

CONTROL COMMANDS

MONITOR COMMANDS

Monitor commands are issued using the ceph utility:

```
ceph [-m monhost] {command}
```

The command is usually (though not always) of the form:

```
ceph {subsystem} {command}
```

SYSTEM COMMANDS

Execute the following to display the current status of the cluster.

```
ceph -s  
ceph status
```

Execute the following to display a running summary of the status of the cluster, and major events.

```
ceph -w
```

Execute the following to show the monitor quorum, including which monitors are participating and which one is the leader.

```
ceph quorum_status
```

Execute the following to query the status of a single monitor, including whether or not it is in the quorum.

```
ceph [-m monhost] mon_status
```

AUTHENTICATION SUBSYSTEM

To add a keyring for an OSD, execute the following:

```
ceph auth add {osd} [--in-file|-i] {path-to-osd-keyring}
```

To list the cluster's keys and their capabilities, execute the following:

```
ceph auth ls
```

PLACEMENT GROUP SUBSYSTEM

To display the statistics for all placement groups, execute the following:

```
ceph pg dump [--format {format}]
```

The valid formats are plain (default) and json.

To display the statistics for all placement groups stuck in a specified state, execute the following:

```
ceph pg dump_stuck inactive|unclean|stale|undersized|degraded [--format {format}] [-t|--thres
```

--format may be plain (default) or json

--threshold defines how many seconds “stuck” is (default: 300)

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.

Unclean Placement groups contain objects that are not replicated the desired number of times. They should be recovering.

Stale Placement groups are in an unknown state - the OSDs that host them have not reported to the monitor cluster in a while (configured by mon_osd_report_timeout).

Delete “lost” objects or revert them to their prior state, either a previous version or delete them if they were just created.

```
ceph pg {pgid} mark_unfound_lost revert|delete
```

OSD SUBSYSTEM

Query OSD subsystem status.

```
ceph osd stat
```

Write a copy of the most recent OSD map to a file. See [osdmactool](#).

```
ceph osd getmap -o file
```

Write a copy of the crush map from the most recent OSD map to file.

```
ceph osd getcrushmap -o file
```

The foregoing functionally equivalent to

```
ceph osd getmap -o /tmp/osdmap  
osdmactool /tmp/osdmap --export-crush file
```

Dump the OSD map. Valid formats for -f are plain and json. If no --format option is given, the OSD map is dumped as plain text.

```
ceph osd dump [--format {format}]
```

Dump the OSD map as a tree with one line per OSD containing weight and state.

```
ceph osd tree [--format {format}]
```

Find out where a specific object is or would be stored in the system:

```
ceph osd map <pool-name> <object-name>
```

Add or move a new item (OSD) with the given id/name/weight at the specified location.

```
ceph osd crush set {id} {weight} [{loc1} [{loc2} ...]]
```

Remove an existing item (OSD) from the CRUSH map.

```
ceph osd crush remove {name}
```

Remove an existing bucket from the CRUSH map.

```
ceph osd crush remove {bucket-name}
```

Move an existing bucket from one position in the hierarchy to another.

```
ceph osd crush move {id} {loc1} [{loc2} ...]
```

Set the weight of the item given by {name} to {weight}.

```
ceph osd crush reweight {name} {weight}
```

Mark an OSD as lost. This may result in permanent data loss. Use with caution.

```
ceph osd lost {id} [--yes-i-really-mean-it]
```

Create a new OSD. If no UUID is given, it will be set automatically when the OSD starts up.

```
ceph osd create [{uuid}]
```

Remove the given OSD(s).

```
ceph osd rm [{id}...]
```

Query the current max_osd parameter in the OSD map.

```
ceph osd getmaxosd
```

Import the given crush map.

```
ceph osd setcrushmap -i file
```

Set the max_osd parameter in the OSD map. This is necessary when expanding the storage cluster.

```
ceph osd setmaxosd
```

Mark OSD {osd-num} down.

```
ceph osd down {osd-num}
```

Mark OSD {osd-num} out of the distribution (i.e. allocated no data).

```
ceph osd out {osd-num}
```

Mark {osd-num} in the distribution (i.e. allocated data).

```
ceph osd in {osd-num}
```

Set or clear the pause flags in the OSD map. If set, no IO requests will be sent to any OSD. Clearing the flags via unpause results in resending pending requests.

```
ceph osd pause  
ceph osd unpause
```

Set the weight of {osd-num} to {weight}. Two OSDs with the same weight will receive roughly the same number of I/O requests and store approximately the same amount of data. `ceph osd reweight` sets an override weight on the OSD. This value is in the range 0 to 1, and forces CRUSH to re-place (1-weight) of the data that would otherwise live on this drive. It does not change the weights assigned to the buckets above the OSD in the crush map, and is a corrective measure in case the normal CRUSH distribution is not working out quite right. For instance, if one of your OSDs is at 90% and the others are at 50%, you could reduce this weight to try and compensate for it.

```
ceph osd reweight {osd-num} {weight}
```

Reweights all the OSDs by reducing the weight of OSDs which are heavily overused. By default it will adjust the weights downward on OSDs which have 120% of the average utilization, but if you include threshold it will use that percentage instead.

```
ceph osd reweight-by-utilization [threshold]
```

Describes what reweight-by-utilization would do.

```
ceph osd test-reweight-by-utilization
```

Adds/removes the address to/from the blacklist. When adding an address, you can specify how long it should be blacklisted in seconds; otherwise, it will default to 1 hour. A blacklisted address is prevented from connecting to any OSD. Blacklisting is most often used to prevent a lagging metadata server from making bad changes to data on the OSDs.

These commands are mostly only useful for failure testing, as blacklists are normally maintained automatically and shouldn't need manual intervention.

```
ceph osd blacklist add ADDRESS[:source_port] [TIME]  
ceph osd blacklist rm ADDRESS[:source_port]
```

Creates/deletes a snapshot of a pool.

```
ceph osd pool mksnap {pool-name} {snap-name}  
ceph osd pool rmsnap {pool-name} {snap-name}
```

Creates/deletes/renames a storage pool.

```
ceph osd pool create {pool-name} pg_num [pgp_num]  
ceph osd pool delete {pool-name} [{pool-name} --yes-i-really-really-mean-it]  
ceph osd pool rename {old-name} {new-name}
```

Changes a pool setting.

```
ceph osd pool set {pool-name} {field} {value}
```

Valid fields are:

- **size**: Sets the number of copies of data in the pool.

- `pg_num`: The placement group number.
- `pgp_num`: Effective number when calculating pg placement.
- `crush_rule`: rule number for mapping placement.

Get the value of a pool setting.

```
ceph osd pool get {pool-name} {field}
```

Valid fields are:

- `pg_num`: The placement group number.
- `pgp_num`: Effective number of placement groups when calculating placement.
- `lpg_num`: The number of local placement groups.
- `lpgp_num`: The number used for placing the local placement groups.

Sends a scrub command to OSD {osd-num}. To send the command to all OSDs, use `*`.

```
ceph osd scrub {osd-num}
```

Sends a repair command to OSD.N. To send the command to all OSDs, use `*`.

```
ceph osd repair N
```

Runs a simple throughput benchmark against OSD.N, writing `TOTAL_DATA_BYTES` in write requests of `BYTES_PER_WRITE` each. By default, the test writes 1 GB in total in 4-MB increments. The benchmark is non-destructive and will not overwrite existing live OSD data, but might temporarily affect the performance of clients concurrently accessing the OSD.

```
ceph tell osd.N bench [TOTAL_DATA_BYTES] [BYTES_PER_WRITE]
```

MDS SUBSYSTEM

Change configuration parameters on a running mds.

```
ceph tell mds.{mds-id} config set {setting} {value}
```

Example:

```
ceph tell mds.0 config set debug_ms 1
```

Enables debug messages.

```
ceph mds stat
```

Displays the status of all metadata servers.

```
ceph mds fail 0
```

Marks the active MDS as failed, triggering failover to a standby if present.

Todo: `ceph mds` subcommands missing docs: `set`, `dump`, `getmap`, `stop`, `setmap`

MON SUBSYSTEM

Show monitor stats:

```
ceph mon stat
```

```
e2: 3 mons at {a=127.0.0.1:40000/0,b=127.0.0.1:40001/0,c=127.0.0.1:40002/0}, election epoch 6
```

The quorum list at the end lists monitor nodes that are part of the current quorum.

This is also available more directly:

```
ceph quorum_status -f json-pretty
```

```
{
  "election_epoch": 6,
  "quorum": [
    0,
    1,
    2
  ],
  "quorum_names": [
    "a",
    "b",
    "c"
  ],
  "quorum_leader_name": "a",
  "monmap": {
    "epoch": 2,
    "fsid": "ba807e74-b64f-4b72-b43f-597dfe60ddbc",
    "modified": "2016-12-26 14:42:09.288066",
    "created": "2016-12-26 14:42:03.573585",
    "features": {
      "persistent": [
        "kraken"
      ],
      "optional": []
    },
    "mons": [
      {
        "rank": 0,
        "name": "a",
        "addr": "127.0.0.1:40000\0",
        "public_addr": "127.0.0.1:40000\0"
      },
      {
        "rank": 1,
        "name": "b",
        "addr": "127.0.0.1:40001\0",
        "public_addr": "127.0.0.1:40001\0"
      },
      {
        "rank": 2,
        "name": "c",
        "addr": "127.0.0.1:40002\0",
        "public_addr": "127.0.0.1:40002\0"
      }
    ]
  }
}
```

The above will block until a quorum is reached.

For a status of just the monitor you connect to (use -m HOST:PORT to select):

```
ceph mon_status -f json-pretty
```

```
{
  "name": "b",
  "rank": 1,
  "state": "peon",
```

```

"election_epoch": 6,
"quorum": [
  0,
  1,
  2
],
"features": {
  "required_con": "9025616074522624",
  "required_mon": [
    "kraken"
  ],
  "quorum_con": "1152921504336314367",
  "quorum_mon": [
    "kraken"
  ]
},
"outside_quorum": [],
"extra_probe_peers": [],
"sync_provider": [],
"monmap": {
  "epoch": 2,
  "fsid": "ba807e74-b64f-4b72-b43f-597dfe60ddbc",
  "modified": "2016-12-26 14:42:09.288066",
  "created": "2016-12-26 14:42:03.573585",
  "features": {
    "persistent": [
      "kraken"
    ],
    "optional": []
  },
  "mons": [
    {
      "rank": 0,
      "name": "a",
      "addr": "127.0.0.1:40000\0",
      "public_addr": "127.0.0.1:40000\0"
    },
    {
      "rank": 1,
      "name": "b",
      "addr": "127.0.0.1:40001\0",
      "public_addr": "127.0.0.1:40001\0"
    },
    {
      "rank": 2,
      "name": "c",
      "addr": "127.0.0.1:40002\0",
      "public_addr": "127.0.0.1:40002\0"
    }
  ]
}
}

```

A dump of the monitor state:

```

ceph mon dump

dumped monmap epoch 2
epoch 2
fsid ba807e74-b64f-4b72-b43f-597dfe60ddbc
last_changed 2016-12-26 14:42:09.288066
created 2016-12-26 14:42:03.573585
0: 127.0.0.1:40000/0 mon.a
1: 127.0.0.1:40001/0 mon.b
2: 127.0.0.1:40002/0 mon.c

```