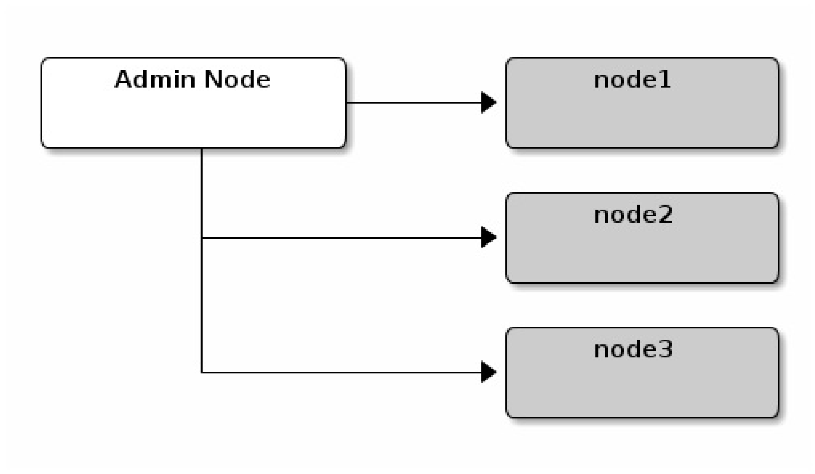
MANUAL DEPLOYMENT

All Ceph clusters require at least one monitor, and at least as many OSDs as copies of an object stored on the cluster. Bootstrapping the initial monitor(s) is the first step in deploying a Ceph Storage Cluster. Monitor deployment also sets important criteria for the entire cluster, such as the number of replicas for pools, the number of placement groups per OSD, the heartbeat intervals, whether authentication is required, etc. Most of these values are set by default, so it's useful to know about them when setting up your cluster for production.

Following the same configuration as Installation (Quick), we will set up a cluster with node1 as the monitor node, and node2 and node3 for OSD nodes.



MONITOR BOOTSTRAPPING

Bootstrapping a monitor (a Ceph Storage Cluster, in theory) requires a number of things:

- **Unique Identifier:** The `fsid` is a unique identifier for the cluster, and stands for File System ID from the days when the Ceph Storage Cluster was principally for the Ceph Filesystem. Ceph now supports native interfaces, block devices, and object storage gateway interfaces too, so `fsid` is a bit of a misnomer.
- **Cluster Name:** Ceph clusters have a cluster name, which is a simple string without spaces. The default cluster name is ceph, but you may specify a different cluster name. Overriding the default cluster name is especially useful when you are working with multiple clusters and you need to clearly understand which cluster your are working with.

  For example, when you run multiple clusters in a federated architecture, the cluster name (e.g., `us-west`, `us-east`) identifies the cluster for the current CLI session. **Note:** To identify the cluster name on the command line interface, specify the Ceph configuration file with the cluster name (e.g., `ceph.conf`, `us-west.conf`, `us-east.conf`, etc.). Also see CLI usage (`ceph --cluster {cluster-name}`).
- **Monitor Name:** Each monitor instance within a cluster has a unique name. In common practice, the Ceph Monitor name is the host name (we recommend one Ceph Monitor per host, and no commingling of Ceph OSD Daemons with Ceph Monitors). You may retrieve the short hostname with `hostname -s`.
- **Monitor Map:** Bootstrapping the initial monitor(s) requires you to generate a monitor map. The monitor map requires the `fsid`, the cluster name (or uses the default), and at least one host name and its IP address.
- **Monitor Keyring**: Monitors communicate with each other via a secret key. You must generate a keyring with a monitor secret and provide it when bootstrapping the initial monitor(s).
- **Administrator Keyring**: To use the ceph CLI tools, you must have a `client.admin` user. So you must generate the admin user and keyring, and you must also add the `client.admin` user to the monitor keyring.

The foregoing requirements do not imply the creation of a Ceph Configuration file. However, as a best practice, we recommend creating a Ceph configuration file and populating it with the `fsid`, the `mon initial members` and the `mon host` settings.

You can get and set all of the monitor settings at runtime as well. However, a Ceph Configuration file may contain only those settings that override the default values. When you add settings to a Ceph configuration file, these settings override the default settings. Maintaining those settings in a Ceph configuration file makes it easier to maintain your cluster.

The procedure is as follows:

1. Log in to the initial monitor node(s):

```
ssh {hostname}
```

For example:

```
ssh node1
```

2. Ensure you have a directory for the Ceph configuration file. By default, Ceph uses `/etc/ceph`. When you install ceph, the installer will create the `/etc/ceph` directory automatically.

   ```
   ls /etc/ceph
   ```

   Note: Deployment tools may remove this directory when purging a cluster (e.g., `ceph-deploy purgedata {node-name}`, `ceph-deploy purge {node-name}`).

3. Create a Ceph configuration file. By default, Ceph uses `ceph.conf`, where ceph reflects the cluster name.

   ```
   sudo vim /etc/ceph/ceph.conf
   ```

4. Generate a unique ID (i.e., `fsid`) for your cluster.

   ```
   uuidgen
   ```

5. Add the unique ID to your Ceph configuration file.

   ```
   fsid = {UUID}
   ```

   For example:

   ```
   fsid = a7f64266-0894-4f1e-a635-d0aeaca0e993
   ```

6. Add the initial monitor(s) to your Ceph configuration file.

   ```
   mon initial members = {hostname}[,{hostname}]
   ```

   For example:

   ```
   mon initial members = node1
   ```

7. Add the IP address(es) of the initial monitor(s) to your Ceph configuration file and save the file.

   ```
   mon host = {ip-address}[,{ip-address}]
   ```

   For example:

   ```
   mon host = 192.168.0.1
   ```

   Note: You may use IPv6 addresses instead of IPv4 addresses, but you must set `ms bind ipv6` to `true`. See Network Configuration Reference for details about network configuration.

8. Create a keyring for your cluster and generate a monitor secret key.

   ```
   ceph-authtool --create-keyring /tmp/ceph.mon.keyring --gen-key -n mon. --cap mon 'allow *
   ```

9. Generate an administrator keyring, generate a `client.admin` user and add the user to the keyring.

```
sudo ceph-authtool --create-keyring /etc/ceph/ceph.client.admin.keyring --gen-key -n clie
```

10. Generate a bootstrap-osd keyring, generate a `client.bootstrap-osd` user and add the user to the keyring.

```
sudo ceph-authtool --create-keyring /var/lib/ceph/bootstrap-osd/ceph.keyring --gen-key -n
```

11. Add the generated keys to the `ceph.mon.keyring`.

```
sudo ceph-authtool /tmp/ceph.mon.keyring --import-keyring /etc/ceph/ceph.client.admin.key
sudo ceph-authtool /tmp/ceph.mon.keyring --import-keyring /var/lib/ceph/bootstrap-osd/cep
```

12. Generate a monitor map using the hostname(s), host IP address(es) and the FSID. Save it as `/tmp/monmap`:

```
monmaptool --create --add {hostname} {ip-address} --fsid {uuid} /tmp/monmap
```

For example:

```
monmaptool --create --add node1 192.168.0.1 --fsid a7f64266-0894-4f1e-a635-d0aeaca0e993 /
```

13. Create a default data directory (or directories) on the monitor host(s).

```
sudo mkdir /var/lib/ceph/mon/{cluster-name}-{hostname}
```

For example:

```
sudo -u ceph mkdir /var/lib/ceph/mon/ceph-node1
```

See Monitor Config Reference - Data for details.

14. Populate the monitor daemon(s) with the monitor map and keyring.

```
sudo -u ceph ceph-mon [--cluster {cluster-name}] --mkfs -i {hostname} --monmap /tmp/monma
```

For example:

```
sudo -u ceph ceph-mon --mkfs -i node1 --monmap /tmp/monmap --keyring /tmp/ceph.mon.keyrin
```

15. Consider settings for a Ceph configuration file. Common settings include the following:

```
[global]
fsid = {cluster-id}
mon initial members = {hostname}[, {hostname}]
mon host = {ip-address}[, {ip-address}]
public network = {network}[, {network}]
cluster network = {network}[, {network}]
auth cluster required = cephx
auth service required = cephx
auth client required = cephx
osd journal size = {n}
osd pool default size = {n}  # Write an object n times.
osd pool default min size = {n} # Allow writing n copies in a degraded state.
osd pool default pg num = {n}
osd pool default pgp num = {n}
osd crush chooseleaf type = {n}
```

In the foregoing example, the `[global]` section of the configuration might look like this:

```
[global]
fsid = a7f64266-0894-4f1e-a635-d0aeaca0e993
mon initial members = node1
mon host = 192.168.0.1
public network = 192.168.0.0/24
auth cluster required = cephx
auth service required = cephx
auth client required = cephx
osd journal size = 1024
osd pool default size = 3
osd pool default min size = 2
osd pool default pg num = 333
osd pool default pgp num = 333
osd crush chooseleaf type = 1
```

16. Touch the done file.

    Mark that the monitor is created and ready to be started:

    ```
    sudo touch /var/lib/ceph/mon/ceph-node1/done
    ```

17. Start the monitor(s).

    For most distributions, services are started via systemd now:

    ```
    sudo systemctl start ceph-mon@node1
    ```

    For Ubuntu Trusty, use Upstart:

    ```
    sudo start ceph-mon id=node1 [cluster={cluster-name}]
    ```

    In this case, to allow the start of the daemon at each reboot you must create two empty files like this:

    ```
    sudo touch /var/lib/ceph/mon/{cluster-name}-{hostname}/upstart
    ```

    For example:

    ```
    sudo touch /var/lib/ceph/mon/ceph-node1/upstart
    ```

    For older Debian/CentOS/RHEL, use sysvinit:

    ```
    sudo /etc/init.d/ceph start mon.node1
    ```

18. Verify that the monitor is running.

    ```
    ceph -s
    ```

    You should see output that the monitor you started is up and running, and you should see a health error indicating that placement groups are stuck inactive. It should look something like this:

    ```
    cluster:
    ```

```
    id:     a7f64266-0894-4f1e-a635-d0aeaca0e993
    health: HEALTH_OK

  services:
    mon: 1 daemons, quorum node1
    mgr: node1(active)
    osd: 0 osds: 0 up, 0 in

  data:
    pools:   0 pools, 0 pgs
    objects: 0 objects, 0 bytes
    usage:   0 kB used, 0 kB / 0 kB avail
    pgs:
```

Note: Once you add OSDs and start them, the placement group health errors should disappear. See Adding OSDs for details.

MANAGER DAEMON CONFIGURATION

On each node where you run a ceph-mon daemon, you should also set up a ceph-mgr daemon.

See ceph-mgr administrator's guide

ADDING OSDS

Once you have your initial monitor(s) running, you should add OSDs. Your cluster cannot reach an `active + clean` state until you have enough OSDs to handle the number of copies of an object (e.g., `osd pool default size = 2` requires at least two OSDs). After bootstrapping your monitor, your cluster has a default CRUSH map; however, the CRUSH map doesn't have any Ceph OSD Daemons mapped to a Ceph Node.

SHORT FORM

Ceph provides the `ceph-volume` utility, which can prepare a logical volume, disk, or partition for use with Ceph. The `ceph-volume` utility creates the OSD ID by incrementing the index. Additionally, `ceph-volume` will add the new OSD to the CRUSH map under the host for you. Execute `ceph-volume -h` for CLI details. The `ceph-volume` utility automates the steps of the Long Form below. To create the first two OSDs with the short form procedure, execute the following on node2 and node3:

BLUESTORE

1. Create the OSD.

   ```
   ssh {node-name}
   sudo ceph-volume lvm create --data {data-path}
   ```

   For example:

   ```
   ssh node1
   sudo ceph-volume lvm create --data /dev/hdd1
   ```

Alternatively, the creation process can be split in two phases (prepare, and activate):

1. Prepare the OSD.

   ```
   ssh {node-name}
   sudo ceph-volume lvm prepare --data {data-path} {data-path}
   ```

   For example:

   ```
   ssh node1
   sudo ceph-volume lvm prepare --data /dev/hdd1
   ```

Once prepared, the `ID` and `FSID` of the prepared OSD are required for activation. These can be obtained by listing OSDs in the current server:

```
sudo ceph-volume lvm list
```

2. Activate the OSD:

```
sudo ceph-volume lvm activate {ID} {FSID}
```

For example:

```
sudo ceph-volume lvm activate 0 a7f64266-0894-4f1e-a635-d0aeaca0e993
```

FILESTORE

1. Create the OSD.

```
ssh {node-name}
sudo ceph-volume lvm create --filestore --data {data-path} --journal {journal-path}
```

For example:

```
ssh node1
sudo ceph-volume lvm create --filestore --data /dev/hdd1 --journal /dev/hdd2
```

Alternatively, the creation process can be split in two phases (prepare, and activate):

1. Prepare the OSD.

```
ssh {node-name}
sudo ceph-volume lvm prepare --filestore --data {data-path} --journal {journal-path}
```

For example:

```
ssh node1
sudo ceph-volume lvm prepare --filestore --data /dev/hdd1 --journal /dev/hdd2
```

Once prepared, the `ID` and `FSID` of the prepared OSD are required for activation. These can be obtained by listing OSDs in the current server:

```
sudo ceph-volume lvm list
```

2. Activate the OSD:

```
sudo ceph-volume lvm activate --filestore {ID} {FSID}
```

For example:

```
sudo ceph-volume lvm activate --filestore 0 a7f64266-0894-4f1e-a635-d0aeaca0e993
```

LONG FORM

Without the benefit of any helper utilities, create an OSD and add it to the cluster and CRUSH map with the following procedure. To create the first two OSDs with the long form procedure, execute the following steps for each OSD.

Note: This procedure does not describe deployment on top of dm-crypt making use of the dm-crypt 'lockbox'.

1. Connect to the OSD host and become root.

   ```
   ssh {node-name}
   sudo bash
   ```

2. Generate a UUID for the OSD.

   ```
   UUID=$(uuidgen)
   ```

3. Generate a cephx key for the OSD.

   ```
   OSD_SECRET=$(ceph-authtool --gen-print-key)
   ```

4. Create the OSD. Note that an OSD ID can be provided as an additional argument to `ceph osd new` if you need to reuse a previously-destroyed OSD id. We assume that the `client.bootstrap-osd` key is present on the machine. You may alternatively execute this command as `client.admin` on a different host where that key is present.:

   ```
   ID=$(echo "{\"cephx_secret\": \"$OSD_SECRET\"}" | \
      ceph osd new $UUID -i - \
      -n client.bootstrap-osd -k /var/lib/ceph/bootstrap-osd/ceph.keyring)
   ```

   It is also possible to include a `crush_device_class` property in the JSON to set an initial class other than the default (`ssd` or `hdd` based on the auto-detected device type).

5. Create the default directory on your new OSD.

   ```
   mkdir /var/lib/ceph/osd/ceph-$ID
   ```

6. If the OSD is for a drive other than the OS drive, prepare it for use with Ceph, and mount it to the directory you just created.

   ```
   mkfs.xfs /dev/{DEV}
   mount /dev/{DEV} /var/lib/ceph/osd/ceph-$ID
   ```

7. Write the secret to the OSD keyring file.

   ```
   ceph-authtool --create-keyring /var/lib/ceph/osd/ceph-$ID/keyring \
        --name osd.$ID --add-key $OSD_SECRET
   ```

8. Initialize the OSD data directory.

   ```
   ceph-osd -i $ID --mkfs --osd-uuid $UUID
   ```

9. Fix ownership.

   ```
   chown -R ceph:ceph /var/lib/ceph/osd/ceph-$ID
   ```

10. After you add an OSD to Ceph, the OSD is in your configuration. However, it is not yet running. You must start your new OSD before it can begin receiving data.

    For modern systemd distributions:

```
systemctl enable ceph-osd@$ID
systemctl start ceph-osd@$ID
```

For example:

```
systemctl enable ceph-osd@12
systemctl start ceph-osd@12
```

ADDING MDS

In the below instructions, `{id}` is an arbitrary name, such as the hostname of the machine.

1. Create the mds data directory.:

```
mkdir -p /var/lib/ceph/mds/{cluster-name}-{id}
```

2. Create a keyring.:

```
ceph-authtool --create-keyring /var/lib/ceph/mds/{cluster-name}-{id}/keyring --gen-key -r
```

3. Import the keyring and set caps.:

```
ceph auth add mds.{id} osd "allow rwx" mds "allow" mon "allow profile mds" -i /var/lib/ce
```

4. Add to ceph.conf.:

```
[mds.{id}]
host = {id}
```

5. Start the daemon the manual way.:

```
ceph-mds --cluster {cluster-name} -i {id} -m {mon-hostname}:{mon-port} [-f]
```

6. Start the daemon the right way (using ceph.conf entry).:

```
service ceph start
```

7. If starting the daemon fails with this error:

```
mds.-1.0 ERROR: failed to authenticate: (22) Invalid argument
```

Then make sure you do not have a keyring set in ceph.conf in the global section; move it to the client section; or add a keyring setting specific to this mds daemon. And verify that you see the same key in the mds data directory and `ceph auth get mds.{id}` output.

8. Now you are ready to create a Ceph filesystem.

SUMMARY

Once you have your monitor and two OSDs up and running, you can watch the placement groups peer by executing the following:

```
ceph -w
```

To view the tree, execute the following:

```
ceph osd tree
```

You should see output that looks something like this:

```
# id    weight  type name       up/down reweight
-1      2       root default
-2      2               host node1
0       1                       osd.0   up      1
-3      1               host node2
1       1                       osd.1   up      1
```

To add (or remove) additional monitors, see Add/Remove Monitors. To add (or remove) additional Ceph OSD Daemons, see Add/Remove OSDs.