# MONITOR BOOTSTRAP

Terminology:

- `cluster`: a set of monitors
- `quorum`: an active set of monitors consisting of a majority of the cluster

In order to initialize a new monitor, it must always be fed:

1. a logical name
2. secret keys
3. a cluster fsid (uuid)

In addition, a monitor needs to know two things:

1. what address to bind to
2. who its peers are (if any)

There are a range of ways to do both.

## LOGICAL ID

The logical id should be unique across the cluster. It will be appended to `mon.` to logically describe the monitor in the Ceph cluster. For example, if the logical id is `foo`, the monitor's name will be `mon.foo`.

For most users, there is no more than one monitor per host, which makes the short hostname logical choice.

## SECRET KEYS

The `mon.` secret key is stored a `keyring` file in the `mon data` directory. It can be generated with a command like:

```
ceph-authtool --create-keyring /path/to/keyring --gen-key -n mon.
```

When creating a new monitor cluster, the keyring should also contain a `client.admin` key that can be used to administer the system:

```
ceph-authtool /path/to/keyring --gen-key -n client.admin --set-uid=0 --cap mon 'allow *' --ca
```

The resulting keyring is fed to `ceph-mon --mkfs` with the `--keyring <keyring>` command-line argument.

## CLUSTER FSID

The cluster fsid is a normal uuid, like that generated by the `uuidgen` command. It can be provided to the monitor in two ways:

1. via the `--fsid <uuid>` command-line argument (or config file option)
2. via a monmap provided to the new monitor via the `--monmap <path>` command-line argument.

## MONITOR ADDRESS

The monitor address can be provided in several ways.

1. via the `--public-addr <ip[:port]>` command-line option (or config file option)
2. via the `--public-network <cidr>` command-line option (or config file option)
3. via the monmap provided via `--monmap <path>`, if it includes a monitor with our name
4. via the bootstrap monmap (provided via `--inject-monmap <path>` or generated from `--mon-host <list>`) if it includes a monitor with no name (`noname-<something>`) and an address configured on the local host.

## PEERS

The monitor peers are provided in several ways:

1. via the initial monmap, provided via `--monmap <filename>`
2. via the bootstrap monmap generated from `--mon-host <list>`
3. via the bootstrap monmap generated from `[mon.*]` sections with `mon addr` in the config file
4. dynamically via the admin socket

However, these methods are not completely interchangeable because of the complexity of creating a new monitor cluster without danger of races.

## CLUSTER CREATION

There are three basic approaches to creating a cluster:

1. Create a new cluster by specifying the monitor names and addresses ahead of time.
2. Create a new cluster by specifying the monitor names ahead of time, and dynamically setting the addresses as `ceph-mon` daemons configure themselves.
3. Create a new cluster by specifying the monitor addresses ahead of time.

### NAMES AND ADDRESSES

Generate a monmap using `monmaptool` with the names and addresses of the initial monitors. The generated monmap will also include a cluster fsid. Feed that monmap to each monitor daemon:

```
ceph-mon --mkfs -i <name> --monmap <initial_monmap> --keyring <initial_keyring>
```

When the daemons start, they will know exactly who they and their peers are.

### ADDRESSES ONLY

The initial monitor addresses can be specified with the `mon host` configuration value, either via a config file or the command-line argument. This method has the advantage that a single global config file for the cluster can have a line like:

```
mon host = a.foo.com, b.foo.com, c.foo.com
```

and will also serve to inform any ceph clients or daemons who the monitors are.

The `ceph-mon` daemons will need to be fed the initial keyring and cluster fsid to initialize themselves:

    ceph-mon –mkfs -i <name> –fsid <uuid> –keyring <initial_keyring>

When the daemons first start up, they will share their names with each other and form a new cluster.

### NAMES ONLY

In dynamic “cloud” environments, the cluster creator may not (yet) know what the addresses of the monitors are going to be. Instead, they may want machines to configure and start themselves in parallel and, as they come up, form a new cluster on their own. The problem is that the monitor cluster relies on strict majorities to keep itself consistent, and in order to “create” a new cluster, it needs to know what the *initial* set of monitors will be.

This can be done with the `mon initial members` config option, which should list the ids of the initial monitors that are allowed to create the cluster:

```
mon initial members = foo, bar, baz
```

The monitors can then be initialized by providing the other pieces of information (they keyring, cluster fsid, and a way of determining their own address). For example:

```
ceph-mon --mkfs -i <name> --mon-initial-hosts 'foo,bar,baz' --keyring <initial_keyring> --pub
```

When these daemons are started, they will know their own address, but not their peers. They can learn those addresses via the admin socket:

```
ceph daemon mon.<id> add_bootstrap_peer_hint <peer ip>
```

Once they learn enough of their peers from the initial member set, they will be able to create the cluster.

## CLUSTER EXPANSION

Cluster expansion is slightly less demanding than creation, because the creation of the initial quorum is not an issue and there is no worry about creating separately independent clusters.

New nodes can be forced to join an existing cluster in two ways:

1. by providing no initial monitor peers addresses, and feeding them dynamically.
2. by specifying the `mon initial members` config option to prevent the new nodes from forming a new, independent cluster, and feeding some existing monitors via any available method.

### INITIALLY PEERLESS EXPANSION

Create a new monitor and give it no peer addresses other than it's own. For example:

```
ceph-mon --mkfs -i <myid> --fsid <fsid> --keyring <mon secret key> --public-addr <ip>
```

Once the daemon starts, you can give it one or more peer addresses to join with:

```
ceph daemon mon.<id> add_bootstrap_peer_hint <peer ip>
```

This monitor will never participate in cluster creation; it can only join an existing cluster.

### EXPANDING WITH INITIAL MEMBERS

You can feed the new monitor some peer addresses initially and avoid badness by also setting `mon initial members`. For example:

```
ceph-mon --mkfs -i <myid> --fsid <fsid> --keyring <mon secret key> --public-addr <ip> --mon-h
```

When the daemon is started, `mon initial members` must be set via the command line or config file:

```
ceph-mon -i <myid> --mon-initial-members foo,bar,baz
```

to prevent any risk of split-brain.