8. Ceph Deployment

Ceph, an open source distributed object, block, and file storage, is a de facto standard block storage solution for OpenStack. Ceph can act as storage backend for OpenStack Block Storage (Cinder), OpenStack Image service (Glance). In addition, Ceph Storage can be a drop-in replacement for OpenStack Object Storage (Swift).

Chapter Details	
Chapter Goal	Deploy and use a standalone Ceph cluster
Chapter Sections	*6.1. Prepare virtual lab* *8.2. Deploy Ceph* *8.3. Use Ceph*

8.1. Prepare virtual lab

You're starting from a clean VM that will act like a physical host. We will create several VMs: master node, 3 controllers, 1 compute. MCP service will be deployed over that set of VMs.

8.1.1. Get the model

Step 1 Download the model and provisioning scripts:

```
stack@lab:~$ git clone git@bitbucket.org:mirantis-training/mcp100-ceph-ha.git
```

Enter Yes if it asks about wrong fingerprints.

Change the directory;

```
stack@lab:~$ cd mcp100-ceph-ha/
```

Step 2 Create virtual networks by executing provided script:

```
stack@lab:~/mcp100-ceph-ha$ ./provision-networks.sh
```

This will create *br_pxe*, *br_mgmt*, *br_ctl*, *br_ext* networks which correspond to PXE, management, control and external

Step 3 Install prerequisites:

```
stack@lab:~/mcp100-ceph-ha$ sudo apt-get update

stack@lab:~/mcp100-ceph-ha$ sudo apt-get install -y git mkisofs curl virtinst cpu-check
```

8.1.2. Prepare cfg01

The deployment will start from preparing configuration ISO which will be used later to bootstrap master node. We will prepare the model, download some handy scripts and generate ISO.

Step 1 Copy the model:

```
stack@lab:~/mcp100-ceph-ha$ sudo su
root@lab:/home/stack/mcp100-ceph-ha# mkdir /root/model
root@lab:/home/stack/mcp100-ceph-ha# cp -rT /home/stack/mcp100-ceph-ha/ /root/model/
```

Step 2 Download *cfg01*:

```
root@lab:/home/stack/mcp100-ceph-ha# wget http://repos/cfg01-day01-2018.8.0.qcow2 -O \
/pool/images/cfg01.ceph-ha.local.img
```

The image will be used to bootstrap master node, however before starting VM we need to generate configuration ISO.

Step 3 Download generation script and cloud-init for ISO:

Choose MCP version:

```
root@lab:/home/stack/mcp100-ceph-ha# export MCP_VERSION="2018.8.0"
```

Download ISO generation script and set permissions:

```
root@lab:/home/stack/mcp100-ceph-ha# wget -O /root/create-config-drive \
http://repos/create_config_drive_${MCP_VERSION}.sh

root@lab:/home/stack/mcp100-ceph-ha# chmod +x /root/create-config-drive
```

Download cloud-init:

```
root@lab:/home/stack/mcp100-ceph-ha# wget -O /root/user_data.yaml \
http://repos/master_config_${MCP_VERSION}.yaml
```

Step 4 Make the following changes to `/root/user_data.yaml`:

```
export SALT_MASTER_DEPLOY_IP=${SALT_MASTER_DEPLOY_IP:-"192.168.10.100"}
export SALT_MASTER_MINION_ID=${SALT_MASTER_MINION_ID:-"cfg01.ceph-ha.local"}
export DEPLOY_NETWORK_GW=${DEPLOY_NETWORK_GW:-"192.168.10.1"}
export DEPLOY_NETWORK_NETMASK=${DEPLOY_NETWORK_NETMASK:-"255.255.255.0"}
export DEPLOY_NETWORK_MTU=${DEPLOY_NETWORK_MTU:-"1450"}
export DNS_SERVERS=${DNS_SERVERS:-"172.19.0.6"}
```

Step 5 Download Drivetrain pipelines:

```
root@lab:~# git clone https://github.com/Mirantis/mk-pipelines.git /root/mk-pipelines
root@lab:~# git clone https://github.com/Mirantis/pipeline-library.git /root/pipeline-l
root@lab:~# pushd /root/mk-pipelines && git checkout tags/${MCP_VERSION} ; popd
root@lab:~# pushd /root/pipeline-library && git checkout tags/${MCP_VERSION} ; popd
```

Step 6 Check that you have all files required:

```
root@lab:/home/stack/mcp100-ceph-ha# ls /root/
create-config-drive  mk-pipelines  model  pipeline-library  user_data.yaml
```

Step 7 Generate configuration ISO:

```
root@lab:/home/stack/mcp100-ceph-ha# /root/create-config-drive -u /root/user_data.yaml
--model /root/model --mk-pipelines /root/mk-pipelines \
--pipeline-library /root/pipeline-library /pool/images/cfg01.ceph-ha.local.iso
```

Configuration ISO should be ready and can be found at `/pool/images/cfg01.ceph-ha.local.iso`

8.1.3. Create cfg01 VM

Step 1 Use *virt-install* to create *cfg01* VM:

```
root@lab:/home/stack/mcp100-ceph-ha# virt-install --name cfg01.ceph-ha.local \
--disk path=/pool/images/cfg01.ceph-ha.local.img,bus=virtio,format=qcow2,cache=none \
--disk path=/pool/images/cfg01.ceph-ha.local.iso,device=cdrom \
--network network:br_pxe,model=virtio \
--network network:br_mgmt,model=virtio \
--network network:br_ctl,model=virtio \
--network network:br_ext,model=virtio \
--ram 8192 --vcpus=4 --accelerate \
--boot hd --vnc --noreboot --autostart

Starting install...
Creating domain...
Domain creation completed. You can restart your domain by running:
  virsh --connect qemu:///system start cfg01.ceph-ha.local
```

The VM is created but not started yet. We will need a small update to configuration.

Step 2 Update networking:

```
root@lab:/home/stack/mcp100-ceph-ha# virsh net-update br_pxe add ip-dhcp-host "\
<host mac='$(virsh domiflist cfg01.ceph-ha.local | grep br_pxe | awk '{print $5}')' nam
" --live --config
```

We've changed the record in libvirt DHCP service to get a fixed IP address for *cfg01*.

Step 3 Start cfg01 VM:

```
root@lab:/home/stack/mcp100-ceph-ha# virsh start cfg01.ceph-ha.local
```

Step 4 Login to *cfg01* and wait until cloud-init is over. The process takes about 10 min to complete. VM should perform a restart at the end:

```
root@cfg01:~# virsh console cfg01.ceph-ha.local
```

Step 5 Perform initial provisioning of salt master:

Patch linux.system.repo state:

```
root@cfg01:~# wget http://repos/repo.sls -O /srv/salt/env/prd/linux/system/repo.sls
```

Sync changes and run initial states:

```
root@cfg01:~# salt '*' saltutil.sync_all

root@cfg01:~# salt-call state.sls linux.system,linux,openssh,salt,reclass,jenkins
```

```
root@cfg01:~# salt '*' saltutil.sync_all
```

Step 6 Update pipelines:

```
root@cfg01:~# pushd /home/repo/mk/mk-pipelines && \
git init && git add -A && git commit -m 'initial commit' ; popd

root@cfg01:~# pushd /home/repo/mcp-ci/pipeline-library && \
git init && git add -A && git commit -m 'initial commit' ; popd
```

Patch linux.system.repo state:

```
root@cfg01:~# wget http://repos/repo.sls -O /srv/salt/env/prd/linux/system/repo.sls
```

Step 7 Exit to the host by pressing *CTRL +]*:

```
root@lab:/home/stack/mcp100-ceph-ha#
```

8.1.4. Create other VMs

Step 1 Create VMs:

```
root@lab:/home/stack/mcp100-ceph-ha# ./provision-cluster.sh
```

The script will spawn several new VMs from the base ubuntu cloud image. The images will be supplied with cloud init that installs salt-minion and targets to the salt-master.

8.2. Deploy Ceph

The deployment will be done using Jenkins pipelines. The pipeline will apply a set of salt states that bring the system to desired state with 3 cmnd nodes and 2 osd nodes.

8.2.1. Run Jenkins pipeline to install ceph

Step 1 Open browser and navigate to http://192.168.10.100:8081

Login as **admin/r00tme**.

Step 2 Launch pipeline “Deploy - OpenStack” and leave all parameters default except STACK_INSTALL:

STACK_INSTALL: core,ceph

The pipeline takes around 10 minutes to complete

8.3. Use Ceph

After successful ceph cluster deployment, in this exercise we:

- check the cluster status
- install ceph client

- create image
- create a block device from the image on the client
- crate and mount filesystem on the block device

8.3.1. Using ceph CLI

Step 1: Login to *cfg01*:

```
root@cfg01:~# virsh console cfg01.ceph-ha.local
```

Step 2 Get the status of cluster:

```
root@cfg01:~# salt "cmn01*" cmd.run 'ceph status'

cmn01.ceph-ha.local:
      cluster:
        id:     a619c5fc-c4ed-4f22-9ed2-66cf2feca23d
        health: HEALTH_WARN
                Degraded data redundancy: 96 pgs undersized

      services:
        mon: 3 daemons, quorum cmn01,cmn02,cmn03
        mgr: cmn01(active), standbys: cmn02, cmn03
        osd: 4 osds: 4 up, 4 in

      data:
        pools:   3 pools, 96 pgs
        objects: 0 objects, 0B
        usage:   4.02GiB used, 15.6GiB / 19.6GiB avail
        pgs:     96 active+undersized
```

Step 3 Show available space:

```
root@cfg01:~# salt "cmn01*" cmd.run 'ceph df'
cmn01.ceph-ha.local:
    GLOBAL:
        SIZE        AVAIL       RAW USED     %RAW USED
        19.6GiB     15.6GiB      4.02GiB         20.49
    POOLS:
        NAME        ID     USED      %USED     MAX AVAIL     OBJECTS
        images      1        0B          0       4.87GiB           0
        vms         2        0B          0       4.87GiB           0
        volumes     3        0B          0       4.87GiB           0
```

8.3.2. Deploy ceph(rbd) client

In the next steps we will install and setup ceph client on *cfg01* node.

Step 1 Add a new user (keyring) with rights to use the pool “volumes” in `/srv/salt/reclass/classes/cluster/ceph-ha/ceph/common.yml`:

```
parameters:
  ceph:
    common:
      keyring:
        cfga:
          caps:
            mon: "allow r"
            osd: "allow class-read object_prefix rbd_children, allow rwx pool=volumes"
```

Edit `/srv/salt/reclass/classes/cluster/ceph-ha/ceph/common.yml`:

```
classes:
- system.linux.system.repo.ceph
- cluster.ceph-ha.infra
parameters:
  ceph:
    common:
      public_network: 172.16.10.0/24
#      cluster_network: 10.16.0.0/24
      keyring:
        cfga:
          caps:
            mon: "allow r"
            osd: "allow class-read object_prefix rbd_children, allow rwx pool=volumes"
```

Step 2 Add ceph client (common) role to node cfg01 in `/srv/salt/reclass/nodes/cfg01.ceph-ha.local.yml`:

```
classes:
- cluster.ceph-ha.infra.config
- system.ceph.common.cluster
- cluster.ceph-ha.ceph.common
```

Step 3 Sync changes:

```
root@cfg01:~# salt '*' saltutil.sync_all
```

Step 4 Add ceph repo on *cfg01*:

```
root@cfg01:~# salt-call state.sls linux
```

Step 5 Add “cfga” keyring:

```
root@cfg01:~# salt "cmn*" state.sls ceph
```

Step 6 Deploy ceph client on *cfg01*:

```
root@cfg01:~# salt-call state.sls ceph
```

8.3.3. Create image

Step 1 Create image “test” on pool “volumes”:

```
root@cfg01:~# rbd -c /etc/ceph/ceph.conf -k /etc/ceph/ceph.client.cfga.keyring --id cfg
```

Step 2 List images on pool “volumes”:

```
root@cfg01:~# rbd -c /etc/ceph/ceph.conf -k /etc/ceph/ceph.client.cfga.keyring --id cfg
test
```

Step 3 Check details of image ‘test’:

```
root@cfg01:~# rbd -c /etc/ceph/ceph.conf -k /etc/ceph/ceph.client.cfga.keyring --id cfg
rbd image 'test':
    size 400 MB in 100 objects
    order 22 (4096 kB objects)
    block_name_prefix: rbd_data.10d06b8b4567
    format: 2
    features: layering, exclusive-lock, object-map, fast-diff, deep-flatten
    flags:
```

Step 4 Disable some image features:

```
root@cfg01:~# rbd -c /etc/ceph/ceph.conf -k /etc/ceph/ceph.client.cfga.keyring --id cfg
volumes/test object-map fast-diff deep-flatten exclusive-lock
```

Step 5 Check details again:

```
root@cfg01:~# rbd -c /etc/ceph/ceph.conf -k /etc/ceph/ceph.client.cfga.keyring --id cfg
rbd image 'test':
    size 400 MB in 100 objects
    order 22 (4096 kB objects)
    block_name_prefix: rbd_data.10d06b8b4567
    format: 2
    features: layering
    flags:
```

8.3.4. Create a block device backed by image

Step 1 Create block device:

```
root@cfg01:~# rbd -c /etc/ceph/ceph.conf -k /etc/ceph/ceph.client.cfga.keyring --id cfg
[81338.815408] libceph: mon2 172.16.10.98:6789 feature set mismatch, my 106b84a842a42 <
rbd: sysfs write failed
In some cases useful info is found in syslog - try "dmesg | tail" or so.
rbd: map failed: (110) Connection timed out
root@cfg01:~# salt "cmn01*" cmd.run 'ceph osd crush tunables hammer'
```

The command will fail due missing required protocol features

Step 2 Adjust tunables profile to hammer to be able to use *cfg01* kernel:

```
root@cfg01:~# salt "cmn01*" cmd.run 'ceph osd crush tunables hammer'
cmn01.ceph-ha.local:
    adjusted tunables profile to hammer
```

Step 3 Repeat the command:

```
root@cfg01:~# rbd -c /etc/ceph/ceph.conf -k /etc/ceph/ceph.client.cfga.keyring --id cfg
/dev/rbd0
```

8.3.5. Create, mount and test filesystem

Step 1 Look up for device:

```
root@cfg01:~# ls -l /dev/rbd0
brw-rw---- 1 root disk 251, 0 Dec 20 03:18 /dev/rbd0
```

Step 2 Format device:

```
root@cfg01:~# mkfs.ext4 -m0 /dev/rbd0
mke2fs 1.42.13 (17-May-2015)
Discarding device blocks: done
Creating filesystem with 409600 1k blocks and 102400 inodes
Filesystem UUID: d21f8cc8-690d-4d60-b96d-8345e5ddd634
Superblock backups stored on blocks:
    8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

Step 3 Mount device:

```
root@cfg01:~# mkdir /var/test

root@cfg01:~# mount /dev/rbd0 /var/test
```

Step 4 Create file and print it:

```
root@cfg01:~# echo test > /var/test/file

root@cfg01:~# cat /var/test/file
test
```

Step 5 Check disk usage again:

```
root@cfg01:~# salt "cmn01*" cmd.run 'ceph df'

cmn01.ceph-ha.local:
GLOBAL:
    SIZE        AVAIL       RAW USED     %RAW USED
    19.6GiB     15.6GiB      4.04GiB         20.59
POOLS:
    NAME        ID     USED        %USED     MAX AVAIL     OBJECTS
    images      1          0B          0       4.86GiB           0
    vms         2          0B          0       4.86GiB           0
    volumes     3      12.6MiB       0.21       4.86GiB          20
```