SUSE Enterprise Storage 7.1

Troubleshooting Guide

Troubleshooting Guide

SUSE Enterprise Storage 7.1

by Tomáš Bažant, Alexandra Settle, and Liam Proven

Publication Date: 12 Dec 2024

https://documentation.suse.com ↗

For SUSE trademarks, see http://www.suse.com/company/legal/. All third-party trademarks are the property of their respective owners. Trademark symbols (®, ™ etc.) denote trademarks of SUSE and its affiliates. Asterisks (*) denote third-party trademarks.

All information found in this book has been compiled with utmost attention to detail. However, this does not guarantee complete accuracy. Neither SUSE LLC, its affiliates, the authors nor the translators shall be held liable for possible errors or the consequences thereof.

Contents

About this guide

This guide takes you through various common problems when running SUSE Enterprise Storage 7.1 and other related issues to relevant components such as Ceph or Object Gateway.

SUSE Enterprise Storage 7.1 is an extension to SUSE Linux Enterprise Server 15 SP3. It combines the capabilities of the Ceph (http://ceph.com/ ↗) storage project with the enterprise engineering and support of SUSE. SUSE Enterprise Storage 7.1 provides IT organizations with the ability to deploy a distributed storage architecture that can support a number of use cases using commodity hardware platforms.

1 Available documentation

Note: Online documentation and latest updates

Documentation for our products is available at https://documentation.suse.com ↗, where you can also find the latest updates, and browse or download the documentation in various formats. The latest documentation updates can be found in the English language version.

In addition, the product documentation is available in your installed system under `/usr/share/doc/manual`. It is included in an RPM package named `ses-manual_LANG_CODE`. Install it if it is not already on your system, for example:

```
# zypper install ses-manual_en
```

The following documentation is available for this product:

***Deployment Guide* (https://documentation.suse.com/ses/html/ses-all/book-storage-deployment.html) ↗**

This guide focuses on deploying a basic Ceph cluster, and how to deploy additional services. It also cover the steps for upgrading to SUSE Enterprise Storage 7.1 from the previous product version.

Administration and Operations Guide (https://documentation.suse.com/ses/html/ses-all/book-storage-admin.html)

This guide focuses on routine tasks that you as an administrator need to take care of after the basic Ceph cluster has been deployed (day 2 operations). It also describes all the supported ways to access data stored in a Ceph cluster.

Security Hardening Guide (https://documentation.suse.com/ses/html/ses-all/book-storage-security.html)

This guide focuses on how to ensure your cluster is secure.

Troubleshooting Guide (https://documentation.suse.com/ses/html/ses-all/book-storage-troubleshooting.html)

This guide takes you through various common problems when running SUSE Enterprise Storage 7.1 and other related issues to relevant components such as Ceph or Object Gateway.

SUSE Enterprise Storage for Windows Guide (https://documentation.suse.com/ses/html/ses-all/book-storage-windows.html)

This guide describes the integration, installation, and configuration of Microsoft Windows environments and SUSE Enterprise Storage using the Windows Driver.

2 Improving the documentation

Your feedback and contributions to this documentation are welcome. The following channels for giving feedback are available:

Service requests and support

For services and support options available for your product, see http://www.suse.com/support/.

To open a service request, you need a SUSE subscription registered at SUSE Customer Center. Go to https://scc.suse.com/support/requests, log in, and click *Create New*.

Bug reports

Report issues with the documentation at https://bugzilla.suse.com/. A Bugzilla account is required.

To simplify this process, you can use the *Report Documentation Bug* links next to headlines in the HTML version of this document. These preselect the right product and category in Bugzilla and add a link to the current section. You can start typing your bug report right away.

Contributions

To contribute to this documentation, use the *Edit Source* links next to headlines in the HTML version of this document. They take you to the source code on GitHub, where you can open a pull request. A GitHub account is required.

Note: *Edit Source* only available for English

The *Edit Source* links are only available for the English version of each document. For all other languages, use the *Report Documentation Bug* links instead.

For more information about the documentation environment used for this documentation, see the repository's README at https://github.com/SUSE/doc-ses.

Mail

You can also report errors and send feedback concerning the documentation to `doc-team@suse.com`. Include the document title, the product version, and the publication date of the document. Additionally, include the relevant section number and title (or provide the URL) and provide a concise description of the problem.

3 Documentation conventions

The following notices and typographic conventions are used in this document:

- `/etc/passwd`: Directory names and file names
- *`PLACEHOLDER`*: Replace *`PLACEHOLDER`* with the actual value
- `PATH`: An environment variable
- **`ls`**, `--help`: Commands, options, and parameters
- `user`: The name of user or group
- `package_name`: The name of a software package

- Alt , Alt – F1 : A key to press or a key combination. Keys are shown in uppercase as on a keyboard.

- *File*, *File › Save As*: menu items, buttons

- AMD/Intel This paragraph is only relevant for the AMD64/Intel 64 architectures. The arrows mark the beginning and the end of the text block.
  IBM Z, POWER This paragraph is only relevant for the architectures `IBM Z` and `POWER`. The arrows mark the beginning and the end of the text block.

- *Chapter 1, "Example chapter"*: A cross-reference to another chapter in this guide.

- Commands that must be run with `root` privileges. Often you can also prefix these commands with the **`sudo`** command to run them as non-privileged user.

```
# command
> sudo command
```

- Commands that can be run by non-privileged users.

```
> command
```

- Notices

Warning: Warning notice

Vital information you must be aware of before proceeding. Warns you about security issues, potential loss of data, damage to hardware, or physical hazards.

Important: Important notice

Important information you should be aware of before proceeding.

Note: Note notice

Additional information, for example about differences in software versions.

Tip: Tip notice

Helpful information, like a guideline or a piece of practical advice.

- Compact Notices

Additional information, for example about differences in software versions.

Helpful information, like a guideline or a piece of practical advice.

4 Support

Find the support statement for SUSE Enterprise Storage and general information about technology previews below. For details about the product lifecycle, see https://www.suse.com/lifecycle.

If you are entitled to support, find details on how to collect information for a support ticket at https://documentation.suse.com/sles-15/html/SLES-all/cha-adm-support.html.

4.1 Support statement for SUSE Enterprise Storage

To receive support, you need an appropriate subscription with SUSE. To view the specific support offerings available to you, go to https://www.suse.com/support/ and select your product.

The support levels are defined as follows:

L1

Problem determination, which means technical support designed to provide compatibility information, usage support, ongoing maintenance, information gathering and basic troubleshooting using available documentation.

L2

Problem isolation, which means technical support designed to analyze data, reproduce customer problems, isolate problem area and provide a resolution for problems not resolved by Level 1 or prepare for Level 3.

L3

Problem resolution, which means technical support designed to resolve problems by engaging engineering to resolve product defects which have been identified by Level 2 Support.

For contracted customers and partners, SUSE Enterprise Storage is delivered with L3 support for all packages, except for the following:

- Technology previews.
- Sound, graphics, fonts, and artwork.
- Packages that require an additional customer contract.
- Some packages shipped as part of the module *Workstation Extension* are L2-supported only.
- Packages with names ending in `-devel` (containing header files and similar developer resources) will only be supported together with their main packages.

SUSE will only support the usage of original packages. That is, packages that are unchanged and not recompiled.

4.2 Technology previews

Technology previews are packages, stacks, or features delivered by SUSE to provide glimpses into upcoming innovations. Technology previews are included for your convenience to give you a chance to test new technologies within your environment. We would appreciate your feedback! If you test a technology preview, please contact your SUSE representative and let them know about your experience and use cases. Your input is helpful for future development.

Technology previews have the following limitations:

- Technology previews are still in development. Therefore, they may be functionally incomplete, unstable, or in other ways *not* suitable for production use.
- Technology previews are *not* supported.
- Technology previews may only be available for specific hardware architectures.

- Details and functionality of technology previews are subject to change. As a result, upgrading to subsequent releases of a technology preview may be impossible and require a fresh installation.
- SUSE may discover that a preview does not meet customer or market needs, or does not comply with enterprise standards. Technology previews can be removed from a product at any time. SUSE does not commit to providing a supported version of such technologies in the future.

For an overview of technology previews shipped with your product, see the release notes at https://www.suse.com/releasenotes/x86_64/SUSE-Enterprise-Storage/7.1.

5 Ceph contributors

The Ceph project and its documentation is a result of the work of hundreds of contributors and organizations. See https://ceph.com/contributors/ for more details.

6 Commands and command prompts used in this guide

As a Ceph cluster administrator, you will be configuring and adjusting the cluster behavior by running specific commands. There are several types of commands you will need:

6.1 Salt-related commands

These commands help you to deploy Ceph cluster nodes, run commands on several (or all) cluster nodes at the same time, or assist you when adding or removing cluster nodes. The most frequently used commands are **`ceph-salt`** and **`ceph-salt config`**. You need to run Salt commands on the Salt Master node as `root`. These commands are introduced with the following prompt:

```
root@master #
```

For example:

```
root@master # ceph-salt config ls
```

6.2 Ceph related commands

These are lower-level commands to configure and fine tune all aspects of the cluster and its gateways on the command line, for example **`ceph`**, **`cephadm`**, **`rbd`**, or **`radosgw-admin`**.

To run Ceph related commands, you need to have read access to a Ceph key. The key's capabilities then define your privileges within the Ceph environment. One option is to run Ceph commands as `root` (or via **`sudo`**) and use the unrestricted default keyring 'ceph.client.admin.key'.

The safer and recommended option is to create a more restrictive individual key for each administrator user and put it in a directory where the users can read it, for example:

```
~/.ceph/ceph.client.USERNAME.keyring
```

Tip: Path to Ceph keys

To use a custom admin user and keyring, you need to specify the user name and path to the key each time you run the **`ceph`** command using the `-n client.USER_NAME` and `--keyring PATH/TO/KEYRING` options.

To avoid this, include these options in the `CEPH_ARGS` variable in the individual users' `~/.bashrc` files.

Although you can run Ceph-related commands on any cluster node, we recommend running them on the Admin Node. This documentation uses the `cephuser` user to run the commands, therefore they are introduced with the following prompt:

```
cephuser@adm >
```

For example:

```
cephuser@adm > ceph auth list
```

Tip: Commands for specific nodes

If the documentation instructs you to run a command on a cluster node with a specific role, it will be addressed by the prompt. For example:

```
cephuser@mon >
```

6.2.1 Running **ceph-volume**

Starting with SUSE Enterprise Storage 7, Ceph services are running containerized. If you need to run **ceph-volume** on an OSD node, you need to prepend it with the **cephadm** command, for example:

```
cephuser@adm > cephadm ceph-volume simple scan
```

6.3 General Linux commands

Linux commands not related to Ceph, such as **mount**, **cat**, or **openssl**, are introduced either with the `cephuser@adm >` or `#` prompts, depending on which privileges the related command requires.

6.4 Additional information

For more information on Ceph key management, refer to *Book "Administration and Operations Guide", Chapter 30 "Authentication with* `cephx`*", Section 30.2 "Key management"*.

1 Reporting software problems

If you come across a problem when running SUSE Enterprise Storage 7.1 related to some of its components, such as Ceph or Object Gateway, report the problem to SUSE Technical Support. The recommended way is with the **supportconfig** utility.

Tip

Because **supportconfig** is modular software, make sure that the `supportutils-plugin-ses` package is installed.

```
> rpm -q supportutils-plugin-ses
```

If it is missing on the Ceph server, install it with:

```
# zypper ref && zypper in supportutils-plugin-ses
```

`supportutils-plugin-ses` will try to capture as much useful information as it can about both the Ceph cluster as a whole, and about the host on which it is running. Some information, including the Ceph daemon's status, **podman** image information, **rpm** verification and some relevant configuration data will be saved to the `plugin-ses.txt` file inside the `supportconfig` **tar** archive. A lot more information, including overall cluster status, logs, and diagnostic data for each currently running Ceph daemon will be saved to files in the Ceph subdirectory inside the supportconfig tar archive.

When **supportconfig** is run on an Admin Node, it will be able to capture the overall status of the entire cluster. For example, **ceph status** and **ceph health**. If the `ceph.conf` file and admin keyring are not present, it will not be able to capture overall cluster status, but still capture diagnostic data for each currently running Ceph daemon.

In general, when using **supportconfig**, you will want to run it on an admin node (to ensure the overall cluster status is captured), as well as on any other nodes that are experiencing problems, to ensure it captures diagnostic data for anything and everything that may be relevant.

Although you can use **supportconfig** on the command line, we recommend using the related YaST module. Find more information about **supportconfig** in https://documentation.suse.com/sles/15-SP3/html/SLES-all/cha-adm-support.html#sec-admsupport-supportconfig ↗.

2 Troubleshooting logging and debugging

Typically, when you add debugging to your Ceph configuration, you do so at runtime. You can also add Ceph debug logging to your Ceph configuration file if you are encountering issues when starting your cluster. You may view Ceph log files under `/var/log/ceph` (the default location).

Tip

When debug output slows down your system, the latency can hide race conditions.

Logging is resource intensive. If you are encountering a problem in a specific area of your cluster, enable logging for that area of the cluster. For example, if your OSDs are running fine, but your metadata servers are not, you should start by enabling debug logging for the specific metadata server instance(s) giving you trouble. Enable logging for each subsystem as needed.

Important

Verbose logging can generate over 1GB of data per hour. If your OS disk reaches its capacity, the node will stop working.

If you enable or increase the rate of Ceph logging, ensure that you have sufficient disk space on your OS disk. See *Section 2.3, "Accelerating log rotation"* for details on rotating log files. When your system is running well, remove unnecessary debugging settings to ensure your cluster runs optimally. Logging debug output messages is relatively slow, and a waste of resources when operating your cluster. See *Section 2.5, "Enable system, log, and debug settings"* for details on available settings.

2.1 Accessing configuration settings at runtime

If you would like to see the configuration settings at runtime, you must log in to a host with a running daemon and execute the following:

```
cephuser@adm > cephadm enter --name osd.ID -- ceph daemon osd.ID config show | less
```

For example:

```
cephuser@adm > cephadm enter --name osd.0 -- ceph daemon osd.0 config show | less
```

To activate Ceph's debugging output (i.e., `dout()`) at runtime, use the **`ceph tell`** command to inject arguments into the runtime configuration:

```
cephuser@adm > ceph tell {daemon-type}.{daemon id or *} config set {name} {value}
```

Replace `{daemon-type}` with one of osd, mon or mds. You may apply the runtime setting to all daemons of a particular type with `*`, or specify a specific daemon's ID. For example, to increase debug logging for a ceph-osd daemon named `osd.0`, execute the following:

```
cephuser@adm > ceph tell osd.0 config set debug_osd 0/5
```

The **`ceph tell`** command goes through the monitors. If you cannot bind to the monitor, you can make the change by logging into the daemon host using ceph daemon. For example:

```
cephuser@adm > cephadm enter --name osd.0 -- ceph daemon osd.0 config set debug_osd 0/5
```

See *Section 2.5, "Enable system, log, and debug settings"* for details on available settings.

2.2 Activating Ceph debugging at boot time

To activate Ceph's debugging output (i.e., `dout()`) at boot time, you must add settings to your Ceph configuration file. Subsystems common to each daemon may be set under `[global]` in your configuration file. Subsystems for particular daemons are set under the daemon section in your configuration file (e.g., `[mon]`, `[osd]`, `[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
        debug mds balancer = 1
```

See *Section 2.5, "Enable system, log, and debug settings"* for details.

2.3 Accelerating log rotation

If your OS disk is relatively full, you can accelerate log rotation by modifying the Ceph log rotation file at `etc/logrotate.d/ceph`. Add a size setting after the rotation frequency to accelerate log rotation (via cronjob) if your logs exceed the size setting. For example, the default setting looks like this:

```
  rotate 7
  weekly
  compress
  sharedscripts
```

Modify it by adding a size setting:

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

Then, start the crontab editor for your user space:

```
crontab -e
```

Finally, add an entry to check the `etc/logrotate.d/ceph` file:

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

The preceding example checks the `etc/logrotate.d/ceph` file every 30 minutes.

2.4 Monitoring memory utilization

To monitor the memory utilization of the Ceph processes, pay attention to the **`top`** command's VIRT or the output of **`ps -u ceph -l`** in the `VSZ` column. While some fluctuation of memory utilization over time is normal, it should not constantly increase and this will help you pinpoint the problematic process if facing low- or out-of-memory concerns.

2.5 Enable system, log, and debug settings

In most cases, you will enable debug logging output via subsystems.

2.5.1 Enabling Ceph subsystems

Each subsystem has a logging level for its output logs, and for its logs in-memory. You may set different values for each of these subsystems by setting a log file level and a memory level for debug logging. Ceph's logging levels operate on a scale of 1 to 20, where 1 is terse and 20 is verbose 1. In general, the logs in-memory are not sent to the output log unless:

- a fatal signal is raised or
- an `assert` in source code is triggered or
- upon request.

A debug logging setting can take a single value for the log level and the memory level, which sets them both as the same value. For example, if you specify debug ms = 5, Ceph will treat it as a log level and a memory level of 5. You may also specify them separately. The first setting is the log level, and the second setting is the memory level. You must separate them with a forward slash (/). For example, if you want to set the ms subsystem's debug logging level to 1 and its memory level to 5, you would specify it as debug ms = 1/5. For example:

```
debug {subsystem} = {log-level}/{memory-level}
#for example
debug mds balancer = 1/20
```

The following table provides a list of Ceph subsystems and their default log and memory levels. Once you complete your logging efforts, restore the subsystems to their default level or to a level suitable for normal operations.

TABLE 2.1: CEPH SUBSYSTEMS

Subsystem	Log Level	Memory Level
`default`	0	5
`lockdep`	0	1
`context`	0	1
`crush`	1	1
`mds`	1	5
`mds balancer`	1	5
`mds locker`	1	5

Subsystem	Log Level	Memory Level
`mds log`	1	5
`mds log expire`	1	5
`mds migrator`	1	5
`buffer`	0	1
`timer`	0	1
`filer`	0	1
`striper`	0	1
`objector`	0	1
`rados`	0	5
`rbd`	0	5
`rbd mirror`	0	5
`rbd replay`	0	5
`journaler`	0	5
`objectcacher`	0	5
`client`	0	5
`osd`	1	5
`optracker`	0	5
`optracker`	0	5
`objclass`	0	5
`filestore`	1	3
`journal`	1	3
`ms`	0	5
`mon`	1	5

Subsystem	Log Level	Memory Level
`monc`	0	10
`paxos`	1	5
`tp`	0	5
`auth`	1	5
`crypto`	1	5
`finisher`	1	1
`reserver`	1	1
`heartbeatmap`	1	5
`perfcounter`	1	5
`rgw`	1	5
`rgw sync`	1	5
`civetweb`	1	10
`javaclient`	1	5
`asok`	1	5
`throttle`	1	1
`refs`	0	0
`compressor`	1	5
`bluestore`	1	5
`bluefs`	1	5
`bdev`	1	3
`kstore`	1	5
`rocksdb`	4	5
`leveldb`	4	5

Subsystem	Log Level	Memory Level
`memdb`	4	5
`fuse`	1	5
`mgr`	1	5
`mgrc`	1	5
`dpdk`	1	5
`eventtrace`	1	5

2.5.2 Logging settings

Logging and debugging settings are not required in a Ceph configuration file, but you may override default settings as needed. Ceph supports the following settings:

`log file`

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

Type: String

Required: No

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

`log max new`

Description: The maximum number of new log files.

Type: Integer

Required: No

Default: 1000

`log max recent`

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

Type: Integer

Required: No

Default: 10000

`log to stderr`

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

Type: Boolean

Required: No

Default: `true`

`err to stderr`

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

Type: Boolean

Required: No

Default: `true`

`log to syslog`

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

Type: Boolean

Required: No

Default: `false`

`err to syslog`

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

Type: Boolean

Required: No

Default: `false`

`log flush on exit`

Description: Determines if Ceph should flush the log files after exit.

Type: Boolean

Required: No

Default: `true`

`clog to monitors`

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

Type: Boolean

Required: No

Default: `true`

`clog to syslog`

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

Type: Boolean

Required: No

Default: `false`

`mon cluster log to syslog`

Description: Determines if the cluster log messages should be output to the syslog.

Type: Boolean

Required: No

Default: `false`

`mon cluster log file`

Description: The locations of the cluster's log files. There are two channels in Ceph: `cluster` and `audit`. This option represents a mapping from channels to log files, where the log entries of that channel are sent to. The default entry is a fallback mapping for channels not explicitly specified. So, the following default setting will send cluster log to `$cluster.log`, and send audit log to `$cluster.audit.log`, where `$cluster` will be replaced with the actual cluster name.

Type: String

Required: No

Default: `default=/var/log/ceph/$cluster.$channel.log,cluster=/var/log/ceph/$cluster.log`

2.5.3 OSD

`osd debug drop ping probability`

Description: ?

Type: Double

Required: No

Default: 0

`osd debug drop ping duration`

Description:

Type: Integer

Required: No

Default: 0

`osd debug drop pg create probability`

Description:

Type: Integer

Required: No

Default: 0

`osd debug drop pg create duration`

Description: ?

Type: Double

Required: No

Default: 1

`osd min pg log entries`

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

Type: 32-bit Unsigned Integer

Required: No

Default: 250

`osd op log threshold`

Description: How many op log messages to show up in one pass.

Type: Integer

Required: No

Default: 5

2.5.4 Filestore

`filestore debug omap check`

Description: Debugging check on synchronization. This is an expensive operation.

Type: Boolean

Required: No

Default: `false`

2.5.5 MDS

`mds debug scatterstat`

Description: Ceph will assert that various recursive stat invariants are true (for developers only).

Type: Boolean

Required: No

Default: `false`

`mds debug frag`

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

Type: Boolean

Required: No

Default: `false`

`mds debug auth pins`

Description: The debug auth pin invariants (for developers only).

Type: Boolean

Required: No

Default: `false`

`mds debug subtrees`

Description: The debug subtree invariants (for developers only).

Type: Boolean

Required: No

Default: `false`

2.5.6 RADOS gateway (RGW)

`rgw log nonexistent bucket`

Description: Should we log a non-existent buckets?

Type: Boolean

Required: No

Default: `false`

`rgw log object name`

Description: Should an object's name be logged. // man date to see codes (a subset are supported)

Type: String

Required: No

Default: `%Y-%m-%d-%H-%i-%n`

`rgw log object name utc`

Description: Object log name contains UTC?

Type: Boolean

Required: No

Default: `false`

`rgw enable ops log`

Description: Enables logging of every RGW operation.

Type: Boolean

Required: No

Default: `true`

`rgw enable usage log`

Description: Enable logging of RGW’s bandwidth usage.

Type: Boolean

Required: No

Default: `false`

`rgw usage log flush threshold`

Description: Threshold to flush pending log data.

Type: Integer

Required: No

Default: 1024

`rgw usage log tick interval`

Description: Flush pending log data every s seconds.

Type: Integer

Required: No

Default: 30

`rgw intent log object name`

Description:

Type: String

Required: No

Default: `%Y-%m-%d-%i-%n`

`rgw intent log object name utc`

Description: Include a UTC timestamp in the intent log object name.

Type: Boolean

Required: No

Default: `false`

2.6 Logging kernel RBD and CephFS clients

The RBD and CephFS kernel clients provide means to enable runtime logging that allow low-level debugging. However, the performance penalty may be high, which is why dynamic debugging is turned off by default. Also, analysing these logs requires comparing them with the kernel source code.

For example, to see the debug messages related with CephFS snapshot handling, you could try the following:

```
# echo "file fs/ceph/snap.c +pf" > /sys/kernel/debug/dynamic_debug/control
```

The kernel logs would now include any debug messages in file `fs/ceph/snap.c`. In order to disable dynamic debugging:

```
# echo "file fs/ceph/snap.c -p" > /sys/kernel/debug/dynamic_debug/control
```

2.7 Per-service and per-daemon events

In order to simplify debugging failed daemon deployments, cephadm stores events on a per-service and per-daemon basis. To view the list of events for a specific *service*, run the following example command:

```
cephuser@adm > ceph orch ls --service_name=alertmanager --format yaml
  service_type: alertmanager
  service_name: alertmanager
  placement:
    hosts:
    - unknown_host
[...]
  events:
  - 2021-02-01T08:58:02.741162 service:alertmanager [INFO] "service was created"
  - 2021-02-01T12:09:25.264584 service:alertmanager [ERROR] "Failed to apply: \
   Cannot place <AlertManagerSpec for service_name=alertmanager> on \
   unknown_host: Unknown hosts"'
```

As opposed to a specific service, to view the list of events for a specific *daemon*, for example, run the following command:

```
cephuser@adm > ceph orch ceph --service-type mds \
 --daemon-id=hostname.ppdhsz --format yaml
  daemon_type: mds
  daemon_id: cephfs.hostname.ppdhsz
  hostname: hostname
[...]
  events:
  - 2021-02-01T08:59:43.845866 daemon:mds.cephfs.hostname.ppdhsz [INFO] \
   "Reconfigured mds.cephfs.hostname.ppdhsz on host 'hostname'"
```

2.8 Debugging a crashing process running in a container

When a process is crashing, it is useful to find out the reason why it is crashing. If the `systemd-coredump` service is active, a dump of the memory of the crashing process—also referred to as *core dump*—is created. It is then possible to attach the **`gdb`** debugger to the dump and inspect the cause of the crash.

The following procedure illustrates attaching the **`gdb`** debugger to a crashing OSD process. Run the commands in this example on the host where the crashing process is running.

1. Verify that the `systemd-coredump` service is started and start it if needed:

   ```
   > sudo systemctl start systemd-coredump.socket
   ```

   Core dump files will be located under the `/var/lib/systemd/coredump` directory. In this example, the core dump file to be inspected is

   ```
   /var/lib/systemd/coredump/core.ceph-
   osd.167.055db41886e04e5e9822dcb5ad3c2aab.48529.1657718534000000.zst
   ```

2. Verify that the **`strings`** and **`unzstd`** commands are installed. If not, install them by running the following command:

   ```
   > sudo zypper install binutils unzstd
   ```

3. Identify the name of the container with the crashing process:

   ```
   > sudo unzstd /var/lib/systemd/coredump/core.ceph-
   osd.167.055db41886e04e5e9822dcb5ad3c2aab.48529.1657718534000000.zst
   > strings -a /var/lib/systemd/coredump/core.ceph-
   osd.167.055db41886e04e5e9822dcb5ad3c2aab.48529.1657718534000000 | grep
    CONTAINER_IMAGE
   CONTAINER_IMAGE=registry.suse.de/devel/storage/7.0/pacific/containers/ses/7.1/ceph/
   ceph@sha256:5d99dbf24e01ebd5319549d4db372113892c57f2d53fbb0363e0781a8c1c09e6
   ```

4. Run the identified container:

   ```
   > sudo podman run -it -v /var/lib/systemd/coredump:/mnt/coredump \
    registry.suse.de/devel/storage/7.0/pacific/containers/ses/7.1/ceph/
   ceph@sha256:5d99dbf24e01ebd5319549d4db372113892c57f2d53fbb0363e0781a8c1c09e6
   ```

5. Add a repository for packages with debugging information and install them together with the `gdb` package:

   ```
   > sudo zypper ar http://download.nue.suse.com/ibs/SUSE/Products/Storage/7.1/x86_64/
   product_debug/ debug
   > sudo zypper ref
   > sudo zypper install gdb ceph-base-debuginfo ceph-debugsource ceph-common-debuginfo
   ```

6. Attach `gdb` to the unpacked core dump:

   ```
   > cd /mnt/coredump
   > gdb /usr/bin/ceph-osd \
    core.ceph-osd.167.055db41886e04e5e9822dcb5ad3c2aab.48529.165771853400000
   ```

3 Troubleshooting cephadm

As cephadm deploys daemons as containers, troubleshooting daemons is slightly different. Here are a few tools and commands to help investigating issues.

3.1 Pausing or disabling cephadm

If something goes wrong and cephadm is behaving in a way you do not like, you can pause most background activity by executing the following:

```
cephuser@adm > ceph orch pause
```

This will stop any changes, but cephadm will still periodically check hosts to refresh its inventory of daemons and devices. You can disable cephadm completely with:

```
cephuser@adm > ceph orch set backend ''
cephuser@adm > ceph mgr module disable cephadm
```

This disables all of the **`ceph orch ...`** CLI commands but the previously deployed daemon containers will continue to exist and start as they did before.

3.2 Checking cephadm logs

You can monitor the cephadm log in real time:

```
cephuser@adm > ceph -W cephadm
```

To view the last few messages, execute:

```
cephuser@adm > ceph log last cephadm
```

3.3 Accessing Ceph daemon logs

SUSE Enterprise Storage 7.1 supports Ceph logging via systemd-journald. To access the logs of Ceph daemons in SUSE Enterprise Storage 7.1, follow the instructions below.

1. Use the **`ceph orch ps`** command (or **`ceph orch ps`** *`node_name`* or **`ceph orch ps --daemon-type`** *`daemon_type`*) to find the cephadm name of the daemon where the host is running.

   ```
   cephuser@adm > ceph orch ps --daemon-type DAEMON_TYPE_NAME
   ```

   For example, if you want to view the logs for Prometheus, run the following command to find the name of the daemon:

   ```
   cephuser@adm > ceph orch ps --daemon-type prometheus
   NAME               HOST    STATUS         REFRESHED  AGE  VERSION  IMAGE NAME
                                               IMAGE ID      CONTAINER ID
   prometheus.main    main    running (65m)  7m ago     2h   2.32.1
    registry.suse.com/ses/7.1/ceph/prometheus-server:2.32.1  e77db6e75e78  7b11062150ab
   ```

   In this example, `prometheus.main` is the name and the host is `main`.

2. To view the daemon's logs, execute the following on the host where the daemon is running:

   ```
   cephuser@adm > cephadm logs --name DAEMON_NAME
   ```

   For example, to view the Prometheus logs from above:

   ```
   cephuser@adm > cephadm logs --name prometheus.main
   ```

3.4 Collecting systemd status

To print the state of a `systemd` unit, run:

```
systemctl status "ceph-$(cephadm shell ceph fsid)@service name.service";
```

To fetch all state of all daemons of a given host, run:

```
  fsid="$(cephadm shell ceph fsid)"
  for name in $(cephadm ls | jq -r '.[].name') ; do
    systemctl status "ceph-$fsid@$name.service" > $name;
  done
```

3.5 Listing configured container images

cephadm has a set of container images configured by default.

To get the value of the configured default container:

```
cephuser@adm > ceph config get mgr mgr/cephadm/OPTION_NAME
```

Where *OPTION_NAME* is any of the following names:

- container_image_base
- container_image_prometheus
- container_image_node_exporter
- container_image_alertmanager
- container_image_grafana
- container_image_haproxy
- container_image_keepalived

For example:

```
cephuser@adm > ceph config get mgr mgr/cephadm/container_image_base \
 registry.suse.com/ses/7.1/ceph/ceph
```

Note

`mgr/cephadm/container_image_base` is the default Ceph container image used by all services except the monitoring stack and the ingress stack.

3.6 Listing all downloaded container images

To list all container images that are downloaded on a host:

Note

`Image` may also be called `ImageID`.

```
  podman ps -a --format json | jq '.[].Image'
  "docker.io/library/centos:8"
  "registry.opensuse.org/opensuse/leap:15.2"
```

3.7 Running containers manually

cephadm writes small wrappers that run a containers. Refer to `/var/lib/ceph/`*`cluster-fsid`*`/`*`service-name`*`/unit.run` for the container execution command.

3.7.1 Assessing SSH errors

If you receive the following error:

```
xxxxxx.gateway_bootstrap.HostNotFound: -F /tmp/cephadm-conf-kbqvkrkw root@10.10.1.2
raise OrchestratorError('Failed to connect to %s (%s).  Check that the host is reachable
 and accepts  connections using the cephadm SSH key' % (host, addr)) from
orchestrator._interface.OrchestratorError: Failed to connect to 10.10.1.2 (10.10.1.2).
 Check that the host is reachable and accepts connections using the cephadm SSH key
```

You can verify the issue by trialing a few different options.

- Ensure cephadm has an SSH identity key:

  ```
  root@master # cephadm shell -- ceph config-key get mgr/cephadm/ssh_identity_key >
   key
  INFO:cephadm:Inferring fsid f8edc08a-7f17-11ea-8707-000c2915dd98
  INFO:cephadm:Using recent ceph image docker.io/ceph/ceph:v15 obtained 'mgr/cephadm/
  ssh_identity_key'
  root@master # chmod 0600 key
  ```

  If this fails, cephadm does not have a key. Fix this by running the following command:

  ```
  root@master # cephadm shell -- ceph cephadm generate-ssh-key
  ```

  Or:

  ```
  root@master # cat key | cephadm shell -- ceph cephadm set-ssk-key -i -
  ```

- Ensure that the SSH config is correct:

  ```
  root@master # cephadm shell -- ceph cephadm get-ssh-config > config
  ```

- Verify the connection to the host:

  ```
  root@master # ssh -F config -i key root@mon1
  ```

3.7.2 Verifying public key is in `authorized_keys`

To verify that the public key is in the `authorized_keys` file, run the following commands:

```
root@master # cephadm shell -- ceph config-key get mgr/cephadm/ssh_identity_pub > key.pub
root@master # grep "`cat key.pub`"  /root/.ssh/authorized_keys
```

3.8 Failing to infer CIDR network error

If you see one of the following errors:

```
ERROR: Failed to infer CIDR network for mon ip ***; pass --skip-mon-network to configure
 it later
```

or

```
Must set public_network config option or specify a CIDR network, ceph addrvec, or plain
 IP
```

You need to specify a subnet for Ceph Monitors:

```
cephuser@adm > ceph config set mon public_network MON_NETWORK
```

3.9 Accessing the admin socket

Each Ceph daemon provides an admin socket that bypasses the MONs. To access the admin socket, enter the daemon container on the host:

```
root@master # cephadm enter --name daemon-name
root@master # ceph --admin-daemon /var/run/ceph/ceph-daemon-name.asok config show
```

3.10 Deploying a Ceph Manager manually

Use the following steps to deploy a new Ceph Manager on a host.

1. Disable the cephadm scheduler, in order to prevent cephadm from removing the new Ceph Manager:

   ```
   cephuser@adm > ceph config set mgr mgr/cephadm/pause true
   ```

2. Create the auth entry:

```
cephuser@adm > ceph auth create DAEMON_NAME mon "profile mgr" osd "allow *" mds
 "allow *"
```

If you already have an auth entry, you can get the entry with the following command:

```
cephuser@adm > ceph auth get DAEMON_NAME mon "profile mgr" osd "allow *" mds "allow
 *"
```

3. Generate the `ceph.conf`:

```
cephuser@adm > ceph config generate-minimal-conf
```

4. Get the container image:

```
cephuser@adm > ceph config get "DAEMON_NAME" container_image
```

5. Create the config json:

```
cat config-json.json
{
"config": "# minimal ceph.conf for 8255263a-a97e-4934-822c-00bfe029b28f
\n[global]\n\tfsid = 8255263a-a97e-4934-822c-00bfe029b28f\n\tmon_host =
 [v2:192.168.178.28:40483/0,v1:192.168.178.28:40484/0]\n",
"keyring": "[mgr.hostname.smfvfd]\n\tkey = AQDSb0ZfJqfDNxAAhgAVuH8Wg0yaRUAHwXdTQA==
\n"
}
```

6. Deploy the daemon:

```
cephuser@adm > cephadm --image container-image deploy --fsid fsid --name DAEMON_NAME
 --config-json config-json.json --tcp-ports 42483
```

3.11 Distributing a program temporary fix (PTF)

Occasionally, in order to fix a given issue, SUSE will provide a set of packages known as a Program Temporary Fix (PTF). Such a PTF is fully supported by SUSE until the Maintenance Update (MU) containing a permanent fix has been released via the regular update repositories. Customers running PTF fixes will be notified through the related Service Request when a permanent patch for a PTF has been released.

PTFs are published on the `registry.suse.com` server through a path similar to:

```
registry.suse.com/ptf/PTF_NUMBER/ses/7.1/ceph/ceph:PTF-PTF_NUMBER
```

The following steps describe how to deploy PTF images on an existing Ceph cluster:

1. Note the *PTF_NUMBER* used in the L3 PTF process.

2. Determine the Ceph Monitor key that corresponds to the relevant service:

   ```
   cephuser@adm > ceph auth ls | grep DAEMON_TYPE
   ```

3. Receive the PTF:

   ```
   # podman pull \
   registry.suse.com/ptf/PTF_NUMBER/ses/7.1/ceph/ceph:PTF-PTF_NUMBER
   ```

4. Verify that the PTF is listed:

   ```
   # podman image ls
   ```

5. Set the PTF image for the relevant service, for example, iSCSI:

   ```
   cephuser@adm > ceph config set client.iscsi container_image \
   registry.suse.com/ptf/PTF_NUMBER/ses/7.1/ceph/ceph:PTF-PTF_NUMBER
   ```

6. Remove the specific iSCSI daemon that is causing problems, for example:

   ```
   cephuser@adm > ceph orch daemon rm iscsi.igw.sesblade16.zkomeh
   ```

7. The removed daemon will be deployed automatically using the PTF.

Important

If you plan to run the **`ceph-salt update`** command after applying a PTF (as described in *Book "Administration and Operations Guide", Chapter 13 "Operational tasks", Section 13.6.4 "Running the update"*), it will not affect the manual changes made by the temporary fix.

Conversely, running the **`ceph orch upgrade`** command (see *Book "Administration and Operations Guide", Chapter 13 "Operational tasks", Section 13.7.1 "Starting the update"*) will upgrade daemons manually deployed by the PTF.

3.12 Failure When Adding Hosts with cephadm

When cephadm fails to add a new cluster host—no matter whether by means of `ceph-salt` (see *Book "Deployment Guide", Chapter 7 "Deploying the bootstrap cluster using* `ceph-salt`*", Section 7.2.2 "Adding Salt Minions"*) or manually using the **`ceph orch host add`** command—you can diagnose the reason for failure. The failure message has the following pattern:

```
Failed to connect to HOSTNAME (FQDN).
```

For example:

```
cephuser@adm > ceph orch host add node5.example.com
Failed to connect to node5 (node5.example.com).
```

If the message includes a fully qualified domain name (FQDN, `node5.example com` in our example) as the *`HOSTNAME`*, try adding the host using its FQDN:

```
cephuser@adm > ceph orch host add node5.example.com
```

If the message includes a short host name (`node5` in our example) as the *`HOSTNAME`*, try adding the host using both its short name and FQDN:

```
cephuser@adm > ceph orch host add node5 node5.example.com
```

If either of the previous two commands succeed, `ceph-salt` is probably causing the problem. Otherwise, try adding the host using its short host name and IP address:

```
cephuser@adm > ceph orch host add node5 192.168.2.102
```

If this succeeds, the cluster does not have proper name resolution. Refer to *Book "Deployment Guide", Chapter 5 "Installing and configuring SUSE Linux Enterprise Server"* for more details.

Tip

For more information about resolving full and bare host names via DNS, refer to https://docs.ceph.com/en/octopus/cephadm/concepts/ ↗.

3.13 Disabling automatic deployment of daemons

By default, Ceph daemons are automatically deployed to new hosts after you add these hosts to the placement specification and apply it (refer to *Book "Deployment Guide", Chapter 8 "Deploying the remaining core services using cephadm"* for more details).

If you need to disable the automated deployment of Ceph daemons, add `unmanaged: true` to its specification file and apply it, for example:

```
cat mgr.yaml
service_type: mgr
unmanaged: true
placement:
  label: mgr
[...]

cephuser@adm > ceph orch apply -i mgr.yaml
```

After applying this specification, cephadm will no longer deploy any new daemons on hosts that match the placement specification.

To manually deploy a daemon on a new host, run:

```
ceph orch daemon add DAEMON_TYPE --placement=PLACEMENT_SPEC
```

For example:

```
cephuser@adm > ceph orch daemon add mgr --placement=ses-node3
```

To manually remove a daemon, run:

```
ceph orch daemon rm DAEMON_NAME [--force]
```

For example:

```
cephuser@adm > ceph orch daemon rm mgr.ses-node3.xietsy
```

4 Troubleshooting OSDs

Before troubleshooting your OSDs, check your monitors and network first. If you execute `ceph health` or `ceph -s` on the command line and Ceph returns a health status, it means that the monitors have a quorum. If you do not have a monitor quorum or if there are errors with the monitor status, see *Chapter 6, Troubleshooting Ceph Monitors and Ceph Managers*. Check your networks to ensure they are running properly, because networks may have a significant impact on OSD operation and performance.

4.1 Obtain OSD data

To begin troubleshooting the OSDs, obtain any information including the information collected from *Book "Administration and Operations Guide", Chapter 12 "Determine the cluster state", Section 12.9 "Monitoring OSDs and placement groups"*.

4.1.1 Finding Ceph logs

If you have not changed the default path, you can find Ceph log files at `/var/log/ceph`:

```
cephuser@adm > ls /var/log/ceph
```

If you do not get enough detail, change your logging level. See *Chapter 2, Troubleshooting logging and debugging* for more information.

4.1.2 Using the admin socket tool

Use the admin socket tool to retrieve runtime information. For details, list the sockets for your Ceph processes:

```
cephuser@adm > ls /var/run/ceph
```

Execute the following:

```
cephuser@adm > ceph daemon DAEMON-NAME help
```

Alternatively, specify a `SOCKET-FILE`:

```
cephuser@adm > ceph daemon SOCKET-FILE help
```

The admin socket, among other things, allows you to:

- List your configuration at runtime
- Dump historic operations
- Dump the operation priority queue state
- Dump operations in flight
- Dump perfcounters

4.1.3 Displaying freespace

Filesystem issues may arise. To display your file system's free space, execute **`df`**.

```
cephuser@adm > df -h
```

Execute **`df --help`** for additional usage.

4.1.4 Identifying I/O statistics

Use iostat to identify I/O-related issues.

```
cephuser@adm > iostat -x
```

4.1.5 Retrieving diagnostic messages

To retrieve diagnostic messages, use **`dmesg`** with `less`, `more`, `grep` or `tail`. For example:

```
cephuser@adm > dmesg | grep scsi
```

4.2 Stopping without rebalancing

Periodically you may need to perform maintenance on a subset of your cluster or resolve a problem that affects a failure domain. If you do not want CRUSH to automatically rebalance the cluster as you stop OSDs for maintenance, set the cluster to `noout` first:

```
cephuser@adm > ceph osd set noout
```

Once the cluster is set to `noout`, you can begin stopping the OSDs within the failure domain that requires maintenance work:

```
cephuser@adm > ceph orch daemon stop osd.ID
```

Note

Placement groups within the OSDs you stop will become degraded while you are addressing issues with within the failure domain.

Once you have completed your maintenance, restart the OSDs:

```
cephuser@adm > ceph orch daemon start osd.ID
```

Finally, unset the cluster from `noout`:

```
cephuser@adm > ceph osd unset noout
```

4.3 OSDs not running

Under normal circumstances, restarting the `ceph-osd` daemon will allow it to rejoin the cluster and recover. If this is not true, continue on to the next section.

4.3.1 An OSD will not start

If an OSD will not start when you start your cluster and you are unable to determine the reason, we recommend generating a supportconfig and open a support ticket. For more information, see *How do I submit a support case?* (https://scc.suse.com/docs/help#submit-support-case) ↗

4.3.2 Failing OSD

When a `ceph-osd` process dies, the monitor learns about the failure from surviving `ceph-osd` daemons and reports it via the **`ceph health`** command:

```
cephuser@adm > ceph health
```

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

A warning displays whenever there are `ceph-osd` processes that are marked `in` and `down`. Identify which `ceph-osd`s are down with:

```
cephuser@adm > 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
```

If there is a disk failure or other fault preventing `ceph-osd` from functioning or restarting, an error message should be present in its log file. You can monitor the cephadm log in real time with the following command:

```
cephuser@adm > ceph -W cephadm
```

You can view the last few messages with:

```
cephuser@adm > ceph log last cephadm
```

If the daemon stopped because of a heartbeat failure, the underlying kernel file system may be unresponsive. Check **`dmesg`** output for disk or other kernel errors.

4.3.3 Preventing write action to OSD

Ceph prevents writing to a full OSD so that you do not lose data. In an operational cluster, you should receive a warning when your cluster is getting near its full ratio. The `mon osd full ratio` defaults to 0.95, or 95% of capacity before it stops clients from writing data. The `mon osd backfillfull ratio` defaults to 0.90, or 90% of capacity when it blocks backfills from starting. The OSD nearfull ratio defaults to 0.85, or 85% of capacity when it generates a health warning. Change this using the following command:

```
cephuser@adm > ceph osd set-nearfull-ratio <float[0.0-1.0]>
```

Full cluster issues usually arise when testing how Ceph handles an OSD failure on a small cluster. When one node has a high percentage of the cluster's data, the cluster can easily eclipse its nearfull and full ratio immediately. If you are testing how Ceph reacts to OSD failures on a small cluster, you should leave ample free disk space and consider temporarily lowering the OSD `full ratio`, OSD `backfillfull` ratio and OSD `nearfull` ratio using these commands:

```
cephuser@adm > ceph osd set-nearfull-ratio <float[0.0-1.0]>
```

```
cephuser@adm > ceph osd set-full-ratio <float[0.0-1.0]>
cephuser@adm > ceph osd set-backfillfull-ratio <float[0.0-1.0]>
```

Full `ceph-osds` will be reported by **`ceph health`**:

```
cephuser@adm > ceph health
  HEALTH_WARN 1 nearfull osd(s)
```

Or:

```
cephuser@adm > ceph health detail
  HEALTH_ERR 1 full osd(s); 1 backfillfull osd(s); 1 nearfull osd(s)
  osd.3 is full at 97%
  osd.4 is backfill full at 91%
  osd.2 is near full at 87%
```

We recommend adding new `ceph-osds` to deal with a full cluster, allowing the cluster to redistribute data to the newly available storage. If you cannot start an OSD because it is full, you may delete some data by deleting some placement group directories in the full OSD.

Important

If you choose to delete a placement group directory on a full OSD, do not delete the same placement group directory on another full OSD, or you may loose data. You must maintain at least one copy of your data on at least one OSD.

4.4 Unresponsive or slow OSDs

A commonly recurring issue involves slow or unresponsive OSDs. Ensure that you have eliminated other troubleshooting possibilities before delving into OSD performance issues. For example, ensure that your network(s) is working properly and your OSDs are running. Check to see if OSDs are throttling recovery traffic.

4.4.1 Identifying bad sectors and fragmented disks

Check your disks for bad sectors and fragmentation. This can cause total throughput to drop substantially.

4.4.2 Co-resident monitors and OSDs

Monitors are generally light-weight processes but often perform fsync(). This can interfere with other workloads, particularly if monitors run on the same drive as the OSDs. Additionally, if you run monitors on the same host as the OSDs, you may incur performance issues related to:

- Running an older kernel (pre-3.0)
- Running a kernel with no syncfs(2) syscall.

In these cases, multiple OSDs running on the same host can drag each other down by doing lots of commits. That often leads to the bursty writes.

4.4.3 Co-resident processes

Spinning up co-resident processes such as a cloud-based solution, virtual machines and other applications that write data to Ceph while operating on the same hardware as OSDs can introduce significant OSD latency. We recommend optimizing a host for use with Ceph and using other hosts for other processes. The practice of separating Ceph operations from other applications may help improve performance and may streamline troubleshooting and maintenance.

4.4.4 Logging levels

If you turned logging levels up to track an issue and then forgot to turn logging levels back down, the OSD may be putting a lot of logs onto the disk. If you intend to keep logging levels high, you may consider mounting a drive to the default path for logging. For example, `/var/log/ceph/$cluster-$name.log`.

4.4.5 Recovery throttling

Depending upon your configuration, Ceph may reduce recovery rates to maintain performance or it may increase recovery rates to the point that recovery impacts OSD performance. Check to see if the OSD is recovering.

4.4.6 Kernel version

Check the kernel version you are running. Older kernels may not receive new backports that Ceph depends upon for better performance.

4.4.7 Kernel issues with syncfs

Try running one OSD per host to see if performance improves. Old kernels might not have a recent enough version of glibc to support **`syncfs(2)`**.

4.4.8 Filesystem issues

Currently, we recommend deploying clusters with XFS.

4.4.9 Insufficient RAM

We recommend 1.5 GB of RAM per TB of raw OSD capacity for each Object Storage Node. You may notice that during normal operations, the OSD only uses a fraction of that amount. Unused RAM makes it tempting to use the excess RAM for co-resident applications, VMs and so forth. However, when OSDs go into recovery mode, their memory utilization spikes. If there is no RAM available, the OSD performance will slow considerably.

4.4.10 Complaining about old or slow requests

If a `ceph-osd` daemon is slow to respond to a request, log messages will generate complaining about requests that are taking too long. The warning threshold defaults to 30 seconds, and is configurable via the `osd op complaint time` option. When this happens, the cluster log receives messages. Legacy versions of Ceph complain about old requests:

```
osd.0 192.168.106.220:6800/18813 312 : [WRN] old request \
  osd_op(client.5099.0:790 fatty_26485_object789 [write 0~4096] 2.5e54f643) \
  v4 received at 2012-03-06 15:42:56.054801 currently waiting for sub ops
```

Newer versions of Ceph complain about `slow requests`:

```
{date} {osd.num} [WRN] 1 slow requests, 1 included below; oldest blocked for > 30.005692
 secs
```

```
{date} {osd.num}  [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]
```

Possible causes include:

- A bad drive (check **`dmesg`** output)
- A bug in the kernel file system (check **`dmesg`** output)
- An overloaded cluster (check system load, iostat, etc.)
- A bug in the `ceph-osd` daemon.

Possible solutions:

- Remove VMs from Ceph hosts
- Upgrade kernel
- Upgrade Ceph
- Restart OSDs

4.4.11 Debugging slow requests

If you run **`ceph daemon osd.<id> dump_historic_ops`** or **`ceph daemon osd.<id> dump_ops_in_flight`**, you see a set of operations and a list of events each operation went through. These are briefly described below.

Events from the Messenger layer:

`header_read`

When the messenger first started reading the message off the wire.

`throttled`

When the messenger tried to acquire memory throttle space to read the message into memory.

`all_read`

When the messenger finished reading the message off the wire.

`dispatched`

When the messenger gave the message to the OSD.

`initiated`

This is identical to `header_read`. The existence of both is a historical oddity.

Events from the OSD as it prepares operations:

`queued_for_pg`

The op has been put into the queue for processing by its PG.

`reached_pg`

The PG has started doing the op.

`waiting for \*`

The op is waiting for some other work to complete before it can proceed (e.g. a new OSDMap; for its object target to scrub; for the PG to finish peering; all as specified in the message).

`started`

The op has been accepted as something the OSD should do and is now being performed.

`waiting for subops from`

The op has been sent to replica OSDs.

Events from the FileStore:

`commit_queued_for_journal_write:`

The op has been given to the FileStore.

`write_thread_in_journal_buffer`

The op is in the journal's buffer and waiting to be persisted (as the next disk write).

`journaled_completion_queued`

The op was journaled to disk and its callback queued for invocation.

Events from the OSD after stuff has been given to local disk:

`op_commit`

The op has been committed by the primary OSD.

`op_applied`

The op has been `write()'en` to the backing FS on the primary.

`sub_op_applied: op_applied`

For a replica's “subop”.

`sub_op_committed: op_commit`

For a replica’s sub-op (only for EC pools).

`sub_op_commit_rec/sub_op_apply_rec from <X>`

The primary marks this when it hears about the above, but for a particular replica (i.e. <X>).

`commit_sent`

We sent a reply back to the client (or primary OSD, for sub ops).

Many of these events are seemingly redundant, but cross important boundaries in the internal code (such as passing data across locks into new threads).

4.5 OSD weight is 0

When OSD starts, it is assigned a weight. The higher the weight, the bigger the chance that the cluster writes data to the OSD. The weight is either specified in a cluster CRUSH Map, or calculated by the OSDs' start-up script.

4.6 OSD is down

OSD daemon is either running, or stopped/down. There are 3 general reasons why an OSD is down:

- Hard disk failure.
- The OSD crashed.
- The server crashed.

You can see the detailed status of OSDs by running

```
cephuser@adm > ceph osd tree
# id  weight  type name up/down reweight
 -1    0.02998  root default
 -2    0.009995   host doc-ceph1
 0     0.009995      osd.0 up  1
 -3    0.009995   host doc-ceph2
 1     0.009995      osd.1 up  1
 -4    0.009995   host doc-ceph3
 2     0.009995      osd.2 down
```

The example listing shows that the `osd.2` is down. Then you may check if the disk where the OSD is located is mounted:

```
# lsblk -f
NAME
                FSTYPE      LABEL UUID                                   FSAVAIL FSUSE%
 MOUNTPOINT
vda
├─vda1
├─vda2
               vfat        EFI   7BDD-40C3                               18.8M     6% /
boot/efi
└─vda3
             ext4        ROOT  144d3eec-c193-4793-b6a9-1ac295259f4b     36.7G     5% /
vdb
             LVM2_member       Fj8nlY-Dnmm-8Y0w-tJrO-8rAe-SUTH-lA2Ux2
└─ceph--6dc6f71f--ff02--4316--b145--c7804d5fb2f4-osd--block--2cb16f65--c87d--405d--
a913--4e4ea6037ca1
vdc
             LVM2_member       0101m1-mc83-K8ce-j4Dv-yxUO-5KPj-6FpuSu
└─ceph--1082eac0--4478--48cf--b8ab--4d3a62ca1c27-osd--data--185c9dd6--ce28--4b98--
a9b6--9d5b1f444a4f
vdd
             LVM2_member       wqE8sC-w5mx-J9t1-FoM1-vIRC-0xxq-5FPyuL
└─ceph--9e71d81a--e394--49b7--bef9--b639083be779-osd--data--eb94f8e6--
af6d--4ef6--911b--7f44f7484c85
```

You can track the reason why the OSD is down by inspecting its log file. See *Section 4.3.2, “Failing OSD”* for instructions on how to source the log files. After you find and fix the reason why the OSD is not running, start it with the following command. To identify the unique FSID of the cluster, run **`ceph fsid`**. To identify the Object Gateway daemon name, run **`ceph orch ps ---hostname`** *`HOSTNAME`*.

```
# systemctl start ceph-FSID@osd.2
```

Do not forget to replace `2` with the actual number of your stopped OSD.

4.7 Finding slow OSDs

When tuning the cluster performance, it is very important to identify slow storage/OSDs within the cluster. The reason is that if the data is written to the slow(est) disk, the complete write operation slows down as it always waits until it is finished on all the related disks.

It is not trivial to locate the storage bottleneck. You need to examine each and every OSD to find out the ones slowing down the write process. To do a benchmark on a single OSD, run:

```
ceph tell osd.OSD_ID_NUMBER bench
```

For example:

```
cephuser@adm > ceph tell osd.0 bench
 { "bytes_written": 1073741824,
   "blocksize": 4194304,
   "bytes_per_sec": "19377779.000000"}
```

Then you need to run this command on each OSD and compare the `bytes_per_sec` value to get the slow(est) OSDs.

4.8 Flapping OSDs

We recommend using both a public (front-end) network and a cluster (back-end) network so that you can better meet the capacity requirements of object replication. Another advantage is that you can run a cluster network such that it is not connected to the internet, thereby preventing some denial of service attacks. When OSDs peer and check heartbeats, they use the cluster (back-end) network when it's available.

However, if the cluster (back-end) network fails or develops significant latency while the public (front-end) network operates optimally, OSDs currently do not handle this situation well. What happens is that OSDs mark each other down on the monitor, while marking themselves up. This is called *flapping*.

If something is causing OSDs to flap (repeatedly getting marked down and then up again), you can force the monitors to stop the flapping with:

```
cephuser@adm > ceph osd set noup      # prevent OSDs from getting marked up
cephuser@adm > ceph osd set nodown    # prevent OSDs from getting marked down
```

These flags are recorded in the osdmap structure:

```
cephuser@adm > ceph osd dump | grep flags
flags no-up,no-down
```

You can clear the flags with:

```
cephuser@adm > ceph osd unset noup
```

```
cephuser@adm > ceph osd unset nodown
```

Two other flags are supported, `noin` and `noout`, which prevent booting OSDs from being marked in (allocated data) or protect OSDs from eventually being marked out (regardless of what the current value for **`mon osd down out`** interval is).

Note

`noup`, `noout`, and `nodown` are temporary in the sense that once the flags are cleared, the action they were blocking should occur shortly after. The `noin` flag, on the other hand, prevents OSDs from being marked `in` on boot, and any daemons that started while the flag was set will remain that way.

5 Troubleshooting placement groups (PGs)

5.1 Identifying troubled placement groups

As previously noted, a placement group is not necessarily problematic because its state is not `active+clean`. Generally, Ceph's ability to self-repair may not be working when placement groups get stuck. The stuck states include:

- **Unclean**: Placement groups contain objects that are not replicated the required number of times. They should be recovering.
- **Inactive**: Placement groups cannot process reads or writes because they are waiting for an OSD with the most up-to-date data to come back up.
- **Stale**: Placement groups are in an unknown state, because the OSDs that host them have not reported to MONs in a while (configured by the `mon osd report timeout` option).

To identify stuck placement groups, run the following:

```
cephuser@adm > ceph pg dump_stuck [unclean|inactive|stale|undersized|degraded]
```

5.2 Placement groups never get clean

When you create a cluster and your cluster remains in `active`, `active+remapped`, or `active+degraded` status and never achieves an `active+clean` status, you likely have a problem with the configuration. As a general rule, you should run your cluster with more than one OSD and a pool size greater than 1 object replica.

5.2.1 Experimenting with a one node cluster

Ceph no longer provides documentation for operating on a single node. Mounting client kernel modules on a single node containing a Ceph daemon can cause a deadlock due to issues with the Linux kernel itself (unless you use VMs for the clients). However, we recommend experimenting with Ceph in a 1-node configuration regardless of the limitations.

If you are trying to create a cluster on a single node, change the default of the `osd crush chooseleaf` type setting from 1 (meaning `host` or `node`) to 0 (meaning `osd`) in your Ceph configuration file before you create your monitors and OSDs. This tells Ceph that an OSD can peer with another OSD on the same host. If you are trying to set up a 1-node cluster and `osd crush chooseleaf` type is greater than 0, Ceph tries to pair the PGs of one OSD with the PGs of another OSD on another node, chassis, rack, row, or even datacenter depending on the setting.

Note

Do not mount kernel clients directly on the same node as your Ceph Storage Cluster, because kernel conflicts can arise. However, you can mount kernel clients within virtual machines (VMs) on a single node.

If you are creating OSDs using a single disk, you must create directories for the data manually first. For example:

```
cephuser@adm > ceph-deploy osd create --data {disk} {host}
```

5.2.2 Fewer OSDs than replicas

If you have brought up two OSDs to an up and in state, but you still do not see `active+clean` placement groups, you may have an `osd pool default size` set to greater than 2. There are a few ways to address this situation. If you want to operate your cluster in an `active+degraded` state with two replicas, you can set the `osd pool default min size` to 2 so that you can write objects in an `active+degraded` state. You may also set the `osd pool default size` setting to 2 so that you only have two stored replicas (the original and one replica), in which case the cluster should achieve an `active+clean` state.

Note

You can make the changes at runtime. If you make the changes in your Ceph configuration file, you may need to restart your cluster.

5.2.3 Forcing pool sizes

If you have the `osd pool default size` set to 1, you only have one copy of the object. OSDs rely on other OSDs to tell them which objects they should have. If an OSD has a copy of an object and there is no second copy, then no second OSD can tell the first OSD that it should have that copy. For each placement group mapped to the first OSD (see **`ceph pg dump`**), you can force the first OSD to notice the placement groups it needs by running:

```
cephuser@adm > ceph osd force-create-pg <pgid>
```

5.2.4 Identifying CRUSH map errors

Another candidate for placement groups remaining unclean involves errors in your CRUSH map.

5.3 Stuck placement groups

It is normal for placement groups to enter states such as `degraded` or `peering` following a failure. These states indicate the normal progression through the failure recovery process. However, if a placement group stays in one of these states for a long time this may be an indication of a larger problem. For this reason, the monitor will warn when placement groups get stuck in a non-optimal state. Specifically, check for:

`inactive`

The placement group has not been active for too long. For example, it has not been able to service read/write requests.

`unclean`

The placement group has not been clean for too long. For exmaple, it has not been able to completely recover from a previous failure.

`stale`

The placement group status has not been updated by a `ceph-osd`, indicating that all nodes storing this placement group may be down.

You can explicitly list stuck placement groups with one of:

```
cephuser@adm > ceph pg dump_stuck stale
```

```
cephuser@adm > ceph pg dump_stuck inactive
cephuser@adm > ceph pg dump_stuck unclean
```

For stuck stale placement groups, ensure you have the right `ceph-osd` daemons running again. For stuck inactive placement groups, it is can be a peering problem. For stuck unclean placement groups, there can be something preventing recovery from completing, like unfound objects.

5.4 Peering failure of placement groups

In certain cases, the `ceph-osd` peering process can run into problems, preventing a PG from becoming active and usable. For example, **`ceph health`** may report:

```
cephuser@adm > ceph health detail
  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
```

Query the cluster to determine exactly why the PG is marked down by executing the following:

```
cephuser@adm > 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"}
    ]
  }
```

`recovery_state` section shows that peering is blocked due to down `ceph-osd` daemons, specifically `osd.1`. In this case, restart the `ceph-osd` to recover. Alternatively, if there is a catastrophic failure of `osd.1` such as a disk failure, tell the cluster that it is lost and to cope as best it can.

Important

The cluster cannot guarantee that the other copies of the data are consistent and up to date.

To instruct Ceph to continue anyway:

```
cephuser@adm > ceph osd lost 1
```

Recovery will proceed.

5.5 Failing unfound objects

Under certain combinations of failures Ceph may complain about unfound objects:

```
cephuser@adm > ceph health detail
  HEALTH_WARN 1 pgs degraded; 78/3778 unfound (2.065%)
  pg 2.4 is active+degraded, 78 unfound
```

This means that the storage cluster knows that some objects (or newer copies of existing objects) exist, but it has not found copies of them. One example of how this might come about for a PG whose data is on `ceph-osd`s 1 and 2:

- 1 goes down
- 2 handles some writes, alone
- 1 comes up
- 1 and 2 repeer, and the objects missing on 1 are queued for recovery.
- Before the new objects are copied, 2 goes down.

In this example, 1 is aware that these object exist, but there is no live `ceph-osd` who has a copy. In this case, I/O to those objects blocks, and the cluster hopes that the failed node comes back soon. This is assumed to be preferable to returning an I/O error to the user.

Identify which objects are unfound by executing the following:

```
cephuser@adm > ceph pg 2.4 list_unfound [starting offset, in json]
  { "offset": { "oid": "",
       "key": "",
       "snapid": 0,
       "hash": 0,
       "max": 0},
   "num_missing": 0,
   "num_unfound": 0,
   "objects": [
      { "oid": "object 1",
        "key": "",
        "hash": 0,
        "max": 0 },
      ...
   ],
   "more": 0}
```

If there are too many objects to list in a single result, the `more` field is true and you can query for more.

Identify which OSDs have been probed or might contain data:

```
cephuser@adm > ceph pg 2.4 query
  "recovery_state": [
       { "name": "Started\/Primary\/Active",
         "enter_time": "2012-03-06 15:15:46.713212",
         "might_have_unfound": [
               { "osd": 1,
                 "status": "osd is down"}]},
```

In this case, for example, the cluster knows that `osd.1` might have data, but it is `down`. The full range of possible states include:

- already probed
- querying
- OSD is down
- not queried (yet)

Sometimes it takes some time for the cluster to query possible locations.

It is possible that there are other locations where the object can exist that are not listed. For example, if a `ceph-osd` is stopped and taken out of the cluster, the cluster fully recovers, and due to some future set of failures ends up with an unfound object, it will not consider the long-departed `ceph-osd` as a potential location to consider.

If all possible locations have been queried and objects are still lost, you may have to give up on the lost objects. This, again, is possible given unusual combinations of failures that allow the cluster to learn about writes that were performed before the writes themselves are recovered. To mark the unfound objects as `lost`:

```
cephuser@adm > ceph pg 2.5 mark_unfound_lost revert|delete
```

This the final argument specifies how the cluster should deal with lost objects. The `delete` option forgets about them entirely. The `revert` option (not available for erasure coded pools) either rolls back to a previous version of the object or (if it was a new object) forgets about it entirely. Use this with caution, as it may confuse applications that expected the object to exist.

5.6 Identifying homeless placement groups

It is possible for all OSDs that had copies of a given placement groups to fail. If that is the case, that subset of the object store is unavailable, and the monitor receives no status updates for those placement groups. To detect this situation, the monitor marks any placement group whose primary OSD has failed as `stale`. For example:

```
cephuser@adm > ceph health
  HEALTH_WARN 24 pgs stale; 3/300 in osds are down
```

Identify which placement groups are `stale`, and what were the last OSDs to store them by executing the following:

```
cephuser@adm > 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
```

For example, to get placement group 2.5 back online, this output shows that it was last managed by `osd.0` and `osd.2`. Restarting the `ceph-osd` daemons allows the cluster to recover that placement group.

5.7 Only a few OSDs receive data

If you have many nodes in your cluster and only a few of them receive data, check the number of placement groups in your pool. See *Book “Administration and Operations Guide”, Chapter 12 “Determine the cluster state”, Section 12.7 “Checking placement group states”* for more information. Since placement groups get mapped to OSDs, a small number of placement groups will not distribute across the cluster. Create a pool with a placement group count that is a multiple of the number of OSDs. See *Book “Administration and Operations Guide”, Chapter 17 “Stored data management”, Section 17.4 “Placement groups”* for details.

5.8 Unable to write data

If your cluster is up but some OSDs are down and you cannot write data, check to ensure that you have the minimum number of OSDs running for the placement group. If you do not have the minimum number of OSDs running, Ceph will not allow you to write data because there is no guarantee that Ceph can replicate your data.

5.9 Identifying inconsistent placement groups

If you receive an `active+clean+inconsistent` state, this may happen due to an error during scrubbing. Identify the inconsistent placement group(s) by executing the following:

```
cephuser@adm > 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
```

Or:

```
cephuser@adm > rados list-inconsistent-pg rbd
  ["0.6"]
```

There is only one consistent state, but in the worst case, there could be different inconsistencies in multiple perspectives found in more than one objects. If an object named `foo` in PG `0.6` is truncated, the output is:

```
cephuser@adm > rados list-inconsistent-obj 0.6 --format=json-pretty
```

```
  {
      "epoch": 14,
      "inconsistents": [
          {
              "object": {
                  "name": "foo",
                  "nspace": "",
                  "locator": "",
                  "snap": "head",
                  "version": 1
              },
              "errors": [
                  "data_digest_mismatch",
                  "size_mismatch"
              ],
              "union_shard_errors": [
                  "data_digest_mismatch_info",
                  "size_mismatch_info"
              ],
              "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_info",
                          "size_mismatch_info"
                      ],
                      "size": 0,
                      "omap_digest": "0xffffffff",
                      "data_digest": "0xffffffff"
                  }
              ]
          }
      ]
```

```
}
```

In this case, we can learn from the output that the only inconsistent object is named `foo`, and it has inconsistencies. The inconsistencies fall into two categories:

`errors`

These errors indicate inconsistencies between `shards` without a determination of which shard(s) are bad. Check for the `errors` in the `shards` array, if available, to pinpoint the problem.

`data_digest_mismatch`

The digest of the replica read from OSD.2 is different from the ones of OSD.0 and OSD.1

`size_mismatch`

The size of the replica read from OSD.2 is 0, while the size reported by OSD.0 and OSD.1 is 968.

`union_shard_errors`

The union of all shard specific `errors` in `shards` array. The `errors` are set for the given shard that has the problem. They include errors like `read_error`. The `errors` ending in `oi` indicate a comparison with `selected_object_info`. Look at the `shards` array to determine which shard has which error(s).

`data_digest_mismatch_info`

The digest stored in the object-info is not 0xffffffff, which is calculated from the shard read from OSD.2

`size_mismatch_info`

The size stored in the object-info is different from the one read from OSD.2. The latter is 0.

Repair the inconsistent placement group by executing:

```
cephuser@adm > ceph pg repair placement-group-ID
```

This command overwrites the bad copies with the authoritative ones. In most cases, Ceph is able to choose authoritative copies from all available replicas using some predefined criteria but this does not always work. For example, the stored data digest could be missing, and the calculated digest will be ignored when choosing the authoritative copies. Use the above command with caution.

If `read_error` is listed in the errors attribute of a shard, the inconsistency is likely due to disk errors. You might want to check your disk used by that OSD.

If you receive `active+clean+inconsistent` states periodically due to clock skew, you may consider configuring your NTP daemons on your monitor hosts to act as peers.

5.10 Identifying inactive erasure coded PGs

When CRUSH fails to find enough OSDs to map to a PG, it will show as a 2147483647 which is `ITEM_NONE` or `no OSD found`. For instance:

```
[2,1,6,0,5,8,2147483647,7,4]
```

5.10.1 Displaying not enough OSDs

If the Ceph cluster only has 8 OSDs and the erasure coded pool needs 9, that is what it will show. You can either create another erasure coded pool that requires less OSDs:

```
cephuser@adm > ceph osd erasure-code-profile set myprofile k=5 m=3
cephuser@adm > ceph osd pool create erasurepool erasure myprofile
```

Or, add a new OSDs and the PG automatically uses them.

5.10.2 Satisfying CRUSH constraints

If the cluster has enough OSDs, it is possible that the CRUSH rule imposes constraints that cannot be satisfied. If there are 10 OSDs on two hosts and the CRUSH rule requires that no two OSDs from the same host are used in the same PG, the mapping may fail because only two OSDs will be found. You can check the constraint by displaying the rule:

```
cephuser@adm > ceph osd crush rule ls
  [
      "replicated_rule",
      "erasurepool"]
  $ ceph osd crush rule dump erasurepool
  { "rule_id": 1,
    "rule_name": "erasurepool",
    "ruleset": 1,
    "type": 3,
    "min_size": 3,
    "max_size": 20,
    "steps": [
          { "op": "take",
            "item": -1,
            "item_name": "default"},
          { "op": "chooseleaf_indep",
            "num": 0,
            "type": "host"},
          { "op": "emit"}]}
```

Resolve the problem by creating a new pool in which PGs are allowed to have OSDs residing on the same host with:

```
cephuser@adm > ceph osd erasure-code-profile set myprofile crush-failure-domain=osd
cephuser@adm > ceph osd pool create erasurepool erasure myprofile
```

5.10.3 Identifying when CRUSH gives up too soon

If the Ceph cluster has just enough OSDs to map the PG (for instance a cluster with a total of 9 OSDs and an erasure coded pool that requires 9 OSDs per PG), it is possible that CRUSH gives up before finding a mapping. It can be resolved by:

- Lowering the erasure coded pool requirements to use less OSDs per PG (that requires the creation of another pool as erasure code profiles cannot be dynamically modified).
- Adding more OSDs to the cluster (that does not require the erasure coded pool to be modified, it will become clean automatically)
- Use a handmade CRUSH rule that tries more times to find a good mapping. This can be done by setting `set_choose_tries` to a value greater than the default.

Verify the problem with crushtool after extracting the crushmap from the cluster so your experiments do not modify the Ceph cluster and only work on a local files:

```
cephuser@adm > ceph osd crush rule dump erasurepool
  { "rule_name": "erasurepool",
    "ruleset": 1,
    "type": 3,
    "min_size": 3,
    "max_size": 20,
    "steps": [
          { "op": "take",
            "item": -1,
            "item_name": "default"},
          { "op": "chooseleaf_indep",
            "num": 0,
            "type": "host"},
          { "op": "emit"}]}
$ ceph osd getcrushmap > crush.map
got crush map from osdmap epoch 13
$ crushtool -i crush.map --test --show-bad-mappings \
   --rule 1 \
   --num-rep 9 \
   --min-x 1 --max-x $((1024 * 1024))
bad mapping rule 8 x 43 num_rep 9 result [3,2,7,1,2147483647,8,5,6,0]
bad mapping rule 8 x 79 num_rep 9 result [6,0,2,1,4,7,2147483647,5,8]
bad mapping rule 8 x 173 num_rep 9 result [0,4,6,8,2,1,3,7,2147483647]
```

Where `--num-rep` is the number of OSDs the erasure code CRUSH rule needs, `--rule` is the value of the ruleset field displayed by **`ceph osd crush rule dump`**. The test tries to map one million values (i.e. the range defined by [--min-x,--max-x]) and must display at least one bad mapping. If it outputs nothing it means all mappings are successful and the problem is elsewhere.

The CRUSH rule can be edited by decompiling the crush map:

```
# crushtool --decompile crush.map > crush.txt
```

Add the following line to the rule:

```
step set_choose_tries 100
```

The relevant part of of the `crush.txt` file should look something like:

```
rule erasurepool {
        ruleset 1
        type erasure
        min_size 3
        max_size 20
        step set_chooseleaf_tries 5
        step set_choose_tries 100
        step take default
        step chooseleaf indep 0 type host
        step emit
}
```

It can then be compiled and tested again:

```
# crushtool --compile crush.txt -o better-crush.map
```

When all mappings succeed, an histogram of the number of tries that were necessary to find all of them can be displayed with the `--show-choose-tries` option of `crushtool`:

```
# crushtool -i better-crush.map --test --show-bad-mappings \
     --show-choose-tries \
     --rule 1 \
     --num-rep 9 \
     --min-x 1 --max-x $((1024 * 1024))
  ...
  11:        42
  12:        44
  13:        54
  14:        45
  15:        35
  16:        34
  17:        30
  18:        25
  19:        19
  20:        22
  21:        20
  22:        17
  23:        13
  24:        16
  25:        13
  26:        11
  27:        11
  28:        13
  29:        11
  30:        10
  31:         6
  32:         5
  33:        10
  34:         3
  35:         7
  36:         5
  37:         2
  38:         5
  39:         5
  40:         2
  41:         5
  42:         4
  43:         1
  44:         2
  45:         2
  46:         3
  47:         1
```

```
48:          0
...
102:          0
103:          1
104:          0
...
```

It takes 11 tries to map 42 PGs, 12 tries to map 44 PGs etc. The highest number of tries is the minimum value of `set_choose_tries` that prevents bad mappings (i.e. 103 in the above output because it did not take more than 103 tries for any PG to be mapped).

6 Troubleshooting Ceph Monitors and Ceph Managers

6.1 Initial troubleshooting

Q: *Are the monitors running?*

A: Ensure the monitors are running, this is an important to check if you have performed an upgrade and not manually restarted the monitors.

Q: *Are you able to connect to the monitor's servers?*

A: Occasionally, you can be running `iptable` rules that block access to monitor servers or monitor ports. This can often be the case from monitor stress-testing that was forgotten. We recommend trying to `ssh` into the server and, if that succeeds, try connecting to the monitor's port using your tool of choice (such as **`telnet`** or **`netcat`**).

Q: *Does* **`ceph -s`** *run and obtain a reply from the cluster?*

A: If the answer is yes, then your cluster is up and running. The monitors will only answer to a status request if there is a formed quorum. If **`ceph -s`** is blocked, without obtaining a reply from the cluster or showing a lot of fault messages, then it is possible that your monitors are either down completely or just a portion is up – a portion that is not enough to form a quorum (keep in mind that a quorum if formed by a majority of monitors).

Q: *What if* **`ceph -s`** *does not finish?*

A: Contact each monitor individually for the status, regardless of a quorum being formed. This can be achieved using **`ceph tell mon.ID mon_status`** with the ID being the monitor's identifier. Perform this for each monitor in the cluster. The section *Section 6.3, "Understanding* `mons_status`*"* explains how to interpret the output of this command.

6.2 Using the monitor's admin socket

The admin socket allows you to interact with a given daemon directly using a Unix socket file. This file can be found in your monitor's run directory. By default, the admin socket will be kept in `/var/run/ceph/ceph-mon.ID.asok` but this can vary if you defined it otherwise. If you are unable to find it there, check your `ceph.conf` for an alternative path or run:

```
cephuser@adm > ceph-conf --name mon.ID --show-config-value admin_socket
```

Keep in mind that the admin socket is only available while the monitor is running. When the monitor is properly shutdown, the admin socket is removed. If however the monitor is not running and the admin socket still persists, it is likely that the monitor was improperly shutdown. Regardless, if the monitor is not running, you will not be able to use the admin socket, with ceph likely returning `Error 111: Connection Refused`. To accessing the admin socket run **`ceph tell`** on the daemon you are interested in. For example:

```
cephuser@adm > ceph tell mon.ID mon_status
```

This passes the command help to the running MON daemon *`ID`* via the admin socket, which is a file ending in `.asok` somewhere under `/var/run/ceph`. When you know the full path to the file, you can run the following:

```
cephuser@adm > ceph --admin-daemon PATH_TO_FILE COMMAND
```

Using help as the command to the **`ceph tool`** shows the supported commands available through the admin socket. Take a look at `config get`, `config show`, `mon stat` and `quorum_status`, as those can be enlightening when troubleshooting a monitor.

6.3 Understanding mons_status

`mon_status` can be obtained via the admin socket. This command outputs a multitude of information about the monitor including the same output you would get with `quorum_status`. For example, the following example output of **`ceph tell mon.c mon_status`**:

```
  { "name": "c",
  "rank": 2,
  "state": "peon",
  "election_epoch": 38,
  "quorum": [
        1,
        2],
  "outside_quorum": [],
  "extra_probe_peers": [],
  "sync_provider": [],
  "monmap": { "epoch": 3,
      "fsid": "5c4e9d53-e2e1-478a-8061-f543f8be4cf8",
      "modified": "2013-10-30 04:12:01.945629",
      "created": "2013-10-29 14:14:41.914786",
      "mons": [
            { "rank": 0,
              "name": "a",
              "addr": "127.0.0.1:6789\/0"},
            { "rank": 1,
              "name": "b",
              "addr": "127.0.0.1:6790\/0"},
            { "rank": 2,
              "name": "c",
              "addr": "127.0.0.1:6795\/0"}]}}
```

This example shows that there are three montiors in the monmap (`a`, `b` and `c`), the quorum is formed by only two monitors, and `c` is in the quorum as a `peon`. This means that monitor `a` is out of quorum. This is because there are two monitors in this set: 1 and 2. These are not monitor names. These are monitor ranks, as established in the current monmap. It shows that the missing monitor is the one with a rank of 0, and according to the monmap that would be `mon.a`.

Ranks (re)calculated whenever you add or remove monitors and follow this rule: the greater the `IP:PORT` combination, the lower the rank is. In this case, considering that `127.0.0.1:6789` is lower than all the remaining `IP:PORT` combinations, `mon.a` has rank 0.

6.4 Restoring the MONs quorum

If the Ceph Monitors cannot form a quorum, cephadm will not be able to manage the cluster until the quorum is restored. In order to restore the Ceph Monitor quorum, remove unhealthy Ceph Monitors form the monmap by following these steps:

1. Stop all Ceph Monitors. Log in to each Ceph Monitor host via SSH and run the following command there:

   ```
   cephuser@adm > cephadm unit --name mon.`hostname` stop
   ```

2. Identify a surviving monitor by logging in to that host via SSH and running:

   ```
   cephuser@adm > cephadm shell --name mon.`hostname`
   ```

3. Extract a copy of the monmap to a file, for example `/tmp/monmap`. Note that *`MON_ID`* is usually identical to the string that the **`hostname`** command returns:

   ```
   ceph-mon -i MON_ID --extract-monmap /tmp/monmap
   ```

4. Remove the non-surviving or problematic monitors. For example, if you have three monitors, `mon.a`, `mon.b`, and `mon.c` where only `mon.a` is surviving, follow this example:

   ```
   cephuser@adm > monmaptool /tmp/monmap --rm b
   cephuser@adm > monmaptool /tmp/monmap --rm c
   ```

5. Inject the surviving map with the removed monitors into the surviving monitor(s). For example, to inject the map into the monitor `mon.a`, follow this example:

   ```
   cephuser@adm > ceph-mon -i a --inject-monmap /tmp/monmap
   ```

6. Start only the surviving monitors, and verify that the monitors form a quorum with the **`ceph -s`** command.

Note

You may wish to archive the removed monitors' data directories from `/var/lib/ceph/mon` in a safe location, or delete it if you are confident the remaining monitors are healthy and are sufficiently redundant.

6.5 Most common monitor issues

6.5.1 Have quorum but at least one monitor is down

When this happens, depending on the version of Ceph you are running, you should be seeing something similar to:

```
cephuser@adm > ceph health detail
[snip]
mon.a (rank 0) addr 127.0.0.1:6789/0 is down (out of quorum)
```

To troubleshoot, make sure that `mon.a` is running. After that, make sure you are able to connect to `mon.a`'s server from the other monitors' servers. Check the ports as well. Check iptables on all your monitor nodes and make sure you are not dropping or rejecting connections. If this initial troubleshooting does not solve your problems, check the problematic monitor's `mon_status` via the admin socket. Considering the monitor is out of the quorum, its state should be one of `probing`, `electing` or `synchronizing`. If it happens to be either `leader` or `peon`, then the monitor believes itself to be in the quorum, while the remaining cluster is sure it is not; or maybe it got into the quorum while we were troubleshooting the monitor. Check using **`ceph -s`** again, just to make sure. Continue if the monitor is not yet in quorum.

1. *What if the state is* `probing`*?*

   This means the monitor is still looking for the other monitors. Every time you start a monitor, the monitor stays in this state for some time while trying to find the rest of the monitors specified in the `monmap`. The time a monitor spends in this state can vary. For instance, when on a single-monitor cluster, the monitor passes through the probing state almost instantaneously, since there are no other monitors around. On a multi-monitor cluster, the monitors will stay in this state until they find enough monitors to form a quorum – this means that if you have 2 out of 3 monitors down, the one remaining monitor stays in this state indefinitely until one of the other monitors is brought up manually.

   If there is a quorum, the monitor should be able to find the remaining monitors as long as they can be reached. If your monitor is stuck probing and you have gone through with all the communication troubleshooting, then there is a chance that the monitor is trying to reach the other monitors on a wrong address. `mon_status` outputs the `monmap` known to the monitor and checks if the other monitor's locations match reality. If they do not, then it may be related to a broken mon map. If they do, then it may be related to severe clock skews amongst the monitor nodes and you should refer to *Section 6.5.2, "Fixing clock skews"*.

2. *What if the state is* `electing`*?*

   This means the monitor is in the middle of an election. These should be fast to complete, but at times the monitors can get stuck electing. This is usually a sign of a clock skew among the monitor nodes. See *Section 6.5.2, "Fixing clock skews"* for more information. This is not a state that is likely to persist and aside from old bugs there is not an obvious reason besides clock skews on why this would happen.

3. *What if the state is* `synchronizing`*?*

   This means the monitor is synchronizing with the rest of the cluster in order to join the quorum. However, if you notice that the monitor jumps from synchronizing to electing and then back to synchronizing, then it can mean that the cluster state is advancing (i.e., generating new maps) way too fast for the synchronization process to keep up.

4. *What if the state is* `leader` *or* `peon`*?*

   This should not happen. If this does happen, it is likely related to clock skews, see *Section 6.5.2, “Fixing clock skews”* for more information. If you see no issue with the clock skews, prepare your logs and reach out to your support representative.

6.5.2 Fixing clock skews

Monitors can be severely affected by significant clock skews across the monitor nodes. This usually translates into weird behavior with no obvious cause. To avoid such issues, run a clock synchronization tool on your monitor nodes.

By default, the maximum tolerated clock skew allows clocks to drift up to 0.05 seconds. This value is configurable via the `mon-clock-drift-allowed` option, however we do not recommend doing this. The clock skew mechanism is in place because clock skewed monitor may not properly behave. Changing this value without testing it first may cause unforeseen effects on the stability of the monitors and overall cluster healthiness, although there is no risk of data loss.

The monitors will warn you if there is a clock skew by sending a `HEALTH_WARN` alert. Run the **`ceph health detail`** command to determine what monitor is flagging a clock skew. For example:

```
mon.c addr 10.10.0.1:6789/0 clock skew 0.08235s > max 0.05s (latency 0.0045s)
```

If you have a clock skew, synchronize your clocks. Running an NTP client may help. If you are already using one and you hit this sort of issue, check if you are using an NTP server remote to your network and consider hosting your own NTP server on your network. This last option tends to reduce the amount of issues with monitor clock skews.

6.5.3 Connecting and mounting to the client

If you cannot connect or mount to the client, check your iptables. Some OS install utilities add a `REJECT` rule to iptables. The rule rejects all clients trying to connect to the host except for SSH. If your monitor host’s IP tables have a `REJECT` rule in place, clients connecting from a separate node will fail to mount with a timeout error. You need to address iptables rules that reject clients trying to connect to Ceph daemons. For example, address rules that look like similar to this:

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

You may also need to add rules to iptables on your Ceph hosts to ensure that clients can access the ports associated with your Ceph monitors (for example, port 6789 by default) and Ceph OSDs (for example, 6800 through 7300 by default). For example:

```
iptables -A INPUT -m multiport -p tcp -s {ip-address}/{netmask} --dports 6789,6800:7300 -
j ACCEPT
```

6.6 Monitor store failures

6.6.1 Identifying symptoms of store corruption

Ceph monitor stores the cluster map in a key/value store such as LevelDB. If a monitor fails due to the key/value store corruption, following error messages might be found in the monitor log:

```
Corruption: error in middle of record
```

Or:

```
Corruption: 1 missing files; e.g.: /var/lib/ceph/mon/mon.foo/store.db/1234567.ldb
```

6.6.2 Recovering using healthy monitors

If there are any survivors, replace the corrupted one with a new one. After booting up, the new joiner will sync up with a healthy peer, and once it is fully synchronized, it will be able to serve the clients.

6.6.3 Recovering using OSDs

But what if all monitors fail at the same time? Since users are encouraged to deploy at least three (and preferably five) monitors in a Ceph cluster, the chance of simultaneous failure is rare. But unplanned power-downs in a data center with improperly configured disk/fs settings could fail the underlying file system, and hence kill all the monitors. In this case, we can recover the monitor store with the information stored in OSDs.

```
  ms=/root/mon-store
  mkdir $ms
  # collect the cluster map from stopped OSDs
  for host in $hosts; do
    rsync -avz $ms/. user@$host:$ms.remote
    rm -rf $ms
    ssh user@$host EOF
      for osd in /var/lib/ceph/osd/ceph-*; do
        ceph-objectstore-tool --data-path \$osd --no-mon-config --op update-mon-db --mon-
store-path $ms.remote
      done
  EOF
    rsync -avz user@$host:$ms.remote/. $ms
  done

  # rebuild the monitor store from the collected map, if the cluster does not
  # use cephx authentication, we can skip the following steps to update the
  # keyring with the caps, and there is no need to pass the "--keyring" option.
  # i.e. just use "ceph-monstore-tool $ms rebuild" instead
  ceph-authtool /path/to/admin.keyring -n mon. \
    --cap mon 'allow *'
  ceph-authtool /path/to/admin.keyring -n client.admin \
    --cap mon 'allow *' --cap osd 'allow *' --cap mds 'allow *'
  # add one or more ceph-mgr's key to the keyring. in this case, an encoded key
  # for mgr.x is added, you can find the encoded key in
  # /etc/ceph/${cluster}.${mgr_name}.keyring on the machine where ceph-mgr is
  # deployed
  ceph-authtool /path/to/admin.keyring --add-key 'AQDN8kBe9PLWARAAZwxXMr
+n85SBYbSlLcZnMA==' -n mgr.x \
    --cap mon 'allow profile mgr' --cap osd 'allow *' --cap mds 'allow *'
  # if your monitors' ids are not single characters like 'a', 'b', 'c', please
  # specify them in the command line by passing them as arguments of the "--mon-ids"
  # option. if you are not sure, please check your ceph.conf to see if there is any
  # sections named like '[mon.foo]'. don't pass the "--mon-ids" option, if you are
  # using DNS SRV for looking up monitors.
  ceph-monstore-tool $ms rebuild -- --keyring /path/to/admin.keyring --mon-ids alpha beta
 gamma

  # make a backup of the corrupted store.db just in case!  repeat for
  # all monitors.
  mv /var/lib/ceph/mon/mon.foo/store.db /var/lib/ceph/mon/mon.foo/store.db.corrupted

  # move rebuild store.db into place.  repeat for all monitors.
  mv $ms/store.db /var/lib/ceph/mon/mon.foo/store.db
  chown -R ceph:ceph /var/lib/ceph/mon/mon.foo/store.db
```

1. Collect the map from all OSD hosts.

2. Rebuild the store.

3. Fill the entities in the keyring file with appropriate caps.

4. Replace the corrupted store on `mon.foo` with the recovered copy.

6.6.3.1 Known limitations

The following information is not recoverable using the steps above:

- Some added keyrings: all the OSD keyrings added using the **`ceph auth add`** command are recovered from the OSD's copy. The `client.admin` keyring is imported using `ceph-monstore-tool`. The MDS keyrings and other keyrings are missing in the recovered monitor store. You may need to re-add them manually.

- Creating pools: If any RADOS pools were in the process of being creating, that state is lost. The recovery tool assumes that all pools have been created. If there are PGs that are stuck in the `unknown` state after the recovery for a partially created pool, you can force creation of the empty PG with the **`ceph osd force-create-pg`** command. This will create an empty PG, so only do this if you know the pool is empty.

- MDS Maps: the MDS maps are lost.

6.7 Next steps

6.7.1 Preparing your logs

Monitor logs are, by default, kept in `/var/log/ceph/ceph-mon.FOO.log*`. However, your logs may not have the necessary information. If you do not find your monitor logs at their default location, you can check where they are by running:

```
cephuser@adm > ceph-conf --name mon.FOO --show-config-value log_file
```

The amount of information in the logs are subject to the debug levels being enforced by your configuration files. If you have not enforced a specific debug level, then Ceph is using the default levels and your logs may not contain important information to track down you issue. A first step in getting relevant information into your logs will be to raise debug levels. Similarly to what happens on other components, different parts of the monitor will output their debug information on different subsystems. You will have to raise the debug levels of those subsystems more closely related to your issue. For most situations, setting the following options on your monitors will be enough to pinpoint a potential source of the issue:

```
debug mon = 10
debug ms = 1
```

6.7.2 Adjusting debug levels

You do not need to restart a monitor to adjust debug legals. You may do it in one of two ways:

- If you have a quorum, either inject the debug option into the monitor you want to debug:

  ```
  cephuser@adm > ceph tell mon.FOO config set debug_mon 10/10
  ```

  Or into all monitors at once:

  ```
  cephuser@adm > ceph tell mon.* config set debug_mon 10/10
  ```

- If you have no quorum, use the monitor's admin socket and directly adjust the configuration options:

  ```
  cephuser@adm > ceph daemon mon.FOO config set debug_mon 10/10
  ```

Going back to default values is as easy as rerunning the above commands using the debug level 1/10 instead. You can check your current values using the admin socket and the following commands:

```
cephuser@adm > ceph daemon mon.FOO config show
```

Or:

```
cephuser@adm > ceph daemon mon.FOO config get 'OPTION_NAME'
```

6.8 Manually deploying a MGR daemon

cephadm requires a MGR daemon in order to manage the cluster. If the last MGR of a cluster was removed, follow these steps to deploy an example MGR daemon named `mgr.hostname.smfvfd` on a random host of your cluster manually:

1. Disable the cephadm scheduler to prevent cephadm from removing the new MGR daemon:

   ```
   cephuser@adm > ceph config-key set mgr/cephadm/pause true
   ```

2. Get or create the auth entry for the new MGR daemon:

   ```
   cephuser@adm > ceph auth get-or-create mgr.hostname.smfvfd \
   mon "profile mgr" osd "allow *" mds "allow *"
   ```

3. Generate a minimal `ceph.conf`:

   ```
   cephuser@adm > ceph config generate-minimal-conf
   ```

4. Find the name of the container image:

   ```
   cephuser@adm > ceph config get "mgr.hostname.smfvfd" container_image
   ```

5. Create a file `config-json.json` which contains the information necessary to deploy the daemon, for example:

   ```
   {
     "config": "# minimal ceph.conf for 8255263a-a97e-4934-822c-00bfe029b28f
   \n[global]\n\tfsid = 8255263a-a97e-4934-822c-00bfe029b28f\n\tmon_host =
    [v2:192.168.0.1:40483/0,v1:192.168.0.1:40484/0]\n",
     "keyring": "[mgr.hostname.smfvfd]\n\tkey = V2VyIGRhcyBsaWVzdCBpc3QgZG9vZi4=\n"
   }
   ```

6. Deploy the daemon:

   ```
   cephuser@adm > cephadm --image IMAGE_NAME \
    deploy --fsid CLUSTER_FSID \
    --name mgr.hostname.smfvfd --config-json config-json.json
   ```

7 Troubleshooting networking

7.1 Identifying OSD networking issues

Ceph is a distributed storage system that depends on networks to peer with OSDs, replicate objects, recover from faults and check heartbeats. Networking issues can cause OSD latency and flapping OSDs. See *Section 4.8, “Flapping OSDs”* for details.

Ensure that Ceph processes and Ceph-dependent processes are connected and listening.

```
ss -a | grep ceph
ss -l | grep ceph
sudo ss -p | grep ceph
```

Check network statistics:

```
ss -s
```

8 Troubleshooting NFS Ganesha

8.1 Debugging NFS Ganesha logs

NFS Ganesha has the following log levels: `NULL`, `FATAL`, `MAJ`, `CRIT`, `WARN`, `EVENT`, `INFO`, `DEBUG`, `MID_DEBUG`, `M_DBG`, `FULL_DEBUG` and `F_DBG`.

Find the logs by running the following command:

```
cephuser@adm > cephadm logs --name nfs.SERVICE_ID.hostname
```

For example:

```
 INFO:cephadm:Inferring fsid e4d48df0-fc2a-11ea-8734-525400e03ff1
 -- Logs begin at Mon 2020-09-21 18:29:25 CEST. --
 Sep 22 04:59:08 node1 bash[29705]: 22/09/2020 02:59:08 : epoch 5f6960b1 : node1 :
ganesha.nfsd-1[svc_6] nfs_rpc_process_request :DISP :F_DBG :About to authenticate
Prog=100003, vers=4, proc=1, xid=770234021, SVCXPRT=0x7>
 Sep 22 04:59:08 node1 bash[29705]: 22/09/2020 02:59:08 : epoch 5f6960b1 : node1 :
ganesha.nfsd-1[svc_6] nfs_rpc_process_request :DISP :F_DBG :Before SVCAUTH_CHECKSUM on
SVCXPRT 0x7f081c0030c0 fd 53
 Sep 22 04:59:08 node1 bash[29705]: 22/09/2020 02:59:08 : epoch 5f6960b1 : node1 :
ganesha.nfsd-1[svc_6] export_check_access :RW LOCK :F_DBG :Got read lock on
0x7f0872a64660 (&export_opt_lock) at /home/abuild/rpmbuild/BU>
 Sep 22 04:59:08 node1 bash[29705]: 22/09/2020 02:59:08 : epoch 5f6960b1 :
node1 : ganesha.nfsd-1[svc_6] export_check_access :EXPORT :M_DBG :EXPORT_DEFAULTS
(options=03303002/00080000 , , , >
 Sep 22 04:59:08 node1 bash[29705]: 22/09/2020 02:59:08 : epoch 5f6960b1 :
node1 : ganesha.nfsd-1[svc_6] export_check_access :EXPORT :M_DBG :default options
(options=03303002/ffffffff root_squash , ----, 34-, UDP, TCP,>
 Sep 22 04:59:08 node1 bash[29705]: 22/09/2020 02:59:08 : epoch 5f6960b1 :
node1 : ganesha.nfsd-1[svc_6] export_check_access :EXPORT :M_DBG :Final options
(options=03303002/ffffffff root_squash , ----, 34-, UDP, TCP,>
 Sep 22 04:59:08 node1 bash[29705]: 22/09/2020 02:59:08 : epoch 5f6960b1 : node1 :
ganesha.nfsd-1[svc_6] export_check_access :RW LOCK :F_DBG :Unlocked 0x7f0872a64660
(&export_opt_lock) at /home/abuild/rpmbuild/BUILD/nfs->
 Sep 22 04:59:08 node1 bash[29705]: 22/09/2020 02:59:08 : epoch 5f6960b1 : node1 :
ganesha.nfsd-1[svc_6] get_gsh_client :RW LOCK :F_DBG :Got read lock on 0x7f0872a5e5d0
(&client_by_ip.lock) at /home/abuild/rpmbuild/BUILD>
 Sep 22 04:59:08 node1 bash[29705]: 22/09/2020 02:59:08 : epoch 5f6960b1 : node1 :
ganesha.nfsd-1[svc_6] get_gsh_client :HT CACHE :DEBUG :client_mgr cache hit slot 859
```

8.1.1 Setting the default log level

To change the default log level, you can use a RADOS object to add a `LOG` configuration block. To enable debug logging, change the log level to `FULL_DEBUG` to increase the log verbosity. Keep in mind that this can produce a large amount of log data and adversely affect performance. For example:

```
LOG {
    # The components block contains one or more logging components
    # and the setting to be used.
    components {
        # The ALL component is special. When set it defines the level
        # for all components and overrides any other setting in this block.
        ALL = FULL_DEBUG; # this will likely kill performance
    }
}
```

Put the file into a RADOS configuration object. In this example, the object is named `debugconf-nfs.foo`:

```
cephuser@adm > rados --pool nfs-ganesha --namespace foo put debugconf-nfs.foo debugconf-
nfs.foo
```

Add a reference to the debug configuration object in the RADOS common configuration file:

```
%url "rados://nfs-ganesha/foo/export-1"
%url "rados://nfs-ganesha/foo/userconf-nfs-foo"
%url "rados://nfs-ganesha/foo/debugconf-nfs.foo"
```

Put the updated common configuration file into the RADOS common configuration object. In this example, the common configuration object is named `conf-nfs.foo`:

```
cephuser@adm > rados --pool nfs-ganesha --namespace foo put conf-nfs.foo conf-nfs.foo
```

Once both configuration objects are placed into the RADOS recovery pool, notify the NFS Ganesha daemons of the change:

```
cephuser@adm > rados --pool nfs-ganesha --namespace foo notify conf-nfs.foo conf-nfs.foo
reply client.35746 cookie 94143722285376 : 0 bytes
reply client.35749 cookie 94618743368000 : 0 bytes
```

8.2 Changing the default port

To change the default port for NFS Ganesha you can use a user defined RADOS configuration object to add an `NFS_CORE_PARAM` block.

Set the `NFS_Port` configuration value to the port you want to change. This example uses port `12345`.

```
NFS_CORE_PARAM {
     Enable_NLM = false;
     Enable_RQUOTA = false;
     Protocols = 4;
     NFS_Port = 12345;
}
```

Put the file into a user defined RADOS configuration object. In this example, the object is named `userconf-nfs.foo`:

```
cephuser@adm > rados --pool nfs-ganesha --namespace foo put userconf-nfs.foo userconf-
nfs.foo
```

Add a reference to the parameter in your RADOS common configuration file:

```
%url "rados://nfs-ganesha/foo/export-1"
%url "rados://nfs-ganesha/foo/userconf-nfs.foo"
```

Note

This action is similar to how exports are included in the common configuration file.

Put the updated common configuration file into the RADOS common configuration object. In this example, the common configuration object is named `conf-nfs.foo`:

```
cephuser@adm > rados --pool nfs-ganesha --namespace foo put conf-nfs.foo conf-nfs.foo
```

Once both configuration objects are placed into the RADOS recovery pool, restart the NFS Ganesha daemons using the Ceph orchestrator:

```
cephuser@adm > ceph orch restart nfs.foo
restart nfs.foo.node1 from host 'node1'
restart nfs.foo.node3 from host 'node3'
```

9 Troubleshooting Ceph health status

The following section details the statues that have been triggered and actions to take when the status is displayed.

MONITOR

`MON_DOWN`

One or more monitor daemons are down. The cluster requires a majority of the monitors in order to function. When one or more monitors are down, clients will initially have difficulty connecting to the cluster.

Restart the monitor daemon that is down as soon as possible to reduce the risk of a subsequent monitor failure.

`MON_CLOCK_SKEW`

The clocks on the hosts running the `ceph-mon` monitor daemons are not well synchronized. This health alert is raised if the cluster detects a clock skew greater than `mon_clock_drift_allowed`. Resolve this by synchronizing the clocks using either `ntpd` or `chrony`. If it is impractical to keep the clocks closely synchronized, the `mon_clock_drift_allowed` threshold can be increased, but this value must stay well below the `mon_lease` interval in order for monitor cluster to function properly.

`MON_MSGR2_NOT_ENABLED`

The `ms_bind_msgr2` option is enabled but one or more monitors is not configured to bind to a v2 port in the cluster's monmap. This means that features specific to the msgr2 protocol (for example, encryption) are not available on some or all connections. In most cases this can be corrected by issuing the following command:

```
cephuser@adm > ceph mon enable-msgr2
```

This command changes any monitor configured for the old default port 6789 to continue to listen for v1 connections on 6789 and also listen for v2 connections on the new default 3300 port. If a monitor is configured to listen for v1 connections on a non-standard port (not 6789), then the monmap needs to be modified manually.

MANAGER

`MGR_MODULE_DEPENDENCY`

An enabled manager module is failing its dependency check. This health check should come with a message from the module about the problem. For example, a module might report that a required package is not installed. In which case, the message will read: "Install the required package and restart your manager daemons." This health check only applies to enabled modules. If a module is not enabled, you can see whether it is reporting dependency issues in the output of **`ceph module ls`**.

`MGR_MODULE_ERROR`

A manager module has experienced an unexpected error. Typically, this means an unhandled exception was raised from the module's `serve` function. The human-readable description of the error may be obscurely worded if the exception did not provide a useful description of itself. This health check may indicate a bug. Open a bug report if you think you have encountered a bug. If you believe the error is transient, you may restart your manager daemon(s), or use **`ceph mgr fail`** on the active daemon to prompt a failover to another daemon.

OSDS

`OSD_DOWN`

One or more OSDs are marked down. The `ceph-osd` daemon may have been stopped, or peer OSDs may be unable to reach the OSD over the network. Common causes include a stopped or crashed daemon, a down host, or a network outage. Verify the host is healthy, the daemon is started, and network is functioning. If the daemon has crashed, the daemon log file (`/var/log/ceph/ceph-osd.*`) may contain debugging information.

OSD_*CRUSH TYPE*_DOWN

For example, `OSD_HOST_DOWN` or `OSD_ROOT_DOWN`. All the OSDs within a particular CRUSH subtree are marked down, for example all OSDs on a host.

`OSD_ORPHAN`

An OSD is referenced in the CRUSH Map hierarchy but does not exist. The OSD can be removed from the CRUSH hierarchy with:

```
cephuser@adm > ceph osd crush rm osd.ID
```

`OSD_OUT_OF_ORDER_FULL`

The utilization thresholds for `backfillfull`, `nearfull`, `full`, and `failsafe_full` are not ascending. The thresholds can be adjusted with:

```
cephuser@adm > ceph osd set-backfillfull-ratio RATIO
cephuser@adm > ceph osd set-nearfull-ratio RATIO
cephuser@adm > ceph osd set-full-ratio RATIO
```

`OSD_FULL`

One or more OSDs have exceeded the `full` threshold and is preventing the cluster from servicing writes. Utilization by pool can be checked with:

```
cephuser@adm > ceph df
```

The currently defined `full` ratio can be seen with:

```
cephuser@adm > ceph osd dump | grep full_ratio
```

A short-term workaround to restore write availability is to raise the full threshold by a small amount:

```
cephuser@adm > ceph osd set-full-ratio RATIO
```

New storage should be added to the cluster by deploying more OSDs or existing data should be deleted in order to free up space.

`OSD_BACKFILLFULL`

One or more OSDs have exceeded the `backfillfull` threshold, preventing data from being allowed to rebalance to this device. This is an early warning that rebalancing may not be able to complete and that the cluster is approaching `full`. Utilization by pool can be checked with:

```
cephuser@adm > ceph df
```

`OSD_NEARFULL`

One or more OSDs have exceeded the `nearfull` threshold. This is an early warning that the cluster is approaching `full`. Utilization by pool can be checked with:

```
cephuser@adm > ceph df
```

`OSDMAP_FLAGS`

One or more cluster flags of interest has been set. These flags include:

full

The cluster is flagged as `full` and cannot serve writes

pauserd, pausewr

Paused reads or writes

noup

OSDs are not allowed to start

nodown

OSD failure reports are being ignored and the monitors are not marking OSDs down

noin

OSDs that were previously marked out are not being marked back in when they start

noout

Down OSDs are not automatically marked out after the configured interval

nobackfill, norecover, norebalance

Recovery or data rebalancing is suspended

noscrub, nodeep_scrub

Scrubbing is disabled

notieragent

Cache tiering activity is suspended

With the exception of `full`, these flags can be set or cleared with:

```
cephuser@adm > ceph osd set FLAG
cephuser@adm > ceph osd unset FLAG
```

`OSD_FLAGS`

One or more OSDs or CRUSH `{nodes,device classes}` has a flag of interest set. These flags include:

noup

These OSDs are not allowed to start

nodown

Failure reports for these OSDs are ignored

noin

If these OSDs were previously marked out automatically after a failure, they are not to be marked in when they start

noout

If these OSDs are down they are not automatically marked out after the configured interval

These flags can be set and cleared in batch with:

```
cephuser@adm > ceph osd set-group FLAG WHO
cephuser@adm > ceph osd unset-group FLAG WHO
```

For example:

```
cephuser@adm > ceph osd set-group noup,noout osd.0 osd.1
cephuser@adm > ceph osd unset-group noup,noout osd.0 osd.1
cephuser@adm > ceph osd set-group noup,noout host-foo
cephuser@adm > ceph osd unset-group noup,noout host-foo
cephuser@adm > ceph osd set-group noup,noout class-hdd
cephuser@adm > ceph osd unset-group noup,noout class-hdd
```

`OLD_CRUSH_TUNABLES`

The CRUSH Map is using old settings and should be updated. The oldest tunables that can be used (for example, the oldest client version that can connect to the cluster) without triggering this health warning are determined by the `mon_crush_min_required_version` config option.

`OLD_CRUSH_STRAW_CALC_VERSION`

The CRUSH Map is using an older, sub-optimal method for calculating intermediate weight values for straw buckets. The CRUSH Map requires an update to use the newer method (`straw_calc_version=1`).

`CACHE_POOL_NO_HIT_SET`

One or more cache pools are not configured with a hit set to track utilization. This prevents the tiering agent from identifying cold objects to flush and evict from the cache. Hit sets can be configured on the cache pool with the following:

```
cephuser@adm > ceph osd pool set POOLNAME hit_set_type TYPE
cephuser@adm > ceph osd pool set POOLNAME hit_set_period PERIOD-IN-SECONDS
cephuser@adm > ceph osd pool set POOLNAME hit_set_count NUMBER-OF-HITSETS
cephuser@adm > ceph osd pool set POOLNAME hit_set_fpp TARGET-FALSE-POSITIVE-RATE
```

`OSD_NO_SORTBITWISE`

No SUSE Enterprise Storage 7 v12.y.z OSDs are running but the `sortbitwise` flag has not been set. Set the `sortbitwise` flag before v12.y.z or newer OSDs can start. You can safely set the flag with:

```
cephuser@adm > ceph osd set sortbitwise
```

`POOL_FULL`

One or more pools have reached the quota and are no longer allowing writes. Pool quotas and utilization can be seen with the following command:

```
cephuser@adm > ceph df detail
```

You can either raise the pool quota with the following commands:

```
cephuser@adm > ceph osd pool set-quota POOLNAME max_objects NUM-OBJECTS
cephuser@adm > ceph osd pool set-quota POOLNAME max_bytes NUM-BYTES
```

Or, you can delete existing data to reduce utilization.

`BLUEFS_SPILLOVER`

One or more OSDs that use the BlueStore backend have been allocated db partitions (storage space for metadata, normally on a faster device) but that space has filled, such that metadata has overflowed onto the normal slow device. This is not necessarily an error condition or even unexpected, but if the administrator's expectation was that all metadata would fit on the faster device, it indicates that not enough space was provided. This warning can be disabled on all OSDs with the following command:

```
cephuser@adm > ceph config set osd bluestore_warn_on_bluefs_spillover false
```

Alternatively, it can be disabled on a specific OSD with the following command:

```
cephuser@adm > ceph config set osd.123 bluestore_warn_on_bluefs_spillover false
```

To provide more metadata space, the OSD in question can be destroyed and reprovisioned. This involves data migration and recovery. It is possible to expand the LVM logical volume backing the db storage. If the underlying LV has been expanded, the OSD daemon needs to be stopped and BlueFS informed of the device size change with the following command:

```
cephuser@adm > ceph-bluestore-tool bluefs-bdev-expand --path /var/lib/ceph/osd/ceph-
$ID
```

`BLUEFS_AVAILABLE_SPACE`

To check how much space is free for BlueFS, execute:

```
cephuser@adm > ceph daemon osd.123 bluestore bluefs available
```

This provides output for up to 3 values; `BDEV_DB free`, `BDEV_SLOW free` and `available_from_bluestore`. `BDEV_DB` and `BDEV_SLOW` report the amount of space that has been acquired by BlueFS and is considered free. Value `available_from_bluestore` denotes ability of BlueStore to leave more space to BlueFS. It is normal that this value is different from amount of BlueStore free space, as BlueFS allocation unit is typically larger than BlueStore allocation unit. This means that only part of BlueStore free space is acceptable for BlueFS.

`BLUEFS_LOW_SPACE`

If BlueFS is running low on available free space and there is little `available_from_bluestore`, consider reducing BlueFS' allocation unit size. To simulate available space when the allocation unit is different, execute:

```
cephuser@adm > ceph daemon osd.123 bluestore bluefs available ALLOC-UNIT-SIZE
```

`BLUESTORE_FRAGMENTATION`

As BlueStore works, free space on underlying storage becomes fragmented. This is normal and unavoidable, but excessive fragmentation can cause slowdown. To inspect BlueStore fragmentation, execute:

```
cephuser@adm > ceph daemon osd.123 bluestore allocator score block
```

Score is given in [0-1] range. [0.0 .. 0.4] tiny fragmentation [0.4 .. 0.7] small, acceptable fragmentation [0.7 .. 0.9] considerable, but safe fragmentation [0.9 .. 1.0] severe fragmentation, can impact BlueFS' ability to get space from BlueStore. If detailed report of free fragments is required, execute:

```
cephuser@adm > ceph daemon osd.123 bluestore allocator dump block
```

If the OSD process does not perform fragmentation, inspect with **`ceph-bluestore-tool`**. Get the fragmentation score:

```
cephuser@adm > ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-123 --allocator
 block free-score
```

Dump detailed free chunks:

```
cephuser@adm > ceph-bluestore-tool --path /var/lib/ceph/osd/ceph-123 --allocator
 block free-dump
```

`BLUESTORE_LEGACY_STATFS`

As of SUSE Enterprise Storage 6, BlueStore tracks its internal usage statistics on a per-pool granular basis and one or more OSDs have BlueStore volumes that were created prior to SUSE Enterprise Storage 7.1. If all OSDs are older than SUSE Enterprise Storage 7.1, the per-pool metrics are not available. However, if there is a mix of pre-SUSE Enterprise Storage 7.1 and post-SUSE Enterprise Storage 7.1 OSDs, the cluster usage statistics reported by **`ceph df`** will not be accurate. The old OSDs can be updated to use the new usage tracking scheme by stopping each OSD, running a repair operation, and the restarting it.

For example, if `osd.123` requires an update, run the following command. To identify the unique FSID of the cluster, run **`ceph fsid`**. To identify the Object Gateway daemon name, run **`ceph orch ps ---hostname`** *`HOSTNAME`*.

```
# systemctl stop ceph-FSID@osd.123
cephuser@adm > ceph-bluestore-tool repair --path /var/lib/ceph/osd/ceph-123
# systemctl start ceph-osd@123
```

This warning can be disabled with:

```
cephuser@adm > ceph config set global bluestore_warn_on_legacy_statfs false
```

`BLUESTORE_DISK_SIZE_MISMATCH`

One or more OSDs using BlueStore has an internal inconsistency between the size of the physical device and the metadata tracking its size. This can lead to the OSD crashing in the future. The OSDs in question should be destroyed and re-deployed. To avoid putting any data at risk, re-deploy only one OSD at a time. For example, if *`OSD_ID`* has the error:

```
cephuser@adm > ceph osd out osd.OSD_ID
cephuser@adm > while ! ceph osd safe-to-destroy osd.OSD_ID ; do sleep 1m ; done
cephuser@adm > ceph osd destroy osd.OSD_ID
cephuser@adm > cephadm ceph-volume lvm zap /path/to/device
cephuser@adm > cephadm ceph-volume lvm create --osd-id  OSD_ID--data /path/to/device
```

DEVICE HEALTH

`DEVICE_HEALTH`

One or more devices are expected to fail. The warning threshold is controlled by the `mgr/devicehealth/warn_threshold` configuration option. This warning only applies to OSDs that are currently marked `in`. The expected response to this failure is to mark the device `out`. The data is then migrated off of the device and the hardware is removed from the system. Marking out is normally done automatically if `mgr/devicehealth/self_heal` is enabled based on the `mgr/devicehealth/mark_out_threshold`. Device health can be checked with:

```
cephuser@adm > ceph device info DEVICE-ID
```

Device life expectancy is set by a prediction model run by the Ceph Manager or by an external tool via the command:

```
cephuser@adm > ceph device set-life-expectancy DEVICE-ID FROM TO
```

You can change the stored life expectancy manually, but that usually does not persist—the tool that originally set it reset and changing the stored value does not affect the actual health of the hardware device.

`DEVICE_HEALTH_IN_USE`

One or more devices are expected to fail and has been marked `out` of the cluster based on `mgr/devicehealth/mark_out_threshold`, but the devices are still participating in one more PGs. This may be because it was only recently marked as `out` and the data is still migrating, or because the data cannot be migrated off for some reason (for example, the cluster is nearly full, or the CRUSH hierarchy is such that there is not another suitable OSD to migrate the data to). This message can be silenced by disabling the self heal behavior (setting `mgr/devicehealth/self_heal` to false), by adjusting the `mgr/devicehealth/mark_out_threshold`, or by addressing what is preventing data from being migrated off of the ailing device.

`DEVICE_HEALTH_TOOMANY`

Too many devices are expected to fail and the `mgr/devicehealth/self_heal` behavior is enabled, such that marking `out` all of the ailing devices would exceed the clusters `mon_osd_min_in_ratio` ratio that prevents too many OSDs from being automatically marked `out`. This can indicates that too many devices in the cluster are expected to fail and action is required to add newer (healthier) devices before too many devices fail and data is lost. The health message can also be silenced by adjusting parameters like `mon_osd_min_in_ratio` or `mgr/devicehealth/mark_out_threshold`, but be warned that this increases the likelihood of unrecoverable data loss in the cluster.

DATA HEALTH (POOLS AND PLACEMENT GROUPS)

`PG_AVAILABILITY`

Data availability is reduced and the cluster is unable to service potential read or write requests for some data in the cluster. Specifically, if one or more PGs are in a state that does not allow IO requests to be serviced. Problematic PG states include peering, stale, incomplete, and in-active (if those conditions do not clear quickly). Detailed information about which PGs are affected is available from:

```
cephuser@adm > ceph health detail
```

In most cases the root cause is that one or more OSDs are currently down; see the discussion for `OSD_DOWN` above. The state of specific problematic PGs can be queried with:

```
cephuser@adm > ceph tell PG_ID query
```

`PG_DEGRADED`

Data redundancy is reduced for some data, meaning the cluster does not have the desired number of replicas for all data (for replicated pools) or erasure code fragments (for erasure coded pools). Specifically, if one or more PGs:

- have a degraded or undersized flag set, meaning there are not enough instances of that placement group in the cluster;
- have not had the clean flag set for some time.

`PG_RECOVERY_FULL`

Data redundancy can be reduced or at risk for some data due to a lack of free space in the cluster. Specifically, one or more PGs have the `recovery_toofull` flag set, meaning that the cluster is unable to migrate or recover data because one or more OSDs are above the full threshold. See the discussion for `OSD_FULL` above for steps to resolve this condition.

`PG_BACKFILL_FULL`

Data redundancy can be reduced or at risk for some data due to a lack of free space in the cluster. Specifically, one or more PGs have the `backfill_toofull` flag set, meaning that the cluster is unable to migrate or recover data because one or more OSDs are above the backfillfull threshold. See the discussion for `OSD_BACKFILLFULL` above for steps to resolve this condition.

`PG_DAMAGED`

Data scrubbing has discovered some problems with data consistency in the cluster. Specifically, one or more PGs have the inconsistent or `snaptrim_error` flag is set, indicating an earlier scrub operation found a problem, or that the repair flag is set and a repair for such an inconsistency is currently in progress.

`OSD_SCRUB_ERRORS`

Recent OSD scrubs have uncovered inconsistencies. This error is generally paired with `PG_DAMAGED`.

`LARGE_OMAP_OBJECTS`

One or more pools contain large omap objects as determined by `osd_deep_scrub_large_omap_object_key_threshold` (threshold for number of keys to determine a large omap object) or `osd_deep_scrub_large_omap_object_value_sum_threshold` (the threshold for summed size (bytes) of all key values to determine a large omap object) or both. More information on the object name, key count, and size in bytes can be found by searching the cluster log for ‘Large omap object found'. Large omap objects can be caused by RGW bucket index objects that do not have automatic resharding enabled. The thresholds can be adjusted with:

```
cephuser@adm > ceph config set osd
 osd_deep_scrub_large_omap_object_key_threshold KEYS
cephuser@adm > ceph config set osd
 osd_deep_scrub_large_omap_object_value_sum_threshold BYTES
```

`CACHE_POOL_NEAR_FULL`

A cache tier pool is nearly full. Full is determined by the `target_max_bytes` and `target_max_objects` properties on the cache pool. Once the pool reaches the target threshold, write requests to the pool may block while data is flushed and evicted from the cache, a state that normally leads to very high latencies and poor performance. The cache pool target size can be adjusted with:

```
cephuser@adm > ceph osd pool set CACHE-POOL-NAME target_max_bytes BYTES
cephuser@adm > ceph osd pool set CACHE-POOL-NAME target_max_objects OBJECTS
```

Normal cache flush and eviction activity can also be throttled due to reduced availability, performance of the base tier, or overall cluster load.

`POOL_TOO_FEW_PGS`

One or more pools should probably have more PGs, based on the amount of data that is currently stored in the pool. This can lead to sub-optimal distribution and balance of data across the OSDs in the cluster, and similarly reduce overall performance. This warning is generated if the `pg_autoscale_mode` property on the pool is set to warn. To disable the warning, you can disable auto-scaling of PGs for the pool entirely with:

```
cephuser@adm > ceph osd pool set POOL-NAME pg_autoscale_mode off
```

To allow the cluster to automatically adjust the number of PGs:

```
cephuser@adm > ceph osd pool set POOL-NAME pg_autoscale_mode on
```

You can also manually set the number of PGs for the pool to the recommended amount with:

```
cephuser@adm > ceph osd pool set POOL_NAME pg_num NEW_PG_NUM
```

`TOO_MANY_PGS`

The number of PGs in use in the cluster is above the configurable threshold of `mon_max_pg_per_osd` PGs per OSD. If this threshold is exceeded, the cluster does not allow new pools to be created, pool `pg_num` to be increased, or pool replication to be increased (any of which would lead to more PGs in the cluster). A large number of PGs can lead to higher memory utilization for OSD daemons, slower peering after cluster state changes (like OSD restarts, additions, or removals), and higher load on the Ceph Manager and Ceph Monitor daemons. The simplest way to mitigate the problem is to increase the number of OSDs in the cluster by adding more hardware. The OSD count used for the purposes of this health check is the number of `in` OSDs, marking `out` OSDs `in` (if there are any) can also help:

```
cephuser@adm > ceph osd in OSD_IDs
```

`POOL_TOO_MANY_PGS`

One or more pools require more PGs based on the amount of data that is currently stored in the pool. This can lead to higher memory utilization for OSD daemons, slower peering after cluster state changes (like OSD restarts, additions, or removals), and higher load on the manager and monitor daemons. This warning is generated if the `pg_autoscale_mode` property on the pool is set to warn. To disable the warning, you can disable auto-scaling of PGs for the pool entirely with:

```
cephuser@adm > ceph osd pool set POOL_NAME pg_autoscale_mode off
```

To allow the cluster to automatically adjust the number of PGs:

```
cephuser@adm > ceph osd pool set POOL_NAME pg_autoscale_mode on
```

You can also manually set the number of PGs for the pool to the recommended amount with:

```
cephuser@adm > ceph osd pool set POOL_NAME pg_num NEW_PG_-NUM
```

`POOL_TARGET_SIZE_RATIO_OVERCOMMITTED`

One or more pools have a `target_size_ratio` property set to estimate the expected size of the pool as a fraction of total storage, but the value(s) exceed the total available storage (either by themselves or in combination with other pools' actual usage). This can indicate that the `target_size_ratio` value for the pool is too large, and should be reduced or set to zero with:

```
cephuser@adm > ceph osd pool set POOL-NAME target_size_ratio 0
```

`POOL_TARGET_SIZE_BYTES_OVERCOMMITTED`

One or more pools have a `target_size_bytes` property set to estimate the expected size of the pool, but the value(s) exceed the total available storage (either by themselves or in combination with other pools' actual usage). This indicates that the `target_size_bytes` value for the pool is too large and should be reduced or set to zero with:

```
cephuser@adm > ceph osd pool set POOL-NAME target_size_bytes 0
```

`TOO_FEW_OSDS`

The number of OSDs in the cluster is below the configurable threshold of `osd_pool_default_size`.

`SMALLER_PGP_NUM`

One or more pools have a `pgp_num` value less than `pg_num`, indicating that the PG count was increased without also increasing the placement behavior. To adjust the placement group number, adjust `pgp_num` and `pg_num`. Ensure that changing `pgp_num` is performed first and does not trigger the rebalance. To resolve, set `pgp_num` to match `pg_num` and trigger the data migration with:

```
cephuser@adm > ceph osd pool set POOL pgp_num PG_NUM_VALUE
```

`MANY_OBJECTS_PER_PG`

One or more pools has an average number of objects per PG that is significantly higher than the overall cluster average. The specific threshold is controlled by the `mon_pg_warn_max_object_skew` configuration value. This indicates that the pool(s) containing most of the data in the cluster have too few PGs, or that other pools that do not contain as much data have too many PGs. The threshold can be raised to silence the health warning by adjusting the `mon_pg_warn_max_object_skew` configuration option on the monitors.

`POOL_APP_NOT_ENABLED`

A pool exists that contains one or more objects but has not been tagged for use by a particular application. Resolve this warning by labeling the pool for use by an application. For example, if the pool is used by RBD:

```
cephuser@adm > rbd pool init POOLNAME
```

If the pool is being used by a custom application *`FOO`*, you can also label via the low-level command:

```
cephuser@adm > ceph osd pool application enable FOO
```

`POOL_FULL`

One or more pools has reached (or is very close to reaching) its quota. The threshold to trigger this error condition is controlled by the `mon_pool_quota_crit_threshold` configuration option. Pool quotas can be adjusted up or down (or removed) with:

```
cephuser@adm > ceph osd pool set-quota POOL max_bytes BYTES
cephuser@adm > ceph osd pool set-quota POOL max_objects OBJECTS
```

Setting the quota value to 0 disables the quota.

`POOL_NEAR_FULL`

One or more pools are approaching its quota. The threshold to trigger this warning condition is controlled by the `mon_pool_quota_warn_threshold` configuration option. Pool quotas can be adjusted up or down (or removed) with:

```
cephuser@adm > ceph osd pool set-quota POOL max_bytes BYTES
cephuser@adm > ceph osd pool set-quota POOL max_objects OBJECTS
```

`OBJECT_MISPLACED`

One or more objects in the cluster is not stored on the node the cluster would like it to be stored on. This is an indication that data migration due to some recent cluster change has not yet completed. Misplaced data is not a dangerous condition in and of itself. Data consistency is not at risk and old copies of objects are not removed until the desired number of new copies (in the desired locations) are present.

`OBJECT_UNFOUND`

One or more objects in the cluster cannot be found. Specifically, the OSDs know that a new or updated copy of an object should exist, but a copy of that version of the object has not been found on OSDs that are currently online. Read or write requests to unfound objects will block. Ideally, a down OSD can be brought back online that has the more recent copy of the unfound object. Candidate OSDs can be identified from the peering state for the PG(s) responsible for the unfound object:

```
cephuser@adm > ceph tell PG_ID query
```

If the latest copy of the object is not available, the cluster can be told to roll back to a previous version of the object.

`SLOW_OPS`

One or more OSD requests is taking a long time to process. This can be an indication of extreme load, a slow storage device, or a software bug. The request queue on the OSD(s) in question can be queried with the following command, executed from the OSD host:

```
cephuser@adm > ceph daemon osd.ID ops
```

A summary of the slowest recent requests can be seen with:

```
cephuser@adm > ceph daemon osd.ID dump_historic_ops
```

The location of an OSD can be found with:

```
cephuser@adm > ceph osd find osd.ID
```

`PG_NOT_SCRUBBED`

One or more PGs have not been scrubbed recently. PGs are normally scrubbed every `mon_scrub_interval` seconds and this warning triggers when `mon_warn_pg_not_deep_scrubbed_ratio` percentage of interval has elapsed without a scrub since it was due. PGs do not scrub if they are not flagged as clean. This can happen if they are misplaced or degraded (see `PG_AVAILABILITY` and `PG_DEGRADED` above). You can manually initiate a scrub of a clean PG with:

```
cephuser@adm > ceph pg scrub PG_ID
```

`PG_NOT_DEEP_SCRUBBED`

One or more PGs have not been deep scrubbed recently. PGs are normally scrubbed every `osd_deep_scrub_interval` seconds and this warning triggers when `mon_warn_pg_not_deep_scrubbed_ratio` percentage of interval has elapsed without a scrub since it was due. PGs do not (deep) scrub if they are not flagged as clean. This can happen if they are misplaced or degraded (see `PG_AVAILABILITY` and `PG_DEGRADED` above). You can manually initiate a scrub of a clean PG with:

```
cephuser@adm > ceph pg deep-scrub PG_ID
```

HEALTH CHECKS

`CEPHADM_PAUSED`

cephadm background work has been paused with **`ceph orch pause`**. cephadm continues to perform passive monitoring activities (for example, checking host and daemon status), but it will not make any changes (for example, deploying or removing daemons).

Resume cephadm work with:

```
cephuser@adm > ceph orch resume
```

`CEPHADM_STRAY_HOST`

One or more hosts have running Ceph daemons but are not registered as hosts managed by cephadm. This means that those services cannot currently be managed by cephadm. For example, restarted, upgraded, included in **`ceph orch ps`**.
You can manage the host(s) with:

```
cephuser@adm > ceph orch host add hostname
```

Note

You may need to configure SSH access to the remote host before this will work.

Alternatively, you can manually connect to the host and ensure that services on that host are removed or migrated to a host that is managed by cephadm.
You can also disable this warning entirely with:

```
cephuser@adm > ceph config set mgr mgr/cephadm/warn_on_stray_hosts false
```

`CEPHADM_STRAY_DAEMON`

One or more Ceph daemons are running but not are not managed by cephadm. This may be because they were deployed using a different tool, or because they were started manually. Those services cannot currently be managed by cephadm. For example, restarted, upgraded, or included in **`ceph orch ps`**..
If the daemon is a stateful one (MON or OSD), it should be adopted by cephadm. For stateless daemons, it is usually easiest to provision a new daemon with the **`ceph orch apply`** command and then stop the unmanaged daemon.
This warning can be disabled entirely with:

```
cephuser@adm > ceph config set mgr mgr/cephadm/warn_on_stray_daemons false
```

`CEPHADM_HOST_CHECK_FAILED`

One or more hosts have failed the basic cephadm host check, which verifies that the host is reachable and **`cephadm`** can be executed there, and that the host satisfies basic prerequisites, like a working container runtime (podman or docker) and working time synchronization. If this test fails, cephadm will not be able to manage services on that host.

You can manually run this check with:

```
cephuser@adm > ceph cephadm check-host hostname
```

You can remove a broken host from management with:

```
cephuser@adm > ceph orch host rm hostname
```

You can disable this health warning with:

```
cephuser@adm > ceph config set mgr mgr/cephadm/warn_on_failed_host_check false
```

MISCELLANEOUS

`RECENT_CRASH`

One or more Ceph daemons have crashed recently, and the crash has not yet been archived or acknowledged by the administrator. This may indicate a software bug, a hardware problem (for example, a failing disk), or some other problem. New crashes can be listed with:

```
cephuser@adm > ceph crash ls-new
```

Information about a specific crash can be examined with:

```
cephuser@adm > ceph crash info CRASH-ID
```

This warning can be silenced by archiving the crash (perhaps after being examined by an administrator) so that it does not generate this warning:

```
cephuser@adm > ceph crash archive CRASH-ID
```

Similarly, all new crashes can be archived with:

```
cephuser@adm > ceph crash archive-all
```

Archived crashes are still visible via **`ceph crash ls`** but not **`ceph crash ls-new`**. The time period for what recent means is controlled by the option `mgr/crash/warn_recent_interval` (default: two weeks). These warnings can be disabled entirely with:

```
cephuser@adm > ceph config set mgr/crash/warn_recent_interval 0
```

`TELEMETRY_CHANGED`

Telemetry has been enabled but the contents of the telemetry report have changed since that time, so telemetry reports are not sent. The Ceph developers periodically revise the telemetry feature to include new and useful information, or to remove information found to be useless or sensitive. If any new information is included in the report, Ceph requires the administrator to re-enable telemetry to ensure they have an opportunity to (re)review what information is shared. To review the contents of the telemetry report:

```
cephuser@adm > ceph telemetry show
```

The telemetry report consists of several optional channels that are independently enabled or disabled. To re-enable telemetry (and make this warning go away):

```
cephuser@adm > ceph telemetry on
```

To disable telemetry (and make this warning go away):

```
cephuser@adm > ceph telemetry soff
```

```
groups:
 - name: cluster health
  rules:
   - alert: health error
    expr: ceph_health_status == 2
    for: 5m
    labels:
     severity: critical
     type: ses_default
    annotations:
     description: Ceph in error for > 5m
   - alert: unhealthy
    expr: ceph_health_status != 0
    for: 15m
    labels:
     severity: warning
     type: ses_default
    annotations:
     description: Ceph not healthy for > 5m
 - name: mon
  rules:
   - alert: low monitor quorum count
    expr: ceph_monitor_quorum_count < 3
    labels:
     severity: critical
     type: ses_default
    annotations:
     description: Monitor count in quorum is low
 - name: osd
  rules:
   - alert: 10% OSDs down
    expr: sum(ceph_osd_down) / count(ceph_osd_in) >= 0.1
    labels:
     severity: critical
     type: ses_default
    annotations:
     description: More then 10% of OSDS are down
   - alert: OSD down
    expr: sum(ceph_osd_down) > 1
    for: 15m
    labels:
     severity: warning
     type: ses_default
    annotations:
     description: One or more OSDS down for more then 15 minutes
   - alert: OSDs near full
    expr: (ceph_osd_utilization unless on(osd) ceph_osd_down) > 80
    labels:
     severity: critical
     type: ses_default
    annotations:
     description: OSD {{ $labels.osd }} is dangerously full, over 80%
   # alert on single OSDs flapping
   - alert: flap osd
    expr: rate(ceph_osd_up[5m])*60 > 1
    labels:
     severity: warning
     type: ses_default
    annotations:
     description: >
       OSD {{ $label.osd }} was marked down at back up at least once a
       minute for 5 minutes.
   # alert on high deviation from average PG count
   - alert: high pg count deviation
    expr: abs(((ceph_osd_pgs > 0) - on (job) group_left avg(ceph_osd_pgs > 0) by
(job)) / on (job) group_left avg(ceph_osd_pgs > 0) by (job)) > 0.35
    for: 5m
    labels:
     severity: warning
     type: ses_default
    annotations:
     description: >
       OSD {{ $labels.osd }} deviates by more then 30% from
       average PG count
   # alert on high commit latency...but how high is too high
 - name: mds
  rules:
  # no mds metrics are exported yet
 - name: mgr
```

```
  rules:
  # no mgr metrics are exported yet
 - name: pgs
  rules:
   - alert: pgs inactive
    expr: ceph_total_pgs - ceph_active_pgs > 0
    for: 5m
    labels:
     severity: critical
     type: ses_default
    annotations:
     description: One or more PGs are inactive for more then 5 minutes.
   - alert: pgs unclean
    expr: ceph_total_pgs - ceph_clean_pgs > 0
    for: 15m
    labels:
     severity: warning
     type: ses_default
    annotations:
     description: One or more PGs are not clean for more then 15 minutes.
 - name: nodes
  rules:
   - alert: root volume full
    expr: node_filesystem_avail{mountpoint="/"} / node_filesystem_size{mountpoint="/"} <
0.1
    labels:
     severity: critical
     type: ses_default
    annotations:
     description: Root volume (OSD and MON store) is dangerously full (< 10% free)
   # alert on nic packet errors and drops rates > 1 packet/s
   - alert: network packets dropped
    expr: irate(node_network_receive_drop{device!="lo"}[5m]) +
irate(node_network_transmit_drop{device!="lo"}[5m]) > 1
    labels:
     severity: warning
     type: ses_default
    annotations:
     description: >
      Node {{ $labels.instance }} experiences packet drop > 1
      packet/s on interface {{ $lables.device }}
   - alert: network packet errors
    expr: irate(node_network_receive_errs{device!="lo"}[5m]) +
irate(node_network_transmit_errs{device!="lo"}[5m]) > 1
    labels:
     severity: warning
     type: ses_default
    annotations:
     description: >
      Node {{ $labels.instance }} experiences packet errors > 1
      packet/s on interface {{ $lables.device }}
   # predict fs fillup times
   - alert: storage filling
    expr: ((node_filesystem_free - node_filesystem_size) /
deriv(node_filesystem_free[2d]) <= 5) > 0
    labels:
     severity: warning
     type: ses_default
    annotations:
     description: >
      Mountpoint {{ $lables.mountpoint }} will be full in less then 5 days
      assuming the average fillup rate of the past 48 hours.
 - name: pools
  rules:
   - alert: pool full
    expr: ceph_pool_used_bytes / ceph_pool_available_bytes > 0.9
    labels:
     severity: critical
     type: ses_default
    annotations:
     description: Pool {{ $labels.pool }} at 90% capacity or over
   - alert: pool filling up
    expr: (-ceph_pool_used_bytes / deriv(ceph_pool_available_bytes[2d]) <= 5 ) > 0
    labels:
     severity: warning
     type: ses_default
    annotations:
     description: >
      Pool {{ $labels.pool }} will be full in less then 5 days
      assuming the average fillup rate of the past 48 hours.
```

10 Troubleshooting the Ceph Dashboard

10.1 Locating the Ceph Dashboard

If you are unsure of the location of the Ceph Dashboard, run the following command:

```
cephuser@adm > ceph mgr services | grep dashboard
"dashboard": "https://host:port"
```

The command returns the URL where the Ceph Dashboard is located: (https://*`host`*:*`port`*/)

10.2 Accessing the Ceph Dashboard

If you are unable to access the Ceph Dashboard, run through the following commands.

1. Verify the Ceph Dashboard module is enabled:

   ```
   cephuser@adm > ceph mgr module ls
   ```

2. Ensure the Ceph Dashboard module is listed in the `enabled_modules` section. Example snipped output from the **`ceph mgr module ls`** command:

   ```
   {
       "always_on_modules": [
           "balancer",
           "crash",
           "devicehealth",
           "orchestrator",
           "osd_support",
           "pg_autoscaler",
           "progress",
           "rbd_support",
           "status",
           "telemetry",
           "volumes"
       ],
       "enabled_modules": [
           "dashboard",
           "iostat",
           "restful"
       ],
   ```

```
      "disabled_modules": [
          {
               ...
          }
      ]
  }
```

3. If it is not listed, activate the module with the following command:

```
cephuser@adm > ceph mgr module enable dashboard
```

4. Check the Ceph Manager log file for any errors. To do this, first you will need the name of the Ceph Manager daemon. Find this with the **`cephadm ls`** command. It will return output similar to the following snippet:

```
{ ... ,
  {
    "style": "cephadm:v1",
    "name": "mgr.master.jqxsqf",
    "fsid": "f09a96d0-2a55-11eb-a87b-525400c955e8",
    ...
  },
  ...
}
```

You can then use the name given by the command above to view the log file:

```
cephuser@adm > cephadm logs --name mgr.master.jqxsqf -- -f
```

5. Ensure the SSL/TSL support is configured properly:

```
cephuser@adm > ceph config-key get mgr/dashboard/key
cephuser@adm > ceph config-key get mgr/dashboard/crt
```

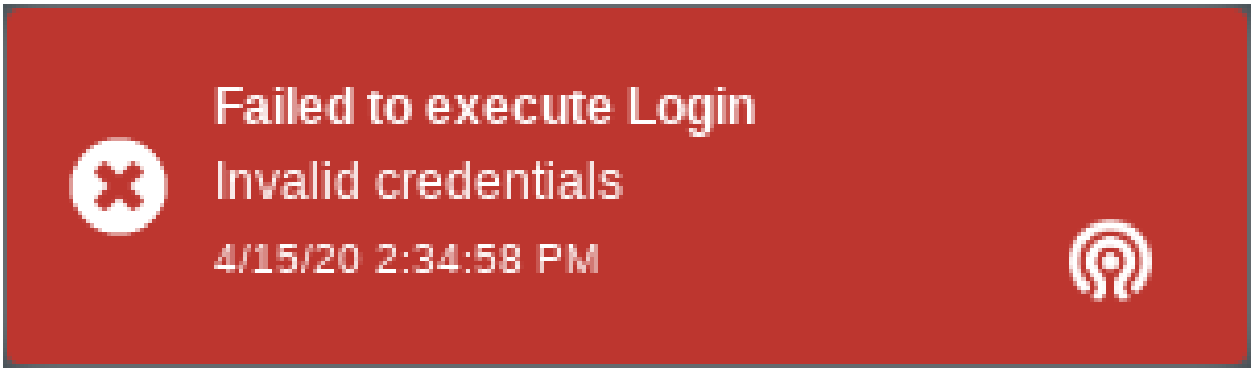
6. Verify the self-signed certificate exists. If not, run:

```
cephuser@adm > ceph dashboard create-self-signed-cert
```

See *Book "Administration and Operations Guide", Chapter 10 "Manual configuration", Section 10.1.1 "Creating self-signed certificates"* for more information on self-signed certificates or *Book "Administration and Operations Guide", Chapter 10 "Manual configuration", Section 10.1.2 "Using certificates signed by CA"* for information on self-signed certificates by CA.

10.3 Troubleshooting logging into the Ceph Dashboard

If you are unable to log into the Ceph Dashboard and you receive the following error, run through the procedural checks below:



Failed to execute Login
Invalid credentials
4/15/20 2:34:58 PM

FIGURE 10.1: FAILED TO EXECUTE LOGIN

1. Check that your user credentials are correct. If you are seeing the notification message above when trying to log into the Ceph Dashboard, it is likely you are using the wrong credentials. Double check your username and password, and ensure the caps lock key is not enabled by accident.

2. If your user credentials are correct, but you are experiencing the same error, check that the user account exists:

   ```
   cephuser@adm > ceph dashboard ac-user-show USERNAME
   ```

   This command returns your user data. If the user does not exist, it will print:

   ```
   Error ENOENT: User USERNAME does not exist
   ```

   In this case, you will need to create the user. See *Book "Administration and Operations Guide", Chapter 11 "Manage users and roles on the command line", Section 11.2 "Managing user accounts"* for more information on user accounts.

3. Check if the user is enabled:

   ```
   cephuser@adm > ceph dashboard ac-user-show USERNAME {..., "enabled": true, ..}
   ```

> Check if `enabled` is set to `true` for your user. If not the user is not enabled, run:
>
> ```
> cephuser@adm > ceph dashboard ac-user-enable USERNAME
> ```

10.4 Determining if a Ceph Dashboard feature is not working

When an error occurs on the backend, you will usually receive an error notification on the frontend. Run through the following scenarios to debug.

1. Check the Ceph Dashboard/mgr logfile(s) for any errors. These can be identified by searching for keywords, such as *500 Internal Server Error*, followed by `traceback`. The end of a traceback contains more details about what exact error occurred.
2. Check your web browser's Javascript Console for any errors.

10.5 Ceph Dashboard logs

10.5.1 Debugging the Ceph Dashboard flag

With this flag enabled, traceback of errors are included in backend responses.

To enable this flag via the Ceph Dashboard, navigate from *Cluster* to *Manager modules*. Select *Dashboard module* and click the edit button. Click the *debug* checkbox and update.

To enable via the CLI, run the following command:

```
cephuser@adm > ceph dashboard debug enable
```

Note

Activating the debug flag temporarily causes a `HEALTH_WARN` cluster state.

10.5.2 Setting logging level of Ceph Dashboard module

Setting the logging level to debug makes the log more verbose and helpful for debugging.

1. Increase the logging level of manager daemons:

   ```
   cephuser@adm > ceph tell mgr config set debug_mgr 20
   ```

2. Adjust the logging level of the Ceph Dashboard module via the Ceph Dashboard or CLI:

   - Navigate from *Cluster* to *Manager modules*. Select *Dashboard module* and click the edit button. Modify the `log_level` configuration.

   - To adjust via the CLI, run the following command:

     ```
     cephuser@adm > bin/ceph config set mgr mgr/dashboard/log_level debug
     ```

10.6 Adding exceptions for self-signed SSL certificates in the Ceph Dashboard

The Web browser shows an error page when accessing any embedded Grafana pages using self-signed SSL certificates. Most browsers do not display the button to add an exception for those pages.

Follow the next steps to add the exception.

1. Open the frame in a separate browser tab to add an exception.
   For example, open the context menu by clicking the right mouse button and select the menu *This Frame* › *Open Frame in New Tab*.

2. In the new opened browser tab, press the *Advanced* button followed by the *Accept the risk and continue* button.

3. Finally, reload the Ceph Dashboard page to see the embedded Grafana pages.

11 Troubleshooting Object Gateway

11.1 Running a basic health check

The most basic health check to test a running Object Gateway process is to simply point your browser or client at the Object Gateway endpoint. This should return an empty bucket results for the anonymous user:

```
root@master # curl -I
```

If the result returns a 405 status, this indicates `MethodNotAllowed`. The output will result in a `NoSuchBucket` XML file. If the result returns a 404 status, this means the Object Gateway DNS name is misconfigured or the requests were not made at an endpoint resolving to Object Gateway DNS name.

11.2 Identifying gateway issues

If the gateway is not running, usually restartinng the gateway (automatic under `systemd`) should restore the service. This can be achieved via:

```
cephuser@adm > ceph orch daemon restart rgw-daemon-name
```

For increasing the log level under the `client.RGW_NAME` section, the Object Gateway configurable can be increased from 1 (default) up to 20 (very verbose). For the messages sent to the cluster itself the messenger debug levels can be raised (off by default). This is controlled via `debug ms`. Usually a level of 1 is sufficient to gather enough information. For injecting arguments in a running process:

Note

This is a temporary change.

```
cephuser@adm > ceph-daemon CLIENT_RGW_NAME config set debug_rgw 20
```

For persistent changes, or a non-running process, it is best to set these lines in the `ceph.conf` file under the `client.rgw-name` section or, alternatively, using the Ceph CLI. In this case, the debug levels are extremely verbose, so it is best to do this only to capture error logs for a short time. To set the logs using the Ceph CLI, run:

```
cephuser@adm > ceph config set CLIENT_RGW_NAME debug_rgw 20
```

11.3 Diagnosing crashed Object Gateway process

If the Object Gateway process dies, you will normally see a connection refused at the client. In that situation, restarting Object Gateway will restore the service.

To diagnose the cause of the crash, check the log in `/var/log/ceph` or the core file (if one was generated).

11.4 Identifying blocked Object Gateway requests

If some (or all) Object Gateway requests appear to be blocked, you can get some insight into the internal state of the Object Gateway daemon via its admin socket. By default, there will be a socket configured to reside in `/var/run/ceph`, and the daemon can be queried with:

```
cephuser@adm > ceph daemon /var/run/ceph/client.rgw-name help
  help                 list available commands
  objecter_requests    show in-progress osd requests
  perfcounters_dump    dump perfcounters value
  perfcounters_schema  dump perfcounters schema
  version              get protocol version
```

Of particular interest:

```
cephuser@adm > ceph daemon /var/run/ceph/client.rgw objecter_requests
...
```

This will dump information about current in-progress requests with the RADOS cluster. This allows you to identify if any requests are blocked by a non-responsive OSD. For example, one might see:

```
{ "ops": [
      { "tid": 1858,
        "pg": "2.d2041a48",
        "osd": 1,
        "last_sent": "2012-03-08 14:56:37.949872",
        "attempts": 1,
        "object_id": "fatty_25647_object1857",
        "object_locator": "@2",
        "snapid": "head",
        "snap_context": "0=[]",
        "mtime": "2012-03-08 14:56:37.949813",
        "osd_ops": [
              "write 0~4096"]},
      { "tid": 1873,
        "pg": "2.695e9f8e",
        "osd": 1,
        "last_sent": "2012-03-08 14:56:37.970615",
        "attempts": 1,
        "object_id": "fatty_25647_object1872",
        "object_locator": "@2",
        "snapid": "head",
        "snap_context": "0=[]",
        "mtime": "2012-03-08 14:56:37.970555",
        "osd_ops": [
              "write 0~4096"]}],
"linger_ops": [],
"pool_ops": [],
"pool_stat_ops": [],
"statfs_ops": []}
```

In this dump, two requests are in progress. The `last_sent` field is the time the RADOS request was sent. If this is a while ago, it suggests that the OSD is not responding. For example, for request 1858, you could check the OSD status with:

```
cephuser@adm > ceph pg map 2.d2041a48
osdmap e9 pg 2.d2041a48 (2.0) -> up [1,0] acting [1,0]
```

This tells you to look at `osd.1`, the primary copy for this PG:

```
  ceph daemon osd.1 ops
  { "num_ops": 651,
   "ops": [
         { "description": "osd_op(client.4124.0:1858 fatty_25647_object1857 [write
 0~4096] 2.d2041a48)",
           "received_at": "1331247573.344650",
           "age": "25.606449",
           "flag_point": "waiting for sub ops",
           "client_info": { "client": "client.4124",
               "tid": 1858}},
  ...
```

The `flag_point` field indicates that the OSD is currently waiting for replicas to respond, in this case `osd.0`.

11.5 Large OMAP issues

If you receive a cluster warning for a large OMAP issue, it means that an object has exceeded one of the OSD deep scrub large OMAP object thresholds, usually set to 200,000 keys and 1 GiB of storage. The relevant pool or PG involved can be found by grepping for *large OMAP object*, as mentioned in the warning message. Depending on the pool and the object key involved it might indicate one of the following:

11.5.1 Resharding issues

If the large OMAP object name happens to be in the `zone.rgw.buckets.index` pool, this means that a bucket has more than 200,000 keys. Dynamic resharding is set on by default in single site clusters which would automatically have resharded the bucket. For multi-site clusters, however, this is not supported and would have to be manually done *Book "Administration and Operations Guide", Chapter 21 "Ceph Object Gateway", Section 21.10.1 "Bucket index resharding"*. Looking at the bucket statistics reveals the total object count and the number of shards. This is helpful to understand if the bucket has already been resharded, in which case the warning is just because the deep scrub happened to run before the reshard process and would be cleared in the next deep scrub.

11.5.2 Reading usage statistics

Usage statistics are also stored as OMAP keys in the `zone.rgw.log` pool. However, these are not automatically trimmed, therefore a manual trimming of usage statistics would have to be done. For example:

```
# radosgw-admin usage trim [--uid=user-id] --start-date=2019-01-01 --end-date=2019-03-03
```

12 Troubleshooting CephFS

12.1 Slow or stuck operations

If you are experiencing apparent hung operations, the first task is to identify where the problem is occurring: in the client, the MDS, or the network connecting them. Start by looking to see if either side has stuck operations and narrow it down from there.

12.2 Checking RADOS health

If part of the CephFS metadata or data pools is unavailable and CephFS is not responding, it is probably because RADOS itself is unhealthy.

Check the cluster's status with the following command:

```
cephuser@adm > ceph status
```

Ceph will print the cluster status. Review *Chapter 2, Troubleshooting logging and debugging*, *Chapter 6, Troubleshooting Ceph Monitors and Ceph Managers*, *Chapter 5, Troubleshooting placement groups (PGs)*, and *Chapter 4, Troubleshooting OSDs* for tips on what may be causing the issue.

12.3 MDS

If an operation is hung inside the MDS, it will eventually show up in **`ceph health`**, identifying “slow requests are blocked”. It may also identify clients as “failing to respond” or misbehaving in other ways. If the MDS identifies specific clients as misbehaving, we recommend investigating the root cause. Often it can be the result of the following:

- Overloading the system
- Running an older (misbehaving) client
- Underlying RADOS issues

12.3.1 Identifying MDS slow requests

You can list current operations via the admin socket by running the following command from the MDS host:

```
cephuser@adm > ceph daemon mds.NAME dump_ops_in_flight
```

Identify the stuck commands and examine why they are stuck. Usually the last event will have been an attempt to gather locks, or sending the operation off to the MDS log. If it is waiting on the OSDs, fix them. If operations are stuck on a specific inode, you probably have a client holding caps which prevent others from using it. This can be because the client is trying to flush out dirty data or because you have encountered a bug in CephFS' distributed file lock code (the file “capabilities” [“caps”] system).

If it is a result of a bug in the capabilities code, restarting the MDS is likely to resolve the problem.

If there are no slow requests reported on the MDS, and it is not reporting that clients are misbehaving, either the client has a problem or its requests are not reaching the MDS.

12.4 Kernel mount debugging

12.4.1 Slow requests

Unfortunately, the kernel client does not support the admin socket, but it has similar (if limited) interfaces if your kernel has `debugfs` enabled. There will be a folder in `sys/kernel/debug/ceph/`, and that folder contains a variety of files that output interesting output when you cat them. These files are described below; the most interesting when debugging slow requests are probably the `mdsc` and `osdc` files.

bdi

BDI info about the Ceph system (blocks dirtied, written, etc)

caps

Counts of file caps structures in-memory and used

client_options

Dumps the options provided to the CephFS mount

dentry_lru

Dumps the CephFS dentries currently in-memory mdsc

Dumps current requests to the MDS

mdsmap

Dumps the current MDSMap epoch and MDSes

mds_sessions

Dumps the current sessions to MDSes

monc

Dumps the current maps from the monitor, and any subscriptions held

monmap

Dumps the current monitor map epoch and monitors

osdc

Dumps the current ops in-flight to OSDs (ie, file data IO)

osdmap

Dumps the current OSDMap epoch, pools, and OSDs

12.5 Disconnecting and remounting the file system

Because CephFS has a consistent cache, if your network connection is disrupted for a long enough time the client will be forcibly disconnected from the system. At this point, the kernel client is in a bind: it cannot safely write back dirty data, and many applications do not handle IO errors correctly on `close()`. At the moment, the kernel client will remount the FS, but outstanding filesystem IO may or may not be satisfied. In these cases, you may need to reboot your client system.

You can identify you are in this situation if `dmesg/kern.log` report something like:

```
 Jul 20 08:14:38 teuthology kernel: [3677601.123718] ceph: mds0 closed our session
 Jul 20 08:14:38 teuthology kernel: [3677601.128019] ceph: mds0 reconnect start
 Jul 20 08:14:39 teuthology kernel: [3677602.093378] ceph: mds0 reconnect denied
 Jul 20 08:14:39 teuthology kernel: [3677602.098525] ceph:  dropping dirty+flushing Fw
state for ffff8802dc150518 1099935956631
 Jul 20 08:14:39 teuthology kernel: [3677602.107145] ceph:  dropping dirty+flushing Fw
state for ffff8801008e8518 1099935946707
 Jul 20 08:14:39 teuthology kernel: [3677602.196747] libceph: mds0 172.21.5.114:6812
socket closed (con state OPEN)
 Jul 20 08:14:40 teuthology kernel: [3677603.126214] libceph: mds0 172.21.5.114:6812
connection reset
```

```
Jul 20 08:14:40 teuthology kernel: [3677603.132176] libceph: reset on mds0
```

This is an area of ongoing work to improve the behavior. Kernels will soon be reliably issuing error codes to in-progress IO, although your application(s) may not deal with them well. In the longer-term, we hope to allow reconnect and reclaim of data in cases where it will not violate POSIX semantics.

12.6 Mounting

12.6.1 Mount I/O error

A `mount 5` (EIO, I/O error) error typically occurs if a MDS server is laggy or if it crashed. Ensure at least one MDS is up and running, and the cluster is `active + healthy`.

12.6.2 Mount out of memory error

A `mount 12` error (ENOMEM, out of memory) with `cannot allocate memory` usually occurs if you have a version mismatch between the Ceph Client version and the Ceph Storage Cluster version. Check the versions using:

```
ceph -v
```

If the Ceph Client is behind the Ceph cluster, try to upgrade it:

```
sudo zypper up
sudo zypper in ceph-common
```

You may need to uninstall, autoclean and autoremove **`ceph-common`** and then reinstall it so that you have the latest version.

12.7 Mounting CephFS using old kernel clients

The kernel since SUSE Linux Enterprise Server 15 SP2 includes a CephFS client that is able to take full advantage of all the features available on an SES7 cluster. All relevant features and bug fixes are backported to this operating system.

However, it may be necessary to access CephFS from other systems that may provide an older CephFS client, which may not support all the features required by an SUSE Enterprise Storage 7.1 cluster. When this happens, the kernel client will fail to mount the file system and will emit messages similar to the one shown below:

```
[ 4187.023633] libceph: mon0 192.168.122.150:6789 feature set mismatch, my 107b84a842aca
 < server's 40107b84a842aca, missing 400000000000000
[ 4187.023838] libceph: mon0 192.168.122.150:6789 missing required protocol features
```

The message above means that the MON identified 0x400000000000000 as the missing feature in the client (the value 0x107b84a842aca represents all the features supported by the client, while 0x40107b84a842aca represents the minimum set of features required by the cluster). From the following table, which shows the complete list of feature bits, we can see that the missing feature bit 58 (2ˆ58 = 0x400000000000000) is CRUSH_TUNABLES5, NEW_OSDOPREPLY_ENCODING, or FS_FILE_LAYOUT_V2 (all these three features share the same feature bit).

TABLE 12.1: CEPH FEATURES

Feature	Bit	Value
UID	0	0x1
NOSRCADDR	1	0x2
FLOCK	3	0x8
SUBSCRIBE2	4	0x10
MONNAMES	5	0x20
RECONNECT_SEQ	6	0x40
DIRLAYOUTHASH	7	0x80
OBJECTLOCATOR	8	0x100
PGID64	9	0x200
INCSUBOSDMAP	10	0x400
PGPOOL3	11	0x800
OSDREPLYMUX	12	0x1000
OSDENC	13	0x2000

Feature	Bit	Value
SERVER_KRAKEN	14	0x4000
MONENC	15	0x8000
CRUSH_TUNABLES	18	0x40000
SERVER_LUMINOUS	21	0x200000
RESEND_ON_SPLIT	21	0x200000
RADOS_BACKOFF	21	0x200000
OSDMAP_PG_UPMAP	21	0x200000
CRUSH_CHOOSE_ARGS	21	0x200000
MSG_AUTH	23	0x800000
CRUSH_TUNABLES2	25	0x2000000
CREATEPOOLID	26	0x4000000
REPLY_CREATE_INODE	27	0x8000000
SERVER_M	28	0x10000000
MDSENC	29	0x20000000
OSDHASHPSPOOL	30	0x40000000
MON_SINGLE_PAXOS	31	0x80000000
OSD_CACHEPOOL	35	0x800000000
CRUSH_V2	36	0x1000000000
EXPORT_PEER	37	0x2000000000
OSD_ERASURE_CODES	38	0x4000000000
OSD_OSD_TMAP2OMAP	38	0x4000000000
OSDMAP_ENC	39	0x8000000000
MDS_INLINE_DATA	40	0x10000000000

Feature	Bit	Value
CRUSH_TUNABLES3	41	0x20000000000
OSD_PRIMARY_AFFINITY	41	0x20000000000
MSGR_KEEPALIVE2	42	0x40000000000
OSD_POOLRESEND	43	0x80000000000
ERASURE_CODE_PLUGINS_V2	44	0x100000000000
OSD_FADVISE_FLAGS	46	0x400000000000
MDS_QUOTA	47	0x800000000000
CRUSH_V4	48	0x1000000000000
MON_METADATA	50	0x4000000000000
OSD_BITWISE_HOBJ_SORT	51	0x8000000000000
OSD_PROXY_WRITE_FEATURES	52	0x10000000000000
ERASURE_CODE_PLUGINS_V3	53	0x20000000000000
OSD_HITSET_GMT	54	0x40000000000000
HAMMER_0_94_4	55	0x80000000000000
NEW_OSDOP_ENCODING	56	0x100000000000000
MON_STATEFUL_SUB	57	0x200000000000000
MON_ROUTE_OSDMAP	57	0x200000000000000
OSDSUBOP_NO_SNAPCONTEXT	57	0x200000000000000
SERVER_JEWEL	57	0x200000000000000
CRUSH_TUNABLES5	58	0x400000000000000

Feature	Bit	Value
NEW_OSDOPREPLY_ENCODING	58	0x400000000000000
FS_FILE_LAYOUT_V2	58	0x400000000000000
FS_BTIME	59	0x800000000000000
FS_CHANGE_ATTR	59	0x800000000000000
MSG_ADDR2	59	0x800000000000000
OSD_RECOVERY_DELETES	60	0x1000000000000000
CEPHX_V2	61	0x2000000000000000
RESERVED	62	0x4000000000000000

A possible solution to allow an old kernel client to mount a recent CephFS is to modify the cluster CRUSH profile. CRUSH profiles define a set of CRUSH tunables that are named after the Ceph versions in which they were introduced. For example, the `firefly` tunables are first supported in the Firefly release (0.80), and older clients will not be able to access the cluster. Thus, to fix the problem shown above, the following command can be used:

```
cephuser@adm > ceph osd crush tunables hammer
```

This will adjust the CRUSH profile to the behaviour it had for the Hammer (0.94) release. Note however that this is not the optimal behaviour for the cluster. To change back to the optimal profile, run the following command:

```
cephuser@adm > ceph osd crush tunables optimal
```

The following table lists the available CRUSH profiles and which CRUSH tunables versions (the CRUSH_TUNABLE feature bits in the previous table) they correspond to. It also identifies the minimum kernel version required to use for each profile. Note however that Operating System vendors may choose to backport features to their kernels, so these kernel versions are valid for mainline kernels only. The kernel client included since SUSE Linux Enterprise Server 15 SP2, for example, includes backports of features and bug fixes relevant for usage in SUSE Enterprise Storage 7.1 clusters.

TABLE 12.2: CRUSH PROFILES

CRUSH Profile	Ceph Release	CRUSH Tunable	Minimum Kernel Version
argonaut	0.48	CRUSH_TUNABLES	3.6
bobtail	0.56	CRUSH_TUNABLES2	3.9
firefly	0.80	CRUSH_TUNABLES3	3.15
hammer	0.94	CRUSH_V4	4.1
jewel	10.2.0	CRUSH_TUNABLES5	4.5

13 Hints and tips

The chapter provides information to help you enhance performance of your Ceph cluster and provides tips how to set the cluster up.

13.1 Identifying orphaned volumes

To identify possibly orphaned journal/WAL/DB volumes, follow these steps:

1. Get a list of OSD IDs for which LVs exist, but not OSD daemon is running on the node.

   ```
   root@minion >  comm -3 <(ceph osd tree | grep up | awk '{print $4}') <(cephadm ceph-
   volume lvm list 2>/dev/null | grep ====== | awk '{print $2}')
   ```

2. If the command outputs one or more OSDs (for example osd.33), take the IDs (33 in the prior example) and zap the associated volumes.

   ```
   root@minion >  ceph-volume lvm zap --destroy --osd-id ID.
   ```

13.2 Adjusting scrubbing

By default, Ceph performs light scrubbing daily (find more details in *Book "Administration and Operations Guide", Chapter 17 "Stored data management", Section 17.6 "Scrubbing placement groups"*) and deep scrubbing weekly. *Light* scrubbing checks object sizes and checksums to ensure that placement groups are storing the same object data. *Deep* scrubbing checks an object's content with that of its replicas to ensure that the actual contents are the same. The price for checking data integrity is increased I/O load on the cluster during the scrubbing procedure.

The default settings allow Ceph OSDs to initiate scrubbing at inappropriate times, such as during periods of heavy loads. Customers may experience latency and poor performance when scrubbing operations conflict with their operations. Ceph provides several scrubbing settings that can limit scrubbing to periods with lower loads or during off-peak hours.

If the cluster experiences high loads during the day and low loads late at night, consider restricting scrubbing to night time hours, such as 11pm till 6am:

```
cephuser@adm > ceph config set osd osd_scrub_begin_hour 23
cephuser@adm > ceph config set osd osd_scrub_end_hour 6
```

If time restriction is not an effective method of determining a scrubbing schedule, consider using the `osd_scrub_load_threshold` option. The default value is 0.5, but it could be modified for low load conditions:

```
cephuser@adm > ceph config set osd osd_scrub_load_threshold 0.25
```

13.3 Stopping OSDs without rebalancing

You may need to stop OSDs for maintenance periodically. If you do not want CRUSH to automatically rebalance the cluster in order to avoid huge data transfers, set the cluster to `noout` first:

```
root@minion > ceph osd set noout
```

When the cluster is set to `noout`, you can begin stopping the OSDs within the failure domain that requires maintenance work. To identify the unique FSID of the cluster, run **`ceph fsid`**. To identify the Object Gateway daemon name, run

```
cephuser@adm > ceph orch ps ---hostname HOSTNAME
```

```
# systemctl stop ceph-FSID@DAEMON_NAME
```

Find more information about operating Ceph services and identifying their names in *Book "Administration and Operations Guide", Chapter 14 "Operation of Ceph services"*.

After you complete the maintenance, start OSDs again:

```
# systemctl start ceph-FSID@osd.SERVICE_ID.service
```

After OSD services are started, unset the cluster from `noout`:

```
cephuser@adm > ceph osd unset noout
```

13.4 Checking for unbalanced data writing

When data is written to OSDs evenly, the cluster is considered balanced. Each OSD within a cluster is assigned its *weight*. The weight is a relative number and tells Ceph how much of the data should be written to the related OSD. The higher the weight, the more data will be written. If an OSD has zero weight, no data will be written to it. If the weight of an OSD is relatively high compared to other OSDs, a large portion of the data will be written there, which makes the cluster unbalanced.

Unbalanced clusters have poor performance, and in the case that an OSD with a high weight suddenly crashes, a lot of data needs to be moved to other OSDs, which slows down the cluster as well.

To avoid this, you should regularly check OSDs for the amount of data writing. If the amount is between 30% and 50% of the capacity of a group of OSDs specified by a given ruleset, you need to reweight the OSDs. Check for individual disks and find out which of them fill up faster than the others (or are generally slower), and lower their weight. The same is valid for OSDs where not enough data is written—you can increase their weight to have Ceph write more data to them. In the following example, you will find out the weight of an OSD with ID 13, and reweight it from 3 to 3.05:

```
cephuser@adm > ceph osd tree | grep osd.13
 13  hdd 3                osd.13  up  1.00000  1.00000

 cephuser@adm > ceph osd crush reweight osd.13 3.05
 reweighted item id 13 name 'osd.13' to 3.05 in crush map

cephuser@adm > ceph osd tree | grep osd.13
 13  hdd 3.05                osd.13  up  1.00000  1.00000
```

Tip: OSD reweight by utilization

The **`ceph osd reweight-by-utilization`** *`threshold`* command automates the process of reducing the weight of OSDs which are heavily overused. By default it will adjust the weights downward on OSDs which reached 120% of the average usage, but if you include threshold it will use that percentage instead.

13.5 Increasing file descriptors

For OSD daemons, the read/write operations are critical to keep the Ceph cluster balanced. They often need to have many files open for reading and writing at the same time. On the OS level, the maximum number of simultaneously open files is called 'maximum number of file descriptors'.

To prevent OSDs from running out of file descriptors, you can override the default and specify an appropriate value, for example:

```
cephuser@adm > ceph config set global max_open_files 131072
```

After you change max_open_files, you need to restart the OSD service on the relevant Ceph node.

13.6 Integration with virtualization software

13.6.1 Storing KVM disks in Ceph cluster

You can create a disk image for a KVM-driven virtual machine, store it in a Ceph pool, optionally convert the content of an existing image to it, and then run the virtual machine with **qemu-kvm** making use of the disk image stored in the cluster. For more detailed information, see *Book "Administration and Operations Guide", Chapter 27 "Ceph as a back-end for QEMU KVM instance"*.

13.6.2 Storing libvirt disks in Ceph cluster

Similar to KVM (see *Section 13.6.1, "Storing KVM disks in Ceph cluster"*), you can use Ceph to store virtual machines driven by libvirt. The advantage is that you can run any libvirt-supported virtualization solution, such as KVM, Xen, or LXC. For more information, see *Book "Administration and Operations Guide", Chapter 26 "libvirt and Ceph"*.

13.6.3 Storing Xen disks in Ceph cluster

One way to use Ceph for storing Xen disks is to make use of libvirt as described in *Book "Administration and Operations Guide", Chapter 26 "libvirt and Ceph"*.

Another option is to make Xen talk to the rbd block device driver directly:

1. If you have no disk image prepared for Xen, create a new one:

   ```
   cephuser@adm > rbd create myimage --size 8000 --pool mypool
   ```

2. List images in the pool mypool and check if your new image is there:

   ```
   cephuser@adm > rbd list mypool
   ```

3. Create a new block device by mapping the myimage image to the rbd kernel module:

   ```
   cephuser@adm > rbd map --pool mypool myimage
   ```

Tip: User name and authentication

To specify a user name, use `--id user-name`. Moreover, if you use `cephx` authentication, you must also specify a secret. It may come from a keyring or a file containing the secret:

```
cephuser@adm > rbd map --pool rbd myimage --id admin --keyring /path/to/
keyring
```

or

```
cephuserrbd map --pool rbd myimage --id admin --keyfile /path/to/file
```

4. List all mapped devices:

```
cephuser@adm > rbd showmapped
 id pool   image   snap device
 0  mypool myimage -    /dev/rbd0
```

5. Now you can configure Xen to use this device as a disk for running a virtual machine. You can for example add the following line to the `xl`-style domain configuration file:

```
disk = [ '/dev/rbd0,,sda', '/dev/cdrom,,sdc,cdrom' ]
```

13.7 Firewall settings for Ceph

We recommend protecting the network cluster communication with SUSE Firewall. You can edit its configuration by selecting *YaST* › *Security and Users* › *Firewall* › *Allowed Services*.

Following is a list of Ceph-related services and numbers of the ports that they normally use:

Ceph Dashboard

The Ceph Dashboard binds to a specific TCP/IP address and TCP port. By default, the currently active Ceph Manager that hosts the dashboard binds to TCP port 8443 (or 8080 when SSL is disabled).

Ceph Monitor

Enable the *Ceph MON* service or ports 3300 and 6789 (TCP).

Ceph OSD or Metadata Server

Enable the *Ceph OSD/MDS* service or ports 6800-7300 (TCP).

This port range needs to be adjusted for dense nodes. See *Book “Deployment Guide”, Chapter 7 “Deploying the bootstrap cluster using* `ceph-salt`*”, Section 7.4 “Reviewing final steps”* for more information.

iSCSI Gateway

Open port 3260 (TCP).

Object Gateway

Open the port where Object Gateway communication occurs. To display it, run the following command:

```
cephuser@adm > ceph config get client.rgw.RGW_DAEMON_NAME rgw_frontends
```

Default is 80 for HTTP and 443 for HTTPS (TCP).

NFS Ganesha

By default, NFS Ganesha uses ports 2049 (NFS service, TCP) and 875 (rquota support, TCP).

SSH

Open port 22 (TCP).

NTP

Open port 123 (UDP).

Salt

Open ports 4505 and 4506 (TCP).

Grafana

Open port 3000 (TCP).

Prometheus

Open port 9095 (TCP).

13.8 Testing network performance

Both intermittent and complete network failures will impact a Ceph cluster. These two utilities can help in tracking down the cause and verifying expectations.

Tip: Sync Runners

Use the **`salt-run saltutil.sync_runners`** command if the Salt runner is reported as `not available`.

13.8.1 Performing basic diagnostics

Try the **`salt-run network.ping`** command to ping between cluster nodes to see if an individual interface can reach to a specific interface and the average response time. Any specific response time much slower than average will also be reported. For example:

```
root@master # salt-run network.ping
Succeeded: 8 addresses from 7 minions average rtt 0.15 ms
```

Or, for IPv6:

```
root@master # salt-run network.ping6
Succeeded: 8 addresses from 7 minions average rtt 0.15 ms
```

Try validating all interfaces with JumboFrame enabled:

```
root@master # salt-run network.jumbo_ping
Succeeded: 8 addresses from 7 minions average rtt 0.26 ms
```

13.8.2 Performing throughput benchmark

Try the **`salt-run network.iperf`** command to test network bandwidth between each pair of interfaces. On a given cluster node, a number of `iperf` processes (according to the number of CPU cores) are started as servers. The remaining cluster nodes will be used as clients to generate network traffic. The accumulated bandwidth of all per-node `iperf` processes is reported. This should reflect the maximum achievable network throughput on all cluster nodes. For example:

```
root@master # salt-run network.iperf
Fastest 2 hosts:
    |_
      - 192.168.31.25
      - 11443 Mbits/sec
    |_
      - 172.16.31.25
      - 10363 Mbits/sec

Slowest 2 hosts:
    |_
      - 172.16.32.14
      - 10217 Mbits/sec
    |_
      - 192.168.121.164
      - 10113 Mbits/sec
```

13.8.3 Useful options

The `output=full` option will list the results of each interface rather than the summary of the two slowest and fastest.

```
root@master # salt-run network.iperf output=full
192.168.128.1:
    8644.0 Mbits/sec
192.168.128.2:
    10360.0 Mbits/sec
192.168.128.3:
    9336.0 Mbits/sec
192.168.128.4:
    9588.56 Mbits/sec
192.168.128.5:
    10187.0 Mbits/sec
192.168.128.6:
    10465.0 Mbits/sec
```

The `remove=network` where network is a comma delimited list of subnets that should not be included.

```
root@master # salt-run network.ping remove="192.168.121.0/24,192.168.1.0/24"
Succeeded: 20 addresses from 10 minions average rtt 14.16 ms
```

13.9 Locating physical disks using LED lights

Ceph tracks which daemons manage which hardware storage devices (HDDs, SSDs), and collects health metrics about those devices in order to provide tools to predict and automatically respond to hardware failure.

You can blink the drive LEDs on hardware enclosures to make the replacement of failed disks easy and less error-prone. Use the following command:

```
cephuser@adm > ceph device light --enable=on --devid=string --light_type=ident --force
```

The `DEVID` parameter is the device identification. You can obtain it by running the following command:

```
cephuser@adm > ceph device ls
```

13.10 Sending large objects with **rados** fails with full OSD

`rados` is a command line utility to manage RADOS object storage. For more information, see **`man 8 rados`**.

If you send a large object to a Ceph cluster with the **`rados`** utility, such as

```
cephuser@adm > rados -p mypool put myobject /file/to/send
```

it can fill up all the related OSD space and cause serious trouble to the cluster performance.

13.11 Managing the 'Too Many PGs per OSD' status message

If you receive a `Too Many PGs per OSD` message after running **`ceph status`**, it means that the `mon_pg_warn_max_per_osd` value (300 by default) was exceeded. This value is compared to the number of PGs per OSD ratio. This means that the cluster setup is not optimal.

The number of PGs cannot be reduced after the pool is created. Pools that do not yet contain any data can safely be deleted and then re-created with a lower number of PGs. Where pools already contain data, the only solution is to add OSDs to the cluster so that the ratio of PGs per OSD becomes lower.

13.12 Managing the '*nn* pg stuck inactive' status message

If you receive a `stuck inactive` status message after running **`ceph status`**, it means that Ceph does not know where to replicate the stored data to fulfill the replication rules. It can happen shortly after the initial Ceph setup and fix itself automatically. In other cases, this may require a manual interaction, such as bringing up a broken OSD, or adding a new OSD to the cluster. In very rare cases, reducing the replication level may help.

If the placement groups are stuck perpetually, you need to check the output of **`ceph osd tree`**. The output should look tree-structured, similar to the example in *Section 4.6, "OSD is down"*.

If the output of **`ceph osd tree`** is rather flat as in the following example

```
cephuser@adm > ceph osd tree
ID  CLASS  WEIGHT   TYPE NAME              STATUS  REWEIGHT  PRI-AFF
-1         0.02939  root default
-3         0.00980      host doc-ses-node2
 0    hdd  0.00980          osd.0              up   1.00000  1.00000
-5         0.00980      host doc-ses-node3
 1    hdd  0.00980          osd.1              up   1.00000  1.00000
-7         0.00980      host doc-ses-node4
 2    hdd  0.00980          osd.2              up   1.00000  1.00000
```

You should check that the related CRUSH map has a tree structure. If it is also flat, or with no hosts as in the above example, it may mean that host name resolution is not working correctly across the cluster.

If the hierarchy is incorrect—for example the root contains hosts, but the OSDs are at the top level and are not themselves assigned to hosts—you will need to move the OSDs to the correct place in the hierarchy. This can be done using the **`ceph osd crush move`** and/or **`ceph osd crush set`** commands. For further details see *Book "Administration and Operations Guide", Chapter 17 "Stored data management", Section 17.5 "CRUSH Map manipulation"*.

13.13 Fixing clock skew warnings

As a general rule, time synchronization must be configured and running on all nodes. Once time synchronization is set up, the clocks should not get skewed. However, if a clock skew is to occur, this is likely due to the `chronyd.service` not running on one or more hosts.

Note

It is also possible that the battery on the motherboard has died, and the clock skew will be more pronounced. If this is the case, be aware that it will take quite some time for `chronyd` to re-synchronize the clocks.

If you receive a clock skew warning, confirm that the `chronyd.service` daemon is running on all cluster nodes. If not, restart the service and wait for chronyd to re-sync the clock.

Find more information on setting up time synchronization in https://documentation.suse.com/sles/15-SP3/html/SLES-all/cha-ntp.html#sec-ntp-yast ↗.

13.14 Determining poor cluster performance caused by network problems

There may be other reasons why cluster performance becomes weak, such as network problems. In such case, you may notice the cluster reaching quorum, OSD and monitor nodes going offline, data transfers taking a long time, or a lot of reconnect attempts.

To check whether cluster performance is degraded by network problems, inspect the Ceph log files under the `/var/log/ceph` directory.

To fix network issues on the cluster, focus on the following points:

- Basic network diagnostics. Try running the `net.ping` diagnostics tool. This tool has cluster nodes send network pings from their network interfaces to the network interfaces of other cluster nodes, and measures the average response time. Any specific response time much slower then average will also be reported. See *Section 13.8.1, "Performing basic diagnostics"* for more information.

- Check firewall settings on cluster nodes. Make sure they do not block ports or protocols required by Ceph operation. See *Section 13.7, "Firewall settings for Ceph"* for more information on firewall settings.

- Check the networking hardware, such as network cards, cables, or switches, for proper operation.

Tip: Separate network

To ensure fast and safe network communication between cluster nodes, set up a separate network used exclusively by the cluster OSD and monitor nodes.

13.15 Managing /var running out of space

By default, the Salt Master saves every minion's result for every job in its *job cache*. The cache can then be used later to look up results from previous jobs. The cache directory defaults to `/var/cache/salt/master/jobs/`.

Each job return from every minion is saved in a single file. Over time this directory can grow very large, depending on the number of published jobs and the value of the `keep_jobs` option in the `/etc/salt/master` file. `keep_jobs` sets the number of hours (24 by default) to keep information about past minion jobs.

```
keep_jobs: 24
```

Important: Do not set `keep_jobs: 0`

Setting `keep_jobs` to '0' will cause the job cache cleaner to *never* run, possibly resulting in a full partition.

If you want to disable the job cache, set `job_cache` to 'False':

```
job_cache: False
```

Tip: Restoring partition full because of job cache

When the partition with job cache files gets full because of wrong `keep_jobs` setting, follow these steps to free disk space and improve the job cache settings:

1. Stop the Salt Master service:

   ```
   root@master # systemctl stop salt-master
   ```

2. Change the Salt Master configuration related to job cache by editing `/etc/salt/master`:

   ```
   job_cache: False
   keep_jobs: 1
   ```

3. Clear the Salt Master job cache:

   ```
   # rm -rfv /var/cache/salt/master/jobs/*
   ```

4. Start the Salt Master service:

   ```
   root@master # systemctl start salt-master
   ```

14 Frequently asked questions

14.1 How does the number of placement groups affect the cluster performance?

When your cluster is becoming 70% to 80% full, it is time to add more OSDs to it. When you increase the number of OSDs, you may consider increasing the number of placement groups as well.

Warning

Changing the number of PGs causes a lot of data transfer within the cluster.

To calculate the optimal value for your newly-resized cluster is a complex task.

A high number of PGs creates small chunks of data. This speeds up recovery after an OSD failure, but puts a lot of load on the monitor nodes as they are responsible for calculating the data location.

On the other hand, a low number of PGs takes more time and data transfer to recover from an OSD failure, but does not impose that much load on monitor nodes as they need to calculate locations for less (but larger) data chunks.

Find more information on the optimal number of PGs for your cluster in *Book "Administration and Operations Guide", Chapter 17 "Stored data management", Section 17.4.2 "Determining the value of PG_NUM"*.

14.2 Can I use SSDs and hard disks on the same cluster?

Solid-state drives (SSD) are generally faster than hard disks. If you mix the two types of disks for the same write operation, the data writing to the SSD disk will be slowed down by the hard disk performance. Thus, you should *never mix SSDs and hard disks* for data writing following *the same rule* (see *Book "Administration and Operations Guide", Chapter 17 "Stored data management", Section 17.3 "Rule sets"* for more information on rules for storing data).

There are generally 2 cases where using SSD and hard disk on the same cluster makes sense:

1. Use each disk type for writing data following different rules. Then you need to have a separate rule for the SSD disk, and another rule for the hard disk.
2. Use each disk type for a specific purpose. For example the SSD disk for journal, and the hard disk for storing data.

14.3 What are the trade-offs of ssing a journal on SSD?

Using SSDs for OSD journal(s) is better for performance as the journal is usually the bottleneck of hard disk-only OSDs. SSDs are often used to share journals of several OSDs.

Following is a list of potential disadvantages of using SSDs for OSD journal:

- SSD disks are more expensive than hard disks. But as one OSD journal requires up to 6GB of disk space only, the price may not be so crucial.
- SSD disk consumes storage slots which can be otherwise used by a large hard disk to extend the cluster capacity.
- SSD disks have reduced write cycles compared to hard disks, but modern technologies are beginning to eliminate the problem.
- If you share more journals on the same SSD disk, you risk losing all the related OSDs after the SSD disk fails. This will require a lot of data to be moved to rebalance the cluster.
- Hotplugging disks becomes more complex as the data mapping is not 1:1 the failed OSD and the journal disk.

14.4 What happens when a disk fails?

When a disk with a stored cluster data has a hardware problem and fails to operate, here is what happens:

- The related OSD crashed and is automatically removed from the cluster.
- The failed disk's data is replicated to another OSD in the cluster from other copies of the same data stored in other OSDs.
- Then you should remove the disk from the cluster CRUSH Map, and physically from the host hardware.

14.5 What happens when a journal disk fails?

Ceph can be configured to store journals or write ahead logs on devices separate from the OSDs. When a disk dedicated to a journal fails, the related OSD(s) fail as well (see *Section 14.4, “What happens when a disk fails?”*).

Warning: Hosting multiple journals on one disk

For performance boost, you can use a fast disk (such as SSD) to store journal partitions for several OSDs. We do not recommend to host journals for more than 4 OSDs on one disk, because in case of the journals' disk failure, you risk losing stored data for all the related OSDs' disks.

A Ceph maintenance updates based on upstream 'Pacific' point releases

Several key packages in SUSE Enterprise Storage 7.1 are based on the Pacific release series of Ceph. When the Ceph project (https://github.com/ceph/ceph ↗) publishes new point releases in the Pacific series, SUSE Enterprise Storage 7.1 is updated to ensure that the product benefits from the latest upstream bug fixes and feature backports.

This chapter contains summaries of notable changes contained in each upstream point release that has been—or is planned to be—included in the product.

Glossary

General

Admin node

The host from which you run the Ceph-related commands to administer cluster hosts.

Alertmanager

A single binary which handles alerts sent by the Prometheus server and notifies the end user.

archive sync module

Module that enables creating an Object Gateway zone for keeping the history of S3 object versions.

Bucket

A point that aggregates other nodes into a hierarchy of physical locations.

Ceph Client

The collection of Ceph components which can access a Ceph Storage Cluster. These include the Object Gateway, the Ceph Block Device, the CephFS, and their corresponding libraries, kernel modules, and FUSE clients.

Ceph Dashboard

A built-in Web-based Ceph management and monitoring application to administer various aspects and objects of the cluster. The dashboard is implemented as a Ceph Manager module.

Ceph Manager

Ceph Manager or MGR is the Ceph manager software, which collects all the state from the whole cluster in one place.

Ceph Monitor

Ceph Monitor or MON is the Ceph monitor software.

Ceph Object Storage

The object storage "product", service or capabilities, which consists of a Ceph Storage Cluster and a Ceph Object Gateway.

Ceph OSD Daemon

The **`ceph-osd`** daemon is the component of Ceph that is responsible for storing objects on a local file system and providing access to them over the network.

Ceph Storage Cluster

The core set of storage software which stores the user's data. Such a set consists of Ceph monitors and OSDs.

`ceph-salt`

Provides tooling for deploying Ceph clusters managed by cephadm using Salt.

cephadm

cephadm deploys and manages a Ceph cluster by connecting to hosts from the manager daemon via SSH to add, remove, or update Ceph daemon containers.

CephFS

The Ceph file system.

CephX

The Ceph authentication protocol. Cephx operates like Kerberos, but it has no single point of failure.

CRUSH rule

The CRUSH data placement rule that applies to a particular pool or pools.

CRUSH, CRUSH Map

Controlled Replication Under Scalable Hashing: An algorithm that determines how to store and retrieve data by computing data storage locations. CRUSH requires a map of the cluster to pseudo-randomly store and retrieve data in OSDs with a uniform distribution of data across the cluster.

DriveGroups

DriveGroups are a declaration of one or more OSD layouts that can be mapped to physical drives. An OSD layout defines how Ceph physically allocates OSD storage on the media matching the specified criteria.

Grafana

Database analytics and monitoring solution.

Metadata Server

Metadata Server or MDS is the Ceph metadata software.

Multi-zone

Node

Any single machine or server in a Ceph cluster.

Object Gateway

The S3/Swift gateway component for Ceph Object Store. Also known as the RADOS Gateway (RGW).

OSD

Object Storage Device: A physical or logical storage unit.

OSD node

A cluster node that stores data, handles data replication, recovery, backfilling, rebalancing, and provides some monitoring information to Ceph monitors by checking other Ceph OSD daemons.

PG

Placement Group: a sub-division of a *pool*, used for performance tuning.

Point Release

Any ad-hoc release that includes only bug or security fixes.

Pool

Logical partitions for storing objects such as disk images.

Prometheus

Systems monitoring and alerting toolkit.

RADOS Block Device (RBD)

The block storage component of Ceph. Also known as the Ceph block device.

Reliable Autonomic Distributed Object Store (RADOS)

The core set of storage software which stores the user's data (MON+OSD).

Routing tree

A term given to any diagram that shows the various routes a receiver can run.

Rule Set

Rules to determine data placement for a pool.

Samba

Windows integration software.

Samba Gateway

The Samba Gateway joins the Active Directory in the Windows domain to authenticate and authorize users.

zonegroup