CEPH-DEPLOY – CEPH DEPLOYMENT TOOL

SYNOPSIS

ceph-deploy new [*initial-monitor-node(s)*]

ceph-deploy install [*ceph-node*] [*ceph-node*…]

ceph-deploy mon *create-initial*

ceph-deploy osd *create –data device ceph-node*

ceph-deploy admin [*admin-node*][*ceph-node*…]

ceph-deploy purgedata [*ceph-node*][*ceph-node*…]

ceph-deploy forgetkeys

DESCRIPTION

ceph-deploy is a tool which allows easy and quick deployment of a Ceph cluster without involving complex and detailed manual configuration. It uses ssh to gain access to other Ceph nodes from the admin node, sudo for administrator privileges on them and the underlying Python scripts automates the manual process of Ceph installation on each node from the admin node itself. It can be easily run on an workstation and doesn't require servers, databases or any other automated tools. With **ceph-deploy**, it is really easy to set up and take down a cluster. However, it is not a generic deployment tool. It is a specific tool which is designed for those who want to get Ceph up and running quickly with only the unavoidable initial configuration settings and without the overhead of installing other tools like `Chef`, `Puppet` or `Juju`. Those who want to customize security settings, partitions or directory locations and want to set up a cluster following detailed manual steps, should use other tools i.e, `Chef`, `Puppet`, `Juju` or `Crowbar`.

With **ceph-deploy**, you can install Ceph packages on remote nodes, create a cluster, add monitors, gather/forget keys, add OSDs and metadata servers, configure admin hosts or take down the cluster.

COMMANDS

NEW

Start deploying a new cluster and write a configuration file and keyring for it. It tries to copy ssh keys from admin node to gain passwordless ssh to monitor node(s), validates host IP, creates a cluster with a new initial monitor node or nodes for monitor quorum, a ceph configuration file, a monitor secret keyring and a log file for the new cluster. It populates the newly created Ceph configuration file with `fsid` of cluster, hostnames and IP addresses of initial monitor members under `[global]` section.

Usage:

```
ceph-deploy new [MON][MON...]
```

Here, [MON] is the initial monitor hostname (short hostname i.e, `hostname -s`).

Other options like **`--no-ssh-copykey`**, **`--fsid`**, **`--cluster-network`** and **`--public-network`** can also be used with this command.

If more than one network interface is used, `public network` setting has to be added under `[global]` section of Ceph configuration file. If the public subnet is given, new command will choose the one IP from the remote host that exists within the subnet range. Public network can also be added at runtime using **`--public-network`** option with the command as mentioned above.

INSTALL

Install Ceph packages on remote hosts. As a first step it installs `yum-plugin-priorities` in admin and other nodes using passwordless ssh and sudo so that Ceph packages from upstream repository get more priority. It then detects the platform and distribution for the hosts and installs Ceph normally by downloading distro compatible packages if adequate repo for Ceph is already added. `--release` flag is used to get the latest release for installation. During detection of platform and distribution before installation, if it finds the `distro.init` to be `sysvinit` (Fedora, CentOS/RHEL etc), it doesn't allow installation with custom cluster name and uses the default name ceph for the cluster.

If the user explicitly specifies a custom repo url with **`--repo-url`** for installation, anything detected from the configuration will be overridden and the custom repository location will be used for installation of Ceph packages. If required, valid custom repositories are also detected and installed. In case of installation from a custom repo a boolean is used to determine the logic needed to proceed with a custom repo installation. A custom repo install helper is used that goes through config checks to retrieve repos (and any extra repos defined) and installs them. `cd_conf` is the object built from `argparse` that holds the flags and information needed to determine what metadata from the configuration is to be used.

A user can also opt to install only the repository without installing Ceph and its dependencies by using **`--repo`** option.

Usage:

```
ceph-deploy install [HOST][HOST...]
```

Here, [HOST] is/are the host node(s) where Ceph is to be installed.

An option `--release` is used to install a release known as CODENAME (default: firefly).

Other options like **`--testing`**, **`--dev`**, **`--adjust-repos`**, **`--no-adjust-repos`**, **`--repo`**, **`--local-mirror`**, **`--repo-url`** and **`--gpg-url`** can also be used with this command.

MDS

Deploy Ceph mds on remote hosts. A metadata server is needed to use CephFS and the `mds` command is used to create one on the desired host node. It uses the subcommand `create` to do so. `create` first gets the hostname and distro information of the desired mds host. It then tries to read the `bootstrap-mds` key for the cluster and deploy it in the desired host. The key generally has a format of `{cluster}.bootstrap-mds.keyring`. If it doesn't finds a keyring, it runs `gatherkeys` to get the keyring. It then creates a mds on the desired host under the path `/var/lib/ceph/mds/` in `/var/lib/ceph/mds/{cluster}-{name}` format and a bootstrap keyring under `/var/lib/ceph/bootstrap-mds/` in `/var/lib/ceph/bootstrap-mds/{cluster}.keyring` format. It then runs appropriate commands based on `distro.init` to start the `mds`.

Usage:

```
ceph-deploy mds create [HOST[:DAEMON-NAME]] [HOST[:DAEMON-NAME]...]
```

The [DAEMON-NAME] is optional.

MON

Deploy Ceph monitor on remote hosts. `mon` makes use of certain subcommands to deploy Ceph monitors on other nodes.

Subcommand `create-initial` deploys for monitors defined in `mon initial members` under `[global]` section in Ceph configuration file, wait until they form quorum and then gatherkeys, reporting the monitor status along the process. If monitors don't form quorum the command will eventually time out.

Usage:

```
ceph-deploy mon create-initial
```

Subcommand `create` is used to deploy Ceph monitors by explicitly specifying the hosts which are desired to be made monitors. If no hosts are specified it will default to use the `mon initial members` defined under `[global]` section of Ceph configuration file. `create` first detects platform and distro for desired hosts and checks if hostname is compatible for deployment. It then uses the monitor keyring initially created using `new` command and deploys the monitor in desired host. If multiple hosts were specified during `new` command i.e, if there are multiple hosts in `mon initial members` and multiple keyrings were created then a concatenated keyring is used for deployment of monitors. In this process a keyring parser is used which looks for `[entity]` sections in monitor keyrings and returns a list of those sections. A helper is then used to collect all keyrings into a single blob that will be used to inject it to monitors with **`--mkfs`** on remote nodes. All keyring files are concatenated to be in a directory ending with `.keyring`. During this process the helper uses list of sections returned by keyring parser to check if an entity is already present in a keyring and if not, adds it. The concatenated keyring is used for deployment of monitors to desired multiple hosts.

Usage:

```
ceph-deploy mon create [HOST] [HOST...]
```

Here, [HOST] is hostname of desired monitor host(s).

Subcommand add is used to add a monitor to an existing cluster. It first detects platform and distro for desired host and checks if hostname is compatible for deployment. It then uses the monitor keyring, ensures configuration for new monitor host and adds the monitor to the cluster. If the section for the monitor exists and defines a mon addr that will be used, otherwise it will fallback by resolving the hostname to an IP. If **`--address`** is used it will override all other options. After adding the monitor to the cluster, it gives it some time to start. It then looks for any monitor errors and checks monitor status. Monitor errors arise if the monitor is not added in `mon initial members`, if it doesn't exist in `monmap` and if neither `public_addr` nor `public_network` keys were defined for monitors. Under such conditions, monitors may not be able to form quorum. Monitor status tells if the monitor is up and running normally. The status is checked by running `ceph daemon mon.hostname mon_status` on remote end which provides the output and returns a boolean status of what is going on. `False` means a monitor that is not fine even if it is up and running, while `True` means the monitor is up and running correctly.

Usage:

```
ceph-deploy mon add [HOST]

ceph-deploy mon add [HOST] --address [IP]
```

Here, [HOST] is the hostname and [IP] is the IP address of the desired monitor node. Please note, unlike other `mon` subcommands, only one node can be specified at a time.

Subcommand `destroy` is used to completely remove monitors on remote hosts. It takes hostnames as arguments. It stops the monitor, verifies if `ceph-mon` daemon really stopped, creates an archive directory `mon-remove` under `/var/lib/ceph/`, archives old monitor directory in `{cluster}-{hostname}-{stamp}` format in it and removes the monitor from cluster by running `ceph remove...` command.

Usage:

```
ceph-deploy mon destroy [HOST] [HOST...]
```

Here, [HOST] is hostname of monitor that is to be removed.

GATHERKEYS

Gather authentication keys for provisioning new nodes. It takes hostnames as arguments. It checks for and fetches `client.admin` keyring, monitor keyring and `bootstrap-mds/bootstrap-osd` keyring from monitor host. These authentication keys are used when new `monitors/OSDs/MDS` are added to the cluster.

Usage:

```
ceph-deploy gatherkeys [HOST] [HOST...]
```

Here, [HOST] is hostname of the monitor from where keys are to be pulled.

DISK

Manage disks on a remote host. It actually triggers the `ceph-disk` utility and it's subcommands to manage disks.

Subcommand `list` lists disk partitions and Ceph OSDs.

Usage:

```
ceph-deploy disk list HOST
```

Subcommand zap zaps/erases/destroys a device’s partition table and contents. It actually uses ceph-volume lvm zap remotely, alternatively allowing someone to remove the Ceph metadata from the logical volume.

OSD

Manage OSDs by preparing data disk on remote host. osd makes use of certain subcommands for managing OSDs.

Subcommand create prepares a device for Ceph OSD. It first checks against multiple OSDs getting created and warns about the possibility of more than the recommended which would cause issues with max allowed PIDs in a system. It then reads the bootstrap-osd key for the cluster or writes the bootstrap key if not found. It then uses **ceph-volume** utility’s lvm create subcommand to prepare the disk, (and journal if using filestore) and deploy the OSD on the desired host. Once prepared, it gives some time to the OSD to start and checks for any possible errors and if found, reports to the user.

Bluestore Usage:

```
ceph-deploy osd create --data DISK HOST
```

Filestore Usage:

```
ceph-deploy osd create --data DISK --journal JOURNAL HOST
```

Note: For other flags available, please see the man page or the –help menu on ceph-deploy osd create

Subcommand list lists devices associated to Ceph as part of an OSD. It uses the ceph-volume lvm list output that has a rich output, mapping OSDs to devices and other interesting information about the OSD setup.

Usage:

```
ceph-deploy osd list HOST
```

ADMIN

Push configuration and client.admin key to a remote host. It takes the {cluster}.client.admin.keyring from admin node and writes it under /etc/ceph directory of desired node.

Usage:

```
ceph-deploy admin [HOST] [HOST...]
```

Here, [HOST] is desired host to be configured for Ceph administration.

CONFIG

Push/pull configuration file to/from a remote host. It uses push subcommand to takes the configuration file from admin host and write it to remote host under /etc/ceph directory. It uses pull subcommand to do the opposite i.e, pull the configuration file under /etc/ceph directory of remote host to admin node.

Usage:

```
ceph-deploy config push [HOST] [HOST...]

ceph-deploy config pull [HOST] [HOST...]
```

Here, [HOST] is the hostname of the node where config file will be pushed to or pulled from.

UNINSTALL

Remove Ceph packages from remote hosts. It detects the platform and distro of selected host and uninstalls Ceph packages from it. However, some dependencies like `librbd1` and `librados2` will not be removed because they can cause issues with `qemu-kvm`.

Usage:

```
ceph-deploy uninstall [HOST] [HOST...]
```

Here, [HOST] is hostname of the node from where Ceph will be uninstalled.

PURGE

Remove Ceph packages from remote hosts and purge all data. It detects the platform and distro of selected host, uninstalls Ceph packages and purges all data. However, some dependencies like `librbd1` and `librados2` will not be removed because they can cause issues with `qemu-kvm`.

Usage:

```
ceph-deploy purge [HOST] [HOST...]
```

Here, [HOST] is hostname of the node from where Ceph will be purged.

PURGEDATA

Purge (delete, destroy, discard, shred) any Ceph data from `/var/lib/ceph`. Once it detects the platform and distro of desired host, it first checks if Ceph is still installed on the selected host and if installed, it won't purge data from it. If Ceph is already uninstalled from the host, it tries to remove the contents of `/var/lib/ceph`. If it fails then probably OSDs are still mounted and needs to be unmounted to continue. It unmount the OSDs and tries to remove the contents of `/var/lib/ceph` again and checks for errors. It also removes contents of `/etc/ceph`. Once all steps are successfully completed, all the Ceph data from the selected host are removed.

Usage:

```
ceph-deploy purgedata [HOST] [HOST...]
```

Here, [HOST] is hostname of the node from where Ceph data will be purged.

FORGETKEYS

Remove authentication keys from the local directory. It removes all the authentication keys i.e, monitor keyring, client.admin keyring, bootstrap-osd and bootstrap-mds keyring from the node.

Usage:

```
ceph-deploy forgetkeys
```

PKG

Manage packages on remote hosts. It is used for installing or removing packages from remote hosts. The package names for installation or removal are to be specified after the command. Two options **`--install`** and **`--remove`** are used for this purpose.

Usage:

```
ceph-deploy pkg --install [PKGs] [HOST] [HOST...]

ceph-deploy pkg --remove [PKGs] [HOST] [HOST...]
```

Here, [PKGs] is comma-separated package names and [HOST] is hostname of the remote node where packages are to be installed or removed from.

CALAMARI

Install and configure Calamari nodes. It first checks if distro is supported for Calamari installation by ceph-deploy. An argument `connect` is used for installation and configuration. It checks for `ceph-deploy` configuration file (cd_conf) and Calamari release repo or `calamari-minion` repo. It relies on default for repo installation as it doesn't install Ceph unless specified otherwise. `options` dictionary is also defined because `ceph-deploy` pops items internally which causes issues when those items are needed to be available for every host. If the distro is Debian/Ubuntu, it is ensured that proxy is disabled for `calamari-minion` repo. `calamari-minion` package is then installed and custom repository files are added. minion config is placed prior to installation so that it is present when the minion first starts. config directory, calamari salt config are created and salt-minion package is installed. If the distro is Redhat/CentOS, the salt-minion service needs to be started.

Usage:

```
ceph-deploy calamari {connect} [HOST] [HOST...]
```

Here, [HOST] is the hostname where Calamari is to be installed.

An option `--release` can be used to use a given release from repositories defined in **ceph-deploy**'s configuration. Defaults to `calamari-minion`.

Another option **`--master`** can also be used with this command.

OPTIONS

`--address`
IP address of the host node to be added to the cluster.

`--adjust-repos`
Install packages modifying source repos.

`--ceph-conf`
Use (or reuse) a given `ceph.conf` file.

`--cluster`
Name of the cluster.

`--dev`
Install a bleeding edge built from Git branch or tag (default: master).

`--cluster-network`
Specify the (internal) cluster network.

`--dmcrypt`
Encrypt [data-path] and/or journal devices with `dm-crypt`.

`--dmcrypt-key-dir`
Directory where `dm-crypt` keys are stored.

`--install`
Comma-separated package(s) to install on remote hosts.

`--fs-type`
Filesystem to use to format disk `(xfs, btrfs or ext4)`. Note that support for btrfs and ext4 is no longer tested or recommended; please use xfs.

`--fsid`
Provide an alternate FSID for `ceph.conf` generation.

`--gpg-url`
Specify a GPG key url to be used with custom repos (defaults to ceph.com).

`--keyrings`
Concatenate multiple keyrings to be seeded on new monitors.

`--local-mirror`

Fetch packages and push them to hosts for a local repo mirror.

`--master`

The domain for the Calamari master server.

`--mkfs`

Inject keys to MONs on remote nodes.

`--no-adjust-repos`

Install packages without modifying source repos.

`--no-ssh-copykey`

Do not attempt to copy ssh keys.

`--overwrite-conf`

Overwrite an existing conf file on remote host (if present).

`--public-network`

Specify the public network for a cluster.

`--remove`

Comma-separated package(s) to remove from remote hosts.

`--repo`

Install repo files only (skips package installation).

`--repo-url`

Specify a repo url that mirrors/contains Ceph packages.

`--testing`

Install the latest development release.

`--username`

The username to connect to the remote host.

`--version`

The current installed version of **ceph-deploy**.

`--zap-disk`

Destroy the partition table and content of a disk.

AVAILABILITY

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

SEE ALSO

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