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 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 Subsystem, Log and Debug Settings for details on available settings.

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:

```
ceph daemon {daemon-name} config show | less
```

For example,:

```
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:

```
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:

```
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 still make the change by logging into the host of the daemon whose configuration you'd like to change using `ceph daemon`. For example:

```
sudo ceph daemon osd.0 config set debug_osd 0/5
```

See Subsystem, Log and Debug Settings for details on available settings.

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 Subsystem, Log and Debug Settings for details.

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.

VALGRIND

Debugging may also require you to track down memory and threading issues. You can run a single daemon, a type of daemon, or the whole cluster with Valgrind. You should only use Valgrind when developing or debugging Ceph. Valgrind is computationally expensive, and will slow down your system otherwise. Valgrind messages are logged to stderr.

SUBSYSTEM, LOG AND DEBUG SETTINGS

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

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 requested. Please consult document on admin socket for more details.

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.

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
`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
`objecter`	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
`objclass`	0	5
`filestore`	1	3
`journal`	1	3
`ms`	0	5
`mon`	1	5
`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
memdb	4	5
fuse	1	5
mgr	1	5
mgrc	1	5
dpdk	1	5
eventtrace	1	5

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 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`

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 tmapput sets uses tmap`

Description:	Uses `tmap`. For debug only.
Type:	Boolean
Required:	No
Default:	`false`

`osd min pg log entries`

Description:	The minimum number of log entries for placement groups.
Type:	32-bit Unsigned Integer
Required:	No
Default:	1000

`osd op log threshold`

Description:	How many op log messages to show up in one pass.
Type:	Integer
Required:	No
Default:	5

FILESTORE

`filestore debug omap check`

Description:	Debugging check on synchronization. This is an expensive operation.
Type:	Boolean
Required:	No
Default:	`false`

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`

RADOS GATEWAY

`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:	`true`

`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`

[1] there are levels >20 in some rare cases and that they are extremely verbose.