph configuration file using the `key` setting, which is not recommended, or a path to a key file using the `keyfile` setting.

Cephx daemon keyrings

Administrative users or deployment tools might generate daemon keyrings in the same way as generating user keyrings. By default, Ceph stores daemons keyrings inside their data directory. The default keyring locations, and the capabilities necessary for the daemon to function.

> **Note:** The monitor keyring contains a key but no capabilities, and is not part of the Ceph storage cluster `auth` database.

The daemon data directory locations default to directories of the form:

```
/var/lib/ceph/$type/CLUSTER-ID
```

For example:

```
/var/lib/ceph/osd/ceph-12
```

These locations can be overridden, but it is preferable to keep the default locations.

Cephx message signatures

Ceph provides fine-grained control so you can enable or disable signatures for service messages between the client and Ceph.

Signatures for messages can be enabled or disabled between Ceph daemons.

Important: Ceph should authenticate all ongoing messages between the entities by using the session key set up for that initial authentication.

Note: Ceph kernel modules do not currently support signatures.

Cephx configuration options

Understand the various Cephx configuration options that can be set up during deployment.

`auth_cluster_required`

Description: Valid settings are `cephx` or `none`.

Type: String

Required: No

Default: `cephx`.

`auth_service_required`

Description: Valid settings are `cephx` or `none`.

Type: String

Required: No

Default: `cephx`.

`auth_client_required`

Description: If enabled, the Red Hat Ceph Storage cluster daemons require Ceph clients to authenticate with the Red Hat Ceph Storage cluster in order to access Ceph services. Valid settings are `cephx` or `none`.

Type: String

Required: No

Default: `cephx`.

`keyring`

Description: The path to the keyring file.

Type: String

Required: No

Default: `/etc/ceph/$cluster.$name.keyring`, `/etc/ceph/$cluster.keyring`, `/etc/ceph/keyring`, `/etc/ceph/keyring.bin`

`keyfile`

Description: The path to a key file (that is. a file containing only the key).

Type: String

Required: No

Default: None

`key`

Description: The key (that is, the text string of the key itself). Not recommended.

Type: String

Required: No

Default: None

ceph-mon

Location: `$mon_data/keyring`

Capabilities: `mon 'allow *'`

ceph-osd

Location: `$osd_data/keyring`

Capabilities: `mon 'allow profile osd' osd 'allow *'`

radosgw

Location: `$rgw_data/keyring`

Capabilities: `mon 'allow rwx' osd 'allow rwx'`

cephx_require_signatures

Description: If set to `true`, Ceph requires signatures on all message traffic between the Ceph client and the Red Hat Ceph Storage cluster, and between daemons comprising the Red Hat Ceph Storage cluster.

Type: Boolean

Required: No

Default: `false`

cephx_cluster_require_signatures

Description: If set to `true`, Ceph requires signatures on all message traffic between Ceph daemons comprising the Red Hat Ceph Storage cluster.

Type: Boolean

Required: No

Default: `false`

cephx_service_require_signatures

Description: If set to `true`, Ceph requires signatures on all message traffic between Ceph clients and the Red Hat Ceph Storage cluster.

Type: Boolean

Required: No

Default: `false`

cephx_sign_messages

Description: If the Ceph version supports message signing, Ceph will sign all messages so they cannot be spoofed.

Type: Boolean

Default: `true`

auth_service_ticket_ttl

Description: When the Red Hat Ceph Storage cluster sends a Ceph client a ticket for authentication, the cluster assigns the ticket a time to live.

Type: Double

Default: `60*60`

Pools, placement groups, and CRUSH configuration

As a storage administrator, you can choose to use the Red Hat Ceph Storage default options for pools, placement groups, and the CRUSH algorithm or customize them for the intended workload.

Pools, placement groups and CRUSH

When you create pools and set the number of placement groups for the pool, Ceph uses default values when you do not specifically override the defaults.

> **Important:** It is best to override some of the defaults. Specifically, set a pool's replica size and override the default number of placement groups.

You can set these values when running pool commands.

By default, Ceph makes three replicas of objects. If you want to set four copies of an object as the default value, a primary copy and three replica copies, reset the default values as shown in `osd_pool_default_size`. If you want to allow Ceph to write a lesser number of copies in a degraded state, set `osd_pool_default_min_size` to a number less than the `osd_pool_default_size` value.

Example

```
[ceph: root@host01 /]# ceph config set global osd_pool_default_size 4  # Write an object
four times.
[ceph: root@host01 /]# ceph config set global osd_pool_default_min_size 1  # Allow writing
one copy in a degraded state.
```

Ensure you have a realistic number of placement groups. Red Hat recommends approximately 100 per OSD. Total number of OSDs multiplied by 100 divided by the number of replicas, that is, `osd_pool_default_size`. For 10 OSDs and `osd_pool_default_size` = 4, we would recommend approximately (100*10)/4=250.

Example

```
[ceph: root@host01 /]# ceph config set global osd_pool_default_pg_num 250
[ceph: root@host01 /]# ceph config set global osd_pool_default_pgp_num 250
```

Pools, placement groups, and CRUSH configuration options

Understand the various Ceph options that govern pools, placement groups, and the CRUSH algorithm.

Options	Description	Type	Default
`mon_allow_pool_delete`	Allows a monitor to delete a pool. In RHCS 3 and later releases, the monitor cannot delete the pool by default as an added measure to protect data.	Boolean	`false`
`mon_max_pool_pg_num`	The maximum number of placement groups per pool.	Integer	`65536`
`mon_pg_create_interval`	Number of seconds between PG creation in the same Ceph OSD Daemon.	Float	`30.0`
`mon_pg_stuck_threshold`	Number of seconds after which PGs can be considered as being stuck.	32-bit Integer	`300`

Options	Description	Type	Default
mon_pg_min_inactive	Ceph issues a HEALTH_ERR status in the cluster log if the number of PGs that remain inactive longer than the mon_pg_stuck_threshold exceeds this setting. The default setting is one PG. A non-positive number disables this setting.	Integer	1
mon_pg_warn_max_per_osd	Ceph issues a HEALTH_WARN status in the cluster log if the average number of PGs per OSD in the cluster is greater than this setting. A non-positive number disables this setting.	Integer	300
mon_pg_warn_min_per_osd	Ceph issues a HEALTH_WARN status in the cluster log if the average number of PGs per OSD in the cluster is less than this setting. A non-positive number disables this setting.	Integer	30
mon_pg_warn_min_objects	Do not warn if the total number of objects in the cluster is below this number.	Integer	1000
mon_pg_warn_min_pool_objects	Do not warn on pools whose object number is below this number.	Integer	1000
mon_pg_check_down_all_threshold	The threshold of down OSDs by percentage after which Ceph checks all PGs to ensure they are not stuck or stale.	Float	0.5
mon_pg_warn_max_object_skew	Ceph issue a HEALTH_WARN status in the cluster log if the average number of objects in a pool is greater than mon pg warn max object skew times the average number of objects for all pools. A non-positive number disables this setting.	Float	10

Options	Description	Type	Default
`mon_delta_reset_interval`	The number of seconds of inactivity before Ceph resets the PG delta to zero. Ceph keeps track of the delta of the used space for each pool to aid administrators in evaluating the progress of recovery and performance.	Integer	`10`
`mon_osd_max_op_age`	The maximum age in seconds for an operation to complete before issuing a `HEALTH_WARN` status.	Float	`32.0`
`osd_pg_bits`	Placement group bits per Ceph OSD Daemon.	32-bit Integer	`6`
`osd_pgp_bits`	The number of bits per Ceph OSD Daemon for Placement Groups for Placement purpose (PGPs).	32-bit Integer	`6`
`osd_crush_chooseleaf_type`	The bucket type to use for `chooseleaf` in a CRUSH rule. Uses ordinal rank rather than name.	32-bit Integer	`1`. Typically a host containing one or more Ceph OSD Daemons.
`osd_pool_default_crush_replicated_ruleset`	The default CRUSH ruleset to use when creating a replicated pool.	8-bit Integer	`0`
`osd_pool_erasure_code_stripe_unit`	Sets the default size, in bytes, of a chunk of an object stripe for erasure coded pools. Every object of size S will be stored as N stripes, with each data chunk receiving `stripe unit` bytes. Each stripe of `N * stripe unit` bytes will be encoded/decoded individually. This option can be overridden by the `stripe_unit` setting in an erasure code profile.	Unsigned 32-bit Integer	`4096`
`osd_pool_default_size`	Sets the number of replicas for objects in the pool. The default value is the same as **`ceph osd pool set POOL NAME size SIZE`**.	32-bit Integer	`3`

Options	Description	Type	Default
`osd_pool_default_min_size`	Sets the minimum number of written replicas for objects in the pool in order to acknowledge a write operation to the client. If the minimum is not met, Ceph will not acknowledge the write to the client. This setting ensures a minimum number of replicas when operating in `degraded` mode.	32-bit Integer	`0`, which means no particular minimum. If `0`, minimum is `size - (size / 2)`.
`osd_pool_default_pg_num`	The default number of placement groups for a pool. The default value is the same as `pg_num` with `mkpool`.	32-bit Integer	`32`
`osd_pool_default_pgp_num`	The default number of placement groups for placement for a pool. The default value is the same as `pgp_num` with `mkpool`. PG and PGP should be equal.	32-bit Integer	`0`
`osd_pool_default_flags`	The default flags for new pools.	32-bit Integer	`0`
`osd_max_pgls`	The maximum number of placement groups to list. A client requesting a large number can tie up the Ceph OSD Daemon.	Unsigned 64-bit Integer	`1024`
`osd_min_pg_log_entries`	The minimum number of placement group logs to maintain when trimming log files.	32-bit Int Unsigned	`250`
`osd_default_data_pool_replay_window`	The time, in seconds, for an OSD to wait for a client to replay a request.	32-bit Integer	`45`

Ceph Object Storage Daemon (OSD) configuration

Configure the Ceph Object Storage Daemon (OSD) to be redundant and optimized based on the intended workload.

For more information, see “Object Storage Daemon (OSD) configuration options” on page 44.

Ceph OSD configuration

All Ceph clusters have a configuration. A deployment tool, such as `cephadm`, will typically create an initial Ceph configuration file for you. However, you can create one yourself if you prefer to bootstrap a cluster without using a deployment tool.

All Ceph clusters have a configuration, which defines:

- Cluster identity
- Authentication settings

- Ceph daemon membership in the cluster
- Network configuration
- Host names and addresses
- Paths to keyrings
- Paths to OSD log files
- Other runtime options

For your convenience, each daemon has a series of default values. Many are set by the `ceph/src/common/config_opts.h` script. You can override these settings with a Ceph configuration file or at runtime by using the monitor `tell` command or connecting directly to a daemon socket on a Ceph node.

> **Important:** Changing the default paths makes it more difficult to troubleshoot Ceph later.

Reference

- For more information about `cephadm` and the Ceph orchestrator, see Operations.

Scrubbing the OSD

In addition to making multiple copies of objects, Ceph ensures data integrity by scrubbing placement groups. Ceph scrubbing is analogous to the **`fsck`** command on the object storage layer.

For each placement group, Ceph generates a catalog of all objects and compares each primary object and its replicas to ensure that no objects are missing or mismatched.

Light scrubbing (daily) checks the object size and attributes. Deep scrubbing (weekly) reads the data and uses checksums to ensure data integrity.

Scrubbing is important for maintaining data integrity, but it can reduce performance. Adjust the following settings to increase or decrease scrubbing operations.

For more information, see “Scrubbing options” on page 62.

Backfilling an OSD

When you add Ceph OSDs to a cluster or remove them from the cluster, the CRUSH algorithm rebalances the cluster by moving placement groups to or from Ceph OSDs to restore the balance. The process of migrating placement groups and the objects they contain can reduce the cluster operational performance considerably. To maintain operational performance, Ceph performs this migration with the *backfill* process, which allows Ceph to set backfill operations to a lower priority than requests to read or write data.

OSD recovery

When the cluster starts or when a Ceph OSD terminates unexpectedly and restarts, the OSD begins peering with other Ceph OSDs before a write operation can occur.

If a Ceph OSD crashes and comes back online, usually it will be out of sync with other Ceph OSDs containing more recent versions of objects in the placement groups. When this happens, the Ceph OSD goes into recovery mode and seeks to get the latest copy of the data and bring its map back up to date. Depending upon how long the Ceph OSD was down, the OSD’s objects and placement groups may be significantly out of date. Also, if a failure domain went down, for example, a rack, more than one Ceph OSD might come back online at the same time. This can make the recovery process time consuming and resource intensive.

To maintain operational performance, Ceph performs recovery with limitations on the number of recovery requests, threads, and object chunk sizes which allows Ceph to perform well in a degraded state.

Object Storage Daemon (OSD) configuration options

Understand the various Ceph Object Storage Daemon (OSD) configuration options that can be set during deployment.

You can set these configuration options with the **`ceph config set osd CONFIGURATION_OPTION VALUE`** command.

`osd_uuid`

Description: The universally unique identifier (UUID) for the Ceph OSD.

Type: UUID

Default: The UUID.

> **Note:** The `osd uuid` applies to a single Ceph OSD. The `fsid` applies to the entire cluster.

`osd_data`

Description: The path to the OSD's data. You must create the directory when deploying Ceph. Mount a drive for OSD data at this mount point.

Type: String

Default: `/var/lib/ceph/osd/$cluster-$id`

> **Important:** Red Hat does not recommend changing the default.

`osd_max_write_size`

Description: The maximum size of a write in megabytes.

Type: 32-bit Integer

Default `90`

`osd_client_message_size_cap`

Description: The largest client data message allowed in memory.

Type: 64-bit Integer Unsigned

Default: 500MB. `500*1024L*1024L`

`osd_class_dir`

Description: The class path for RADOS class plug-ins.

Type: String

Default `$libdir/rados-classes`

`osd_max_scrubs`

Description: The maximum number of simultaneous scrub operations for a Ceph OSD.

Type: 32-bit Int

Default `1`

`osd_scrub_thread_timeout`

Description: The maximum time in seconds before timing out a scrub thread.

Type: 32-bit Integer

Default `60`

`osd_scrub_finalize_thread_timeout`

Description: The maximum time in seconds before timing out a scrub finalize thread.

Type: 32-bit Integer

Default `60*10`

`osd_scrub_begin_hour`

Description: This restricts scrubbing to this hour of the day or later. Use `osd_scrub_begin_hour = 0` and `osd_scrub_end_hour = 0` to allow scrubbing the entire day. Along with `osd_scrub_end_hour`, they define a time window, in which the scrubs can happen. But a scrub is performed no matter whether the time window allows or not, as long as the placement group's scrub interval exceeds `osd_scrub_max_interval`.

Type: Integer

Default: `0`

Allowed range: `[0, 23]`

`osd_scrub_end_hour`

Description: This restricts scrubbing to the hour earlier than this. Use `osd_scrub_begin_hour=0` and `osd_scrub_end_hour=0` to allow scrubbing for the entire day. Along with `osd_scrub_begin_hour`, they define a time window, in which the scrubs can happen. But a scrub is performed no matter whether the time window allows or not, as long as the placement group's scrub interval exceeds `osd_scrub_max_interval`.

Type: Integer

Default: `0`

Allowed range: `[0, 23]`

`osd_scrub_load_threshold`

Description: The maximum load. Ceph will not scrub when the system load (as defined by the `getloadavg()` function) is higher than this number. Default is `0.5`.

Type: Float

Default `0.5`

Allowed range: `[0, 23]`

`osd_scrub_min_interval`

Description: The minimum interval in seconds for scrubbing the Ceph OSD when the Red Hat Ceph Storage cluster load is low.

Type: Float

Default: Once per day. `60*60*24`

`osd_scrub_max_interval`

Description: The maximum interval in seconds for scrubbing the Ceph OSD irrespective of cluster load.

Type: Float

Default: Once per week. `7*60*60*24`

`osd_scrub_interval_randomize_ratio`

Description: Takes the ratio and randomizes the scheduled scrub between `osd scrub min interval` and `osd scrub max interval`.

Type: Float

Default: `0.5`.

`mon_warn_not_scrubbed`

Description: Number of seconds after `osd_scrub_interval` to warn about any PGs that were not scrubbed.

Type: Integer

Default: `0` (no warning).

`osd_scrub_chunk_min`

Description: The object store is partitioned into chunks which end on hash boundaries. For chunky scrubs, Ceph scrubs objects one chunk at a time with writes blocked for that chunk. The `osd scrub chunk min` setting represents the minimum number of chunks to scrub.

Type: 32-bit Integer

Default `5`

`osd_scrub_chunk_max`

Description: The maximum number of chunks to scrub.

Type: 32-bit Integer

Default `25`

`osd_scrub_sleep`

Description: The time to sleep between deep scrub operations.

Type: Float

Default: `0` (or off).

`osd_scrub_during_recovery`

Description: Allows scrubbing during recovery.

Type: Boolean

Default `false`

`osd_scrub_invalid_stats`

Description: Forces extra scrub to fix stats marked as invalid.

Type: Boolean

Default `true`

`osd_scrub_priority`

Description: Controls queue priority of scrub operations versus client I/O.

Type: Unsigned 32-bit Integer

Default `5`

`osd_requested_scrub_priority`

Description: The priority set for user requested scrub on the work queue. If this value were to be smaller than `osd_client_op_priority`, it can be boosted to the value of `osd_client_op_priority` when scrub is blocking client operations.

Type: Unsigned 32-bit Integer

Default `120`

`osd_scrub_cost`

Description: Cost of scrub operations in megabytes for queue scheduling purposes.

Type: Unsigned 32-bit Integer

Default `52428800`

`osd_deep_scrub_interval`

Description: The interval for deep scrubbing, that is fully reading all data. The `osd scrub load threshold` parameter does not affect this setting.

Type: Float

Default: Once per week. `60*60*24*7`

`osd_deep_scrub_stride`

Description: Read size when doing a deep scrub.

Type: 32-bit Integer

Default: 512 KB. `524288`

`mon_warn_not_deep_scrubbed`

Description: Number of seconds after `osd_deep_scrub_interval` to warn about any PGs that were not scrubbed.

Type: Integer

Default: `0` (no warning)

`osd_deep_scrub_randomize_ratio`

Description: The rate at which scrubs will randomly become deep scrubs (even before `osd_deep_scrub_interval` has passed).

Type: Float

Default: `0.15` or 15%

`osd_deep_scrub_update_digest_min_age`

Description: How many seconds old objects must be before scrub updates the whole-object digest.

Type: Integer

Default: `7200` (120 hours)

`osd_deep_scrub_large_omap_object_key_threshold`

Description: Warning when you encounter an object with more OMAP keys than this.

Type: Integer

Default `200000`

`osd_deep_scrub_large_omap_object_value_sum_threshold`

Description: Warning when you encounter an object with more OMAP key bytes than this.

Type: Integer

Default `1 G`

`osd_delete_sleep`

Description: Time in seconds to sleep before the next removal transaction. This throttles the placement group deletion process.

Type: Float

Default `0.0`

`osd_delete_sleep_hdd`

Description: Time in seconds to sleep before the next removal transaction for HDDs.

Type: Float

Default `5.0`

`osd_delete_sleep_ssd`

Description: Time in seconds to sleep before the next removal transaction for SSDs.

Type: Float

Default `1.0`

`osd_delete_sleep_hybrid`

Description: Time in seconds to sleep before the next removal transaction when Ceph OSD data is on HDD and OSD journal or WAL and DB is on SSD.

Type: Float

Default `1.0`

`osd_op_num_shards`

Description: The number of shards for client operations.

Type: 32-bit Integer

Default `0`

`osd_op_num_threads_per_shard`

Description: The number of threads per shard for client operations.

Type: 32-bit Integer

Default `0`

`osd_op_num_shards_hdd`

Description: The number of shards for HDD operations.

Type: 32-bit Integer

Default `5`

`osd_op_num_threads_per_shard_hdd`

Description: The number of threads per shard for HDD operations.

Type: 32-bit Integer

Default `1`

`osd_op_num_shards_ssd`

Description: The number of shards for SSD operations.

Type: 32-bit Integer

Default `8`

`osd_op_num_threads_per_shard_ssd`

Description: The number of threads per shard for SSD operations.

Type: 32-bit Integer

Default `2`

`osd_client_op_priority`

Description: The priority set for client operations. It is relative to `osd recovery op priority`.

Type: 32-bit Integer

Default `63`

Valid Range: 1-63

`osd_recovery_op_priority`

Description: The priority set for recovery operations. It is relative to `osd client op priority`.

Type: 32-bit Integer

Default `3`

Valid Range: 1-63

`osd_op_thread_timeout`

Description: The Ceph OSD operation thread timeout in seconds.

Type: 32-bit Integer

Default `15`

`osd_op_complaint_time`

Description: An operation becomes complaint worthy after the specified number of seconds have elapsed.

Type: Float

Default `30`

`osd_disk_threads`

Description: The number of disk threads, which are used to perform background disk intensive OSD operations such as scrubbing and snap trimming.

Type: 32-bit Integer

Default `1`

`osd_op_history_size`

Description: The maximum number of completed operations to track.

Type: 32-bit Unsigned Integer

Default `20`

`osd_op_history_duration`

Description: The oldest completed operation to track.

Type: 32-bit Unsigned Integer

Default `600`

`osd_op_log_threshold`

Description: How many operations logs to display at once.

Type: 32-bit Integer

Default `5`

`osd_op_timeout`

Description: The time in seconds after which running OSD operations time out.

Type: Integer

Default `0`

> **Important:** Do not set the `osd op timeout` option unless your clients can handle the consequences. For example, setting this parameter on clients running in virtual machines can lead to data corruption because the virtual machines interpret this timeout as a hardware failure.

`osd_max_backfills`

Description: The maximum number of backfill operations allowed to or from a single OSD.

Type: 64-bit Unsigned Integer

Default `1`

`osd_backfill_scan_min`

Description: The minimum number of objects per backfill scan.

Type: 32-bit Integer

Default `64`

`osd_backfill_scan_max`

Description: The maximum number of objects per backfill scan.

Type: 32-bit Integer

Default `512`

`osd_backfill_full_ratio`

Description: Refuse to accept backfill requests when the Ceph OSD's full ratio is above this value.

Type: Float

Default `0.85`

`osd_backfill_retry_interval`

Description: The number of seconds to wait before retrying backfill requests.

Type: Double

Default `30.000000`

`osd_map_dedup`

Description: Enable removing duplicates in the OSD map.

Type: Boolean

Default `true`

`osd_map_cache_size`

Description: The size of the OSD map cache in megabytes.

Type: 32-bit Integer

Default `50`

`osd_map_cache_bl_size`

Description: The size of the in-memory OSD map cache in OSD daemons.

Type: 32-bit Integer

Default `50`

`osd_map_cache_bl_inc_size`

Description: The size of the in-memory OSD map cache incrementals in OSD daemons.

Type: 32-bit Integer

Default `100`

`osd_map_message_max`

Description: The maximum map entries allowed per MOSDMap message.

Type: 32-bit Integer

Default `40`

`osd_snap_trim_thread_timeout`

Description: The maximum time in seconds before timing out a snap trim thread.

Type: 32-bit Integer

Default `60*60*1`

`osd_pg_max_concurrent_snap_trims`

Description: The max number of parallel snap trims/PG. This controls how many objects per PG to trim at once.

Type: 32-bit Integer

Default `2`

`osd_snap_trim_sleep`

Description: Insert a sleep between every trim operation a PG issues.

Type: 32-bit Integer

Default `0`

`osd_max_trimming_pgs`

Description: The max number of trimming PGs

Type: 32-bit Integer

Default `2`

`osd_backlog_thread_timeout`

Description: The maximum time in seconds before timing out a backlog thread.

Type: 32-bit Integer

Default `60*60*1`

`osd_default_notify_timeout`

Description: The OSD default notification timeout (in seconds).

Type: 32-bit Integer Unsigned

Default `30`

`osd_check_for_log_corruption`

Description: Check log files for corruption. Can be computationally expensive.

Type: Boolean

Default `false`

`osd_remove_thread_timeout`

Description: The maximum time in seconds before timing out a remove OSD thread.

Type: 32-bit Integer

Default `60*60`

`osd_command_thread_timeout`

Description: The maximum time in seconds before timing out a command thread.

Type: 32-bit Integer

Default `10*60`

`osd_command_max_records`

Description: Limits the number of lost objects to return.

Type: 32-bit Integer

Default `256`

`osd_auto_upgrade_tmap`

Description: Uses `tmap` for `omap` on old objects.

Type: Boolean

Default `true`

`osd_tmapput_sets_users_tmap`

Description: Uses `tmap` for debugging only.

Type: Boolean

Default `false`

`osd_preserve_trimmed_log`

Description: Preserves trimmed log files, but uses more disk space.

Type: Boolean

Default `false`

`osd_recovery_delay_start`

Description: After peering completes, Ceph delays for the specified number of seconds before starting to recover objects.

Type: Float

Default `0`

`osd_recovery_max_active`

Description: The number of active recovery requests per OSD at one time. More requests will accelerate recovery, but the requests place an increased load on the cluster.

Type: 32-bit Integer

Default `0`

`osd_recovery_max_chunk`

Description: The maximum size of a recovered chunk of data to push.

Type: 64-bit Integer Unsigned

Default `8388608`

`osd_recovery_threads`

Description: The number of threads for recovering data.

Type: 32-bit Integer

Default `1`

`osd_recovery_thread_timeout`

Description: The maximum time in seconds before timing out a recovery thread.

Type: 32-bit Integer

Default `30`

`osd_recover_clone_overlap`

Description: Preserves clone overlap during recovery. Should always be set to `true`.

Type: Boolean

Default `true`

`rados_osd_op_timeout`

Description: Number of seconds that RADOS waits for a response from the OSD before returning an error from a RADOS operation. A value of 0 means no limit.

Type: Double

Default: 0

Ceph Monitor and OSD interaction configuration

Be sure to properly configure the interactions between the Ceph Monitors and OSDs to ensure a stable working environment.

For more information, see "Ceph Monitor and OSD configuration options" on page 57.

Ceph Monitor and OSD interaction

After you have completed your initial Ceph configuration, you can deploy and run Ceph. When you run a command such as **`ceph health`** or **`ceph -s`**, the Ceph Monitor reports on the current state of the Ceph storage cluster.

The Ceph Monitor knows about the Ceph storage cluster by requiring reports from each Ceph OSD daemon, and by receiving reports from Ceph OSD daemons about the status of their neighboring Ceph OSD daemons. If the Ceph Monitor does not receive reports, or if it receives reports of changes in the Ceph storage cluster, the Ceph Monitor updates the status of the Ceph cluster map.

Ceph provides reasonable default settings for Ceph Monitor and OSD interaction. However, you can override the defaults. The following sections describe how Ceph Monitors and Ceph OSD daemons interact for the purposes of monitoring the Ceph storage cluster.

OSD heartbeat

Each Ceph OSD daemon checks the heartbeat of other Ceph OSD daemons every 6 seconds. The heartbeat interval can be changed.

To change the heartbeat interval, change the value at runtime:

Syntax

```
ceph config set osd osd_heartbeat_interval TIME_IN_SECONDS
```

Example

```
[ceph: root@host01 /]# ceph config set osd osd_heartbeat_interval 60
```

If a neighboring Ceph OSD daemon does not send heartbeat packets within a 20 second grace period, the Ceph OSD daemon might consider the neighboring Ceph OSD daemon `down`. It can report it back to a Ceph Monitor, which updates the Ceph cluster map. To change the grace period, set the value at runtime:

Syntax

```
ceph config set osd osd_heartbeat_grace TIME_IN_SECONDS
```

Example

```
[ceph: root@host01 /]# ceph config set osd osd_heartbeat_grace 30
```

Figure: Check heartbeats

Reporting an OSD as down

By default, two Ceph OSD Daemons *from different hosts* must report to the Ceph Monitors that another Ceph OSD Daemon is `down` before the Ceph Monitors acknowledge that the reported Ceph OSD Daemon is `down`.

There is the possibility that all the OSDs reporting the failure are in different hosts in a rack with a bad switch that causes connection problems between OSDs.

To avoid a "false alarm," Ceph considers the peers reporting the failure as a proxy for a "subcluster" that is similarly laggy. While this is not always the case, it may help administrators localize the grace correction to a subset of the system that is performing poorly.

Ceph uses the `mon_osd_reporter_subtree_level` setting to group the peers into the "subcluster" by their common ancestor type in the CRUSH map.

By default, only two reports from **a different subtree** are required to report another Ceph OSD Daemon `down`. Administrators can change the number of reporters from unique subtrees and the common ancestor type required to report a Ceph OSD Daemon `down` to a Ceph Monitor by setting the `mon_osd_min_down_reporters` and `mon_osd_reporter_subtree_level` values at runtime:

Syntax

```
ceph config set mon mon_osd_min_down_reporters NUMBER
```

Example

```
[ceph: root@host01 /]# ceph config set mon mon_osd_min_down_reporters 4
```

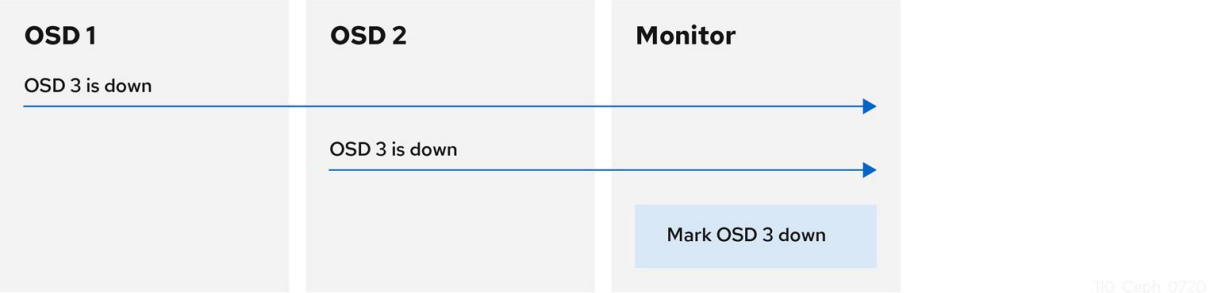
Syntax

```
ceph config set mon mon_osd_reporter_subtree_level CRUSH_ITEM
```

Example

```
[ceph: root@host01 /]# ceph config set mon mon_osd_reporter_subtree_level host
[ceph: root@host01 /]# ceph config set mon mon_osd_reporter_subtree_level rack
[ceph: root@host01 /]# ceph config set mon mon_osd_reporter_subtree_level osd
```

Figure: Report down OSDs



Reporting a peering failure

If a Ceph OSD daemon cannot peer with any of the Ceph OSD daemons defined in its Ceph configuration file or the cluster map, it pings a Ceph Monitor for the most recent copy of the cluster map every 30 seconds. The Ceph Monitor heartbeat interval can be changed.

You can change the Ceph Monitor heartbeat interval by setting the value at runtime:

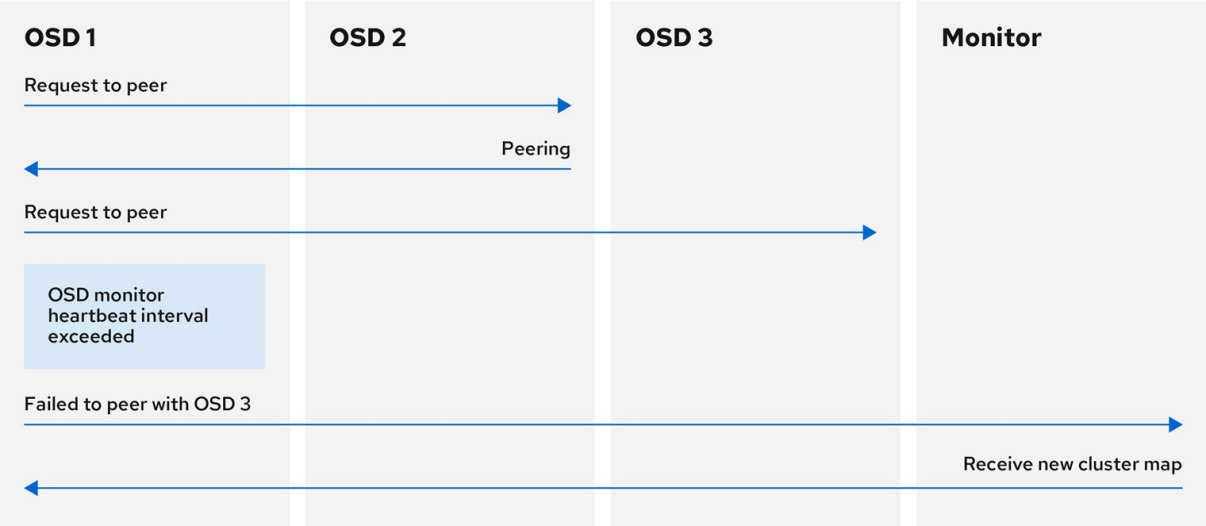
Syntax

```
ceph config set osd osd_mon_heartbeat_interval TIME_IN_SECONDS
```

Example

```
[ceph: root@host01 /]# ceph config set osd osd_mon_heartbeat_interval 60
```

Figure: Report peering failure



OSD reporting status

If a Ceph OSD Daemon does not report to a Ceph Monitor, the Ceph Monitor marks the Ceph OSD Daemon `down` after the `mon_osd_report_timeout`, which is 900 seconds, elapses. A Ceph OSD Daemon sends a report to a Ceph Monitor when a reportable event such as a failure, a change in placement group stats, a change in `up_thru` or when it boots within 5 seconds. the minimum report interval can be changed.

You can change the Ceph OSD Daemon minimum report interval by setting the `osd_mon_report_interval` value at runtime:

Syntax

```
ceph config set osd osd_mon_report_interval TIME_IN_SECONDS
```

To get, set, and verify the config you can use the following example:

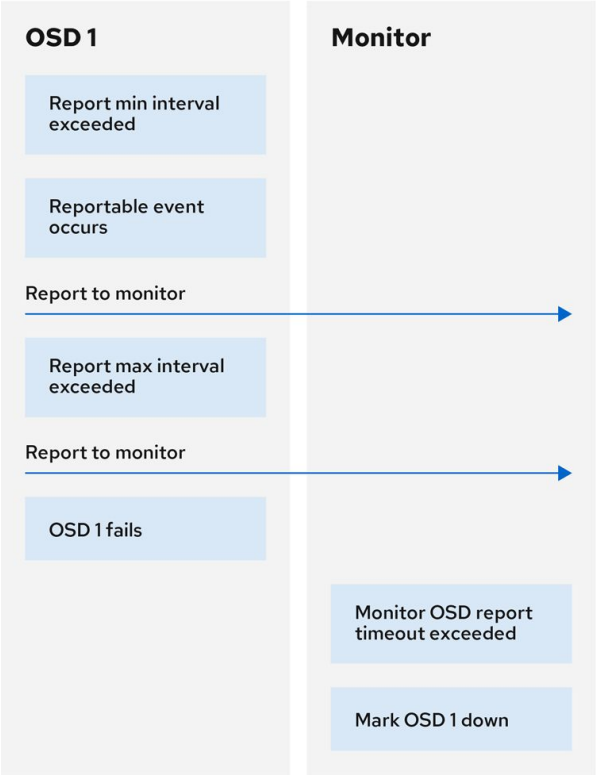
Example

```
[ceph: root@host01 /]# ceph config get osd osd_mon_report_interval
5

[ceph: root@host01 /]# ceph config set osd osd_mon_report_interval
20

[ceph: root@host01 /]# ceph config dump | grep osd
global                  advanced  osd_pool_default_crush_rule                -1
osd                   basic     osd_memory_target                          4294967296
osd                   advanced  osd_mon_report_interval                    20
```

Figure: Ceph configuration update



Ceph Monitor and OSD configuration options

Understand the various Ceph Monitor and OSD configuration options.

When modifying heartbeat settings, include them in the `[global]` section of the Ceph configuration file.

`mon_osd_min_up_ratio`

Description: The minimum ratio of `up` Ceph OSD Daemons before Ceph will mark Ceph OSD Daemons `down`.

Type: Double

Default: `.3`

`mon_osd_min_in_ratio`

Description: The minimum ratio of `in` Ceph OSD Daemons before Ceph will mark Ceph OSD Daemons `out`.

Type Double

Default: `0.750000`

`mon_osd_laggy_halflife`

Description: The number of seconds `laggy` estimates will decay.

Type: Integer

Default: `60*60`

`mon_osd_laggy_weight`

Description: The weight for new samples in `laggy` estimation decay.

Type: Double

Default: `0.3`

`mon_osd_laggy_max_interval`

Description: Maximum value of `laggy_interval` in laggy estimations (in seconds). The monitor uses an adaptive approach to evaluate the `laggy_interval` of a certain OSD. This value will be used to calculate the grace time for that OSD.

Type: Integer

Default: `300`

`mon_osd_adjust_heartbeat_grace`

Description: If set to `true`, Ceph will scale based on `laggy` estimations.

Type: Boolean

Default: `true`

`mon_osd_adjust_down_out_interval`

Description: If set to `true`, Ceph will scaled based on `laggy` estimations.

Type: Boolean

Default: `true`

`mon_osd_auto_mark_in`

Description: Ceph will mark any booting Ceph OSD Daemons as `in` the Ceph Storage Cluster.

Type: Boolean

Default: `false`

`mon_osd_auto_mark_auto_out_in`

Description: Ceph will mark booting Ceph OSD Daemons auto marked `out` of the Ceph Storage Cluster as `in` the cluster.

Type: Boolean

Default: `true`

`mon_osd_auto_mark_new_in`

Description: Ceph will mark booting new Ceph OSD Daemons as `in` the Ceph Storage Cluster.

Type: Boolean

Default: `true`

`mon_osd_down_out_interval`

Description: The number of seconds Ceph waits before marking a Ceph OSD Daemon `down` and `out` if it does not respond.

Type: 32-bit Integer

Default: `600`

`mon_osd_downout_subtree_limit`

Description: The largest CRUSH unit type that Ceph will automatically mark `out`.

Type: String

Default: `rack`

`mon_osd_reporter_subtree_level`

Description: This setting defines the parent CRUSH unit type for the reporting OSDs. The OSDs send failure reports to the monitor if they find an unresponsive peer. The monitor may mark the reported OSD `down` and then `out` after a grace period.

Type: String

Default: `host`

`mon_osd_report_timeout`

Description: The grace period in seconds before declaring unresponsive Ceph OSD Daemons `down`.

Type: 32-bit Integer

Default: `900`

`mon_osd_min_down_reporters`

Description: The minimum number of Ceph OSD Daemons required to report a `down` Ceph OSD Daemon.

Type: 32-bit Integer

Default: `2`

`osd_heartbeat_address`

Description: A Ceph OSD Daemon's network address for heartbeats.

Type: Address

Default: The host address.

`osd_heartbeat_interval`

Description: How often a Ceph OSD Daemon pings its peers (in seconds).

Type: 32-bit Integer

Default: `6`

`osd_heartbeat_grace`

Description: The elapsed time when a Ceph OSD Daemon has not shown a heartbeat that the Ceph Storage Cluster considers it `down`.

Type: 32-bit Integer

Default: `20`

`osd_mon_heartbeat_interval`

Description: Frequency of Ceph OSD Daemon pinging a Ceph Monitor if it has no Ceph OSD Daemon peers.

Type: 32-bit Integer

Default: `30`

`osd_mon_report_interval_max`

Description The maximum time in seconds that a Ceph OSD Daemon can wait before it must report to a Ceph Monitor.

Type: 32-bit Integer

Default: `120`

`osd_mon_report_interval_min`

Description: The minimum number of seconds a Ceph OSD Daemon may wait from startup or another reportable event before reporting to a Ceph Monitor.

Type: 32-bit Integer

Default: `5`

Valid Range: Should be less than `osd mon report interval max`

`osd_mon_ack_timeout`

Description: The number of seconds to wait for a Ceph Monitor to acknowledge a request for statistics.

Type: 32-bit Integer

Default: `30`

Debugging and logging configuration

Increase the amount of debugging and logging information in `cephadm` to help diagnose problems with Red Hat Ceph Storage.

Be sure to have a running Red Hat Ceph Storage cluster before beginning to debugging and logging information in `cephadm`.

- For more information about troubleshooting cephadm, see `cephadm` troubleshooting.
- For more information about `cephadm` logging, see `cephadm` operations.

Debugging and logging configuration options

Logging and debugging settings are not required in a Ceph configuration file, but you can override default settings as needed.

The options take a single item that is assumed to be the default for all daemons regardless of channel. For example, specifying "info" is interpreted as "default=info". However, options can also take key/value pairs. For example, `default=daemon audit=local0` is interpreted as "default all to *daemon*, override *audit* with *local0*."

`log_file`

Description: The location of the logging file for the cluster.

Type: String

Required: No

Default: `/var/log/ceph/$cluster-$name.log`

`mon_cluster_log_file`

Description The location of the monitor cluster's log file.

Type String

Required No

Default: `/var/log/ceph/$cluster.log`

`log_max_new`

Description: The maximum number of new log files.

Type: Integer

Required: No

Default: `1000`

`log_max_recent`

Description: The maximum number of recent events to include in a log file.

Type: Integer

Required: No

Default: `10000`

`log_flush_on_exit`

Description: Determines if Ceph flushes the log files after exit.

Type: Boolean

Required: No

Default: `true`

`mon_cluster_log_file_level`

Description: The level of file logging for the monitor cluster. Valid settings include "debug", "info", "sec", "warn", and "error".

Type: String

Default: `"info"`

`log_to_stderr`

Description: Determines if logging messages appear in `stderr`.

Type: Boolean

Required: No

Default: `true`

`err_to_stderr`

Description: Determines if error messages appear in `stderr`.

Type: Boolean

Required: No

Default: `true`

`log_to_syslog`

Description: Determines if logging messages appear in `syslog`.

Type: Boolean

Required: No

Default: `false`

`err_to_syslog`

Description: Determines if error messages appear in `syslog`.

Type: Boolean

Required: No

Default: `false`

`clog_to_syslog`

Description: Determines if `clog` messages will be sent to `syslog`.

Type: Boolean

Required: No

Default: `false`

`mon_cluster_log_to_syslog`

Description: Determines if the cluster log will be output to `syslog`.

Type: Boolean

Required: No

Default: `false`

`mon_cluster_log_to_syslog_level`

Description: The level of syslog logging for the monitor cluster. Valid settings include "debug", "info", "sec", "warn", and "error".

Type: String

Default: `"info"`

`mon_cluster_log_to_syslog_facility`

Description: The facility generating the syslog output. This is usually set to "daemon" for the Ceph daemons.

Type: String

Default: `"daemon"`

`clog_to_monitors`

Description: Determines if `clog` messages will be sent to monitors.

Type: Boolean

Required: No

Default: `true`

`mon_cluster_log_to_graylog`

Description: Determines if the cluster will output log messages to graylog.

Type: String

Default: `"false"`

`mon_cluster_log_to_graylog_host`

Description: The IP address of the graylog host. If the graylog host is different from the monitor host, override this setting with the appropriate IP address.

Type: String

Default: `"127.0.0.1"`

`mon_cluster_log_to_graylog_port`

Description: Graylog logs will be sent to this port. Ensure the port is open for receiving data.

Type: String

Default: `"12201"`

`osd_preserve_trimmed_log`

Description: Preserves trimmed logs after trimming.

Type: Boolean

Required: No

Default: `false`

`osd_tmapput_sets_uses_tmap`

Description: Uses `tmap`. For debug only.

Type: Boolean

Required: No

Default: `false`

`osd_min_pg_log_entries`

Description: The minimum number of log entries for placement groups.

Type: 32-bit Unsigned Integer

Required: No

Default: 1000

`osd_op_log_threshold`

Description: Number of op log messages to show up in one pass.

Type: Integer

Required: No

Default: 5

Scrubbing options

Ceph ensures data integrity by scrubbing placement groups.

The following are the Ceph scrubbing options that you can adjust to increase or decrease scrubbing operations.

You can set these configuration options with the **`ceph config set global CONFIGURATION_OPTION VALUE`** command.

`mds_max_scrub_ops_in_progress`

Description: The maximum number of scrub operations performed in parallel. You can set this value with **`ceph config set mds_max_scrub_ops_in_progress VALUE`** command.

Type: integer

Default: 5

`osd_max_scrubs`

Description: The maximum number of simultaneous scrub operations for a Ceph OSD Daemon.

Type: integer

Default: 1

`osd_scrub_begin_hour`

Description: The specific hour at which the scrubbing begins. Along with `osd_scrub_end_hour`, you can define a time window in which the scrubs can happen. Use `osd_scrub_begin_hour = 0` and `osd_scrub_end_hour = 0` to allow scrubbing the entire day.

Type: integer

Default: 0

Allowed range: `[0, 23]`

`osd_scrub_end_hour`

Description: The specific hour at which the scrubbing ends. Along with `osd_scrub_begin_hour`, you can define a time window, in which the scrubs can happen. Use `osd_scrub_begin_hour = 0` and `osd_scrub_end_hour = 0` to allow scrubbing for the entire day.

Type: integer

Default: 0

Allowed range: `[0, 23]`

`osd_scrub_begin_week_day`

Description: The specific day on which the scrubbing begins. 0 = Sunday, 1 = Monday, etc. Along with `osd_scrub_end_week_day`, you can define a time window in which scrubs can happen. Use `osd_scrub_begin_week_day = 0` and `osd_scrub_end_week_day = 0` to allow scrubbing for the entire week.

Type: integer

Default: 0

Allowed range: `[0, 6]`

`osd_scrub_end_week_day`

Description: This defines the day on which the scrubbing ends. 0 = Sunday, 1 = Monday, etc. Along with `osd_scrub_begin_week_day`, they define a time window, in which the scrubs can happen. Use `osd_scrub_begin_week_day = 0` and `osd_scrub_end_week_day = 0` to allow scrubbing for the entire week.

Type: integer

Default: 0

Allowed range: `[0, 6]`

`osd_scrub_during_recovery`

Description: Allow scrub during recovery. Setting this to `false` disables scheduling new scrub, and deep-scrub, while there is an active recovery. The already running scrubs continue which is useful to reduce load on busy storage clusters.

Type: boolean

Default: false

`osd_scrub_extended_sleep`

Description: Duration to inject a delay during scrubbing out of scrubbing hours or seconds.

Type: float

Default: 0.0

`osd_scrub_backoff_ratio`

Description: Backoff ratio for scheduling scrubs. This is the percentage of ticks that do NOT schedule scrubs, 66% means that 1 out of 3 ticks schedules scrubs.

Type: float

Default: 0.66

`osd_scrub_load_threshold`

Description: The normalized maximum load. Scrubbing does not happen when the system load, as defined by `getloadavg()`/number of online CPUs, is higher than this defined number.

Type: float

Default: 0.5

`osd_scrub_min_interval`

Description: The minimal interval in seconds for scrubbing the Ceph OSD daemon when the Ceph storage Cluster load is low.

Type: float

Default: 1 day

`osd_scrub_max_interval`

Description: The maximum interval in seconds for scrubbing the Ceph OSD daemon irrespective of cluster load.

Type: float

Default: 7 days

`osd_scrub_chunk_min`

Description: The minimal number of object store chunks to scrub during a single operation. Ceph blocks writes to a single chunk during scrub.

Type: integer

Default: 5

`osd_scrub_chunk_max`

Description: The maximum number of object store chunks to scrub during a single operation.

Type: integer

Default: 25

`osd_scrub_sleep`

Description: Time to sleep before scrubbing the next group of chunks. Increasing this value slows down the overall rate of scrubbing, so that client operations are less impacted.

Type: float

Default: 0.0

`osd_deep_scrub_interval`

Description: The interval for `deep` scrubbing, fully reading all data. The `osd_scrub_load_threshold` does not affect this setting.

Type: float

Default: 7 days

`osd_debug_deep_scrub_sleep`

Description: Inject an expensive sleep during deep scrub IO to make it easier to induce preemption.

Type: float

Default: 0

`osd_scrub_interval_randomize_ratio`

Description: Add a random delay to `osd_scrub_min_interval` when scheduling the next scrub job for a placement group. The delay is a random value less than `osd_scrub_min_interval` * `osd_scrub_interval_randomized_ratio`. The default setting spreads scrubs throughout the allowed time window of `[1, 1.5]` * `osd_scrub_min_interval`.

Type: float

Default: 0.5

`osd_deep_scrub_stride`

Description: Read size when doing a deep scrub.

Type: size

Default: 512 KB

`osd_scrub_auto_repair_num_errors`

Description: Auto repair does not occur if more than this many errors are found.

Type: integer

Default: 5

`osd_scrub_auto_repair`

Description: Setting this to `true` enables automatic Placement Group (PG) repair when errors are found by scrubs or deep-scrubs. However, if more than `osd_scrub_auto_repair_num_errors` errors are found, a repair is NOT performed.

Type: boolean

Default: false

`osd_scrub_max_preemptions`

Description: Set the maximum number of times you need to preempt a deep scrub due to a client operation before blocking client IO to complete the scrub.

Type: integer

Default: 5

`osd_deep_scrub_keys`

Description: Number of keys to read from an object at a time during deep scrub.

Type: integer

Default: 1024

BlueStore configuration options

Ceph BlueStore configuration options can be configured during deployment.

"BlueStore configuration options" on page 65 describes the available BlueStore configuration options.

BlueStore configuration options

Option	Description	Type	Default
`rocksdb_cache_size`	The size of the RocksDB cache in MB.	32-bit integer	`512`
`bluestore_throttle_bytes`	Maximum bytes available before the user throttles input or output (I/O) submission.	Size	64 MB
`bluestore_throttle_deferred_bytes`	Maximum bytes for deferred writes before the user throttles I/O submission.	Size	128 MB
`bluestore_throttle_cost_per_io`	Overhead added to the transaction cost in bytes for each I/O.	Size	0 B

Option	Description	Type	Default
`bluestore_throttle_cost_per_io_hdd`	Default `bluestore_throttle_cost_per_io` value for HDDs.	Unsigned integer	`67 000`
`bluestore_throttle_cost_per_io_ssd`	Default `bluestore_throttle_cost_per_io` value for SSDs.	Unsigned integer	`4 000`
`bluestore_debug_enforce_settings`	`hdd` enforces settings intended for BlueStore on rotational drives. `ssd` enforces settings intended for BlueStore on solid-state drives.	`default`, `hdd`, `ssd`	`default` **Note:** After changing this option, restart the OSD.