CEPH – CEPH ADMINISTRATION TOOL

SYNOPSIS

ceph auth [*add* | *caps* | *del* | *export* | *get* | *get-key* | *get-or-create* | *get-or-create-key* | *import* | *list* | *print-key* | *print_key*] ...

ceph compact

ceph config-key [*del* | *exists* | *get* | *list* | *dump* | *put*] ...

ceph daemon <*name*> | <*path*> <*command*> ...

ceph daemonperf <*name*> | <*path*> [*interval* [*count*]]

ceph df *{detail}*

ceph fs [*ls* | *new* | *reset* | *rm*] ...

ceph fsid

ceph health *{detail}*

ceph heap [*dump* | *start_profiler* | *stop_profiler* | *release* | *stats*] ...

ceph injectargs <*injectedargs*> [<*injectedargs*>...]

ceph log <*logtext*> [<*logtext*>...]

ceph mds [*compat* | *deactivate* | *fail* | *rm* | *rmfailed* | *set_state* | *stat* | *repaired*] ...

ceph mon [*add* | *dump* | *getmap* | *remove* | *stat*] ...

ceph mon_status

ceph osd [*blacklist* | *blocked-by* | *create* | *new* | *deep-scrub* | *df* | *down* | *dump* | *erasure-code-profile* | *find* | *getcrushmap* | *getmap* | *getmaxosd* | *in* | *ls* | *lspools* | *map* | *metadata* | *ok-to-stop* | *out* | *pause* | *perf* | *pg-temp* | *force-create-pg* | *primary-affinity* | *primary-temp* | *repair* | *reweight* | *reweight-by-pg* | *rm* | *destroy* | *purge* | *safe-to-destroy* | *scrub* | *set* | *setcrushmap* | *setmaxosd* | *stat* | *tree* | *unpause* | *unset*] ...

ceph osd crush [*add* | *add-bucket* | *create-or-move* | *dump* | *get-tunable* | *link* | *move* | *remove* | *rename-bucket* | *reweight* | *reweight-all* | *reweight-subtree* | *rm* | *rule* | *set* | *set-tunable* | *show-tunables* | *tunables* | *unlink*] ...

ceph osd pool [*create* | *delete* | *get* | *get-quota* | *ls* | *mksnap* | *rename* | *rmsnap* | *set* | *set-quota* | *stats*] ...

ceph osd tier [*add* | *add-cache* | *cache-mode* | *remove* | *remove-overlay* | *set-overlay*] ...

ceph pg [*debug* | *deep-scrub* | *dump* | *dump_json* | *dump_pools_json* | *dump_stuck* | *getmap* | *ls* | *ls-by-osd* | *ls-by-pool* | *ls-by-primary* | *map* | *repair* | *scrub* | *stat*] ...

ceph quorum [*enter* | *exit*]

ceph quorum_status

ceph report { <*tags*> [<*tags*>...] }

ceph scrub

ceph status

ceph sync force {–yes-i-really-mean-it} {–i-know-what-i-am-doing}

ceph tell <*name (type.id)*> <*command*> *[options...]*

ceph version

DESCRIPTION

ceph is a control utility which is used for manual deployment and maintenance of a Ceph cluster. It provides a diverse set of commands that allows deployment of monitors, OSDs, placement groups, MDS and overall maintenance, administration of the cluster.

COMMANDS

AUTH

Manage authentication keys. It is used for adding, removing, exporting or updating of authentication keys for a particular entity such as a monitor or OSD. It uses some additional subcommands.

Subcommand add adds authentication info for a particular entity from input file, or random key if no input is given and/or any caps specified in the command.

Usage:

```
ceph auth add <entity> {<caps> [<caps>...]}
```

Subcommand caps updates caps for **name** from caps specified in the command.

Usage:

```
ceph auth caps <entity> <caps> [<caps>...]
```

Subcommand del deletes all caps for name.

Usage:

```
ceph auth del <entity>
```

Subcommand export writes keyring for requested entity, or master keyring if none given.

Usage:

```
ceph auth export {<entity>}
```

Subcommand get writes keyring file with requested key.

Usage:

```
ceph auth get <entity>
```

Subcommand get-key displays requested key.

Usage:

```
ceph auth get-key <entity>
```

Subcommand get-or-create adds authentication info for a particular entity from input file, or random key if no input given and/or any caps specified in the command.

Usage:

```
ceph auth get-or-create <entity> {<caps> [<caps>...]}
```

Subcommand get-or-create-key gets or adds key for name from system/caps pairs specified in the command. If key already exists, any given caps must match the existing caps for that key.

Usage:

```
ceph auth get-or-create-key <entity> {<caps> [<caps>...]}
```

Subcommand `import` reads keyring from input file.

Usage:

```
ceph auth import
```

Subcommand `ls` lists authentication state.

Usage:

```
ceph auth ls
```

Subcommand `print-key` displays requested key.

Usage:

```
ceph auth print-key <entity>
```

Subcommand `print_key` displays requested key.

Usage:

```
ceph auth print_key <entity>
```

COMPACT

Causes compaction of monitor's leveldb storage.

Usage:

```
ceph compact
```

CONFIG-KEY

Manage configuration key. It uses some additional subcommands.

Subcommand `del` deletes configuration key.

Usage:

```
ceph config-key del <key>
```

Subcommand `exists` checks for configuration keys existence.

Usage:

```
ceph config-key exists <key>
```

Subcommand `get` gets the configuration key.

Usage:

```
ceph config-key get <key>
```

Subcommand `list` lists configuration keys.

Usage:

```
ceph config-key ls
```

Subcommand dump dumps configuration keys and values.

Usage:

```
ceph config-key dump
```

Subcommand set puts configuration key and value.

Usage:

```
ceph config-key set <key> {<val>}
```

DAEMON

Submit admin-socket commands.

Usage:

```
ceph daemon {daemon_name|socket_path} {command} ...
```

Example:

```
ceph daemon osd.0 help
```

DAEMONPERF

Watch performance counters from a Ceph daemon.

Usage:

```
ceph daemonperf {daemon_name|socket_path} [{interval} [{count}]]
```

DF

Show cluster’s free space status.

Usage:

```
ceph df {detail}
```

FEATURES

Show the releases and features of all connected daemons and clients connected to the cluster, along with the numbers of them in each bucket grouped by the corresponding features/releases. Each release of Ceph supports a different set of features, expressed by the features bitmask. New cluster features require that clients support the feature, or else they are not allowed to connect to these new features. As new features or capabilities are enabled after an upgrade, older clients are prevented from connecting.

Usage:

```
ceph features
```

FS

Manage cephfs filesystems. It uses some additional subcommands.

Subcommand `ls` to list filesystems

Usage:

```
ceph fs ls
```

Subcommand new to make a new filesystem using named pools <metadata> and <data>

Usage:

```
ceph fs new <fs_name> <metadata> <data>
```

Subcommand `reset` is used for disaster recovery only: reset to a single-MDS map

Usage:

```
ceph fs reset <fs_name> {--yes-i-really-mean-it}
```

Subcommand `rm` to disable the named filesystem

Usage:

```
ceph fs rm <fs_name> {--yes-i-really-mean-it}
```

FSID

Show cluster's FSID/UUID.

Usage:

```
ceph fsid
```

HEALTH

Show cluster's health.

Usage:

```
ceph health {detail}
```

HEAP

Show heap usage info (available only if compiled with tcmalloc)

Usage:

```
ceph heap dump|start_profiler|stop_profiler|release|stats
```

INJECTARGS

Inject configuration arguments into monitor.

Usage:

```
ceph injectargs <injected_args> [<injected_args>...]
```

LOG

Log supplied text to the monitor log.

Usage:

```
ceph log <logtext> [<logtext>...]
```

MDS

Manage metadata server configuration and administration. It uses some additional subcommands.

Subcommand `compat` manages compatible features. It uses some additional subcommands.

Subcommand `rm_compat` removes compatible feature.

Usage:

```
ceph mds compat rm_compat <int[0-]>
```

Subcommand `rm_incompat` removes incompatible feature.

Usage:

```
ceph mds compat rm_incompat <int[0-]>
```

Subcommand `show` shows mds compatibility settings.

Usage:

```
ceph mds compat show
```

Subcommand `deactivate` stops mds.

Usage:

```
ceph mds deactivate <role>
```

Subcommand `fail` forces mds to status fail.

Usage:

```
ceph mds fail <role|gid>
```

Subcommand `rm` removes inactive mds.

Usage:

```
ceph mds rm <int[0-]> <name> (type.id)>
```

Subcommand `rmfailed` removes failed mds.

Usage:

```
ceph mds rmfailed <int[0-]>
```

Subcommand `set_state` sets mds state of <gid> to <numeric-state>.

Usage:

```
ceph mds set_state <int[0-]> <int[0-20]>
```

Subcommand `stat` shows MDS status.

Usage:

```
ceph mds stat
```

Subcommand `repaired` mark a damaged MDS rank as no longer damaged.

Usage:

```
ceph mds repaired <role>
```

MON

Manage monitor configuration and administration. It uses some additional subcommands.

Subcommand add adds new monitor named <name> at <addr>.

Usage:

```
ceph mon add <name> <IPaddr[:port]>
```

Subcommand `dump` dumps formatted monmap (optionally from epoch)

Usage:

```
ceph mon dump {<int[0-]>}
```

Subcommand `getmap` gets monmap.

Usage:

```
ceph mon getmap {<int[0-]>}
```

Subcommand `remove` removes monitor named <name>.

Usage:

```
ceph mon remove <name>
```

Subcommand `stat` summarizes monitor status.

Usage:

```
ceph mon stat
```

MON_STATUS

Reports status of monitors.

Usage:

```
ceph mon_status
```

MGR

Ceph manager daemon configuration and management.

Subcommand `dump` dumps the latest MgrMap, which describes the active and standby manager daemons.

Usage:

```
ceph mgr dump
```

Subcommand `fail` will mark a manager daemon as failed, removing it from the manager map. If it is the active manager daemon a standby will take its place.

Usage:

```
ceph mgr fail <name>
```

Subcommand `module ls` will list currently enabled manager modules (plugins).

Usage:

```
ceph mgr module ls
```

Subcommand `module enable` will enable a manager module. Available modules are included in MgrMap and visible via `mgr dump`.

Usage:

```
ceph mgr module enable <module>
```

Subcommand `module disable` will disable an active manager module.

Usage:

```
ceph mgr module disable <module>
```

Subcommand `metadata` will report metadata about all manager daemons or, if the name is specified, a single manager daemon.

Usage:

```
ceph mgr metadata [name]
```

Subcommand `versions` will report a count of running daemon versions.

Usage:

```
ceph mgr versions
```

Subcommand `count-metadata` will report a count of any daemon metadata field.

Usage:

```
ceph mgr count-metadata <field>
```

OSD

Manage OSD configuration and administration. It uses some additional subcommands.

Subcommand `blacklist` manage blacklisted clients. It uses some additional subcommands.

Subcommand add add <addr> to blacklist (optionally until <expire> seconds from now)

Usage:

```
ceph osd blacklist add <EntityAddr> {<float[0.0-]>}
```

Subcommand `ls` show blacklisted clients

Usage:

```
ceph osd blacklist ls
```

Subcommand `rm` remove <addr> from blacklist

Usage:

```
ceph osd blacklist rm <EntityAddr>
```

Subcommand `blocked-by` prints a histogram of which OSDs are blocking their peers

Usage:

```
ceph osd blocked-by
```

Subcommand `create` creates new osd (with optional UUID and ID).

This command is DEPRECATED as of the Luminous release, and will be removed in a future release.

Subcommand `new` should instead be used.

Usage:

```
ceph osd create {<uuid>} {<id>}
```

Subcommand `new` can be used to create a new OSD or to recreate a previously destroyed OSD with a specific *id*. The new OSD will have the specified *uuid*, and the command expects a JSON file containing the base64 cephx key for auth entity *client.osd.<id>*, as well as optional base64 cepx key for dm-crypt lockbox access and a dm-crypt key. Specifying a dm-crypt requires specifying the accompanying lockbox cephx key.

Usage:

```
ceph osd new {<uuid>} {<id>} -i {<params.json>}
```

The parameters JSON file is optional but if provided, is expected to maintain a form of the following format:

```
{
    "cephx_secret": "AQBWtwhZdBO5ExAAIDyjK2Bh16ZXylmzgYYEjg==",
    "crush_device_class": "myclass"
}
```

Or:

```
{
    "cephx_secret": "AQBWtwhZdBO5ExAAIDyjK2Bh16ZXylmzgYYEjg==",
    "cephx_lockbox_secret": "AQDNCglZuaeVCRAAYr76PzR1Anh7A0jswkODIQ==",
    "dmcrypt_key": "<dm-crypt key>",
    "crush_device_class": "myclass"
}
```

Or:

```
{
    "crush_device_class": "myclass"
}
```

The “crush_device_class” property is optional. If specified, it will set the initial CRUSH device class for the new OSD.

Subcommand `crush` is used for CRUSH management. It uses some additional subcommands.

Subcommand add adds or updates crushmap position and weight for <name> with <weight> and location <args>.

Usage:

```
ceph osd crush add <osdname (id|osd.id)> <float[0.0-]> <args> [<args>...]
```

Subcommand `add-bucket` adds no-parent (probably root) crush bucket <name> of type <type>.

Usage:

```
ceph osd crush add-bucket <name> <type>
```

Subcommand `create-or-move` creates entry or moves existing entry for <name> <weight> at/to location <args>.

Usage:

```
ceph osd crush create-or-move <osdname (id|osd.id)> <float[0.0-]> <args>
[<args>...]
```

Subcommand `dump` dumps crush map.

Usage:

```
ceph osd crush dump
```

Subcommand `get-tunable` get crush tunable straw_calc_version

Usage:

```
ceph osd crush get-tunable straw_calc_version
```

Subcommand `link` links existing entry for <name> under location <args>.

Usage:

```
ceph osd crush link <name> <args> [<args>...]
```

Subcommand `move` moves existing entry for <name> to location <args>.

Usage:

```
ceph osd crush move <name> <args> [<args>...]
```

Subcommand `remove` removes <name> from crush map (everywhere, or just at <ancestor>).

Usage:

```
ceph osd crush remove <name> {<ancestor>}
```

Subcommand rename-bucket renames buchket <srcname> to <stname>

Usage:

```
ceph osd crush rename-bucket <srcname> <dstname>
```

Subcommand reweight change <name>'s weight to <weight> in crush map.

Usage:

```
ceph osd crush reweight <name> <float[0.0-]>
```

Subcommand reweight-all recalculate the weights for the tree to ensure they sum correctly

Usage:

```
ceph osd crush reweight-all
```

Subcommand reweight-subtree changes all leaf items beneath <name> to <weight> in crush map

Usage:

```
ceph osd crush reweight-subtree <name> <weight>
```

Subcommand rm removes <name> from crush map (everywhere, or just at <ancestor>).

Usage:

```
ceph osd crush rm <name> {<ancestor>}
```

Subcommand rule is used for creating crush rules. It uses some additional subcommands.

Subcommand create-erasure creates crush rule <name> for erasure coded pool created with <profile> (default default).

Usage:

```
ceph osd crush rule create-erasure <name> {<profile>}
```

Subcommand create-simple creates crush rule <name> to start from <root>, replicate across buckets of type <type>, using a choose mode of <firstn|indep> (default firstn; indep best for erasure pools).

Usage:

```
ceph osd crush rule create-simple <name> <root> <type> {firstn|indep}
```

Subcommand dump dumps crush rule <name> (default all).

Usage:

```
ceph osd crush rule dump {<name>}
```

Subcommand ls lists crush rules.

Usage:

```
ceph osd crush rule ls
```

Subcommand rm removes crush rule <name>.

Usage:

```
ceph osd crush rule rm <name>
```

Subcommand set used alone, sets crush map from input file.

Usage:

```
ceph osd crush set
```

Subcommand set with osdname/osd.id update crushmap position and weight for <name> to <weight> with location <args>.

Usage:

```
ceph osd crush set <osdname (id|osd.id)> <float[0.0-]> <args> [<args>...]
```

Subcommand set-tunable set crush tunable <tunable> to <value>. The only tunable that can be set is straw_calc_version.

Usage:

```
ceph osd crush set-tunable straw_calc_version <value>
```

Subcommand show-tunables shows current crush tunables.

Usage:

```
ceph osd crush show-tunables
```

Subcommand tree shows the crush buckets and items in a tree view.

Usage:

```
ceph osd crush tree
```

Subcommand tunables sets crush tunables values to <profile>.

Usage:

```
ceph osd crush tunables legacy|argonaut|bobtail|firefly|hammer|optimal|default
```

Subcommand unlink unlinks <name> from crush map (everywhere, or just at <ancestor>).

Usage:

```
ceph osd crush unlink <name> {<ancestor>}
```

Subcommand df shows OSD utilization

Usage:

```
ceph osd df {plain|tree}
```

Subcommand deep-scrub initiates deep scrub on specified osd.

Usage:

```
ceph osd deep-scrub <who>
```

Subcommand down sets osd(s) <id> [<id>...] down.

Usage:

```
ceph osd down <ids> [<ids>...]
```

Subcommand dump prints summary of OSD map.

Usage:

```
ceph osd dump {<int[0-]>}
```

Subcommand erasure-code-profile is used for managing the erasure code profiles. It uses some additional subcommands.

Subcommand get gets erasure code profile <name>.

Usage:

```
ceph osd erasure-code-profile get <name>
```

Subcommand ls lists all erasure code profiles.

Usage:

```
ceph osd erasure-code-profile ls
```

Subcommand rm removes erasure code profile <name>.

Usage:

```
ceph osd erasure-code-profile rm <name>
```

Subcommand set creates erasure code profile <name> with [<key[=value]> ...] pairs. Add a –force at the end to override an existing profile (IT IS RISKY).

Usage:

```
ceph osd erasure-code-profile set <name> {<profile> [<profile>...]}
```

Subcommand find find osd <id> in the CRUSH map and shows its location.

Usage:

```
ceph osd find <int[0-]>
```

Subcommand getcrushmap gets CRUSH map.

Usage:

```
ceph osd getcrushmap {<int[0-]>}
```

Subcommand getmap gets OSD map.

Usage:

```
ceph osd getmap {<int[0-]>}
```

Subcommand getmaxosd shows largest OSD id.

Usage:

```
ceph osd getmaxosd
```

Subcommand `in` sets osd(s) <id> [<id>...] in.

Usage:

```
ceph osd in <ids> [<ids>...]
```

Subcommand `lost` marks osd as permanently lost. THIS DESTROYS DATA IF NO MORE REPLICAS EXIST, BE CAREFUL.

Usage:

```
ceph osd lost <int[0-]> {--yes-i-really-mean-it}
```

Subcommand `ls` shows all OSD ids.

Usage:

```
ceph osd ls {<int[0-]>}
```

Subcommand `lspools` lists pools.

Usage:

```
ceph osd lspools {<int>}
```

Subcommand `map` finds pg for <object> in <pool>.

Usage:

```
ceph osd map <poolname> <objectname>
```

Subcommand `metadata` fetches metadata for osd <id>.

Usage:

```
ceph osd metadata {int[0-]} (default all)
```

Subcommand `out` sets osd(s) <id> [<id>...] out.

Usage:

```
ceph osd out <ids> [<ids>...]
```

Subcommand `ok-to-stop` checks whether the list of OSD(s) can be stopped without immediately making data unavailable. That is, all data should remain readable and writeable, although data redundancy may be reduced as some PGs may end up in a degraded (but active) state. It will return a success code if it is okay to stop the OSD(s), or an error code and informative message if it is not or if no conclusion can be drawn at the current time.

Usage:

```
ceph osd ok-to-stop <id> [<ids>...]
```

Subcommand `pause` pauses osd.

Usage:

```
ceph osd pause
```

Subcommand `perf` prints dump of OSD perf summary stats.

Usage:

```
ceph osd perf
```

Subcommand `pg-temp` set pg_temp mapping pgid:[<id> [<id>...]] (developers only).

Usage:

```
ceph osd pg-temp <pgid> {<id> [<id>...]}
```

Subcommand `force-create-pg` forces creation of pg <pgid>.

Usage:

```
ceph osd force-create-pg <pgid>
```

Subcommand `pool` is used for managing data pools. It uses some additional subcommands.

Subcommand `create` creates pool.

Usage:

```
ceph osd pool create <poolname> <int[0-]> {<int[0-]>} {replicated|erasure}
{<erasure_code_profile>} {<rule>} {<int>}
```

Subcommand `delete` deletes pool.

Usage:

```
ceph osd pool delete <poolname> {<poolname>} {--yes-i-really-really-mean-it}
```

Subcommand `get` gets pool parameter <var>.

Usage:

```
ceph osd pool get <poolname> size|min_size|pg_num|
pgp_num|crush_rule|auid|write_fadvise_dontneed
```

Only for tiered pools:

```
ceph osd pool get <poolname> hit_set_type|hit_set_period|hit_set_count|hit_set_fpp|
target_max_objects|target_max_bytes|cache_target_dirty_ratio|cache_target_dirty_high_ratio|
cache_target_full_ratio|cache_min_flush_age|cache_min_evict_age|
min_read_recency_for_promote|hit_set_grade_decay_rate|hit_set_search_last_n
```

Only for erasure coded pools:

```
ceph osd pool get <poolname> erasure_code_profile
```

Use `all` to get all pool parameters that apply to the pool's type:

```
ceph osd pool get <poolname> all
```

Subcommand get-quota obtains object or byte limits for pool.

Usage:

```
ceph osd pool get-quota <poolname>
```

Subcommand ls list pools

Usage:

```
ceph osd pool ls {detail}
```

Subcommand mksnap makes snapshot <snap> in <pool>.

Usage:

```
ceph osd pool mksnap <poolname> <snap>
```

Subcommand rename renames <srcpool> to <destpool>.

Usage:

```
ceph osd pool rename <poolname> <poolname>
```

Subcommand rmsnap removes snapshot <snap> from <pool>.

Usage:

```
ceph osd pool rmsnap <poolname> <snap>
```

Subcommand set sets pool parameter <var> to <val>.

Usage:

```
ceph osd pool set <poolname> size|min_size|pg_num|
pgp_num|crush_rule|hashpspool|nodelete|nopgchange|nosizechange|
hit_set_type|hit_set_period|hit_set_count|hit_set_fpp|debug_fake_ec_pool|
target_max_bytes|target_max_objects|cache_target_dirty_ratio|
cache_target_dirty_high_ratio|
cache_target_full_ratio|cache_min_flush_age|cache_min_evict_age|auid|
min_read_recency_for_promote|write_fadvise_dontneed|hit_set_grade_decay_rate|
hit_set_search_last_n
<val> {--yes-i-really-mean-it}
```

Subcommand set-quota sets object or byte limit on pool.

Usage:

```
ceph osd pool set-quota <poolname> max_objects|max_bytes <val>
```

Subcommand stats obtain stats from all pools, or from specified pool.

Usage:

```
ceph osd pool stats {<name>}
```

Subcommand primary-affinity adjust osd primary-affinity from 0.0 <=<weight> <= 1.0

Usage:

```
ceph osd primary-affinity <osdname (id|osd.id)> <float[0.0-1.0]>
```

Subcommand `primary-temp` sets primary_temp mapping pgid:<id>|-1 (developers only).

Usage:

```
ceph osd primary-temp <pgid> <id>
```

Subcommand `repair` initiates repair on a specified osd.

Usage:

```
ceph osd repair <who>
```

Subcommand `reweight` reweights osd to 0.0 < <weight> < 1.0.

Usage:

```
osd reweight <int[0-]> <float[0.0-1.0]>
```

Subcommand `reweight-by-pg` reweight OSDs by PG distribution [overload-percentage-for-consideration, default 120].

Usage:

```
ceph osd reweight-by-pg {<int[100-]>} {<poolname> [<poolname...]}
{--no-increasing}
```

Subcommand `reweight-by-utilization` reweight OSDs by utilization [overload-percentage-for-consideration, default 120].

Usage:

```
ceph osd reweight-by-utilization {<int[100-]>}
{--no-increasing}
```

Subcommand `rm` removes osd(s) <id> [<id>...] from the OSD map.

Usage:

```
ceph osd rm <ids> [<ids>...]
```

Subcommand `destroy` marks OSD *id* as *destroyed*, removing its cephx entity's keys and all of its dm-crypt and daemon-private config key entries.

This command will not remove the OSD from crush, nor will it remove the OSD from the OSD map. Instead, once the command successfully completes, the OSD will show marked as *destroyed*.

In order to mark an OSD as destroyed, the OSD must first be marked as **lost**.

Usage:

```
ceph osd destroy <id> {--yes-i-really-mean-it}
```

Subcommand `purge` performs a combination of `osd destroy`, `osd rm` and `osd crush remove`.

Usage:

```
ceph osd purge <id> {--yes-i-really-mean-it}
```

Subcommand `safe-to-destroy` checks whether it is safe to remove or destroy an OSD without reducing overall data

redundancy or durability. It will return a success code if it is definitely safe, or an error code and informative message if it is not or if no conclusion can be drawn at the current time.

Usage:

```
ceph osd safe-to-destroy <id> [<ids>...]
```

Subcommand `scrub` initiates scrub on specified osd.

Usage:

```
ceph osd scrub <who>
```

Subcommand `set` sets <key>.

Usage:

```
ceph osd set full|pause|noup|nodown|noout|noin|nobackfill|
norebalance|norecover|noscrub|nodeep-scrub|notieragent
```

Subcommand `setcrushmap` sets crush map from input file.

Usage:

```
ceph osd setcrushmap
```

Subcommand `setmaxosd` sets new maximum osd value.

Usage:

```
ceph osd setmaxosd <int[0-]>
```

Subcommand `set-require-min-compat-client` enforces the cluster to be backward compatible with the specified client version. This subcommand prevents you from making any changes (e.g., crush tunables, or using new features) that would violate the current setting. Please note, This subcommand will fail if any connected daemon or client is not compatible with the features offered by the given <version>. To see the features and releases of all clients connected to cluster, please see ceph features.

Usage:

```
ceph osd set-require-min-compat-client <version>
```

Subcommand `stat` prints summary of OSD map.

Usage:

```
ceph osd stat
```

Subcommand `tier` is used for managing tiers. It uses some additional subcommands.

Subcommand add adds the tier <tierpool> (the second one) to base pool <pool> (the first one).

Usage:

```
ceph osd tier add <poolname> <poolname> {--force-nonempty}
```

Subcommand `add-cache` adds a cache <tierpool> (the second one) of size <size> to existing pool <pool> (the first one).

Usage:

```
ceph osd tier add-cache <poolname> <poolname> <int[0-]>
```

Subcommand cache-mode specifies the caching mode for cache tier <pool>.

Usage:

```
ceph osd tier cache-mode <poolname> none|writeback|forward|readonly|
readforward|readproxy
```

Subcommand remove removes the tier <tierpool> (the second one) from base pool <pool> (the first one).

Usage:

```
ceph osd tier remove <poolname> <poolname>
```

Subcommand remove-overlay removes the overlay pool for base pool <pool>.

Usage:

```
ceph osd tier remove-overlay <poolname>
```

Subcommand set-overlay set the overlay pool for base pool <pool> to be <overlaypool>.

Usage:

```
ceph osd tier set-overlay <poolname> <poolname>
```

Subcommand tree prints OSD tree.

Usage:

```
ceph osd tree {<int[0-]>}
```

Subcommand unpause unpauses osd.

Usage:

```
ceph osd unpause
```

Subcommand unset unsets <key>.

Usage:

```
ceph osd unset full|pause|noup|nodown|noout|noin|nobackfill|
norebalance|norecover|noscrub|nodeep-scrub|notieragent
```

PG

It is used for managing the placement groups in OSDs. It uses some additional subcommands.

Subcommand debug shows debug info about pgs.

Usage:

```
ceph pg debug unfound_objects_exist|degraded_pgs_exist
```

Subcommand deep-scrub starts deep-scrub on <pgid>.

Usage:

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

Subcommand dump shows human-readable versions of pg map (only ‘all’ valid with plain).

Usage:

```
ceph pg dump {all|summary|sum|delta|pools|osds|pgs|pgs_brief} [{all|summary|sum|delta|pools|o
```

Subcommand dump_json shows human-readable version of pg map in json only.

Usage:

```
ceph pg dump_json {all|summary|sum|delta|pools|osds|pgs|pgs_brief} [{all|summary|sum|delta|po
```

Subcommand dump_pools_json shows pg pools info in json only.

Usage:

```
ceph pg dump_pools_json
```

Subcommand dump_stuck shows information about stuck pgs.

Usage:

```
ceph pg dump_stuck {inactive|unclean|stale|undersized|degraded [inactive|unclean|stale|unders
{<int>}
```

Subcommand getmap gets binary pg map to -o/stdout.

Usage:

```
ceph pg getmap
```

Subcommand ls lists pg with specific pool, osd, state

Usage:

```
ceph pg ls {<int>} {active|clean|down|replay|splitting|
scrubbing|scrubq|degraded|inconsistent|peering|repair|
recovery|backfill_wait|incomplete|stale| remapped|
deep_scrub|backfill|backfill_toofull|recovery_wait|
undersized [active|clean|down|replay|splitting|
scrubbing|scrubq|degraded|inconsistent|peering|repair|
recovery|backfill_wait|incomplete|stale|remapped|
deep_scrub|backfill|backfill_toofull|recovery_wait|
undersized...]}
```

Subcommand ls-by-osd lists pg on osd [osd]

Usage:

```
ceph pg ls-by-osd <osdname (id|osd.id)> {<int>}
{active|clean|down|replay|splitting|
scrubbing|scrubq|degraded|inconsistent|peering|repair|
recovery|backfill_wait|incomplete|stale| remapped|
deep_scrub|backfill|backfill_toofull|recovery_wait|
undersized [active|clean|down|replay|splitting|
```

```
scrubbing|scrubq|degraded|inconsistent|peering|repair|
recovery|backfill_wait|incomplete|stale|remapped|
deep_scrub|backfill|backfill_toofull|recovery_wait|
undersized...]}
```

Subcommand `ls-by-pool` lists pg with pool = [poolname]

Usage:

```
ceph pg ls-by-pool <poolstr> {<int>} {active|
clean|down|replay|splitting|
scrubbing|scrubq|degraded|inconsistent|peering|repair|
recovery|backfill_wait|incomplete|stale| remapped|
deep_scrub|backfill|backfill_toofull|recovery_wait|
undersized [active|clean|down|replay|splitting|
scrubbing|scrubq|degraded|inconsistent|peering|repair|
recovery|backfill_wait|incomplete|stale|remapped|
deep_scrub|backfill|backfill_toofull|recovery_wait|
undersized...]}
```

Subcommand `ls-by-primary` lists pg with primary = [osd]

Usage:

```
ceph pg ls-by-primary <osdname (id|osd.id)> {<int>}
{active|clean|down|replay|splitting|
scrubbing|scrubq|degraded|inconsistent|peering|repair|
recovery|backfill_wait|incomplete|stale| remapped|
deep_scrub|backfill|backfill_toofull|recovery_wait|
undersized [active|clean|down|replay|splitting|
scrubbing|scrubq|degraded|inconsistent|peering|repair|
recovery|backfill_wait|incomplete|stale|remapped|
deep_scrub|backfill|backfill_toofull|recovery_wait|
undersized...]}
```

Subcommand `map` shows mapping of pg to osds.

Usage:

```
ceph pg map <pgid>
```

Subcommand `repair` starts repair on <pgid>.

Usage:

```
ceph pg repair <pgid>
```

Subcommand `scrub` starts scrub on <pgid>.

Usage:

```
ceph pg scrub <pgid>
```

Subcommand `stat` shows placement group status.

Usage:

```
ceph pg stat
```

QUORUM

Cause MON to enter or exit quorum.

Usage:

```
ceph quorum enter|exit
```

Note: this only works on the MON to which the ceph command is connected. If you want a specific MON to enter or exit quorum, use this syntax:

```
ceph tell mon.<id> quorum enter|exit
```

QUORUM_STATUS

Reports status of monitor quorum.

Usage:

```
ceph quorum_status
```

REPORT

Reports full status of cluster, optional title tag strings.

Usage:

```
ceph report {<tags> [<tags>...]}
```

SCRUB

Scrubs the monitor stores.

Usage:

```
ceph scrub
```

STATUS

Shows cluster status.

Usage:

```
ceph status
```

SYNC FORCE

Forces sync of and clear monitor store.

Usage:

```
ceph sync force {--yes-i-really-mean-it} {--i-know-what-i-am-doing}
```

TELL

Sends a command to a specific daemon.

Usage:

```
```

```
ceph tell <name (type.id)> <command> [options...]
```

List all available commands.

Usage:

```
ceph tell <name (type.id)> help
```

VERSION

Show mon daemon version

Usage:

```
ceph version
```

OPTIONS

-i `infile`
: will specify an input file to be passed along as a payload with the command to the monitor cluster. This is only used for specific monitor commands.

-o `outfile`
: will write any payload returned by the monitor cluster with its reply to outfile. Only specific monitor commands (e.g. osd getmap) return a payload.

-c `ceph.conf`, **--conf**=`ceph.conf`
: Use ceph.conf configuration file instead of the default `/etc/ceph/ceph.conf` to determine monitor addresses during startup.

--id `CLIENT_ID`, **--user** `CLIENT_ID`
: Client id for authentication.

--name `CLIENT_NAME`, **-n** `CLIENT_NAME`
: Client name for authentication.

--cluster `CLUSTER`
: Name of the Ceph cluster.

--admin-daemon `ADMIN_SOCKET`, **daemon** `DAEMON_NAME`
: Submit admin-socket commands via admin sockets in /var/run/ceph.

--admin-socket `ADMIN_SOCKET_NOPE`
: You probably mean –admin-daemon

-s, **--status**
: Show cluster status.

-w, **--watch**
: Watch live cluster changes.

--watch-debug
: Watch debug events.

--watch-info
: Watch info events.

--watch-sec
: Watch security events.

--watch-warn

Watch warning events.

--watch-error
Watch error events.

--version, **-v**
Display version.

--verbose
Make verbose.

--concise
Make less verbose.

-f `{json,json-pretty,xml,xml-pretty,plain}`, **--format**
Format of output.

--connect-timeout `CLUSTER_TIMEOUT`
Set a timeout for connecting to the cluster.

--no-increasing
`--no-increasing` is off by default. So increasing the osd weight is allowed using the `reweight-by-utilization` or `test-reweight-by-utilization` commands. If this option is used with these commands, it will help not to increase osd weight even the osd is under utilized.

--block
block until completion (scrub and deep-scrub only)

AVAILABILITY

ceph is part of Ceph, a massively scalable, open-source, distributed storage system. Please refer to the Ceph documentation at http://ceph.com/docs for more information.

SEE ALSO

ceph-mon(8), ceph-osd(8), ceph-mds(8)