Red Hat Ceph Storage 3

Troubleshooting Guide

Troubleshooting Red Hat Ceph Storage

Last Updated: 2018-09-07

Red Hat Ceph Storage 3 Troubleshooting Guide

Troubleshooting Red Hat Ceph Storage

Legal Notice

Abstract

This document describes how to resolve common problems with Red Hat Ceph Storage.

Table of Contents

CHAPTER 1. INITIAL TROUBLESHOOTING

This chapter includes information on:

- How to start troubleshooting Ceph errors (Section 1.1, "Identifying Problems")
- Most common **`ceph health`** error messages (Section 1.2, "Understanding the Output of the **`ceph health`** Command")
- Most common Ceph log error messages (Section 1.3, "Understanding Ceph Logs")

1.1. IDENTIFYING PROBLEMS

To determine possible causes of the error with Red Hat Ceph Storage you encounter, answer the following question:

1. Certain problems can arise when using unsupported configurations. Ensure that your configuration is supported. See the Red Hat Ceph Storage: Supported configurations article for details.
2. Do you know what Ceph component causes the problem?
   a. No. Follow Section 1.1.1, "Diagnosing the Health of a Ceph Storage Cluster".
   b. Monitors. See Chapter 4, *Troubleshooting Monitors*.
   c. OSDs. See Chapter 5, *Troubleshooting OSDs*.
   d. Placement groups. See Chapter 6, *Troubleshooting Placement Groups*.

1.1.1. Diagnosing the Health of a Ceph Storage Cluster

This procedure lists basic steps to diagnose the health of a Ceph Storage Cluster.

1. Check the overall status of the cluster:

   ```
   # ceph health detail
   ```

   If the command returns **`HEALTH_WARN`** or **`HEALTH_ERR`** see Section 1.2, "Understanding the Output of the **`ceph health`** Command" for details.

2. Check the Ceph logs for any error messages listed in Section 1.3, "Understanding Ceph Logs". The logs are located by default in the **`/var/log/ceph/`** directory.
3. If the logs do not include sufficient amount of information, increase the debugging level and try to reproduce the action that failed. See Chapter 2, *Configuring Logging* for details.
4. Use the **`ceph-medic`** utility to diagnose the storage cluster. See Chapter 7, *Installing and Using **`ceph-medic`** to Diagnose a Ceph Storage Cluster*for more details.

1.2. UNDERSTANDING THE OUTPUT OF THE `CEPH HEALTH` COMMAND

The **`ceph health`** command returns information about the status of the Ceph Storage Cluster:

- **HEALTH_OK** indicates that the cluster is healthy.
- **HEALTH_WARN** indicates a warning. In some cases, the Ceph status returns to **HEALTH_OK** automatically, for example when Ceph finishes the rebalancing process. However, consider further troubleshooting if a cluster is in the **HEALTH_WARN** state for longer time.
- **HEALTH_ERR** indicates a more serious problem that requires your immediate attention.

Use the **ceph health detail** and **ceph -s** commands to get a more detailed output.

The following tables list the most common **HEALTH_ERR** and **HEALTH_WARN** error messages related to Monitors, OSDs, and placement groups. The tables provide links to corresponding sections that explain the errors and point to specific procedures to fix problems.

Table 1.1. Error Messages Related to Monitors

Error message	See
HEALTH_WARN	
mon.X is down (out of quorum)	Section 4.1.1, “A Monitor Is Out of Quorum”
clock skew	Section 4.1.2, “Clock Skew”
store is getting too big!	Section 4.1.3, “The Monitor Store is Getting Too Big”

Table 1.2. Error Messages Related to Ceph Manager Daemons

Error message	See
HEALTH_WARN	
unknown pgs	Opening Ports for Ceph Manager

Table 1.3. Error Messages Related to OSDs

Error message	See
HEALTH_ERR	
full osds	Section 5.1.1, “Full OSDs”
HEALTH_WARN	
nearfull osds	Section 5.1.2, “Nearfull OSDs”
osds are down	Section 5.1.3, “One or More OSDs Are Down” Section 5.1.4, “Flapping OSDs”

Error message	See
`requests are blocked`	Section 5.1.5, “Slow Requests, and Requests are Blocked”
`slow requests`	Section 5.1.5, “Slow Requests, and Requests are Blocked”

Table 1.4. Error Messages Related to Placement Groups

Error message	See
`HEALTH_ERR`	
`pgs down`	Section 6.1.5, “Placement Groups Are **`down`**”
`pgs inconsistent`	Section 6.1.2, “Inconsistent Placement Groups”
`scrub errors`	Section 6.1.2, “Inconsistent Placement Groups”
`HEALTH_WARN`	
`pgs stale`	Section 6.1.1, “Stale Placement Groups”
`unfound`	Section 6.1.6, “Unfound Objects”

1.3. UNDERSTANDING CEPH LOGS

By default, Ceph stores its logs in the **`/var/log/ceph/`** directory.

The **`<cluster-name>.log`** is the main cluster log file that includes the global cluster events. By default, this log is named **`ceph.log`**. Only the Monitor hosts include the main cluster log.

Each OSD and Monitor has its own log file, named **`<cluster-name>-osd.<number>.log`** and **`<cluster-name>-mon.<hostname>.log`**.

When you increase debugging level for Ceph subsystems, Ceph generates a new log files for those subsystems as well. For details about logging, see Chapter 2, *Configuring Logging*.

The following tables list the most common Ceph log error messages related to Monitors and OSDs. The tables provide links to corresponding sections that explain the errors and point to specific procedures to fix them.

Table 1.5. Common Error Messages in Ceph Logs Related to Monitors

Error message	Log file	See
`clock skew`	Main cluster log	Section 4.1.2, “Clock Skew”

Error message	Log file	See
`clocks not synchronized`	Main cluster log	Section 4.1.2, “Clock Skew”
`Corruption: error in middle of record`	Monitor log	Section 4.1.1, “A Monitor Is Out of Quorum” Section 4.3, “Recovering the Monitor Store”
`Corruption: 1 missing files`	Monitor log	Section 4.1.1, “A Monitor Is Out of Quorum” Section 4.3, “Recovering the Monitor Store”
`Caught signal (Bus error)`	Monitor log	Section 4.1.1, “A Monitor Is Out of Quorum”

Table 1.6. Common Error Messages in Ceph Logs Related to OSDs

Error message	Log file	See
`heartbeat_check: no reply from osd.X`	Main cluster log	Section 5.1.4, “Flapping OSDs”
`wrongly marked me down`	Main cluster log	Section 5.1.4, “Flapping OSDs”
`osds have slow requests`	Main cluster log	Section 5.1.5, “Slow Requests, and Requests are Blocked”
`FAILED assert(!m_filestore_fail_eio)`	OSD log	Section 5.1.3, “One or More OSDs Are Down”
`FAILED assert(0 == "hit suicide timeout")`	OSD log	Section 5.1.3, “One or More OSDs Are Down”

CHAPTER 2. CONFIGURING LOGGING

This chapter describes how to configure logging for various Ceph subsystems.

IMPORTANT

Logging is resource intensive. Also, verbose logging can generate a huge amount of data in a relatively short time. It you are encountering problems in a specific subsystem of the cluster, enable logging only of that subsystem. See Section 2.1, “Ceph Subsystems” for more information.

In addition, consider setting up a rotation of log files. See Section 2.4, “Accelerating Log Rotation” for details.

Once you fix any problems you encounter, change the subsystems log and memory levels to their default values. See Appendix A, *Subsystems Default Logging Levels Values* for list of all Ceph subsystems and their default values.

You can configure Ceph logging by:

- Using the **`ceph`** command at runtime. This is the most common approach. See Section 2.2, “Configuring Logging at Runtime” for details.

- Updating the Ceph configuration file. Use this approach if you are encountering problems when starting the cluster. See Section 2.3, “Configuring Logging in the Ceph Configuration File” for details.

2.1. CEPH SUBSYSTEMS

This section contains information about Ceph subsystems and their logging levels.

Understanding Ceph Subsystems and Their Logging Levels

Ceph consists of several subsystems. Each subsystem has a logging level of its:

- Output logs that are stored by default in **`/var/log/ceph/`** directory (log level)

- Logs that are stored in a memory cache (memory level)

In general, Ceph does not send logs stored in memory to the output logs unless:

- A fatal signal is raised

- An assert in source code is triggered

- You request it

You can set different values for each of these subsystems. Ceph logging levels operate on scale of **`1`** to **`20`**, where **`1`** is terse and **`20`** is verbose.

Use a single value for the log level and memory level to set them both to the same value. For example, **`debug_osd = 5`** sets the debug level for the **`ceph-osd`** daemon to **`5`**.

To use different values for the output log level and the memory level, separate the values with a forward slash (**`/`**). For example, **`debug_mon = 1/5`** sets the debug log level for the **`ceph-mon`** daemon to **`1`** and its memory log level to **`5`**.

The Most Used Ceph Subsystems and Their Default Values

Subsystem	Log Level	Memory Level	Description
asok	1	5	The administration socket
auth	1	5	Authentication
client	0	5	Any application or library that uses **librados** to connect to the cluster
filestore	1	5	The FileStore OSD back end
journal	1	5	The OSD journal
mds	1	5	The Metadata Servers
monc	0	5	The Monitor client handles communication between most Ceph daemons and Monitors
mon	1	5	Monitors
ms	0	5	The messaging system between Ceph components
osd	0	5	The OSD Daemons
paxos	0	5	The algorithm that Monitors use to establish a consensus
rados	0	5	Reliable Autonomic Distributed Object Store, a core component of Ceph
rbd	0	5	The Ceph Block Devices
rgw	1	5	The Ceph Object Gateway

Example Log Outputs

The following examples show the type of messages in the logs when you increase the verbosity for the Monitors and OSDs.

Monitor Debug Settings

```
debug_ms = 5
debug_mon = 20
debug_paxos = 20
debug_auth = 20
```

Example Log Output of Monitor Debug Settings

```
2016-02-12 12:37:04.278761 7f45a9afc700 10 mon.cephn2@0(leader).osd e322
e322: 2 osds: 2 up, 2 in
2016-02-12 12:37:04.278792 7f45a9afc700 10 mon.cephn2@0(leader).osd e322
min_last_epoch_clean 322
2016-02-12 12:37:04.278795 7f45a9afc700 10 mon.cephn2@0(leader).log
v1010106 log
2016-02-12 12:37:04.278799 7f45a9afc700 10 mon.cephn2@0(leader).auth v2877
auth
2016-02-12 12:37:04.278811 7f45a9afc700 20 mon.cephn2@0(leader) e1
sync_trim_providers
2016-02-12 12:37:09.278914 7f45a9afc700 11 mon.cephn2@0(leader) e1 tick
2016-02-12 12:37:09.278949 7f45a9afc700 10 mon.cephn2@0(leader).pg v8126
v8126: 64 pgs: 64 active+clean; 60168 kB data, 172 MB used, 20285 MB /
20457 MB avail
2016-02-12 12:37:09.278975 7f45a9afc700 10
mon.cephn2@0(leader).paxosservice(pgmap 7511..8126) maybe_trim trim_to
7626 would only trim 115 < paxos_service_trim_min 250
2016-02-12 12:37:09.278982 7f45a9afc700 10 mon.cephn2@0(leader).osd e322
e322: 2 osds: 2 up, 2 in
2016-02-12 12:37:09.278989 7f45a9afc700  5
mon.cephn2@0(leader).paxos(paxos active c 1028850..1029466) is_readable =
1 - now=2016-02-12 12:37:09.278990 lease_expire=0.000000 has v0 lc 1029466
....
2016-02-12 12:59:18.769963 7f45a92fb700  1 -- 192.168.0.112:6789/0 <==
osd.1 192.168.0.114:6800/2801 5724 ==== pg_stats(0 pgs tid 3045 v 0) v1
==== 124+0+0 (2380105412 0 0) 0x5d96300 con 0x4d5bf40
2016-02-12 12:59:18.770053 7f45a92fb700  1 -- 192.168.0.112:6789/0 -->
192.168.0.114:6800/2801 -- pg_stats_ack(0 pgs tid 3045) v1 -- ?+0
0x550ae00 con 0x4d5bf40
2016-02-12 12:59:32.916397 7f45a9afc700  0
mon.cephn2@0(leader).data_health(1) update_stats avail 53% total 1951 MB,
used 780 MB, avail 1053 MB
....
2016-02-12 13:01:05.256263 7f45a92fb700  1 -- 192.168.0.112:6789/0 -->
192.168.0.113:6800/2410 -- mon_subscribe_ack(300s) v1 -- ?+0 0x4f283c0 con
0x4d5b440
```

OSD Debug Settings

```
debug_ms = 5
debug_osd = 20
debug_filestore = 20
debug_journal = 20
```

Example Log Output of OSD Debug Settings

```
2016-02-12 11:27:53.869151 7f5d55d84700  1 -- 192.168.17.3:0/2410 -->
192.168.17.4:6801/2801 -- osd_ping(ping e322 stamp 2016-02-12
11:27:53.869147) v2 -- ?+0 0x63baa00 con 0x578dee0
2016-02-12 11:27:53.869214 7f5d55d84700  1 -- 192.168.17.3:0/2410 -->
192.168.0.114:6801/2801 -- osd_ping(ping e322 stamp 2016-02-12
11:27:53.869147) v2 -- ?+0 0x638f200 con 0x578e040
2016-02-12 11:27:53.870215 7f5d6359f700  1 -- 192.168.17.3:0/2410 <==
osd.1 192.168.0.114:6801/2801 109210 ==== osd_ping(ping_reply e322 stamp
2016-02-12 11:27:53.869147) v2 ==== 47+0+0 (261193640 0 0) 0x63c1a00 con
```

```
0x578e040
2016-02-12 11:27:53.870698 7f5d6359f700  1 -- 192.168.17.3:0/2410 <==
osd.1 192.168.17.4:6801/2801 109210 ==== osd_ping(ping_reply e322 stamp
2016-02-12 11:27:53.869147) v2 ==== 47+0+0 (261193640 0 0) 0x6313200 con
0x578dee0
....
2016-02-12 11:28:10.432313 7f5d6e71f700  5 osd.0 322 tick
2016-02-12 11:28:10.432375 7f5d6e71f700 20 osd.0 322 scrub_random_backoff
lost coin flip, randomly backing off
2016-02-12 11:28:10.432381 7f5d6e71f700 10 osd.0 322 do_waiters -- start
2016-02-12 11:28:10.432383 7f5d6e71f700 10 osd.0 322 do_waiters -- finish
```

See Also

- Section 2.2, “Configuring Logging at Runtime”
- Section 2.3, “Configuring Logging in the Ceph Configuration File”

2.2. CONFIGURING LOGGING AT RUNTIME

To activate the Ceph debugging output, **`dout()`**, at runtime:

```
ceph tell <type>.<id> injectargs --debug-<subsystem> <value> [--<name>
<value>]
```

Replace:

- **`<type>`** with the type of Ceph daemons (**`osd`**, **`mon`**, or **`mds`**)
- **`<id>`** with a specific ID of the Ceph daemon. Alternatively, use **`*`** to apply the runtime setting to all daemons of a particular type.
- **`<subsystem>`** with a specific subsystem. See Section 2.1, “Ceph Subsystems” for details.
- **`<value>`** with a number from **`1`** to **`20`**, where **`1`** is terse and **`20`** is verbose

For example, to set the log level for the OSD subsystem on the OSD named **`osd.0`** to 0 and the memory level to 5:

```
# ceph tell osd.0 injectargs --debug-osd 0/5
```

To see the configuration settings at runtime:

1. Log in to the host with a running Ceph daemon, for example **`ceph-osd`** or **`ceph-mon`**.
2. Display the configuration:

   ```
   ceph daemon <name> config show | less
   ```

   Specify the name of the Ceph daemon, for example:

   ```
   # ceph daemon osd.0 config show | less
   ```

See Also

- Section 2.3, “Configuring Logging in the Ceph Configuration File”
- The *Logging Configuration Reference* chapter in the *Configuration Guide* for Red Hat Ceph Storage 3

2.3. CONFIGURING LOGGING IN THE CEPH CONFIGURATION FILE

To activate Ceph debugging output, **`dout()`** at boot time, add the debugging settings to the Ceph configuration file.

- For subsystems common to each daemon, add the settings under the **`[global]`** section.
- For subsystems for particular daemons, add the settings under a daemon section, such as **`[mon]`**, **`[osd]`**, or **`[mds]`**.

For example:

```
[global]
        debug_ms = 1/5

[mon]
        debug_mon = 20
        debug_paxos = 1/5
        debug_auth = 2

[osd]
        debug_osd = 1/5
        debug_filestore = 1/5
        debug_journal = 1
        debug_monc = 5/20

[mds]
        debug_mds = 1
```

See Also

- Section 2.1, “Ceph Subsystems”
- Section 2.2, “Configuring Logging at Runtime”
- The *Logging Configuration Reference* chapter in the *Configuration Guide* for Red Hat Ceph Storage 3

2.4. ACCELERATING LOG ROTATION

Increasing debugging level for Ceph components might generate a huge amount of data. If you have almost full disks, you can accelerate log rotation by modifying the Ceph log rotation file at **`/etc/logrotate.d/ceph`**. The Cron job scheduler uses this file to schedule log rotation.

Procedure: Accelerating Log Rotation

1. Add the size setting after the rotation frequency to the log rotation file:

   ```
   rotate 7
   weekly
   ```

```
size <size>
compress
sharedscripts
```

For example, to rotate a log file when it reaches 500 MB:

```
rotate 7
weekly
size 500 MB
compress
sharedscripts
size 500M
```

2. Open the **`crontab`** editor:

   ```
   $ crontab -e
   ```

3. Add an entry to check the **`/etc/logrotate.d/ceph`** file. For example, to instruct Cron to check **`/etc/logrotate.d/ceph`** every 30 minutes:

   ```
   30 * * * * /usr/sbin/logrotate /etc/logrotate.d/ceph >/dev/null 2>&1
   ```

See Also

- The Scheduling a Recurring Job Using Cron section in the System Administrator’s Guide for Red Hat Enterprise Linux 7.

CHAPTER 3. TROUBLESHOOTING NETWORKING ISSUES

This chapter lists basic troubleshooting procedures connected with networking and Network Time Protocol (NTP).

3.1. BASIC NETWORKING TROUBLESHOOTING

Red Hat Ceph Storage depends heavily on a reliable network connection. Ceph nodes use the network for communicating with each other. Networking issues can cause many problems with OSDs, such as flapping OSD, or OSD incorrectly reported as `down`. Networking issues can also cause Monitor `clock skew` errors. In addition, packet loss, high latency, or limited bandwidth can impact the cluster performance and stability.

Procedure: Basic Networking Troubleshooting

1. Verify that the `cluster_network` and `public_network` parameters in the Ceph configuration file include correct values.

2. Verify that the network interfaces are up. See the Basic Network troubleshooting solution on the Customer Portal for details.

3. Verify that the Ceph nodes are able to reach each other using their short host names.

4. If you use a firewall, ensure that Ceph nodes are able to reach other on their appropriate ports. See the *Configuring Firewall* section in the Red Hat Ceph Storage 3 *Installation Guide for Red Hat Enterprise Linux* or *Installation Guide for Ubuntu*.

5. Verify that there are no errors on the interface counters and that the network connectivity between hosts has expected latency and no packet loss. See the What is the "ethtool" command and how can I use it to obtain information about my network devices and interfaces and RHEL network interface dropping packets solutions on the Customer Portal for details.

6. For performance issues, in addition to the latency checks, also use the `iperf` utility to verify the network bandwidth between all nodes of the cluster. For details, see the What are the performance benchmarking tools available for Red Hat Ceph Storage? solution on the Customer Portal.

7. Ensure that all hosts have equal speed network interconnects, otherwise slow attached nodes could slow down the faster connected ones. Also, ensure that the inter switch links can handle the aggregated bandwidth of the attached nodes.

See Also

- The Networking Guide for Red Hat Enterprise Linux 7

- Knowledgebase articles and solutions related to troubleshooting networking issues on the Customer Portal

3.2. BASIC NTP TROUBLESHOOTING

This section includes basic NTP troubleshooting steps.

Procedure: Basic NTP Troubleshooting

1. Verify that the `ntpd` daemon is running on the Monitor hosts:

```
# systemctl status ntpd
```

2. If **ntpd** is not running, enable and start it:

   ```
   # systemctl enable ntpd
   # systemctl start ntpd
   ```

3. Ensure that **ntpd** is synchronizing the clocks correctly:

   ```
   $ ntpq -p
   ```

4. See the How to troubleshoot NTP issues solution on the Red Hat Customer Portal for advanced NTP troubleshooting steps.

See Also

- Section 4.1.2, “Clock Skew”

CHAPTER 4. TROUBLESHOOTING MONITORS

This chapter contains information on how to fix the most common errors related to the Ceph Monitors.

Before You Start

- Verify your network connection. See Chapter 3, *Troubleshooting Networking Issues* for details.

4.1. THE MOST COMMON ERROR MESSAGES RELATED TO MONITORS

The following tables list the most common error messages that are returned by the **`ceph health detail`** command, or included in the Ceph logs. The tables provide links to corresponding sections that explain the errors and point to specific procedures to fix the problems.

Table 4.1. Error Messages Related to Monitors

Error message	See
`HEALTH_WARN`	
`mon.X is down (out of quorum)`	Section 4.1.1, “A Monitor Is Out of Quorum”
`clock skew`	Section 4.1.2, “Clock Skew”
`store is getting too big!`	Section 4.1.3, “The Monitor Store is Getting Too Big”

Table 4.2. Common Error Messages in Ceph Logs Related to Monitors

Error message	Log file	See
`clock skew`	Main cluster log	Section 4.1.2, “Clock Skew”
`clocks not synchronized`	Main cluster log	Section 4.1.2, “Clock Skew”
`Corruption: error in middle of record`	Monitor log	Section 4.1.1, “A Monitor Is Out of Quorum” Section 4.3, “Recovering the Monitor Store”
`Corruption: 1 missing files`	Monitor log	Section 4.1.1, “A Monitor Is Out of Quorum” Section 4.3, “Recovering the Monitor Store”
`Caught signal (Bus error)`	Monitor log	Section 4.1.1, “A Monitor Is Out of Quorum”

4.1.1. A Monitor Is Out of Quorum

One or more Monitors are marked as **down** but the other Monitors are still able to form a quorum. In addition, the **ceph health detail** command returns an error message similar to the following one:

```
HEALTH_WARN 1 mons down, quorum 1,2 mon.b,mon.c
mon.a (rank 0) addr 127.0.0.1:6789/0 is down (out of quorum)
```

What This Means

Ceph marks a Monitor as **down** due to various reasons.

If the **ceph-mon** daemon is not running, it might have a corrupted store or some other error is preventing the daemon from starting. Also, the **/var/** partition might be full. As a consequence, **ceph-mon** is not able to perform any operations to the store located by default at **/var/lib/ceph/mon-<short-host-name>/store.db** and terminates.

If the **ceph-mon** daemon is running but the Monitor is out of quorum and marked as **down**, the cause of the problem depends on the Monitor state:

- If the Monitor is in the *probing* state longer than expected, it cannot find the other Monitors. This problem can be caused by networking issues, or the Monitor can have an outdated Monitor map (**monmap**) and be trying to reach the other Monitors on incorrect IP addresses. Alternatively, if the **monmap** is up-to-date, Monitor's clock might not be synchronized.

- If the Monitor is in the *electing* state longer than expected, the Monitor's clock might not be synchronized.

- If the Monitor changes its state from *synchronizing* to *electing* and back, the cluster state is advancing. This means that it is generating new maps faster than the synchronization process can handle.

- If the Monitor marks itself as the *leader* or a *peon*, then it believes to be in a quorum, while the remaining cluster is sure that it is not. This problem can be caused by failed clock synchronization.

To Troubleshoot This Problem

1. Verify that the **ceph-mon** daemon is running. If not, start it:

   ```
   systemctl status ceph-mon@<host-name>
   systemctl start ceph-mon@<host-name>
   ```

   Replace **<host-name>** with the short name of the host where the daemon is running. Use the **hostname -s** command when unsure.

2. If you are not able to start **ceph-mon**, follow the steps in The **ceph-mon** Daemon Cannot Start.

3. If you are able to start the **ceph-mon** daemon but is is marked as **down**, follow the steps in The **ceph-mon** Daemon Is Running, but Still Marked as **down**.

The ceph-mon Daemon Cannot Start

1. Check the corresponding Monitor log, by default located at **/var/log/ceph/ceph-mon.<host-name>.log**.

2. If the log contains error messages similar to the following ones, the Monitor might have a corrupted store.

   ```
   Corruption: error in middle of record
   Corruption: 1 missing files; e.g.:
   /var/lib/ceph/mon/mon.0/store.db/1234567.ldb
   ```

   To fix this problem, replace the Monitor. See Section 4.4, “Replacing a Failed Monitor”.

3. If the log contains an error message similar to the following one, the **/var/** partition might be full. Delete any unnecessary data from **/var/**.

   ```
   Caught signal (Bus error)
   ```

   IMPORTANT

   Do not delete any data from the Monitor directory manually. Instead, use the **ceph-monstore-tool** to compact it. See Section 4.5, “Compacting the Monitor Store” for details.

4. If you see any other error messages, open a support ticket. See Chapter 8, *Contacting Red Hat Support Service* for details.

The ceph-mon Daemon Is Running, but Still Marked as down

1. From the Monitor host that is out of the quorum, use the **mon_status** command to check its state:

   ```
   ceph daemon <id> mon_status
   ```

   Replace **<id>** with the ID of the Monitor, for example:

   ```
   # ceph daemon mon.a mon_status
   ```

2. If the status is *probing*, verify the locations of the other Monitors in the **mon_status** output.

   a. If the addresses are incorrect, the Monitor has incorrect Monitor map (**monmap**). To fix this problem, see Section 4.2, “Injecting a Monitor Map”.

   b. If the addresses are correct, verify that the Monitor clocks are synchronized. See Section 4.1.2, “Clock Skew” for details. In addition, troubleshoot any networking issues, see Chapter 3, *Troubleshooting Networking Issues*.

3. If the status is *electing*, verify that the Monitor clocks are synchronized. See Section 4.1.2, “Clock Skew”.

4. If the status changes from *electing* to *synchronizing*, open a support ticket. See Chapter 8, *Contacting Red Hat Support Service* for details.

5. If the Monitor is the *leader* or a *peon*, verify that the Monitor clocks are synchronized. See Section 4.1.2, “Clock Skew”. Open a support ticket if synchronizing the clocks does not solve the problem. See Chapter 8, *Contacting Red Hat Support Service* for details.

See Also

- Section 4.1.4, “Understanding Monitor Status”
- The *Starting, Stopping, Restarting a Daemon by Instances* section in the *Administration Guide* for Red Hat Ceph Storage 3
- The *Using the Administration Socket* section in the *Administration Guide* for Red Hat Ceph Storage 3

4.1.2. Clock Skew

A Ceph Monitor is out of quorum, and the **`ceph health detail`** command output contains error messages similar to these:

```
mon.a (rank 0) addr 127.0.0.1:6789/0 is down (out of quorum)
mon.a addr 127.0.0.1:6789/0 clock skew 0.08235s > max 0.05s (latency
0.0045s)
```

In addition, Ceph logs contain error messages similar to these:

```
2015-06-04 07:28:32.035795 7f806062e700 0 log [WRN] : mon.a
127.0.0.1:6789/0 clock skew 0.14s > max 0.05s
2015-06-04 04:31:25.773235 7f4997663700 0 log [WRN] : message from mon.1
was stamped 0.186257s in the future, clocks not synchronized
```

What This Means

The **`clock skew`** error message indicates that Monitors' clocks are not synchronized. Clock synchronization is important because Monitors depend on time precision and behave unpredictably if their clocks are not synchronized.

The **`mon_clock_drift_allowed`** parameter determines what disparity between the clocks is tolerated. By default, this parameter is set to 0.05 seconds.

IMPORTANT

Do not change the default value of **`mon_clock_drift_allowed`** without previous testing. Changing this value might affect the stability of the Monitors and the Ceph Storage Cluster in general.

Possible causes of the **`clock skew`** error include network problems or problems with Network Time Protocol (NTP) synchronization if that is configured. In addition, time synchronization does not work properly on Monitors deployed on virtual machines.

To Troubleshoot This Problem

1. Verify that your network works correctly. For details, see Chapter 3, *Troubleshooting Networking Issues*. In particular, troubleshoot any problems with NTP clients if you use NTP. See Section 3.2, “Basic NTP Troubleshooting” for more information.
2. If you use a remote NTP server, consider deploying your own NTP server on your network. For details, see the *Configuring NTP Using ntpd* chapter in the *System Administrator’s Guide* for Red Hat Enterprise Linux 7.

3. If you do not use an NTP client, set one up. For details, see the *Configuring the Network Time Protocol for Red Hat Ceph Storage* section in the Red Hat Ceph Storage 3 *Installation Guide for Red Hat Enterprise Linux* or *Ubuntu*.

4. If you use virtual machines for hosting the Monitors, move them to bare metal hosts. Using virtual machines for hosting Monitors is not supported. For details, see the Red Hat Ceph Storage: Supported configurations article on the Red Hat Customer Portal.

NOTE

Ceph evaluates time synchronization every five minutes only so there will be a delay between fixing the problem and clearing the **`clock skew`** messages.

See Also

- Section 4.1.4, “Understanding Monitor Status”
- Section 4.1.1, “A Monitor Is Out of Quorum”

4.1.3. The Monitor Store is Getting Too Big

The **`ceph health`** command returns an error message similar to the following one:

```
mon.ceph1 store is getting too big! 48031 MB >= 15360 MB -- 62% avail
```

What This Means

Ceph Monitors store is in fact a LevelDB database that stores entries as key–values pairs. The database includes a cluster map and is located by default at **`/var/lib/ceph/mon/<cluster-name>-<short-host-name>/store.db`**.

Querying a large Monitor store can take time. As a consequence, the Monitor can be delayed in responding to client queries.

In addition, if the **`/var/`** partition is full, the Monitor cannot perform any write operations to the store and terminates. See Section 4.1.1, “A Monitor Is Out of Quorum” for details on troubleshooting this issue.

To Troubleshoot This Problem

1. Check the size of the database:

   ```
   du -sch /var/lib/ceph/mon/<cluster-name>-<short-host-name>/store.db
   ```

   Specify the name of the cluster and the short host name of the host where the **`ceph-mon`** is running, for example:

   ```
   # du -sch /var/lib/ceph/mon/ceph-host1/store.db
   47G     /var/lib/ceph/mon/ceph-ceph1/store.db/
   47G     total
   ```

2. Compact the Monitor store. For details, see Section 4.5, “Compacting the Monitor Store”.

See Also

- Section 4.1.1, “A Monitor Is Out of Quorum”

4.1.4. Understanding Monitor Status

The **`mon_status`** command returns information about a Monitor, such as:

- State
- Rank
- Elections epoch
- Monitor map (**`monmap`**)

If Monitors are able to form a quorum, use **`mon_status`** with the **`ceph`** command-line utility.

If Monitors are not able to form a quorum, but the **`ceph-mon`** daemon is running, use the administration socket to execute **`mon_status`**. For details, see the *Using the Administration Socket* section in the *Administration Guide* for Red Hat Ceph Storage 3.

An example output of `mon_status`

```
{
    "name": "mon.3",
    "rank": 2,
    "state": "peon",
    "election_epoch": 96,
    "quorum": [
        1,
        2
    ],
    "outside_quorum": [],
    "extra_probe_peers": [],
    "sync_provider": [],
    "monmap": {
        "epoch": 1,
        "fsid": "d5552d32-9d1d-436c-8db1-ab5fc2c63cd0",
        "modified": "0.000000",
        "created": "0.000000",
        "mons": [
            {
                "rank": 0,
                "name": "mon.1",
                "addr": "172.25.1.10:6789\/0"
            },
            {
                "rank": 1,
                "name": "mon.2",
                "addr": "172.25.1.12:6789\/0"
            },
            {
                "rank": 2,
                "name": "mon.3",
                "addr": "172.25.1.13:6789\/0"
            }
        ]
    }
}
```

Monitor States

Leader

During the electing phase, Monitors are electing a leader. The leader is the Monitor with the highest rank, that is the rank with the lowest value. In the example above, the leader is **mon.1**.

Peon

Peons are the Monitors in the quorum that are not leaders. If the leader fails, the peon with the highest rank becomes a new leader.

Probing

A Monitor is in the probing state if it is looking for other Monitors. For example after you start the Monitors, they are *probing* until they find enough Monitors specified in the Monitor map (**monmap**) to form a quorum.

Electing

A Monitor is in the electing state if it is in the process of electing the leader. Usually, this status changes quickly.

Synchronizing

A Monitor is in the synchronizing state if it is synchronizing with the other Monitors to join the quorum. The smaller the Monitor store it, the faster the synchronization process. Therefore, if you have a large store, synchronization takes longer time.

4.2. INJECTING A MONITOR MAP

If a Monitor has an outdated or corrupted Monitor map (**monmap**), it cannot join a quorum because it is trying to reach the other Monitors on incorrect IP addresses.

The safest way to fix this problem is to obtain and inject the actual Monitor map from other Monitors. Note that this action overwrites the existing Monitor map kept by the Monitor.

This procedure shows how to inject the Monitor map when the other Monitors are able to form a quorum, or when at least one Monitor has a correct Monitor map. If all Monitors have corrupted store and therefore also the Monitor map, see Section 4.3, “Recovering the Monitor Store”.

Procedure: Injecting a Monitor Map

1. If the remaining Monitors are able to form a quorum, get the Monitor map by using the **ceph mon getmap** command:

   ```
   # ceph mon getmap -o /tmp/monmap
   ```

2. If the remaining Monitors are not able to form the quorum and you have at least one Monitor with a correct Monitor map, copy it from that Monitor:

   a. Stop the Monitor which you want to copy the Monitor map from:

   ```
   systemctl stop ceph-mon@<host-name>
   ```

   For example, to stop the Monitor running on a host with the **host1** short host name:

   ```
   # systemctl stop ceph-mon@host1
   ```

   b. Copy the Monitor map:

```
ceph-mon -i <id> --extract-monmap /tmp/monmap
```

Replace **<id>** with the ID of the Monitor which you want to copy the Monitor map from, for example:

```
# ceph-mon -i mon.a  --extract-monmap /tmp/monmap
```

3. Stop the Monitor with the corrupted or outdated Monitor map:

```
systemctl stop ceph-mon@<host-name>
```

For example, to stop a Monitor running on a host with the **host2** short host name:

```
# systemctl stop ceph-mon@host2
```

4. Inject the Monitor map:

```
ceph-mon -i <id> --inject-monmap /tmp/monmap
```

Replace **<id>** with the ID of the Monitor with the corrupted or outdated Monitor map, for example:

```
# ceph-mon -i mon.c --inject-monmap /tmp/monmap
```

5. Start the Monitor, for example:

```
# systemctl start ceph-mon@host2
```

If you copied the Monitor map from another Monitor, start that Monitor, too, for example:

```
# systemctl start ceph-mon@host1
```

See Also

- Section 4.1.1, “A Monitor Is Out of Quorum”
- Section 4.3, “Recovering the Monitor Store”

4.3. RECOVERING THE MONITOR STORE

Ceph Monitors store the cluster map in a key–value store such as LevelDB. If the store is corrupted on a Monitor, the Monitor terminates unexpectedly and fails to start again. The Ceph logs might include the following errors:

```
Corruption: error in middle of record
Corruption: 1 missing files; e.g.:
/var/lib/ceph/mon/mon.0/store.db/1234567.ldb
```

Production clusters must use at least three Monitors so that if one fails, it can be replaced with another one. However, under certain circumstances, all Monitors can have corrupted stores. For example, when the Monitor nodes have incorrectly configured disk or file system settings, a power outage can corrupt the underlying file system.

If the store is corrupted on all Monitors, you can recover it with information stored on the OSD nodes by using utilities called **`ceph-monstore-tool`** and **`ceph-objectstore-tool`**.

IMPORTANT

This procedure cannot recover the following information:

- Metadata Daemon Server (MDS) keyrings and maps
- Placement Group settings:
  - **`full ratio`** set by using the **`ceph pg set_full_ratio`** command
  - **`nearfull ratio`** set by using the **`ceph pg set_nearfull_ratio`** command

Before You Start

- Ensure that you have the **`rsync`** utility and the **`ceph-test`** package installed.

Procedure: Recovering the Monitor Store

Use the following commands from the Monitor node with the corrupted store.

1. Collect the cluster map from all OSD nodes:

```
ms=<directory>
mkdir $ms

for host in $host_list; do
  rsync -avz "$ms" root@$host:"$ms"; rm -rf "$ms"
  ssh root@$host <<EOF
  for osd in  /var/lib/ceph/osd/ceph-*; do
    ceph-objectstore-tool --data-path \$osd --op update-mon-db --
mon-store-path $ms
  done
EOF
rsync -avz root@$host:$ms $ms; done
```

Replace **`<directory>`** with a temporary directory to store the collected cluster map, for example:

```
$ ms=/tmp/monstore/
$ mkdir $ms
$ for host in $host_list; do
  rsync -avz "$ms" root@$host:"$ms"; rm -rf "$ms"
  ssh root@$host <<EOF
  for osd in  /var/lib/ceph/osd/ceph-*; do
    ceph-objectstore-tool --data-path \$osd --op update-mon-db --
mon-store-path $ms
  done
EOF
rsync -avz root@$host:$ms $ms; done
```

2. Set appropriate capabilities:

```
ceph-authtool <keyring>  -n mon. --cap mon 'allow *'
ceph-authtool <keyring>  -n client.admin --cap mon 'allow *' --cap
osd 'allow *' --cap mds 'allow *'
```

Replace **`<keyring>`** with the path to the client administration keyring, for example:

```
$ ceph-authtool /etc/ceph/ceph.client.admin.keyring  -n mon. --cap
mon 'allow *'
$ ceph-authtool /etc/ceph/ceph.client.admin.keyring  -n client.admin
--cap mon 'allow *' --cap osd 'allow *' --cap mds 'allow *'
```

3. Rebuild the Monitor store from the collected map:

   ```
   ceph-monstore-tool <directory> rebuild -- --keyring <keyring>
   ```

   Replace **`<directory>`** with the temporary directory from the first step and **`<keyring>`** with the path to the client administration keyring, for example:

   ```
   $ ceph-monstore-tool /tmp/mon-store rebuild -- --keyring
   /etc/ceph/ceph.client.admin.keyring
   ```

   ### NOTE

   If you do not use the **`cephfx`** authentication, omit the **`--keyring`** option:

   ```
   $ ceph-monstore-tool /tmp/mon-store rebuild
   ```

4. Back up the corrupted store:

   ```
   mv /var/lib/ceph/mon/<mon-ID>/store.db \
      /var/lib/ceph/mon/<mon-ID>/store.db.corrupted
   ```

   Replace **`<mon-ID>`** with the Monitor ID, for example **`<mon.0>`**:

   ```
   # mv /var/lib/ceph/mon/mon.0/store.db \
        /var/lib/ceph/mon/mon.0/store.db.corrupted
   ```

5. Replace the corrupted store:

   ```
   mv /tmp/mon-store/store.db /var/lib/ceph/mon/<mon-ID>/store.db
   ```

   Replace **`<mon-ID>`** with the Monitor ID, for example **`<mon.0>`**:

   ```
   # mv /tmp/mon-store/store.db /var/lib/ceph/mon/mon.0/store.db
   ```

   Repeat this step for all Monitors with corrupted store.

6. Change the owner of the new store:

   ```
   chown -R ceph:ceph /var/lib/ceph/mon/<mon-ID>/store.db
   ```

Replace **<mon-ID>** with the Monitor ID, for example **<mon.0>**:

```
# chown -R ceph:ceph /var/lib/ceph/mon/mon.0/store.db
```

Repeat this step for all Monitors with corrupted store.

See also

- Section 4.4, “Replacing a Failed Monitor”

4.4. REPLACING A FAILED MONITOR

When a Monitor has a corrupted store, the recommended way to fix this problem is to replace the Monitor by using the Ansible automation application.

Before You Start

- Before removing a Monitor, ensure that the other Monitors are running and able to form a quorum.

Procedure: Replacing a Failed Monitor

1. From the Monitor host, remove the Monitor store by default located at **/var/lib/ceph/mon/<cluster-name>-<short-host-name>**:

   ```
   rm -rf /var/lib/ceph/mon/<cluster-name>-<short-host-name>
   ```

   Specify the short host name of the Monitor host and the cluster name. For example, to remove the Monitor store of a Monitor running on **host1** from a cluster called **remote**:

   ```
   # rm -rf /var/lib/ceph/mon/remote-host1
   ```

2. Remove the Monitor from the Monitor map (**monmap**):

   ```
   ceph mon remove <short-host-name> --cluster <cluster-name>
   ```

   Specify the short host name of the Monitor host and the cluster name. For example, to remove the Monitor running on **host1** from a cluster called **remote**:

   ```
   # ceph mon remove host1 --cluster remote
   ```

3. Troubleshoot and fix any problems related to the underlying file system or hardware of the Monitor host.

4. From the Ansible administration node, redeploy the Monitor by running the **ceph-ansible** playbook:

   ```
   $ /usr/share/ceph-ansible/ansible-playbook site.yml
   ```

See Also

- Section 4.1.1, “A Monitor Is Out of Quorum”
- The *Managing Cluster Size* chapter in the *Administration Guide* for Red Hat Ceph Storage 3

- The *Deploying Red Hat Ceph Storage* chapter in the Red Hat Ceph Storage 3 *Installation Guide for Red Hat Enterprise Linux*

4.5. COMPACTING THE MONITOR STORE

When the Monitor store has grown big in size, you can compact it:

- Dynamically by using the **ceph tell** command. See the Compacting the Monitor Store Dynamically procedure for details.

- Upon the start of the **ceph-mon** daemon. See the Compacting the Monitor Store at Startup procedure for details.

- By using the **ceph-monstore-tool** when the **ceph-mon** daemon is not running. Use this method when the previously mentioned methods fail to compact the Monitor store or when the Monitor is out of quorum and its log contains the **Caught signal (Bus error)** error message. See the Compacting the Monitor Store with **ceph-monstore-tool** procedure for details.

IMPORTANT

Monitor store size changes when the cluster is not in the **active+clean** state or during the rebalancing process. For this reason, compact the Monitor store when rebalancing is completed. Also, ensure that the placement groups are in the **active+clean** state.

Procedure: Compacting the Monitor Store Dynamically

To compact the Monitor store when the **ceph-mon** daemon is running:

```
ceph tell mon.<host-name> compact
```

Replace **<host-name>** with the short host name of the host where the **ceph-mon** is running. Use the **hostname -s** command when unsure.

```
# ceph tell mon.host1 compact
```

Procedure: Compacting the Monitor Store at Startup

1. Add the following parameter to the Ceph configuration under the **[mon]** section:

   ```
   [mon]
   mon_compact_on_start = true
   ```

2. Restart the **ceph-mon** daemon:

   ```
   systemctl restart ceph-mon@<host-name>
   ```

   Replace **<host-name>** with the short name of the host where the daemon is running. Use the **hostname -s** command when unsure.

   ```
   # systemctl restart ceph-mon@host1
   ```

3. Ensure that Monitors have formed a quorum:

```
# ceph mon stat
```

4. Repeat these steps on other Monitors if needed.

Procedure: Compacting Monitor Store with `ceph-monstore-tool`

NOTE

Before you start, ensure that you have the **`ceph-test`** package installed.

1. Verify that the **`ceph-mon`** daemon with the large store is not running. Stop the daemon if needed.

   ```
   systemctl status ceph-mon@<host-name>
   systemctl stop ceph-mon@<host-name>
   ```

   Replace **`<host-name>`** with the short name of the host where the daemon is running. Use the **`hostname -s`** command when unsure.

   ```
   # systemctl status ceph-mon@host1
   # systemctl stop ceph-mon@host1
   ```

2. Compact the Monitor store:

   ```
   ceph-monstore-tool /var/lib/ceph/mon/mon.<host-name> compact
   ```

   Replace **`<host-name>`** with a short host name of the Monitor host.

   ```
   # ceph-monstore-tool /var/lib/ceph/mon/mon.node1 compact
   ```

3. Start **`ceph-mon`** again:

   ```
   systemctl start ceph-mon@<host-name>
   ```

   For example:

   ```
   # systemctl start ceph-mon@host1
   ```

See Also

- Section 4.1.3, “The Monitor Store is Getting Too Big”
- Section 4.1.1, “A Monitor Is Out of Quorum”

4.6. OPENING PORTS FOR CEPH MANAGER

The **`ceph-mgr`** daemons receive placement group information from OSDs on the same range of ports as the **`ceph-osd`** daemons. If these ports are not open, a cluster will devolve from **`HEALTH_OK`** to **`HEALTH_WARN`** and will indicate that PGs are **`unknown`** with a percentage count of the PGs unknown.

To resolve this situation, for each host running **`ceph-mgr`** daemons, open ports **`6800:7300`**. For example:

```
[root@ceph-mgr] # firewall-cmd --add-port 6800:7300/tcp
[root@ceph-mgr] # firewall-cmd --add-port 6800:7300/tcp --permanent
```

Then, restart the **`ceph-mgr`** daemons.

CHAPTER 5. TROUBLESHOOTING OSDS

This chapter contains information on how to fix the most common errors related to Ceph OSDs.

Before You Start

- Verify your network connection. See Chapter 3, *Troubleshooting Networking Issues* for details.
- Verify that Monitors have a quorum by using the **`ceph health`** command. If the command returns a health status (**`HEALTH_OK`**, **`HEALTH_WARN`**, or **`HEALTH_ERR`**), the Monitors are able to form a quorum. If not, address any Monitor problems first. See Chapter 4, *Troubleshooting Monitors* for details. For details about **`ceph health`** see Section 1.2, "Understanding the Output of the **`ceph health`** Command".
- Optionally, stop the rebalancing process to save time and resources. See Section 5.2, "Stopping and Starting Rebalancing" for details.

5.1. THE MOST COMMON ERROR MESSAGES RELATED TO OSDS

The following tables list the most common error messages that are returned by the **`ceph health detail`** command, or included in the Ceph logs. The tables provide links to corresponding sections that explain the errors and point to specific procedures to fix the problems.

Table 5.1. Error Messages Related to OSDs

Error message	See
`HEALTH_ERR`	
`full osds`	Section 5.1.1, "Full OSDs"
`HEALTH_WARN`	
`nearfull osds`	Section 5.1.2, "Nearfull OSDs"
`osds are down`	Section 5.1.3, "One or More OSDs Are Down" Section 5.1.4, "Flapping OSDs"
`requests are blocked`	Section 5.1.5, "Slow Requests, and Requests are Blocked"
`slow requests`	Section 5.1.5, "Slow Requests, and Requests are Blocked"

Table 5.2. Common Error Messages in Ceph Logs Related to OSDs

Error message	Log file	See
`heartbeat_check: no reply from osd.X`	Main cluster log	Section 5.1.4, "Flapping OSDs"

Error message	Log file	See
`wrongly marked me down`	Main cluster log	Section 5.1.4, “Flapping OSDs”
`osds have slow requests`	Main cluster log	Section 5.1.5, “Slow Requests, and Requests are Blocked”
`FAILED assert(!m_filestore_fail_eio)`	OSD log	Section 5.1.3, “One or More OSDs Are Down”
`FAILED assert(0 == "hit suicide timeout")`	OSD log	Section 5.1.3, “One or More OSDs Are Down”

5.1.1. Full OSDs

The **`ceph health detail`** command returns an error message similar to the following one:

```
HEALTH_ERR 1 full osds
osd.3 is full at 95%
```

What This Means

Ceph prevents clients from performing I/O operations on full OSD nodes to avoid losing data. It returns the **`HEALTH_ERR full osds`** message when the cluster reaches the capacity set by the **`mon_osd_full_ratio`** parameter. By default, this parameter is set to **`0.95`** which means 95% of the cluster capacity.

To Troubleshoot This Problem

Determine how many percent of raw storage (**`%RAW USED`**) is used:

```
# ceph df
```

If **`%RAW USED`** is above 70-75%, you can:

- Delete unnecessary data. This is a short-term solution to avoid production downtime. See Section 5.6, “Deleting Data from a Full Cluster” for details.

- Scale the cluster by adding a new OSD node. This is a long-term solution recommended by Red Hat. For details, see the *Adding and Removing OSD Nodes* chapter in the *Administration Guide* for Red Hat Ceph Storage 3.

See Also

- Section 5.1.2, “Nearfull OSDs”

5.1.2. Nearfull OSDs

The **`ceph health detail`** command returns an error message similar to the following one:

```
HEALTH_WARN 1 nearfull osds
osd.2 is near full at 85%
```

What This Means

Ceph returns the **`nearfull osds`** message when the cluster reaches the capacity set by the **`mon osd nearfull ratio defaults`** parameter. By default, this parameter is set to **`0.85`** which means 85% of the cluster capacity.

Ceph distributes data based on the CRUSH hierarchy in the best possible way but it cannot guarantee equal distribution. The main causes of the uneven data distribution and the **`nearfull osds`** messages are:

- The OSDs are not balanced among the OSD nodes in the cluster. That is, some OSD nodes host significantly more OSDs than others, or the weight of some OSDs in the CRUSH map is not adequate to their capacity.

- The Placement Group (PG) count is not proper as per the number of the OSDs, use case, target PGs per OSD, and OSD utilization.

- The cluster uses inappropriate CRUSH tunables.

- The back-end storage for OSDs is almost full.

To Troubleshoot This Problem:

1. Verify that the PG count is sufficient and increase it if needed. See Section 6.5, “Increasing the PG Count” for details.

2. Verify that you use CRUSH tunables optimal to the cluster version and adjust them if not. For details, see the *CRUSH Tunables* section in the *Storage Strategies* guide for Red Hat Ceph Storage 3 and the How can I test the impact CRUSH map tunable modifications will have on my PG distribution across OSDs in Red Hat Ceph Storage? solution on the Red Hat Customer Portal.

3. Change the weight of OSDs by utilization. See the *Set an OSD’s Weight by Utilization* section in the *Storage Strategies* guide for Red Hat Ceph Storage 3.

4. Determine how much space is left on the disks used by OSDs.

   a. To view how much space OSDs use in general:

   ```
   # ceph osd df
   ```

   b. To view how much space OSDs use on particular nodes. Use the following command from the node containing **`nearful`** OSDs:

   ```
   $ df
   ```

   c. If needed, add a new OSD node. See the *Adding and Removing OSD Nodes* chapter in the *Administration Guide* for Red Hat Ceph Storage 3.

See Also

- Section 5.1.1, “Full OSDs”

5.1.3. One or More OSDs Are Down

The **ceph health** command returns an error similar to the following one:

```
HEALTH_WARN 1/3 in osds are down
```

What This Means
One of the **ceph-osd** processes is unavailable due to a possible service failure or problems with communication with other OSDs. As a consequence, the surviving **ceph-osd** daemons reported this failure to the Monitors.

If the **ceph-osd** daemon is not running, the underlying OSD drive or file system is either corrupted, or some other error, such as a missing keyring, is preventing the daemon from starting.

In most cases, networking issues cause the situation when the **ceph-osd** daemon is running but still marked as **down**.

To Troubleshoot This Problem

1. Determine which OSD is **down**:

   ```
   # ceph health detail
   HEALTH_WARN 1/3 in osds are down
   osd.0 is down since epoch 23, last address
   192.168.106.220:6800/11080
   ```

2. Try to restart the **ceph-osd** daemon:

   ```
   systemctl restart ceph-osd@<OSD-number>
   ```

   Replace **<OSD-number>** with the ID of the OSD that is **down**, for example:

   ```
   # systemctl restart ceph-osd@0
   ```

   a. If you are not able start **ceph-osd**, follow the steps in The **ceph-osd** daemon cannot start.

   b. If you are able to start the **ceph-osd** daemon but it is marked as **down**, follow the steps in The **ceph-osd** daemon is running but still marked as **down**.

The ceph-osd daemon cannot start

1. If you have a node containing a number of OSDs (generally, more that twelve), verify that the default maximum number of threads (PID count) is sufficient. See Section 5.5, “Increasing the PID count” for details.

2. Verify that the OSD data and journal partitions are mounted properly:

   ```
   # ceph-disk list
   ...
   /dev/vdb :
    /dev/vdb1 ceph data, prepared
    /dev/vdb2 ceph journal
   /dev/vdc :
    /dev/vdc1 ceph data, active, cluster ceph, osd.1, journal /dev/vdc2
    /dev/vdc2 ceph journal, for /dev/vdc1
   ```

```
/dev/sdd1 :
 /dev/sdd1 ceph data, unprepared
 /dev/sdd2 ceph journal
```

A partition is mounted if **`ceph-disk`** marks it as **`active`**. If a partition is **`prepared`**, mount it. See Section 5.3, “Mounting the OSD Data Partition” for details. If a partition is **`unprepared`**, you must prepare it first before mounting. See the *Preparing the OSD Data and Journal Drives* section in the *Administration Guide* Red Hat Ceph Storage 3.

3. If you got the **`ERROR: missing keyring, cannot use cephx for authentication`** error message, the OSD is a missing keyring. See the *Keyring Management* section in the *Administration Guide* for Red Hat Ceph Storage 3.

4. If you got the **`ERROR: unable to open OSD superblock on /var/lib/ceph/osd/ceph-1`** error message, the **`ceph-osd`** daemon cannot read the underlying file system. See the following steps for instructions on how to troubleshoot and fix this error.

NOTE

If this error message is returned during boot time of the OSD host, open a support ticket as this might indicate a known issue tracked in the Red Hat Bugzilla 1439210. See Chapter 8, *Contacting Red Hat Support Service* for details.

5. Check the corresponding log file to determine the cause of the failure. By default, Ceph stores log files in the **`/var/log/ceph/`** directory.

   a. An **`EIO`** error message similar to the following one indicates a failure of the underlying disk:

   ```
   FAILED assert(!m_filestore_fail_eio || r != -5)
   ```

   To fix this problem replace the underlying OSD disk. See Section 5.4, “Replacing an OSD Drive” for details.

   b. If the log includes any other **`FAILED assert`** errors, such as the following one, open a support ticket. See Chapter 8, *Contacting Red Hat Support Service* for details.

   ```
   FAILED assert(0 == "hit suicide timeout")
   ```

6. Check the **`dmesg`** output for the errors with the underlying file system or disk:

   ```
   $ dmesg
   ```

   a. The **`error -5`** error message similar to the following one indicates corruption of the underlying XFS file system. For details on how to fix this problem, see the What is the meaning of "xfs_log_force: error -5 returned"? solution on the Red Hat Customer Portal.

   ```
   xfs_log_force: error -5 returned
   ```

   b. If the **`dmesg`** output includes any **`SCSI error`** error messages, see the SCSI Error Codes Solution Finder solution on the Red Hat Customer Portal to determine the best way to fix the problem.

   c. Alternatively, if you are unable to fix the underlying file system, replace the OSD drive. See Section 5.4, “Replacing an OSD Drive” for details.

7. If the OSD failed with a segmentation fault, such as the following one, gather the required information and open a support ticket. See Chapter 8, *Contacting Red Hat Support Service* for details.

   ```
   Caught signal (Segmentation fault)
   ```

The `ceph-osd` is running but still marked as `down`

1. Check the corresponding log file to determine the cause of the failure. By default, Ceph stores log files in the **`/var/log/ceph/`** directory.

   a. If the log includes error messages similar to the following ones, see Section 5.1.4, “Flapping OSDs”.

      ```
      wrongly marked me down
      heartbeat_check: no reply from osd.2 since back
      ```

   b. If you see any other errors, open a support ticket. See Chapter 8, *Contacting Red Hat Support Service* for details.

See Also

- Section 5.1.4, “Flapping OSDs”
- Section 6.1.1, “Stale Placement Groups”
- The *Starting, Stopping, Restarting a Daemon by Instances* section in the *Administration Guide* for Red Hat Ceph Storage 3

5.1.4. Flapping OSDs

The **`ceph -w | grep osds`** command shows OSDs repeatedly as **down** and then **up** again within a short period of time:

```
# ceph -w | grep osds
2017-04-05 06:27:20.810535 mon.0 [INF] osdmap e609: 9 osds: 8 up, 9 in
2017-04-05 06:27:24.120611 mon.0 [INF] osdmap e611: 9 osds: 7 up, 9 in
2017-04-05 06:27:25.975622 mon.0 [INF] HEALTH_WARN; 118 pgs stale; 2/9 in
osds are down
2017-04-05 06:27:27.489790 mon.0 [INF] osdmap e614: 9 osds: 6 up, 9 in
2017-04-05 06:27:36.540000 mon.0 [INF] osdmap e616: 9 osds: 7 up, 9 in
2017-04-05 06:27:39.681913 mon.0 [INF] osdmap e618: 9 osds: 8 up, 9 in
2017-04-05 06:27:43.269401 mon.0 [INF] osdmap e620: 9 osds: 9 up, 9 in
2017-04-05 06:27:54.884426 mon.0 [INF] osdmap e622: 9 osds: 8 up, 9 in
2017-04-05 06:27:57.398706 mon.0 [INF] osdmap e624: 9 osds: 7 up, 9 in
2017-04-05 06:27:59.669841 mon.0 [INF] osdmap e625: 9 osds: 6 up, 9 in
2017-04-05 06:28:07.043677 mon.0 [INF] osdmap e628: 9 osds: 7 up, 9 in
2017-04-05 06:28:10.512331 mon.0 [INF] osdmap e630: 9 osds: 8 up, 9 in
2017-04-05 06:28:12.670923 mon.0 [INF] osdmap e631: 9 osds: 9 up, 9 in
```

In addition the Ceph log contains error messages similar to the following ones:

```
2016-07-25 03:44:06.510583 osd.50 127.0.0.1:6801/149046 18992 : cluster
[WRN] map e600547 wrongly marked me down
```

```
2016-07-25 19:00:08.906864 7fa2a0033700 -1 osd.254 609110 heartbeat_check:
no reply from osd.2 since back 2016-07-25 19:00:07.444113 front 2016-07-25
18:59:48.311935 (cutoff 2016-07-25 18:59:48.906862)
```

What This Means

The main causes of flapping OSDs are:

- Certain cluster operations, such as scrubbing or recovery, take an abnormal amount of time, for example if you perform these operations on objects with a large index or large placement groups. Usually, after these operations finish, the flapping OSDs problem is solved.
- Problems with the underlying physical hardware. In this case, the **`ceph health detail`** command also returns the **`slow requests`** error message. For details, see Section 5.1.5, “Slow Requests, and Requests are Blocked”.
- Problems with network.

OSDs cannot handle well the situation when the cluster (back-end) network fails or develops significant latency while the public (front-end) network operates optimally.

OSDs use the cluster network for sending heartbeat packets to each other to indicate that they are **`up`** and **`in`**. If the cluster network does not work properly, OSDs are unable to send and receive the heartbeat packets. As a consequence, they report each other as being **down** to the Monitors, while marking themselves as **up**.

The following parameters in the Ceph configuration file influence this behavior:

Parameter	Description	Default value
`osd_heartbeat_grace_time`	How long OSDs wait for the heartbeat packets to return before reporting an OSD as **down** to the Monitors.	20 seconds
`mon_osd_min_down_reporters`	How many OSDs must report another OSD as **down** before the Monitors mark the OSD as **down**	1
`mon_osd_min_down_reports`	How many times an OSD must be reported as **down** before the Monitors mark the OSD as **down**	3

This table shows that in default configuration, the Monitors mark an OSD as **down** if only one OSD made three distinct reports about the first OSD being **down**. In some cases, if one single host encounters network issues, the entire cluster can experience flapping OSDs. This is because the OSDs that reside on the host will report other OSDs in the cluster as **down**.

NOTE

The flapping OSDs scenario does not include the situation when the OSD processes are started and then immediately killed.

To Troubleshoot This Problem

1. Check the output of the **`ceph health detail`** command again. If it includes the **`slow requests`** error message, see Section 5.1.5, "Slow Requests, and Requests are Blocked" for details on how to troubleshoot this issue.

   ```
   # ceph health detail
   HEALTH_WARN 30 requests are blocked > 32 sec; 3 osds have slow
   requests
   30 ops are blocked > 268435 sec
   1 ops are blocked > 268435 sec on osd.11
   1 ops are blocked > 268435 sec on osd.18
   28 ops are blocked > 268435 sec on osd.39
   3 osds have slow requests
   ```

2. Determine which OSDs are marked as **down** and on what nodes they reside:

   ```
   # ceph osd tree | grep down
   ```

3. On the nodes containing the flapping OSDs, troubleshoot and fix any networking problems. For details, see Chapter 3, *Troubleshooting Networking Issues*.

4. Alternatively, you can temporary force Monitors to stop marking the OSDs as **down** and **up** by setting the **noup** and **nodown** flags:

   ```
   # ceph osd set noup
   # ceph osd set nodown
   ```

IMPORTANT

Using the **noup** and **nodown** flags does not fix the root cause of the problem but only prevents OSDs from flapping. Open a support ticket, if you are unable to fix and troubleshoot the error by yourself. See Chapter 8, *Contacting Red Hat Support Service* for details.

See Also

- The *Verifying the Network Configuration for Red Hat Ceph Storage* section in the Red Hat Ceph Storage 3 *Installation Guide for Red Hat Enterprise Linux* or *Installation Guide for Ubuntu*

- The {architecture-guide}#heartbeating[*Heartbeating*] section in the *Administration Guide* for Red Hat Ceph Storage 3

5.1.5. Slow Requests, and Requests are Blocked

The **ceph-osd** daemon is slow to respond to a request and the **`ceph health detail`** command returns an error message similar to the following one:

```
HEALTH_WARN 30 requests are blocked > 32 sec; 3 osds have slow requests
30 ops are blocked > 268435 sec
1 ops are blocked > 268435 sec on osd.11
1 ops are blocked > 268435 sec on osd.18
28 ops are blocked > 268435 sec on osd.39
3 osds have slow requests
```

In addition, the Ceph logs include an error message similar to the following ones:

```
2015-08-24 13:18:10.024659 osd.1 127.0.0.1:6812/3032 9 : cluster [WRN] 6
slow requests, 6 included below; oldest blocked for > 61.758455 secs
```

```
2016-07-25 03:44:06.510583 osd.50 [WRN] slow request 30.005692 seconds
old, received at {date-time}: osd_op(client.4240.0:8 benchmark_data_ceph-
1_39426_object7 [write 0~4194304] 0.69848840) v4 currently waiting for
subops from [610]
```

What This Means

An OSD with slow requests is every OSD that is not able to service the I/O operations per second (IOPS) in the queue within the time defined by the **`osd_op_complaint_time`** parameter. By default, this parameter is set to 30 seconds.

The main causes of OSDs having slow requests are:

- Problems with the underlying hardware, such as disk drives, hosts, racks, or network switches
- Problems with network. These problems are usually connected with flapping OSDs. See Section 5.1.4, “Flapping OSDs” for details.
- System load

The following table shows the types of slow requests. Use the **`dump_historic_ops`** administration socket command to determine the type of a slow request. For details about the administration socket, see the *Using the Administration Socket* section in the *Administration Guide* for Red Hat Ceph Storage 3.

Slow request type	Description
`waiting for rw locks`	The OSD is waiting to acquire a lock on a placement group for the operation.
`waiting for subops`	The OSD is waiting for replica OSDs to apply the operation to the journal.
`no flag points reached`	The OSD did not reach any major operation milestone.
`waiting for degraded object`	The OSDs have not replicated an object the specified number of times yet.

To Troubleshoot This Problem

1. Determine if the OSDs with slow or block requests share a common piece of hardware, for example a disk drive, host, rack, or network switch.
2. If the OSDs share a disk:
   a. Use the **`smartmontools`** utility to check the health of the disk or the logs to determine any errors on the disk.

> **NOTE**
>
> The **smartmontools** utility is included in the **smartmontools** package.

b. Use the **iostat** utility to get the I/O wait report (**%iowai**) on the OSD disk to determine if the disk is under heavy load.

> **NOTE**
>
> The **iostat** utility is included in the **sysstat** package.

3. If the OSDs share a host:

   a. Check the RAM and CPU utilization

   b. Use the **netstat** utility to see the network statistics on the Network Interface Controllers (NICs) and troubleshoot any networking issues. See also Chapter 3, *Troubleshooting Networking Issues* for further information.

4. If the OSDs share a rack, check the network switch for the rack. For example, if you use jumbo frames, verify that the NIC in the path has jumbo frames set.

5. If you are unable to determine a common piece of hardware shared by OSDs with slow requests, or to troubleshoot and fix hardware and networking problems, open a support ticket. See Chapter 8, *Contacting Red Hat Support Service* for details.

See Also

- The Using the Administration Socket section in the *Administration Guide* for Red Hat Ceph Storage 3

5.2. STOPPING AND STARTING REBALANCING

When an OSD fails or you stop it, the CRUSH algorithm automatically starts the rebalancing process to redistribute data across the remaining OSDs.

Rebalancing can take time and resources, therefore, consider stopping rebalancing during troubleshooting or maintaining OSDs. To do so, set the **noout** flag before stopping the OSD:

```
# ceph osd set noout
```

When you finish troubleshooting or maintenance, unset the **noout** flag to start rebalancing:

```
# ceph osd unset noout
```

> **NOTE**
>
> Placement groups within the stopped OSDs become **degraded** during troubleshooting and maintenance.

See Also

- The {architecture-guide}#rebalancing_and_recovery[*Rebalancing and Recovery*] section in the *Architecture Guide* for Red Hat Ceph Storage 3

5.3. MOUNTING THE OSD DATA PARTITION

If the OSD data partition is not mounted correctly, the **ceph-osd** daemon cannot start. If you discover that the partition is not mounted as expected, follow the steps in this section to mount it.

Procedure: Mounting the OSD Data Partition

1. Mount the partition:

   ```
   # mount -o noatime <partition> /var/lib/ceph/osd/<cluster-name>-
   <osd-number>
   ```

   Replace **<partition>** with the path to the partition on the OSD drive dedicated to OSD data. Specify the cluster name and the OSD number, for example:

   ```
   # mount -o noatime /dev/sdd1 /var/lib/ceph/osd/ceph-0
   ```

2. Try to start the failed **ceph-osd** daemon:

   ```
   # systemctl start ceph-osd@<OSD-number>
   ```

   Replace the **<OSD-number>** with the ID of the OSD, for example:

   ```
   # systemctl start ceph-osd@0
   ```

See Also

- Section 5.1.3, “One or More OSDs Are Down”

5.4. REPLACING AN OSD DRIVE

Ceph is designed for fault tolerance, which means that it can operate in a **degraded** state without losing data. Consequently, Ceph can operate even if a data storage drive fails. In the context of a failed drive, the **degraded** state means that the extra copies of the data stored on other OSDs will backfill automatically to other OSDs in the cluster. However, if this occurs, replace the failed OSD drive and recreate the OSD manually.

When a drive fails, Ceph reports the OSD as **down**:

```
HEALTH_WARN 1/3 in osds are down
osd.0 is down since epoch 23, last address 192.168.106.220:6800/11080
```

NOTE

Ceph can mark an OSD as **down** also as a consequence of networking or permissions problems. See Section 5.1.3, “One or More OSDs Are Down” for details.

Modern servers typically deploy with hot-swappable drives so you can pull a failed drive and replace it with a new one without bringing down the node. The whole procedure includes these steps:

1. Remove the OSD from the Ceph cluster. For details, see the Removing an OSD from the Ceph Cluster procedure.

2. Replace the drive. For details see, the Replacing the Physical Drive section.

3. Add the OSD to the cluster. For details, see the Adding an OSD to the Ceph Cluster procedure.

Before You Start

1. Determine which OSD is **down**:

```
# ceph osd tree | grep -i down
ID WEIGHT  TYPE NAME      UP/DOWN REWEIGHT PRIMARY-AFFINITY
 0 0.00999         osd.0     down  1.00000          1.00000
```

2. Ensure that the OSD process is stopped. Use the following command from the OSD node:

```
# systemctl status ceph-osd@<OSD-number>
```

Replace **`<OSD-number>`** with the ID of the OSD marked as **down**, for example:

```
# systemctl status ceph-osd@osd.0
...
   Active: inactive (dead)
```

If the **`ceph-osd`** daemon is running. See Section 5.1.3, “One or More OSDs Are Down” for more details about troubleshooting OSDs that are marked as **down** but their corresponding **`ceph-osd`** daemon is running.

Procedure: Removing an OSD from the Ceph Cluster

1. Mark the OSD as **`out`**:

```
# ceph osd out osd.<OSD-number>
```

Replace **`<OSD-number>`** with the ID of the OSD that is marked as **down**, for example:

```
# ceph osd out osd.0
marked out osd.0.
```

> **NOTE**
>
> If the OSD is **down**, Ceph marks it as **`out`** automatically after 900 seconds when it does not receive any heartbeat packet from the OSD. When this happens, other OSDs with copies of the failed OSD data begin backfilling to ensure that the required number of copies exists within the cluster. While the cluster is backfilling, the cluster will be in a **`degraded`** state.

2. Ensure that the failed OSD is backfilling. The output will include information similar to the following one:

```
# ceph -w | grep backfill
2017-06-02 04:48:03.403872 mon.0 [INF] pgmap v10293282: 431 pgs: 1
```

```
active+undersized+degraded+remapped+backfilling, 28
active+undersized+degraded, 49
active+undersized+degraded+remapped+wait_backfill, 59
stale+active+clean, 294 active+clean; 72347 MB data, 101302 MB used,
1624 GB / 1722 GB avail; 227 kB/s rd, 1358 B/s wr, 12 op/s;
10626/35917 objects degraded (29.585%); 6757/35917 objects misplaced
(18.813%); 63500 kB/s, 15 objects/s recovering
2017-06-02 04:48:04.414397 mon.0 [INF] pgmap v10293283: 431 pgs: 2
active+undersized+degraded+remapped+backfilling, 75
active+undersized+degraded+remapped+wait_backfill, 59
stale+active+clean, 295 active+clean; 72347 MB data, 101398 MB used,
1623 GB / 1722 GB avail; 969 kB/s rd, 6778 B/s wr, 32 op/s;
10626/35917 objects degraded (29.585%); 10580/35917 objects
misplaced (29.457%); 125 MB/s, 31 objects/s recovering
2017-06-02 04:48:00.380063 osd.1 [INF] 0.6f starting backfill to
osd.0 from (0'0,0'0] MAX to 2521'166639
2017-06-02 04:48:00.380139 osd.1 [INF] 0.48 starting backfill to
osd.0 from (0'0,0'0] MAX to 2513'43079
2017-06-02 04:48:00.380260 osd.1 [INF] 0.d starting backfill to
osd.0 from (0'0,0'0] MAX to 2513'136847
2017-06-02 04:48:00.380849 osd.1 [INF] 0.71 starting backfill to
osd.0 from (0'0,0'0] MAX to 2331'28496
2017-06-02 04:48:00.381027 osd.1 [INF] 0.51 starting backfill to
osd.0 from (0'0,0'0] MAX to 2513'87544
```

3. Remove the OSD from the CRUSH map:

```
# ceph osd crush remove osd.<OSD-number>
```

Replace **`<OSD-number>`** with the ID of the OSD that is marked as **down**, for example:

```
# ceph osd crush remove osd.0
removed item id 0 name 'osd.0' from crush map
```

4. Remove authentication keys related to the OSD:

```
# ceph auth del osd.<OSD-number>
```

Replace **`<OSD-number>`** with the ID of the OSD that is marked as **down**, for example:

```
# ceph auth del osd.0
updated
```

5. Remove the OSD from the Ceph Storage Cluster:

```
# ceph osd rm osd.<OSD-number>
```

Replace **`<OSD-number>`** with the ID of the OSD that is marked as **down**, for example:

```
# ceph osd rm osd.0
removed osd.0
```

If you have removed the OSD successfully, it is not present in the output of the following command:

```
# ceph osd tree
```

6. Unmount the failed drive:

```
# umount /var/lib/ceph/osd/<cluster-name>-<OSD-number>
```

Specify the name of the cluster and the ID of the OSD, for example:

```
# umount /var/lib/ceph/osd/ceph-0/
```

If you have unmounted the drive successfully, it is not present in the output of the following command:

```
# df -h
```

Procedure: Replacing the Physical Drive

1. See the documentation for the hardware node for details on replacing the physical drive.

   a. If the drive is hot-swappable, replace the failed drive with a new one.

   b. If the drive is not hot-swappable and the node contains multiple OSDs, you might have to shut down the whole node and replace the physical drive. Consider preventing the cluster from backfilling. See Section 5.2, “Stopping and Starting Rebalancing” for details.

2. When the drive appears under the **/dev/** directory, make a note of the drive path.

3. If you want to add the OSD manually, find the OSD drive and format the disk.

Procedure: Adding an OSD to the Ceph Cluster

1. Add the OSD again.

   a. If you used Ansible to deploy the cluster, run the **ceph-ansible** playbook again from the Ceph administration server:

   ```
   # ansible-playbook /usr/share/ceph-ansible site.yml
   ```

   b. If you added the OSD manually, see the *Adding an OSD with the Command-line Interface* section in the _Administration Guid_e for Red Hat Ceph Storage 3.

2. Ensure that the CRUSH hierarchy is accurate:

```
# ceph osd tree
```

3. If you are not satisfied with the location of the OSD in the CRUSH hierarchy, move the OSD to a desired location:

```
ceph osd crush move <bucket-to-move> <bucket-type>=<parent-bucket>
```

For example, to move the bucket located at **sdd:row1** to the root bucket:

```
# ceph osd crush move ssd:row1 root=ssd:root
```

See Also

- Section 5.1.3, “One or More OSDs Are Down”
- The *Managing the Cluster Size* chapter in the *Administration Guide* for Red Hat Ceph Storage 3
- The Red Hat Ceph Storage 3 *Installation Guide for Red Hat Enterprise Linux* or the *Installation Guide for Ubuntu*

5.5. INCREASING THE PID COUNT

If you have a node containing more than 12 Ceph OSDs, the default maximum number of threads (PID count) can be insufficient, especially during recovery. As a consequence, some **`ceph-osd`** daemons can terminate and fail to start again. If this happens, increase the maximum possible number of threads allowed.

To temporary increase the number:

```
# sysctl -w kernel.pid.max=4194303
```

To permanently increase the number, update the **`/etc/sysctl.conf`** file as follows:

```
kernel.pid.max = 4194303
```

5.6. DELETING DATA FROM A FULL CLUSTER

Ceph automatically prevents any I/O operations on OSDs that reached the capacity specified by the **`mon_osd_full_ratio`** parameter and returns the **`full osds`** error message.

This procedure shows how to delete unnecessary data to fix this error.

NOTE

The **`mon_osd_full_ratio`** parameter sets the value of the **`full_ratio`** parameter when creating a cluster. You cannot change the value of **`mon_osd_full_ratio`** afterwards. To temporarily increase the **`full_ratio`** value, increase the **`pg_full_ratio`** instead.

Procedure: Deleting Data from a Full Cluster

1. Determine the current value of **`full_ratio`**, by default it is set to **`0.95`**:

   ```
   # ceph pg dump | grep -i full
   full_ratio 0.95
   ```

2. Temporarily increase the value of **`pg_full_ratio`** to **`0.97`**:

   ```
   # ceph pg set_full_ratio 0.97
   ```

IMPORTANT

Red Hat strongly recommends to not set the **pg_full_ratio** to a value higher than 0.97. Setting this parameter to a higher value makes the recovery process harder. As a consequence, you might not be able to recover full OSDs at all.

3. Verify that you successfully set the parameter to **0.97**:

   ```
   # ceph pg dump | grep -i full
   full_ratio 0.97
   ```

4. Monitor the cluster state:

   ```
   # ceph -w
   ```

   As soon as the cluster changes its state from **full** to **nearfull**, delete any unnecessary data.

5. Set the value of **full ratio** back to **0.95**:

   ```
   # ceph pg set_full_ratio 0.95
   ```

6. Verify that you successfully set the parameter to **0.95**:

   ```
   # ceph pg dump | grep -i full
   full_ratio 0.95
   ```

See Also

- Section 5.1.1, “Full OSDs”
- Section 5.1.2, “Nearfull OSDs”

CHAPTER 6. TROUBLESHOOTING PLACEMENT GROUPS

This section contains information about fixing the most common errors related to the Ceph Placement Groups (PGs).

Before You Start

- Verify your network connection. See Chapter 3, *Troubleshooting Networking Issues* for details.
- Ensure that Monitors are able to form a quorum. See Chapter 4, *Troubleshooting Monitors* for details about troubleshooting the most common errors related to Monitors.
- Ensure that all healthy OSDs are `up` and `in`, and the backfilling and recovery processes are finished. See Chapter 5, *Troubleshooting OSDs* for details about troubleshooting the most common errors related to OSDs.

6.1. THE MOST COMMON ERROR MESSAGES RELATED TO PLACEMENT GROUPS

The following table lists the most common errors messages that are returned by the `ceph health detail` command. The table provides links to corresponding sections that explain the errors and point to specific procedures to fix the problems.

In addition, you can list placement groups that are stuck in a state that is not optimal. See Section 6.2, “Listing Placement Groups in `stale`, `inactive`, or `unclean` State” for details.

Table 6.1. Error Messages Related to Placement Groups

Error message	See
`HEALTH_ERR`	
`pgs down`	Section 6.1.5, “Placement Groups Are `down`”
`pgs inconsistent`	Section 6.1.2, “Inconsistent Placement Groups”
`scrub errors`	Section 6.1.2, “Inconsistent Placement Groups”
`HEALTH_WARN`	
`pgs stale`	Section 6.1.1, “Stale Placement Groups”
`unfound`	Section 6.1.6, “Unfound Objects”

6.1.1. Stale Placement Groups

The `ceph health` command lists some Placement Groups (PGs) as `stale`:

```
HEALTH_WARN 24 pgs stale; 3/300 in osds are down
```

What This Means

The Monitor marks a placement group as **`stale`** when it does not receive any status update from the primary OSD of the placement group's acting set or when other OSDs reported that the primary OSD is **`down`**.

Usually, PGs enter the **`stale`** state after you start the storage cluster and until the peering process completes. However, when the PGs remain **`stale`** for longer than expected, it might indicate that the primary OSD for those PGs is **`down`** or not reporting PG statistics to the Monitor. When the primary OSD storing **`stale`** PGs is back **`up`**, Ceph starts to recover the PGs.

The **`mon_osd_report_timeout`** setting determines how often OSDs report PGs statistics to Monitors. Be default, this parameter is set to **`0.5`**, which means that OSDs report the statistics every half a second.

To Troubleshoot This Problem

1. Identify which PGs are **`stale`** and on what OSDs they are stored. The error message will include information similar to the following example:

```
# ceph health detail
HEALTH_WARN 24 pgs stale; 3/300 in osds are down
...
pg 2.5 is stuck stale+active+remapped, last acting [2,0]
...
osd.10 is down since epoch 23, last address
192.168.106.220:6800/11080
osd.11 is down since epoch 13, last address
192.168.106.220:6803/11539
osd.12 is down since epoch 24, last address
192.168.106.220:6806/11861
```

2. Troubleshoot any problems with the OSDs that are marked as **`down`**. For details, see Section 5.1.3, "One or More OSDs Are Down".

See Also

- The {administration-guide}#identifying_troubled_placement_groups[*Monitoring Placement Group States*] section in the *Administration Guide* for Red Hat Ceph Storage 3

6.1.2. Inconsistent Placement Groups

Some placement groups are marked as **`active + clean + inconsistent`** and the **`ceph health detail`** returns an error messages similar to the following one:

```
HEALTH_ERR 1 pgs inconsistent; 2 scrub errors
pg 0.6 is active+clean+inconsistent, acting [0,1,2]
2 scrub errors
```

What This Means

When Ceph detects inconsistencies in one or more replicas of an object in a placement group, it marks the placement group as **`inconsistent`**. The most common inconsistencies are:

- Objects have an incorrect size.
- Objects are missing from one replica after a recovery finished.

In most cases, errors during scrubbing cause inconsistency within placement groups.

To Troubleshoot This Problem

1. Determine which placement group is in the **inconsistent** state:

   ```
   # ceph health detail
   HEALTH_ERR 1 pgs inconsistent; 2 scrub errors
   pg 0.6 is active+clean+inconsistent, acting [0,1,2]
   2 scrub errors
   ```

2. Determine why the placement group is **inconsistent**.

   a. Start the deep scrubbing process on the placement group:

      ```
      ceph pg deep-scrub <id>
      ```

      Replace **<id>** with the ID of the **inconsistent** placement group, for example:

      ```
      # ceph pg deep-scrub 0.6
      instructing pg 0.6 on osd.0 to deep-scrub
      ```

   b. Search the output of the **ceph -w** for any messages related to that placement group:

      ```
      ceph -w | grep <id>
      ```

      Replace **<id>** with the ID of the **inconsistent** placement group, for example:

      ```
      # ceph -w | grep 0.6
      2015-02-26 01:35:36.778215 osd.106 [ERR] 0.6 deep-scrub stat
      mismatch, got 636/635 objects, 0/0 clones, 0/0 dirty, 0/0 omap,
      0/0 hit_set_archive, 0/0 whiteouts, 1855455/1854371 bytes.
      2015-02-26 01:35:36.788334 osd.106 [ERR] 0.6 deep-scrub 1 errors
      ```

3. If the output includes any error messages similar to the following ones, you can repair the **inconsistent** placement group. See Section 6.4, “Repairing Inconsistent Placement Groups” for details.

   ```
   <pg.id> shard <osd>: soid <object> missing attr _, missing attr
   <attr type>
   <pg.id> shard <osd>: soid <object> digest 0 != known digest
   <digest>, size 0 != known size <size>
   <pg.id> shard <osd>: soid <object> size 0 != known size <size>
   <pg.id> deep-scrub stat mismatch, got <mismatch>
   <pg.id> shard <osd>: soid <object> candidate had a read error,
   digest 0 != known digest <digest>
   ```

4. If the output includes any error messages similar to the following ones, it is not safe to repair the **inconsistent** placement group because you can lose data. Open a support ticket in this situation. See Chapter 8, *Contacting Red Hat Support Service* for details.

   ```
   <pg.id> shard <osd>: soid <object> digest <digest> != known digest
   <digest>
   <pg.id> shard <osd>: soid <object> omap_digest <digest> != known
   omap_digest <digest>
   ```

See Also

- Section 6.4, “Repairing Inconsistent Placement Groups”
- Section 6.3, “Listing Inconsistencies”
- The {architecture-guide}#scrubbing[*Scrubbing*] section in the *Architecture Guide* for Red Hat Ceph Storage 3
- The *Scrubbing* section in the *Configuration Guide* for Red Hat Ceph Storage 3

6.1.3. Unclean Placement Groups

The **ceph health** command returns an error message similar to the following one:

```
HEALTH_WARN 197 pgs stuck unclean
```

What This Means
Ceph marks a placement group as **unclean** if it has not achieved the **active+clean** state for the number of seconds specified in the **mon_pg_stuck_threshold** parameter in the Ceph configuration file. The default value of **mon_pg_stuck_threshold** is **300** seconds.

If a placement group is **unclean**, it contains objects that are not replicated the number of times specified in the **osd_pool_default_size** parameter. The default value of **osd_pool_default_size** is **3**, which means that Ceph creates three replicas.

Usually, **unclean** placement groups indicate that some OSDs might be **down**.

To Troubleshoot This Problem

1. Determine which OSDs are **down**:

   ```
   # ceph osd tree
   ```

2. Troubleshoot and fix any problems with the OSDs. See Section 5.1.3, “One or More OSDs Are Down” for details.

See Also

- Section 6.2, “Listing Placement Groups in **stale**, **inactive**, or **unclean** State”

6.1.4. Inactive Placement Groups

The **ceph health** command returns a error message similar to the following one:

```
HEALTH_WARN 197 pgs stuck inactive
```

What This Means
Ceph marks a placement group as **inactive** if it has not be active for the number of seconds specified in the **mon_pg_stuck_threshold** parameter in the Ceph configuration file. The default value of **mon_pg_stuck_threshold** is **300** seconds.

Usually, **inactive** placement groups indicate that some OSDs might be **down**.

To Troubleshoot This Problem

1. Determine which OSDs are **down**:

   ```
   # ceph osd tree
   ```

2. Troubleshoot and fix any problems with the OSDs. See Section 5.1.3, “One or More OSDs Are Down” for details.

See Also

- Section 6.2, “Listing Placement Groups in **`stale`**, **`inactive`**, or **`unclean`** State”

6.1.5. Placement Groups Are `down`

The **`ceph health detail`** command reports that some placement groups are **down**:

```
HEALTH_ERR 7 pgs degraded; 12 pgs down; 12 pgs peering; 1 pgs recovering; 6
pgs stuck unclean; 114/3300 degraded (3.455%); 1/3 in osds are down
...
pg 0.5 is down+peering
pg 1.4 is down+peering
...
osd.1 is down since epoch 69, last address 192.168.106.220:6801/8651
```

What This Means

In certain cases, the peering process can be blocked, which prevents a placement group from becoming active and usable. Usually, a failure of an OSD causes the peering failures.

To Troubleshoot This Problem

Determine what blocks the peering process:

```
ceph pg <id> query
```

Replace **`<id>`** with the ID of the placement group that is **down**, for example:

```
# ceph pg 0.5 query

{ "state": "down+peering",
  ...
  "recovery_state": [
       { "name": "Started\/Primary\/Peering\/GetInfo",
         "enter_time": "2012-03-06 14:40:16.169679",
         "requested_info_from": []},
       { "name": "Started\/Primary\/Peering",
         "enter_time": "2012-03-06 14:40:16.169659",
         "probing_osds": [
               0,
               1],
         "blocked": "peering is blocked due to down osds",
         "down_osds_we_would_probe": [
               1],
         "peering_blocked_by": [
               { "osd": 1,
                 "current_lost_at": 0,
```

```
                 "comment": "starting or marking this osd lost may let us
proceed"}]},
       { "name": "Started",
         "enter_time": "2012-03-06 14:40:16.169513"}
   ]
}
```

The **recovery_state** section includes information why the peering process is blocked.

- If the output includes the **peering is blocked due to down osds** error message, see Section 5.1.3, "One or More OSDs Are Down".

- If you see any other error message, open a support ticket. See Chapter 8, *Contacting Red Hat Support Service* for details.

See Also

- The *Peering* section in the *Administration Guide* for Red Hat Ceph Storage 3

6.1.6. Unfound Objects

The **ceph health** command returns an error message similar to the following one, containing the **unfound** keyword:

```
HEALTH_WARN 1 pgs degraded; 78/3778 unfound (2.065%)
```

What This Means
Ceph marks objects as **unfound** when it knows these objects or their newer copies exist but it is unable to find them. As a consequence, Ceph cannot recover such objects and proceed with the recovery process.

An Example Situation
A placement group stores data on **osd.1** and **osd.2**.

1. **osd.1** goes **down**.

2. **osd.2** handles some write operations.

3. **osd.1** comes **up**.

4. A peering process between **osd.1** and **osd.2** starts, and the objects missing on **osd.1** are queued for recovery.

5. Before Ceph copies new objects, **osd.2** goes **down**.

As a result, **osd.1** knows that these objects exist, but there is no OSD that has a copy of the objects.

In this scenario, Ceph is waiting for the failed node to be accessible again, and the **unfound** objects blocks the recovery process.

To Troubleshoot This Problem

1. Determine which placement group contain **unfound** objects:

   ```
   # ceph health detail
   ```

```
HEALTH_WARN 1 pgs recovering; 1 pgs stuck unclean; recovery 5/937611
objects degraded (0.001%); 1/312537 unfound (0.000%)
pg 3.8a5 is stuck unclean for 803946.712780, current state
active+recovering, last acting [320,248,0]
pg 3.8a5 is active+recovering, acting [320,248,0], 1 unfound
recovery 5/937611 objects degraded (0.001%); **1/312537 unfound
(0.000%)**
```

2. List more information about the placement group:

```
# ceph pg <id> query
```

Replace **`<id>`** with the ID of the placement group containing the **`unfound`** objects, for example:

```
# ceph pg 3.8a5 query
{ "state": "active+recovering",
  "epoch": 10741,
  "up": [
        320,
        248,
        0],
  "acting": [
        320,
        248,
        0],
<snip>
  "recovery_state": [
        { "name": "Started\/Primary\/Active",
          "enter_time": "2015-01-28 19:30:12.058136",
          "might_have_unfound": [
                { "osd": "0",
                  "status": "already probed"},
                { "osd": "248",
                  "status": "already probed"},
                { "osd": "301",
                  "status": "already probed"},
                { "osd": "362",
                  "status": "already probed"},
                { "osd": "395",
                  "status": "already probed"},
                { "osd": "429",
                  "status": "osd is down"}],
          "recovery_progress": { "backfill_targets": [],
              "waiting_on_backfill": [],
              "last_backfill_started": "0\/\/0\/\/-1",
              "backfill_info": { "begin": "0\/\/0\/\/-1",
                  "end": "0\/\/0\/\/-1",
                  "objects": []},
              "peer_backfill_info": [],
              "backfills_in_flight": [],
              "recovering": [],
              "pg_backend": { "pull_from_peer": [],
                  "pushing": []}},
          "scrub": { "scrubber.epoch_start": "0",
              "scrubber.active": 0,
                "scrubber.block_writes": 0,
                "scrubber.finalizing": 0,
                "scrubber.waiting_on": 0,
                "scrubber.waiting_on_whom": []}},
          { "name": "Started",
            "enter_time": "2015-01-28 19:30:11.044020"}],
```

The **might_have_unfound** section includes OSDs where Ceph tried to locate the **unfound** objects:

- The **already probed** status indicates that Ceph cannot locate the **unfound** objects in that OSD.
- The **osd is down** status indicates that Ceph cannot contact that OSD.

3. Troubleshoot the OSDs that are marked as **down**. See Section 5.1.3, “One or More OSDs Are Down” for details.

4. If you are unable to fix the problem that causes the OSD to be **down**, open a support ticket. See Chapter 8, *Contacting Red Hat Support Service* for details.

6.2. LISTING PLACEMENT GROUPS IN STALE, INACTIVE, OR UNCLEAN STATE

After a failure, placement groups enter states like **degraded** or **peering**. This states indicate normal progression through the failure recovery process.

However, if a placement group stays in one of these states for a longer time than expected, it can be an indication of a larger problem. The Monitors reports when placement groups get stuck in a state that is not optimal.

The following table lists these states together with a short explanation.

State	What it means	Most common causes	See
inactive	The PG has not been able to service read/write requests.	• Peering problems	Section 6.1.4, “Inactive Placement Groups”
unclean	The PG contains objects that are not replicated the desired number of times. Something is preventing the PG from recovering.	• **unfound** objects • OSDs are **down** • Incorrect configuration	Section 6.1.3, “Unclean Placement Groups”
stale	The status of the PG has not been updated by a **ceph-osd** daemon.	• OSDs are **down**	Section 6.1.1, “Stale Placement Groups”

The **mon_pg_stuck_threshold** parameter in the Ceph configuration file determines the number of seconds after which placement groups are considered **inactive**, **unclean**, or **stale**.

List the stuck PGs:

```
# ceph pg dump_stuck inactive
# ceph pg dump_stuck unclean
# ceph pg dump_stuck stale
```

See Also

- The *Monitoring Placement Group States* section in the *Administration Guide* for Red Hat Ceph Storage 3

6.3. LISTING INCONSISTENCIES

Use the **rados** utility to list inconsistencies in various replicas of an objects. Use the **--format=json-pretty** option to list a more detailed output.

You can list:

- Inconsistent placement group in a pool
- Inconsistent objects in a placement group
- Inconsistent snapshot sets in a placement group

Listing Inconsistent Placement Groups in a Pool

```
rados list-inconsistent-pg <pool> --format=json-pretty
```

For example, list all inconsistent placement groups in a pool named **data**:

```
# rados list-inconsistent-pg data --format=json-pretty
[0.6]
```

Listing Inconsistent Objects in a Placement Group

```
rados list-inconsistent-obj <placement-group-id>
```

For example, list inconsistent objects in a placement group with ID **0.6**:

```
# rados list-inconsistent-obj 0.6
{
    "epoch": 14,
    "inconsistents": [
        {
            "object": {
                "name": "image1",
                "nspace": "",
                "locator": "",
                "snap": "head",
                "version": 1
            },
            "errors": [
                "data_digest_mismatch",
                "size_mismatch"
            ],
            "union_shard_errors": [
                "data_digest_mismatch_oi",
                "size_mismatch_oi"
            ],
            "selected_object_info": "0:602f83fe:::foo:head(16'1
client.4110.0:1 dirty|data_digest|omap_digest s 968 uv 1 dd e978e67f od
ffffffff alloc_hint [0 0 0])",
            "shards": [
                {
                    "osd": 0,
                    "errors": [],
                    "size": 968,
                    "omap_digest": "0xffffffff",
                    "data_digest": "0xe978e67f"
                },
                {
                    "osd": 1,
                    "errors": [],
                    "size": 968,
                    "omap_digest": "0xffffffff",
                    "data_digest": "0xe978e67f"
                },
                {
                    "osd": 2,
                    "errors": [
                        "data_digest_mismatch_oi",
                        "size_mismatch_oi"
                    ],
                    "size": 0,
                    "omap_digest": "0xffffffff",
                    "data_digest": "0xffffffff"
                }
            ]
        }
    ]
}
```

The following fields are important to determine what causes the inconsistency:

- **`name`**: The name of the object with inconsistent replicas.
- **`nspace`**: The namespace that is a logical separation of a pool. It’s empty by default.
- **`locator`**: The key that is used as the alternative of the object name for placement.
- **`snap`**: The snapshot ID of the object. The only writable version of the object is called **`head`**. If an object is a clone, this field includes its sequential ID.
- **`version`**: The version ID of the object with inconsistent replicas. Each write operation to an object increments it.

- **`errors`**: A list of errors that indicate inconsistencies between shards without determining which shard or shards are incorrect. See the **`shard`** array to further investigate the errors.
    - **`data_digest_mismatch`**: The digest of the replica read from one OSD is different from the other OSDs.
    - **`size_mismatch`**: The size of a clone or the **`head`** object does not match the expectation.
    - **`read_error`**: This error indicates inconsistencies caused most likely by disk errors.
- **`union_shard_error`**: The union of all errors specific to shards. These errors are connected to a faulty shard. The errors that end with **`oi`** indicate that you have to compare the information from a faulty object to information with selected objects. See the **`shard`** array to further investigate the errors.
  In the above example, the object replica stored on **`osd.2`** has different digest than the replicas stored on **`osd.0`** and **`osd.1`**. Specifically, the digest of the replica is not**`0xffffffff`** as calculated from the shard read from **`osd.2`**, but **`0xe978e67f`**. In addition, the size of the replica read from **`osd.2`** is 0, while the size reported by **`osd.0`** and **`osd.1`** is 968.

Listing Inconsistent Snapshot Sets in a Placement Group

```
rados list-inconsistent-snapset <placement-group-id>
```

For example, list inconsistent sets of snapshots (**`snapsets`**) in a placement group with ID **`0.23`**:

```
# rados list-inconsistent-snapset 0.23 --format=json-pretty
{
    "epoch": 64,
    "inconsistents": [
        {
            "name": "obj5",
            "nspace": "",
            "locator": "",
            "snap": "0x00000001",
            "headless": true
        },
        {
            "name": "obj5",
            "nspace": "",
            "locator": "",
            "snap": "0x00000002",
            "headless": true
        },
        {
            "name": "obj5",
            "nspace": "",
            "locator": "",
            "snap": "head",
            "ss_attr_missing": true,
            "extra_clones": true,
            "extra clones": [
                2,
                1
            ]
        }
    ]
```

The command returns the following errors:

- **ss_attr_missing**: One or more attributes are missing. Attributes are information about snapshots encoded into a snapshot set as a list of key-value pairs.
- **ss_attr_corrupted**: One or more attributes fail to decode.
- **clone_missing**: A clone is missing.
- **snapset_mismatch**: The snapshot set is inconsistent by itself.
- **head_mismatch**: The snapshot set indicates that **head** exists or not, but the scrub results report otherwise.
- **headless**: The **head** of the snapshot set is missing.
- **size_mismatch**: The size of a clone or the **head** object does not match the expectation.

See Also

- Section 6.1.2, “Inconsistent Placement Groups”
- Section 6.4, “Repairing Inconsistent Placement Groups”

6.4. REPAIRING INCONSISTENT PLACEMENT GROUPS

Due to an error during deep scrubbing, some placement groups can include inconsistencies. Ceph reports such placement groups as **inconsistent**:

```
HEALTH_ERR 1 pgs inconsistent; 2 scrub errors
pg 0.6 is active+clean+inconsistent, acting [0,1,2]
2 scrub errors
```

WARNING

You can repair only certain inconsistencies. Do not repair the placement groups if the Ceph logs include the following errors:

```
<pg.id> shard <osd>: soid <object> digest <digest> != known
digest <digest>
<pg.id> shard <osd>: soid <object> omap_digest <digest> !=
known omap_digest <digest>
```

Open a support ticket instead. See Chapter 8, *Contacting Red Hat Support Service* for details.

Repair the **`inconsistent`** placement groups:

```
ceph pg repair <id>
```

Replace **`<id>`** with the ID of the **`inconsistent`** placement group.

See Also

- Section 6.1.2, “Inconsistent Placement Groups”
- Section 6.3, “Listing Inconsistencies”

6.5. INCREASING THE PG COUNT

Insufficient Placement Group (PG) count impacts the performance of the Ceph cluster and data distribution. It is one of the main causes of the **`nearfull osds`** error messages.

The recommended ratio is between 100 and 300 PGs per OSD. This ratio can decrease when you add more OSDs to the cluster.

The **`pg_num`** and **`pgp_num`** parameters determine the PG count. These parameters are configured per each pool, and therefore, you must adjust each pool with low PG count separately.

IMPORTANT

Increasing the PG count is the most intensive process that you can perform on a Ceph cluster. This process might have serious performance impact if not done in a slow and methodical way. Once you increase **`pgp_num`**, you will not be able to stop or reverse the process and you must complete it.

Consider increasing the PG count outside of business critical processing time allocation, and alert all clients about the potential performance impact.

Do not change the PG count if the cluster is in the **`HEALTH_ERR`** state.

Procedure: Increasing the PG Count

1. Reduce the impact of data redistribution and recovery on individual OSDs and OSD hosts:

   a. Lower the value of the **`osd max backfills`**, **`osd_recovery_max_active`**, and **`osd_recovery_op_priority`** parameters:

   ```
   # ceph tell osd.* injectargs '--osd_max_backfills 1 --
   osd_recovery_max_active 1 --osd_recovery_op_priority 1'
   ```

   b. Disable the shallow and deep scrubbing:

   ```
   # ceph osd set noscrub
   # ceph osd set nodeep-scrub
   ```

2. Use the Ceph Placement Groups (PGs) per Pool Calculator to calculate the optimal value of the **`pg_num`** and **`pgp_num`** parameters.

3. Increase the **`pg_num`** value in small increments until you reach the desired value.

   a. Determine the starting increment value. Use a very low value that is a power of two, and increase it when you determine the impact on the cluster. The optimal value depends on the pool size, OSD count, and client I/O load.

   b. Increment the **pg_num** value:

      ```
      ceph osd pool set <pool> pg_num <value>
      ```

      Specify the pool name and the new value, for example:

      ```
      # ceph osd pool set data pg_num 4
      ```

   c. Monitor the status of the cluster:

      ```
      # ceph -s
      ```

      The PGs state will change from **creating** to **active+clean**. Wait until all PGs are in the **active+clean** state.

4. Increase the **pgp_num** value in small increments until you reach the desired value:

   a. Determine the starting increment value. Use a very low value that is a power of two, and increase it when you determine the impact on the cluster. The optimal value depends on the pool size, OSD count, and client I/O load.

   b. Increment the **pgp_num** value:

      ```
      ceph osd pool set <pool> pgp_num <value>
      ```

      Specify the pool name and the new value, for example:

      ```
      # ceph osd pool set data pgp_num 4
      ```

   c. Monitor the status of the cluster:

      ```
      # ceph -s
      ```

      The PGs state will change through **peering**, **wait_backfill**, **backfilling**, **recover**, and others. Wait until all PGs are in the **active+clean** state.

5. Repeat the previous steps for all pools with insufficient PG count.

6. Set **osd max backfills**, **osd_recovery_max_active**, and **osd_recovery_op_priority** to their default values:

   ```
   # ceph tell osd.* injectargs '--osd_max_backfills 1 --
   osd_recovery_max_active 3 --osd_recovery_op_priority 3'
   ```

7. Enable the shallow and deep scrubbing:

   ```
   # ceph osd unset noscrub
   # ceph osd unset nodeep-scrub
   ```

See also

- Section 5.1.2, “Nearfull OSDs”
- The *Monitoring Placement Group States* section in the *Administration Guide* for Red Hat Ceph Storage 3

CHAPTER 7. INSTALLING AND USING CEPH-MEDIC TO DIAGNOSE A CEPH STORAGE CLUSTER

Use the **ceph-medic** utility to detect common issues with a Ceph Storage Cluster that prevents it from functioning properly.

7.1. PREREQUISITE

- A working Red Hat Ceph Storage cluster

7.2. INSTALLING THE CEPH-MEDIC UTILITY

Prerequisite

- Access to the Red Hat Ceph Storage 3 software repositories

Procedure

Do the following steps on the Ansible administration node, as the **root** user.

1. Install the **ceph-medic** package:

   ```
   [root@admin ~]# yum install ceph-medic
   ```

2. Verify the installation of **ceph-medic**:

   ```
   [root@admin ~]# ceph-medic --help
   ```

Additional Resources

- The *Enabling the Red Hat Ceph Storage Repositories* section in the Red Hat Ceph Storage 3 *Installation Guide for Red Hat Enterprise Linux*

7.3. USING THE CEPH-MEDIC UTILITY

The **ceph-medic** utility performs checks against a running Ceph Storage Cluster to identify potential problems.

The **ceph-medics** utility for example checks:

- The correct ownership of files and directories
- If the **fsid** is the same for all nodes in the storage cluster
- If the secret key in the keyring is different than other nodes in the storage cluster

7.3.1. Prerequisites

- A working Red Hat Ceph Storage cluster
- SSH and **sudo** access to storage nodes

7.3.2. Running a Diagnostic Check

A basic check for potential problems with a Ceph Storage Cluster.

Prerequisites

- A working Red Hat Ceph Storage cluster
- SSH and **sudo** access to storage nodes

Procedure

Do the following step from the Ansible administration node, as a normal user.

- Use the **`ceph-medic check`** command:

```
[admin@admin ~]$ ceph-medic check
Host: mon0                  connection: [connected  ]
Host: mon1                  connection: [connected  ]
Host: mon2                  connection: [connected  ]
Host: osd0                  connection: [connected  ]
Host: osd1                  connection: [connected  ]
Host: osd2                  connection: [connected  ]
Collection completed!

=======================  Starting remote check session
========================
Version: 1.0.2    Cluster Name: "example"
Total hosts: [6]
OSDs:    3    MONs:    3     Clients:    0
MDSs:    0    RGWs:    0     MGRs:       0

====================================================================
============

------------ osds ------------
 osd0
 osd1
 osd2

------------ mons ------------
 mon0
 mon1
 mon2

17 passed, 0 errors, on 6 hosts
```

Additional Resources

- Appendix B, *Error Code Definitions for* ***`ceph-medic`***
- The *Enabling Passwordless SSH for Ansible* section in the Red Hat Ceph Storage 3 *Installation Guide for Red Hat Enterprise Linux*

7.3.3. Using a Custom Inventory File

The **ceph-medic** utility must know the storage cluster topology. By default, **ceph-medic** uses the Ansible inventory file (**/etc/ansible/hosts**) to detect nodes.

Prerequisites

- A working Red Hat Ceph Storage cluster
- SSH and **sudo** access to storage nodes

Procedure
To use a custom inventory file, do the following steps on the Ansible administration node, as a user.

1. Create a custom **hosts** file:

   ```
   [admin@admin ~]$ touch ~/example/hosts
   ```

2. Open the **hosts** file for editing. Add the nodes in the storage cluster under the appropriate node group type. The **ceph-medic** tool supports the following node group types: **mons**, **osds**, **rgws**, **mdss**, **mgrs** and **clients**.
   For example:

   ```
   [mons]
   mon0
   mon1
   mon2

   [osds]
   osd0
   osd1
   osd2
   ```

3. When doing a diagnostic check, to specify a custom inventory file, use the **--inventory** option:

   ```
   ceph-medic --inventory $PATH_TO_HOSTS_FILE check
   ```

   Replace

   - **$PATH_TO_HOSTS_FILE** with the full path to the **hosts** file.
     For example:

     ```
     [admin@admin ~]$ ceph-medic --inventory ~/example/hosts check
     ```

Additional Resources

- Appendix B, *Error Code Definitions for* ***ceph-medic***
- The *Enabling Passwordless SSH for Ansible* section in the Red Hat Ceph Storage 3 *Installation Guide for Red Hat Enterprise Linux*

7.3.4. Configuring a Custom Logging Path

The **`ceph-medic`** log file contains more verbosity than the command output on the terminal. The **`ceph-medic`** tool by default writes logs to the current working directory.

Prerequisites

- A working Red Hat Ceph Storage cluster
- SSH and **`sudo`** access to storage nodes

Procedure

To change where these logs are written, do the following step on the Ansible administration node, as a normal user.

1. Open the **`~/.cephmedic.conf`** file for editing.
2. Change the **`--log-path`** option, from a **`.`** to a custom log location.
   For example:

   ```
   --log-path = /var/log/ceph-medic/
   ```

Additional Resources

- Appendix B, *Error Code Definitions for* ***`ceph-medic`***
- The *Enabling Passwordless SSH for Ansible* section in the Red Hat Ceph Storage 3 *Installation Guide for Red Hat Enterprise Linux*

7.3.5. Additional Resources

- Appendix B, *Error Code Definitions for* ***`ceph-medic`***

CHAPTER 8. CONTACTING RED HAT SUPPORT SERVICE

If the information in this guide did not help you to solve the problem, this chapter explains how you contact the Red Hat Support Service.

8.1. PROVIDING INFORMATION TO RED HAT SUPPORT ENGINEERS

If you are unable to fix problems related to Red Hat Ceph Storage by yourself, contact the Red Hat Support Service and provide sufficient amount of information that helps the support engineers to faster troubleshoot the problem you encounter.

Procedure: Providing Information to Red Hat Support Engineers

1. Open a support ticket on the Red Hat Customer Portal.

2. Ideally, attach an **sosreport** to the ticket. See the What is a sosreport and how to create one in Red Hat Enterprise Linux 4.6 and later? solution for details.

3. If the Ceph daemons failed with a segmentation fault, consider generating a human-readable core dump file. See Section 8.2, “Generating Readable Core Dump Files” for details.

8.2. GENERATING READABLE CORE DUMP FILES

When a Ceph daemon terminates unexpectedly with a segmentation fault, gather the information about its failure and provide it to the Red Hat Support Engineers.

Such information speeds up the initial investigation. Also, the Support Engineers can compare the information from the core dump files with Red Hat Ceph Storage known issues.

Before You Start

1. Install the **ceph-debuginfo** package if it is not installed already.

   a. Enable the repository containing the **ceph-debuginfo** package:

   ```
   subscription-manager repos --enable=rhel-7-server-rhceph-3-
   <daemon>-debug-rpms
   ```

   Replace **<daemon>** with **osd** or **mon** depending on the type of the node.

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

   ```
   # yum install ceph-debuginfo
   ```

2. Ensure that the **gdb** package is installed and if it is not, install it:

   ```
   # yum install gdb
   ```

Procedure: Generating Readable Core Dump Files

1. Enable generating core dump files for Ceph.

   a. Set the proper **ulimits** for the core dump files by adding the following parameter to the **/etc/systemd/system.conf** file:

```
DefaultLimitCORE=infinity
```

b. Comment out the **PrivateTmp=true** parameter in the Ceph daemon service file, by default located at **/lib/systemd/system/<cluster-name>-<daemon>@.service**:

```
# PrivateTmp=true
```

c. Set the **suid_dumpable** flag to **2** to allow the Ceph daemons to generate dump core files:

```
# sysctl fs.suid_dumpable=2
```

d. Adjust the core dump files location:

```
# sysctl kernel.core_pattern=/tmp/core
```

e. Reload the **systemd** service for the changes to take effect:

```
# systemctl daemon-reload
```

f. Restart the Ceph daemon for the changes to take effect:

```
systemctl restart ceph-<daemon>@<ID>
```

Specify the daemon type (**osd** or **mon**) and its ID (numbers for OSDs, or short host names for Monitors) for example:

```
# systemctl restart ceph-osd@1
```

2. Reproduce the failure, for example try to start the daemon again.

3. Use the GNU Debugger (GDB) to generate a readable backtrace from an application core dump file:

```
gdb /usr/bin/ceph-<daemon> /tmp/core.<PID>
```

Specify the daemon type and the PID of the failed process, for example:

```
$ gdb /usr/bin/ceph-osd /tmp/core.123456
```

In the GDB command prompt disable paging and enable logging to a file by entering the commands **set pag off** and **set log on**:

```
(gdb) set pag off
(gdb) set log on
```

Apply the **backtrace** command to all threads of the process by entering **thr a a bt full**:

```
(gdb) thr a a bt full
```

After the backtrace is generated turn off logging by entering **set log off**:

```
(gdb) set log off
```

4. Transfer the log file **`gdb.txt`** to the system you access the Red Hat Customer Portal from and attach it to a support ticket.

See Also

- The How to use gdb to generate a readable backtrace from an application core solution on the Red Hat Customer Portal
- The How to enable core file dumps when an application crashes or segmentation faults solution on the Red Hat Customer Portal

APPENDIX A. SUBSYSTEMS DEFAULT LOGGING LEVELS VALUES

Subsystem	Log Level	Memory Level
`asok`	1	5
`auth`	1	5
`buffer`	0	0
`client`	0	5
`context`	0	5
`crush`	1	5
`default`	0	5
`filer`	0	5
`filestore`	1	5
`finisher`	1	5
`heartbeatmap`	1	5
`javaclient`	1	5
`journaler`	0	5
`journal`	1	5
`lockdep`	0	5
`mds balancer`	1	5
`mds locker`	1	5
`mds log expire`	1	5
`mds log`	1	5
`mds migrator`	1	5
`mds`	1	5

Subsystem	Log Level	Memory Level
`monc`	0	5
`mon`	1	5
`ms`	0	5
`objclass`	0	5
`objectcacher`	0	5
`objecter`	0	0
`optracker`	0	5
`osd`	0	5
`paxos`	0	5
`perfcounter`	1	5
`rados`	0	5
`rbd`	0	5
`rgw`	1	5
`throttle`	1	5
`timer`	0	5
`tp`	0	5

APPENDIX B. ERROR CODE DEFINITIONS FOR `CEPH-MEDIC`

The **`ceph-medic`** utility displays an error code and message for any failing checks. These error codes are a warning or an error and apply to common issues specific to the Ceph daemons. The error codes start with *E* and warning codes start with *W*.

COMMON ERROR MESSAGES

`ECOM1`

A Ceph configuration file can not be found at **`/etc/ceph/$CLUSTER_NAME.conf`**.

`ECOM2`

The **`ceph`** executable was not found.

`ECOM3`

The **`/var/lib/ceph/`** directory does not exist or could not be accessed.

`ECOM4`

The **`/var/lib/ceph/`** directory is not owned by the **`ceph`** user.

`ECOM5`

The **`fsid`** in the Ceph configuration is different between the nodes in the storage cluster.

MONITOR ERROR MESSAGES

`EMON1`

The secret key used in the keyring is different between the nodes in the storage cluster.

MONITOR WARNING MESSAGES

`WMON1`

Multiple Monitor directories are found on the same node.

`WMON2`

Collocated OSDs where found on the Monitors nodes.

OSD WARNING MESSAGES

`WOSD1`

Multiple **`ceph_fsid`** values where found in the **`/var/lib/ceph/osd/`** directory.